Food Delivery Time Prediction

Objective
The goal is to predict whether a food delivery will be "Fast" or "Delayed" based on various features like customer location, restaurant location, weather, traffic conditions, etc. This dataset will be used to explore **CNN** and **evaluation/validation** techniques.

Phase 1 - Data Preprocessing and Feature Engineering

- **Data Import and Cleaning**
  - Load the dataset (`Food_Delivery_Time_Prediction.csv`).
  - **Handle Missing Data** Identify and handle missing values either by imputation (mean, median) or by removal, depending on the dataset's size and missingness.
  - **Encode Categorical Features** Apply One-Hot Encoding or Label Encoding to transform categorical variables such as `weather conditions` and `traffic`.
  - **Normalize Numerical Features** Normalize continuous features like `Distance` and `Delivery Time` to ensure they are on a similar scale for model processing.
- **Feature Engineering**
  - **Geographical Distance Calculation** If not provided, calculate the geographical distance between the customer and restaurant using the Haversine formula.
  - **Time-based Features:** Create new features related to the time of day, such as whether the delivery falls within "rush hour" or "non-rush hour."
  - **Weather Impact Analysis:** Incorporate weather-related features (e.g., temperature, humidity) to assess their impact on delivery time predictions.

Phase 2 - Convolutional Neural Network (CNN)

1. **Introduction to CNN**
   - **Objective** Use a CNN model for predicting if a food delivery will be "Fast" or "Delayed" based on features extracted from images (e.g., customer and restaurant location maps, delivery route images).
2. **Implementation**
   - **Dataset Preparation**
     - Prepare the data by converting location and delivery details into image-based representations (e.g., using maps or custom images).
   - **CNN Architecture**
     - Build a simple CNN with several convolutional layers, pooling layers, and dense layers to classify deliveries as "Fast" or "Delayed".
   - **Evaluation Metrics**
     - Accuracy, Precision, Recall, F1-score.
3. **Model Improvement**
   - Tune hyperparameters (e.g., number of filters, kernel size, learning rate) for improved model performance.
   - Evaluate CNN's performance against simpler machine learning models like Logistic Regression.

Phase 3 - Model Evaluation and Validation

1. **Cross-Validation**

    ○ **Objective** Perform K-fold cross-validation to ensure the model generalizes well to unseen data.
    ○ **Implementation** Split the dataset into training and validation sets (e.g., 5-fold cross-validation) and assess model performance on each fold.

2. **Evaluation Metrics**

    ○ Evaluate the CNN model using accuracy, confusion matrix, and ROC curve.
    ○ Compare these metrics to traditional models (Logistic Regression) to validate the CNN model's superior performance.

3. **Hyperparameter Tuning**

    ○ Use GridSearchCV or RandomizedSearchCV to fine-tune the CNN model.
    ○ Explore the best combinations of kernel sizes, activation functions, and learning rates.

Final Deliverables

1. **Jupyter Notebook (.ipynb)**
    ○ Full code for CNN implementation, evaluation/validation methods.
2. **Data Visualizations**
    ○ Visualizations such as confusion matrix for CNN performance, ROC curve for model evaluation.
3. **Final Report**
    ○ A comprehensive report that covers the methodology, model performance, and key findings from CNN, model validation techniques.
