Exploratory data analysis (EDA) report:

1. Basic Information

Purpose:

`df.info()`: Displays the basic structure of the dataset including column names, data types, and non-null counts.

`df.isnull().sum()`: Checks for missing values in each column, crucial for data quality and preparation.

`df.describe()`: Provides summary statistics (mean, standard deviation, percentiles) for numeric columns, offering insights into the data distribution.

2. Distribution of Numeric Features

Purpose: Histogram plots visualize the distribution of numeric variables (`Time`, `Income`) and use KDE (Kernel Density Estimation) to smooth the distribution curves. This helps understand the overall patterns (e.g., skewness, spread) of the data.

3. Outlier Detection with Box Plots

Purpose: Box plots are used to detect outliers by visualizing the spread of the data, the interquartile range (IQR), and identifying extreme values (outliers). This is important for spotting any anomalies in `Time` and `Income`.

4. Label Distribution (`IsFinished`)

Purpose: Visualize the distribution of the `Label` column, which indicates whether a ride was completed or not. Understanding the balance of labels is important for classification problems and general data analysis.

5.Origin and Destination Distribution:

Purpose: This step visualizes the most frequent `Origin` and `Destination` locations in the dataset. These plots help identify the most common routes, which can be important for optimizing services or identifying trends.

6. Sentiment Analysis

Purpose: This basic sentiment analysis classifies the text into positive, negative, or neutral sentiment by checking for predefined words. Sentiment analysis helps in understanding customer satisfaction and identifying areas for improvement.

Feature Engineering & Feature Selection:

1.Ride Duration
Create duration buckets (e.g., short, medium, long rides)

2.Income per Minute
Create a new feature that shows how much income is generated per minute of the ride

Training models:

First of all, the data is highly imbalanced. In the training dataset, we have about 8,311 samples for class 0 and only 189 for class 1. The most important approach for handling imbalanced datasets is the SMOTE method. However, the final results are poor, and the reason is clear: SMOTE oversamples the minority class, and when the difference between the classes is significant, it tends to overfit to noise. This is because SMOTE relies on the features of the minority class, and the lack of representation for this class leads to poor results.

I used multiple algorithms, such as Random Forest and AdaBoost, but the results were not satisfactory. For example, in the AdaBoost algorithm, the precision, recall, and F1-score for class 0 are 0.98, 0.78, and 0.87, respectively, while for class 1, they are 0.03, 0.28, and 0.05. This indicates that the model has learned nothing useful and may have overfitted the data by memorizing the features of class 0. For the Random Forest algorithm, I also implemented weighted classes to help address the imbalance, but the improvement was minimal.

As you mentioned in your documents, I also used a neural network with binary cross-entropy loss and a sigmoid activation function, both of which are suitable for binary classification. Unfortunately, the results were poor, as expected.

I also performed some preprocessing techniques, such as dropping missing values and selecting informative features like 'Income', 'Time', 'Origin', and 'Destination'. I found these features to be the most informative, as other attributes like ID and timestamps are merely indicators and not meaningful. Additionally, I added features such as ride duration and income per minute, as you suggested in feature engineering. I also employed cross-fold validation.

I believe that the only way to achieve better results is to obtain more data for class 1. I also used grid search for hyperparameter tuning, but it takes a significant amount of time.

class	precision	recall	f1-score
0	0.98	0.78	0.87
1	0.03	0.28	0.05

AdaBoost Classification Report