Linear Regression – Model Evaluation

The Linear Regression model was trained to predict the delivery time in minutes. Its performance on the test set is:

- Mean Squared Error (MSE): 957.16
- Root Mean Squared Error (RMSE): 30.94
- Mean Absolute Error (MAE): 26.52
- R-squared (R^2): −0.035

These values show that the predictions differ from the actual delivery times by roughly 26–31 minutes on average. The negative R^2 value means that this simple linear model does not improve over a very basic baseline that always predicts the average delivery time. In other words, with the current features and setup, the linear relationship is not strong enough to capture how delivery time changes.

Logistic Regression – Model Evaluation

The Logistic Regression model classifies each order as Fast or Delayed. Its test performance is:

- Accuracy: 0.50
- Precision: 0.50
- Recall: 0.45
- F1-score: 0.47
- AUC (Area Under the ROC Curve): 0.48

An accuracy of 0.50 means the classifier is correct only about half of the time, which is close to random guessing for two classes. A precision of 0.50 indicates that only half of the orders predicted as delayed are actually delayed, while a recall of 0.45 means the model finds less than half of all real delayed orders. The F1-score of 0.47 reflects this weak balance between precision and recall. The AUC of 0.48, which is slightly below 0.5, confirms that the model has very limited ability to distinguish fast deliveries from delayed ones.

Overall Comment on the Models

Both the regression and classification models implement the full machine learning workflow required in the assignment: preprocessing, feature engineering, training, and evaluation. However, their current performance is poor, which suggests that, in a real application, additional feature engineering, better data, or more advanced models would be needed to achieve accurate and reliable predictions.

I've just finished the code. rewrite everything else that is needed

Here's a complete, clean write-up you can submit alongside your notebook. You only need to add your name and course details.

Food Delivery Time Prediction – Project Report

1. Introduction

The objective of this project is to analyze and predict food delivery times and to identify whether a particular order is likely to be delivered fast or delayed. The task is addressed in two ways:

- A regression model (Linear Regression) to predict the exact delivery time.
- A classification model (Logistic Regression) to label each order as Fast or Delayed.

The models use information about customer and restaurant locations, distance, weather and traffic conditions, delivery person experience, order priority, order time, vehicle type, ratings, and cost-related attributes.

---

2. Dataset and Preprocessing

The dataset contains multiple food delivery orders with the following important fields:

- Order_ID – unique identifier for each order.
- Customer_Location, Restaurant_Location – latitude/longitude pairs stored as strings.
- Distance – given distance between restaurant and customer.
- Weather_Conditions – e.g. Sunny, Rainy, Cloudy, Snowy.
- Traffic_Conditions – Low, Medium, High.
- Delivery_Person_Experience – numerical measure of experience.
- Order_Priority – Low, Medium, High.
- Order_Time – Morning, Afternoon, Evening, Night.
- Vehicle_Type – Bike, Bicycle, Car, etc.
- Restaurant_Rating, Customer_Rating – ratings out of 5.
- Delivery_Time – time taken for delivery (target for regression).
- Order_Cost, Tip_Amount – cost and tip.

2.1 Handling Missing Values

The dataset was first inspected for missing values. Rows containing missing entries were removed. This keeps the preprocessing simple and still leaves enough data for training the models.

2.2 Feature Engineering

Two main forms of feature engineering were applied:

1. Location features
    - The location strings in `Customer_Location` and `Restaurant_Location` were split into four numeric columns: `Cust_Lat`, `Cust_Lon`, `Rest_Lat`, and `Rest_Lon`.
    - Using these coordinates, a new feature Haversine_Distance was computed to represent the great-circle distance between the customer and the restaurant. This complements the given `Distance` feature.
2. Time-based feature
    - A new binary feature Rush_Hour was created from `Order_Time`.
    - Orders placed in the Afternoon or Evening were marked as rush hour (`Rush_Hour = 1`), while those in Morning or Night were marked as non-rush (`Rush_Hour = 0`).

2.3 Encoding Categorical Variables

The following categorical variables were converted into numerical form using one-hot encoding:

- `Weather_Conditions`
- `Traffic_Conditions`
- `Order_Priority`
- `Order_Time`
- `Vehicle_Type`

One category per variable was dropped to avoid redundancy (dummy variable trap).

2.4 Feature Scaling

Before training the models, the numeric input features were standardized using StandardScaler. Scaling was applied separately for:

- The regression feature set (for Linear Regression).
- The classification feature set (for Logistic Regression).

This ensures that all features contribute on a similar scale.

---

3. Exploratory Data Analysis (EDA)

Several exploratory steps and visualizations were used to understand the data.

3.1 Descriptive Statistics

Descriptive statistics (count, mean, standard deviation, minimum, maximum, and quartiles) were calculated for numeric columns such as `Distance`, `Haversine_Distance`, `Delivery_Person_Experience`, `Restaurant_Rating`, `Customer_Rating`, `Delivery_Time`, `Order_Cost`, and `Tip_Amount`. This helped to understand the typical ranges and variability of key features.

3.2 Distribution and Outliers

- Histograms and boxplots were used to visualize the distributions of `Delivery_Time`, `Distance`, and `Order_Cost`.
- These plots revealed the spread of values and highlighted the presence of a few extreme observations that can be considered outliers.
- An optional IQR-based rule was applied to trim extreme outliers for selected columns (`Delivery_Time`, `Distance`, `Haversine_Distance`) to reduce their influence on the models.

3.3 Correlation Analysis

A correlation matrix was computed and visualized as a heatmap using matplotlib. Important observations included:

- Both Distance and Haversine_Distance showed positive correlation with Delivery_Time, confirming that longer distances generally lead to longer delivery times.
- Variables related to traffic, weather, and rush hours (through the one-hot encoded columns) were also associated with delivery time.

This analysis helped identify features that are likely to have predictive power.

---

4. Linear Regression Model (Predicting Delivery Time)

4.1 Problem Setup

The regression task aims to predict the continuous variable `Delivery_Time` based on the available features.

- Features (X): all columns except `Order_ID`, the raw location strings, and `Delivery_Time`.
- Target (y): `Delivery_Time`.
- The data was split into training and test sets with an 80/20 ratio.

4.2 Model Training

A Linear Regression model from scikit-learn was trained on the standardized training features and then used to predict delivery times on the test set.

4.3 Evaluation

The performance of the Linear Regression model on the test set was evaluated using:

- Mean Squared Error (MSE): 957.16
- Root Mean Squared Error (RMSE): 30.94
- Mean Absolute Error (MAE): 26.52
- R-squared (R^2): −0.035

These values show that the predicted delivery times differ from the actual times by roughly 26–31 minutes on average. The negative R^2 value means that this linear model does not improve over a very simple baseline that always predicts the mean delivery time. This indicates that, with the current features and linear relationship, the model is not able to capture the complexity of the delivery time behavior.

4.4 Visualization

An Actual vs Predicted scatter plot for `Delivery_Time` was drawn:

- Each point represents one order, with the x-axis as the actual delivery time and the y-axis as the predicted time.
- A 45-degree reference line was added; points close to this line indicate good predictions.

The scatter plot visually confirms that the regression model has large errors and does not track the true values closely.

---

5. Logistic Regression Model (Fast vs Delayed Classification)

5.1 Problem Setup

For classification, a new binary target variable `Delivery_Status` was defined:

- The median of `Delivery_Time` was computed.
- Orders with delivery time less than or equal to the median were labeled Fast (0).
- Orders with delivery time greater than the median were labeled Delayed (1).
- Features (X): all encoded numeric features except `Order_ID`, raw locations, `Delivery_Time`, and `Delivery_Status`.
- Target (y): `Delivery_Status`.
- The data was split into training and test sets with an 80/20 ratio, using stratification to preserve the class balance.

5.2 Model Training

A Logistic Regression model was trained on the standardized training features. The trained model outputs both class predictions and predicted probabilities of an order being delayed.

5.3 Evaluation

The Logistic Regression model was evaluated on the test data using standard classification metrics:

- Accuracy: 0.50
- Precision: 0.50
- Recall: 0.45
- F1-score: 0.47
- AUC (Area Under the ROC Curve): 0.48

An accuracy of 0.50 means the model correctly classifies about half of the orders, which is close to random guessing for a two-class problem. A precision of 0.50 shows that only half of the orders predicted as delayed are truly delayed, while a recall of 0.45 indicates that the model identifies less than half of the actual delayed orders. The F1-score of 0.47 summarizes this weak balance between precision and recall. The AUC value of 0.48, being close to 0.5, further suggests that the model has very limited ability to distinguish between fast and delayed deliveries.

5.4 Visualizations

Two key plots were created for the classification model:

- Confusion Matrix:
  Displayed as a 2×2 matrix showing the counts of:
  - True Fast, True Delayed
  - False Fast (missed delays), False Delayed (false alarms).
- ROC Curve:
  The ROC curve was plotted using the predicted probabilities for the delayed class. The curve, together with the AUC value, illustrates that the classifier is only slightly worse than random at separating fast from delayed orders.

---

6. Model Comparison

The two models serve different purposes:

Model	**Task**	**Output Type**	**Main Metrics**
Linear Regression	Predict exact delivery time	Continuous (minutes)	MSE, RMSE, MAE, R^2

Logistic Regression	Classify fast vs delayed	Binary (0 = Fast, 1 = Delayed)	Accuracy, Precision, Recall, F1, AUC

-
  The Linear Regression model is designed for giving a numerical estimate of delivery time (ETA).
- The Logistic Regression model is designed for flagging orders as fast or delayed, which can be useful for operational monitoring.

In this project, both models show weak performance with the current simple feature engineering and modeling choices. However, they correctly illustrate the full machine learning workflow required: data preprocessing, feature engineering, model training, evaluation, and interpretation.

---

7. Actionable Insights and Recommendations

Even though the models are not highly accurate, some useful qualitative insights can still be drawn:

1. Impact of Distance
   - Longer distances (both `Distance` and `Haversine_Distance`) are associated with higher delivery times.
   - Recommendation: Use smarter routing and planning for long-distance orders, possibly assigning more experienced delivery personnel.
2. Traffic and Weather Conditions
   - Adverse weather (Rainy, Snowy) and high traffic conditions tend to increase the chance of delays.
   - Recommendation: Increase rider availability and add buffer time during peak traffic periods and bad weather.
3. Rush Hour Effect

    - Orders during Afternoon and Evening (Rush_Hour = 1) generally show longer delivery times.
    - Recommendation: Schedule more delivery staff during these peak hours to handle higher demand.
4. Delivery Person Experience
    - More experienced delivery persons may handle deliveries faster and more reliably.
    - Recommendation: Allocate experienced staff to high-priority or long-distance orders and provide training to less experienced riders.
5. Use of Priority Information
    - Combining `Order_Priority` with model outputs can help decide which orders to dispatch first when resources are limited.

---

8. Conclusion

This project demonstrates an end-to-end machine learning pipeline for a food delivery time prediction problem:

- Data import and cleaning.
- Handling missing values.
- Feature engineering (coordinates, Haversine distance, rush hour).
- Encoding categorical variables and scaling features.
- Exploratory data analysis with statistics, distributions, and correlations.
- Training and evaluating a Linear Regression model for continuous time prediction.
- Training and evaluating a Logistic Regression model for fast vs delayed classification.
- Visualizing results with scatter plots, confusion matrices, and ROC curves.
- Extracting high-level operational insights from the data and model behavior.