Food Delivery Time Prediction

Using Naive Bayes, K-Nearest Neighbors, and Decision Tree

Machine Learning Mini Project

Introduction

The Digital Food Revolution

Online food delivery platforms have experienced exponential growth in recent years, transforming how consumers access meals. With millions of orders processed daily, these platforms face increasing pressure to meet customer expectations.

The ability to accurately predict delivery times has become a critical competitive advantage in this fast-paced industry.

Customer Satisfaction

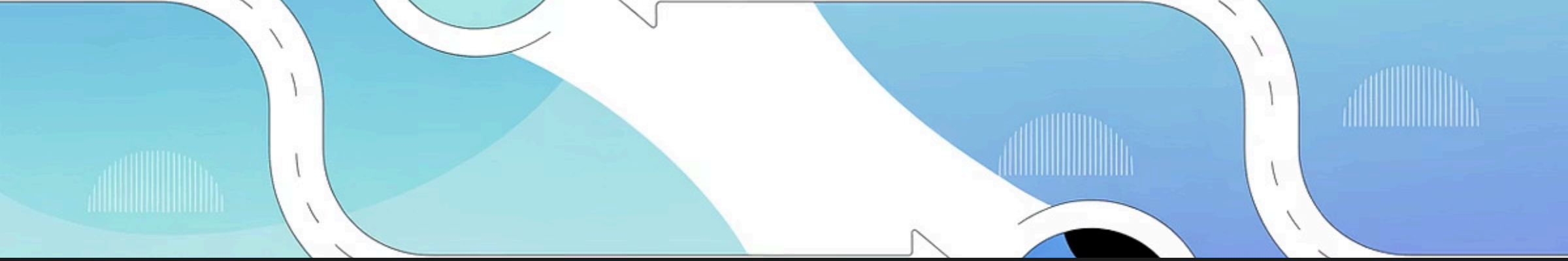
Accurate delivery predictions directly impact user experience and retention rates

Operational Efficiency

Better forecasting enables optimal resource allocation and route planning

Business Growth

Reliable service builds trust and drives platform adoption



Problem Statement & Objective

Research Question

Can we accurately predict whether a food delivery will arrive on time or be delayed based on order characteristics and environmental factors?

Binary Classification

Two-class prediction problem

Fast Delivery (0)

Orders arriving on time

Delayed Delivery (1)

Orders arriving late

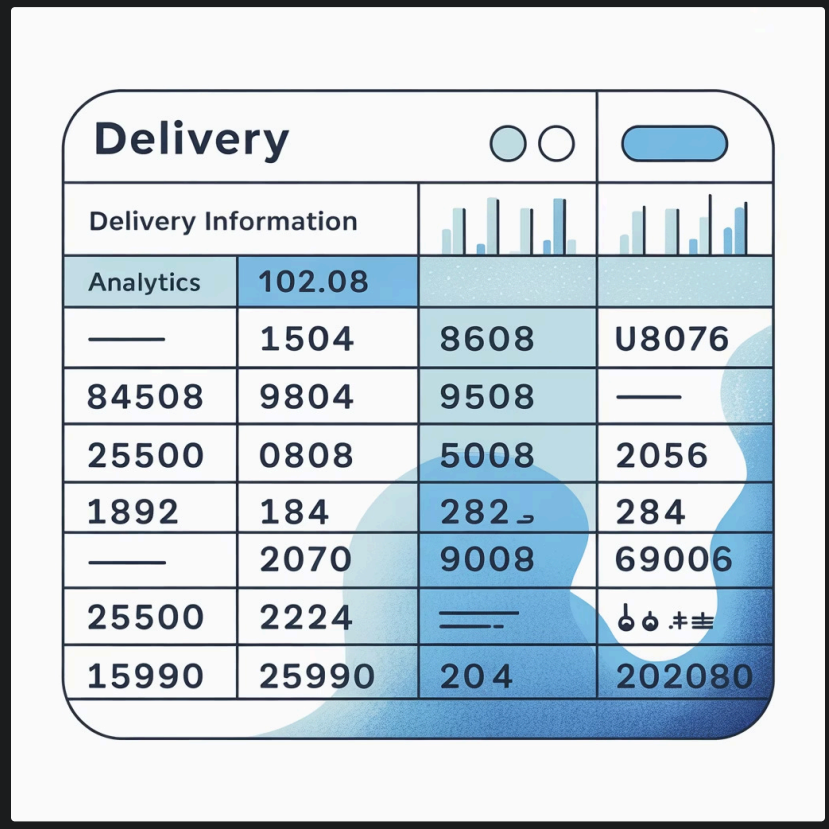
This classification enables proactive communication with customers and helps delivery platforms optimize their operations by identifying potential delays before they occur.

Dataset Overview

Data Characteristics

Our analysis utilizes a comprehensive food delivery dataset containing real-world order information. The dataset captures multiple dimensions of the delivery process, from order placement through final delivery.

Each record represents a single delivery transaction with associated features that influence delivery time outcomes.



Distance

Geographical separation between restaurant and customer location

Weather Conditions

Environmental factors affecting delivery speed

Traffic Density

Road congestion levels during delivery window

Order Priority

Urgency classification of delivery requests

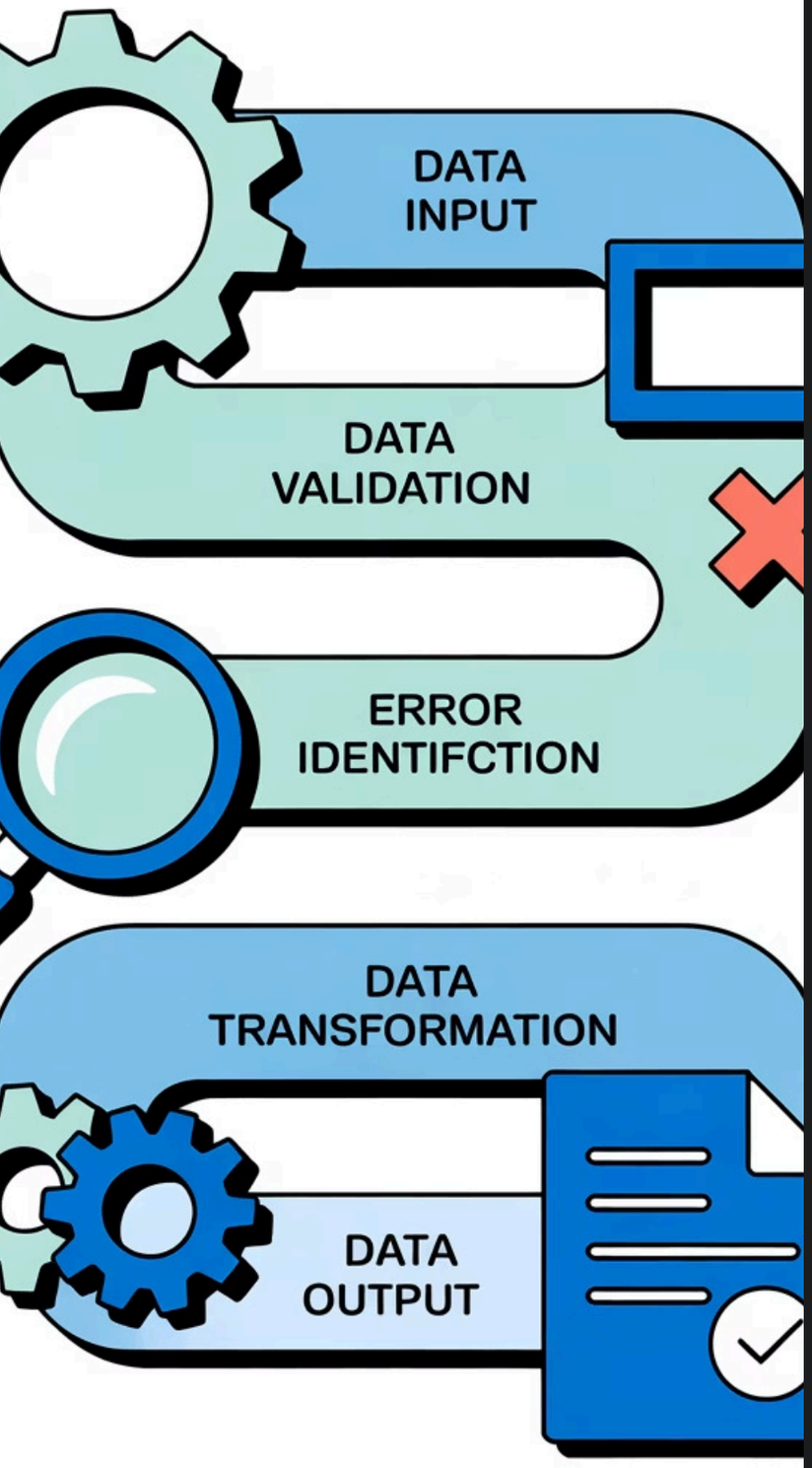
Vehicle Type

Mode of transportation used for delivery

Ratings

Historical performance metrics of delivery personnel

Target Variable: Derived from actual delivery time measurements, classified as binary outcome (Fast or Delayed)



Data Preprocessing

Robust data preprocessing is essential for building reliable machine learning models. Our preprocessing pipeline ensures data quality and prepares features for optimal model performance.

01

Data Cleaning

Removed non-predictive ID column that provides no informational value for delivery time prediction

02

Missing Value Treatment

Identified and handled missing data points using appropriate imputation strategies to maintain dataset integrity

03

Categorical Encoding

Transformed categorical variables (Weather, Traffic, Vehicle Type) into numerical representations suitable for model training

04

Feature Scaling

Applied standardization after train-test split to prevent data leakage and ensure fair model evaluation

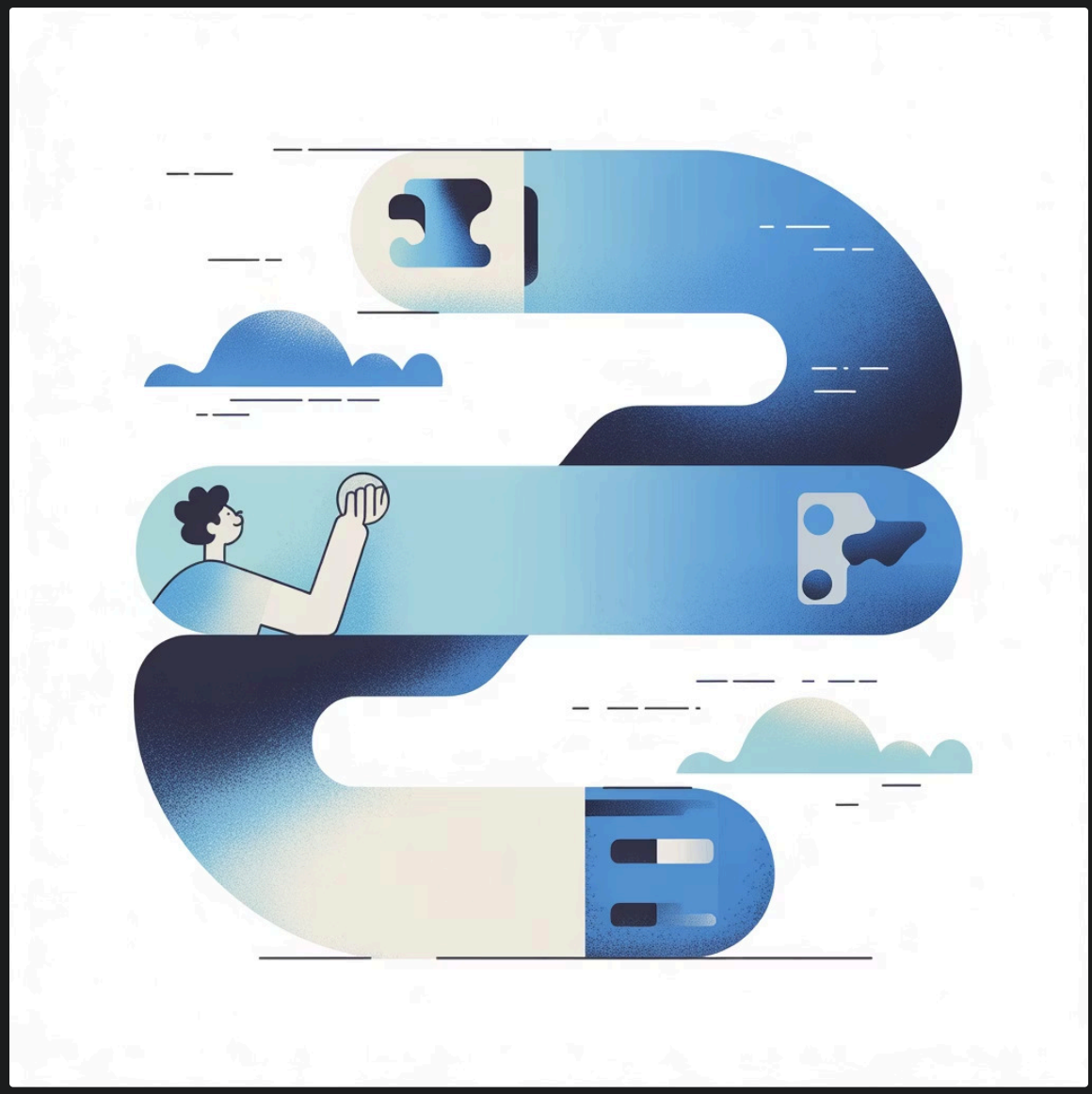
Critical Note: Feature scaling performed *after* splitting data to avoid information leakage from test set into training process

Feature Engineering

Target Variable Creation

We engineered a binary target variable from continuous delivery time measurements. Orders were classified based on whether they met expected delivery windows.

- **Fast (0):** Deliveries within standard time threshold
- **Delayed (1):** Deliveries exceeding expected duration



Distance Feature

Utilized pre-calculated distance metric from dataset rather than computing from raw coordinates

- More accurate than Euclidean distance
- Accounts for actual road networks
- Reduces computational complexity

Geospatial Considerations

Latitude and longitude coordinates excluded from direct modeling

- Distance already captures spatial relationship
- Avoids redundant information
- Simplifies model interpretation

Machine Learning Models Used

We implemented three distinct classification algorithms, each with unique strengths and approaches to the prediction problem. This multi-model strategy enables comprehensive evaluation and comparison of performance characteristics.

Naive Bayes

Probabilistic classifier based on Bayes' theorem with strong independence assumptions

- Fast training and prediction
- Works well with high-dimensional data
- Effective for baseline comparisons

K-Nearest Neighbors

Instance-based learning algorithm that classifies based on similarity to training examples

- Non-parametric approach
- Captures local patterns effectively
- No explicit training phase

Decision Tree

Tree-structured model that makes decisions through hierarchical feature splits

- Highly interpretable results
- Handles non-linear relationships
- No feature scaling required

Naive Bayes Model

Working Principle

Naive Bayes applies Bayes' theorem with the "naive" assumption that features are conditionally independent given the class label. Despite this simplification, it often performs surprisingly well in practice.

The algorithm calculates the probability of each class given the input features and selects the class with highest posterior probability.

Mathematical Foundation

P(Class|Features) = P(Features|Class) × P(Class) / P(Features)

Accuracy

Overall correctness of predictions across all classes

Precision

Proportion of positive predictions that were actually correct

Recall

Proportion of actual positives that were correctly identified

Fl-Score

Harmonic mean of precision and recall for balanced evaluation

Confusion Matrix: Visual representation of model performance showing true positives, false positives, true negatives, and false negatives will be inserted here

K-Nearest Neighbors Model

Core Concept

KNN predicts the class of a new data point by examining the classes of its K nearest neighbors in the feature space, using majority voting to make the final classification decision.

Algorithm Mechanics

For each prediction, KNN:

1. Calculates distance to all training points
2. Identifies K closest neighbors
3. Takes majority vote among neighbors
4. Assigns predicted class

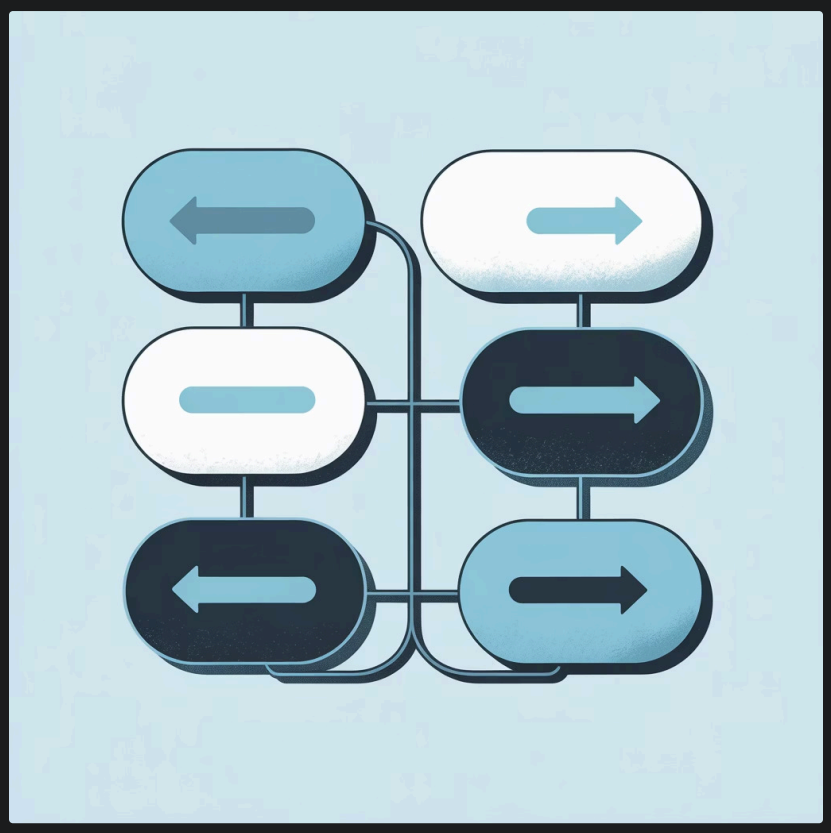
Hyperparameter Selection

K = 5 was chosen as our neighborhood size

- Balances bias-variance tradeoff
- Reduces noise sensitivity
- Commonly used baseline value
- Provides stable predictions

Confusion Matrix: Performance metrics visualizing classification accuracy for fast and delayed delivery predictions will be inserted here

Decision Tree Model



Tree-Based Classification

Decision Trees create a flowchart-like structure where internal nodes represent feature tests, branches represent outcomes, and leaf nodes represent class predictions.

The algorithm recursively partitions the feature space to create homogeneous regions for each class.

1 **Root Node**
Initial split on most informative feature

2 **Internal Nodes**
Successive splits refine classification boundaries

3 **Leaf Nodes**
Final class predictions at terminal points

4 **Pruning Strategy**
To prevent overfitting and improve generalization, we applied regularization through pruning parameters:

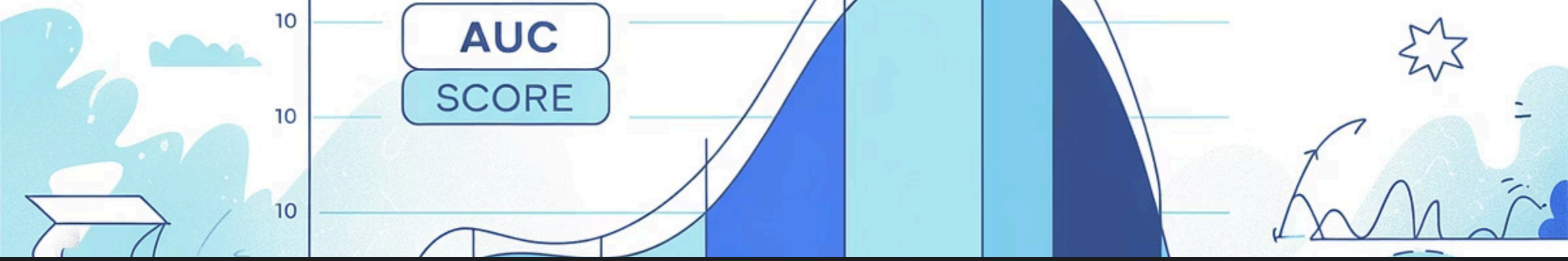
max_depth

Limits tree height to control model complexity and prevent excessive branching

min_samples_split

Requires minimum samples before allowing node splits, ensuring statistical significance

Confusion Matrix: Decision Tree classification results showing predicted versus actual delivery outcomes will be inserted here



ROC Curve Analysis

Understanding ROC and AUC

The Receiver Operating Characteristic (ROC) curve is a fundamental tool for evaluating binary classification models. It plots the True Positive Rate against the False Positive Rate across various classification thresholds.

Key Metrics

- **True Positive Rate (TPR):** Sensitivity or recall of the model
- **False Positive Rate (FPR):** Proportion of negatives incorrectly classified
- **AUC Score:** Area Under the Curve, ranging from 0 to 1

Interpretation Guide

- AUC = 0.5: Random guessing
- AUC > 0.7: Acceptable performance
- AUC > 0.8: Excellent performance
- AUC = 1.0: Perfect classification

Model Comparison: ROC curves for all three models (Naive Bayes, KNN, Decision Tree) will be displayed here for visual comparison of discriminative ability

Model Comparison

Model Performance Comparison

Model	Accuracy	Precision	Recall
Naive Bayes	0.85	0.85	1.00
K-Nearest Neighbors	0.825	0.85	0.97
Decision Tree	0.825	0.85	0.97

Class Imbalance Impact

Uneven distribution of Fast vs. Delayed deliveries affects model performance. Accuracy alone may be misleading when one class dominates the dataset.

Metric Selection

F1-score provides more balanced evaluation than accuracy for imbalanced datasets, considering both precision and recall equally.

Model Strengths

Each algorithm exhibits different performance patterns across metrics, revealing trade-offs between false positives and false negatives.

Results & Discussion

Key Findings

Our comparative analysis revealed important insights about model performance in the context of food delivery time prediction.

While all three models achieved reasonable accuracy, their behavior varied significantly when examining detailed performance metrics.

Model Performance

Decision Tree demonstrated superior overall performance with balanced precision-recall characteristics

Class Imbalance Effects

Unequal class distribution led to biased predictions favoring the majority class across all models

Evaluation Complexity

Accuracy alone insufficient—comprehensive metrics essential for true performance understanding

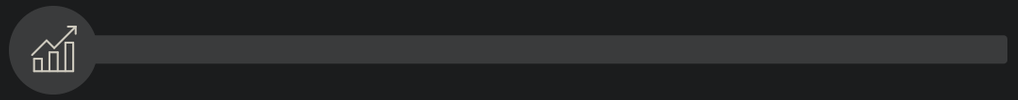
Critical Observations

The presence of class imbalance significantly influenced model behavior. Models tended to predict the majority class more frequently, achieving high accuracy while potentially missing important patterns in the minority class. This highlights the importance of using multiple evaluation metrics and considering the real-world costs of different types of misclassification in food delivery scenarios.

Conclusion

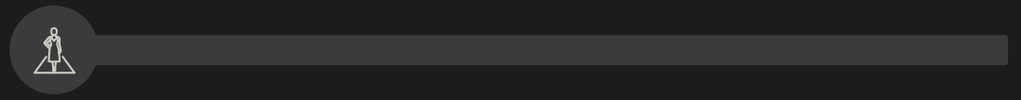
Project Outcomes

Successfully developed and evaluated three machine learning models for predicting food delivery time classifications, demonstrating practical application of classification algorithms to real-world logistics challenges.



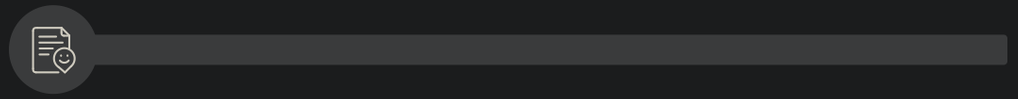
Data Pipeline

Implemented comprehensive preprocessing and feature engineering workflow



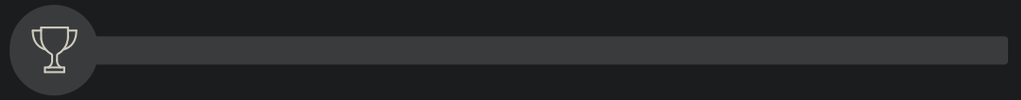
Model Development

Built and trained three distinct classification algorithms with different theoretical foundations



Performance Analysis

Conducted rigorous evaluation using multiple metrics and visualization techniques



Best Model Identified

Decision Tree emerged as top performer with optimal balance of accuracy and interpretability

Practical Significance

This project demonstrates that machine learning can effectively predict delivery delays, enabling food delivery platforms to proactively manage customer expectations, optimize delivery operations, and improve overall service quality. The Decision Tree model's interpretability makes it particularly valuable for understanding which factors most strongly influence delivery times.

Future Enhancements

While our current models provide valuable insights, several opportunities exist to enhance prediction accuracy and practical applicability. These improvements would address current limitations and extend the model's capabilities.

Address Class Imbalance

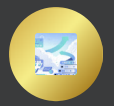
Implement techniques like SMOTE, class weights, or undersampling to balance the dataset and improve minority class prediction

Hyperparameter Optimization

Conduct systematic grid search or random search to identify optimal parameter configurations for each model

Ensemble Methods

Explore Random Forests, Gradient Boosting, or Stacking to combine multiple models for improved accuracy



Advanced Feature Engineering

Create interaction terms, temporal features, and domain-specific variables to capture complex relationships

Deep Learning Approaches

Investigate neural network architectures for potentially capturing more complex non-linear patterns

Production Deployment

Develop real-time prediction API and monitoring system for operational use in delivery platforms

Thank You!

We appreciate your time and interest in our Food Delivery Time Prediction project.

Contact Us

For any questions or further discussion, please reach out to:

abhishawhaval@gmail.com