## **Low Level Design (LLD) for Order Review Classification Project**

This LLD details the implementation of the Python code for classifying customer reviews as positive (score >= 4) or negative (score < 4) using a Random Forest Classifier.

### **Modules**

The code utilizes the following modules:

* pandas (pd): for data manipulation and analysis
* sklearn.model\_selection: for splitting data into training and testing sets
* sklearn.preprocessing: for data scaling
* sklearn.ensemble: for building the Random Forest Classifier
* sklearn.metrics: for evaluating model performance

### **Functions**

* **Data Loading Functions**
  + These functions (not explicitly shown) handle loading each dataset (orders, products, etc.) using pd.read\_csv.
* **Data Merging Function**
  + This function (not explicitly shown) iteratively merges the loaded dataframes using data.merge based on specified columns.
* **Data Preprocessing Functions**
  + data.dropna(): Removes rows with missing values.
  + data['review\_score'] = data['review\_score'].apply(lambda x: 1 if x >= 4 else 0): Converts review score into binary classification (positive: 1, negative: 0).
* **Feature Engineering Functions**
  + These functions (not explicitly shown) might involve creating new features from existing data.
* **Feature Selection Function**
  + features = data[['product\_id', 'seller\_id', 'price', 'freight\_value', 'product\_category\_name']]: Selects specific features for model training.
* **Target Selection Function**
  + target = data['review\_score']: Defines the target variable for classification.
* **Data Encoding Function**
  + features = pd.get\_dummies(features, columns=['product\_category\_name']): One-hot encodes the categorical feature "product\_category\_name".
* **Data Splitting Function**
  + train\_test\_split(features, target, test\_size=0.2, random\_state=42): Splits data into training and testing sets with a 20% test size and sets a random state for reproducibility.
* **Data Normalization Function**
  + scaler = StandardScaler(): Initializes a StandardScaler object for normalization.
  + X\_train = scaler.fit\_transform(X\_train): Fits the scaler on the training data and transforms it.
  + X\_test = scaler.transform(X\_test): Transforms the testing data using the fitted scaler.

### **Algorithms**

The code utilizes the following algorithms:

* **Random Forest Classifier**: A machine learning algorithm for classification tasks that combines multiple decision trees for improved performance.