**Assignment: Predicting Customer's Favorite Dish using XGBoost**

**Objective:**

To build and evaluate a machine learning model that predicts the favorite dish of a customer based on their dining preferences and behavior. The goal is to apply proper data preprocessing, feature engineering, one-hot encoding, and model training while ensuring the correct handling of time-based data.

**Dataset**

You are provided with a dataset named dining\_info.xlsx, which contains the following columns:

| **Column Name** | **Description** |
| --- | --- |
| customer\_id | Unique identifier for each customer. |
| transaction\_id | Unique identifier for each transaction. |
| Preferred Cusine | The cuisine preferred by the customer in the transaction. |
| dish | The dish ordered in the transaction (Target Variable). |
| price\_for\_1 | Price for one unit of the dish. |
| order\_time | Timestamp of when the order was placed. |
| Qty | Quantity of the dish ordered. |
| stay\_duration | Duration of the customer’s stay. |
| check\_in\_date | Date of check-in. |
| check\_out\_date | Date of check-out. |

**Assignment Tasks**

**1. Data Preparation**

* Load and explore the dataset.
* Split the dataset into three parts based on order\_time:

**(a) Feature Extraction Dataset (features\_df)**

* + Rows where order\_time is before **January 1, 2024**.
  + This dataset is used for calculating customer preferences (e.g., total spend, average stay).

**(b) Training Dataset (train\_df)**

* + Rows where order\_time is between **January 1, 2024, and October 1, 2024**.
  + This dataset is used to train the model.

**(c) Testing Dataset (test\_df)**

* + Rows where order\_time is after **October 1, 2024**.
  + This dataset is used for evaluating the model's performance.

**Why Split the Data This Way?**

* **Feature Extraction (Before Jan 1, 2024):**  
  Historical data helps compute meaningful customer preferences, such as their most frequently ordered dish or average spend.
* **Training (Jan–Oct 2024):**  
  The model learns from the most recent patterns.
* **Testing (Post-Oct 2024):**  
  Ensures we evaluate performance on unseen future data, simulating real-world predictions.

**Example:**

Imagine a restaurant wants to predict what a customer will order next based on their previous dining habits.

* Data before 2024 helps determine things their favorite cuisine, average spends etc.
* Data from Jan–Oct 2024 trains the model on recent trends.
* Data after Oct 2024 tests how well the model generalizes.

**2. Feature Engineering**

Using features\_df, compute features such as

**Customer-Level Features:**

* total\_orders\_per\_customer: Total number of transactions per customer.
* avg\_spend\_per\_customer: Average amount spent per transaction.
* total\_qty\_per\_customer: Total number of dishes ordered by the customer.
* Some other similar features

**Cuisine-Level Features:**

* avg\_price\_per\_cuisine: Average price of dishes within each preferred customer cuisine. For example, this tells how much a customer who prefers North Indian spends on average.
* total\_orders\_per\_cuisine: Which is the most preferred cuisine?
* Other features that can relate cuisine with customer preferences.

**Features directly from training data:**

* Remember, you are allowed to use any data that you will know **before the customer makes the booking or when the customer makes the booking. You CANNOT use data that you won’t know beforehand.**
* You are allowed to take some features such as age, duration of stay , check\_in day, check\_in month etc directly from the training data (since this information is not available in the features data – you wont know how long a person is going to stay even before the person makes the booking. Also, age may have changed, so you cant use past age. Days, months differ. So, you cant use days and months present in feature dataset. You have to use them from training dataset. Also, remember that these are features that will be available to you even in the future. If a person makes a booking for March 2025, they will give their age, check\_in and check\_out etc. So, you can directly use these features)

**3. Data Integration**

* Merge train\_df and test\_df with the customer and cuisine features created from features set.
* Drop unnecessary columns:

**Columns from training data you are not supposed to use and hence, drop:**

* + transaction\_id: Not relevant for prediction.
  + customer\_id: Using aggregated features instead.
  + price\_for\_1: Using this feature directly from the training set will be cheating because it will directly tell you which dish as well, since each dish is associated to a price. **Moreover, remember, you wont know how much the customer will purchase for beforehand or when he/she makes the booking.**
  + order\_time: Ensures the model doesn’t “cheat” by using time-based trends directly. **You will not know beforehand at what time the customer will make the purchase.**
  + Qty: **You will not know the quantity the customer will order beforehand!**

**4. Encoding Categorical Data**

* Use **One-Hot Encoding** for categorical variables such as:
  + Preferred Cusine
  + fav\_dish\_per\_customer (if you have created the feature)
  + Any other categorical variable you have created
* Convert the target variable ‘dish’ using **Label Encoding**.

**5. Model Training**

Train an **XGBoost Classifier**. The reason we are using XGBoost is because we will be encountering many missing values. We wont have a lot of information for new customers, since they have never visited before. So the features such as favourite dish, average purchase price etc, will be NaN (missing)

* **Features (X\_train)**: All engineered features.
* **Target (y\_train)**: Encoded dish column.

Hyperparameters:

* **Objective:** multi:softmax (Multi-class classification).
* **Evaluation Metric:** mlogloss.
* **Learning Rate:** experiment with different values between 0.01 to 1.
* **Max Depth:** experiment with different values between 1 and 5
* **Number of Estimators:** experiment with different values between 50 and 500.

**6. Model Evaluation**

* **Predict:** Make the prediction on the test set. Now you will have y\_test and y\_test\_pred.
* **Calculate error Metrics:**
  + **Accuracy**: Measures correct predictions.
  + **Log Loss**: Penalizes incorrect predictions with high confidence.
* **Feature Importance Analysis:**
  + Identify the most influential features for predicting dish.

**🚫 Features NOT Allowed in Training & Why**

| **Feature** | **Reason for Exclusion** |
| --- | --- |
| transaction\_id | Unique identifier with no predictive value. |
| customer\_id | Using aggregated customer behavior instead. |
| order\_time | We wont know this beforehand |
| Qty | We wont know this beforehand |

**📌 Deliverables**

1. **Python Script**:
   * Well-documented code implementing all steps.
   * Achieve an accuracy of atleast 13% (0.13).

**Bonus Challenge 🚀**

* Try different hyperparameters for **XGBoost** to improve accuracy.
* Try to achieve accuracy of 20%

**🔍 Summary**

This assignment teaches how to:  
✅ Preprocess time-based data correctly.  
✅ Engineer useful customer and cuisine features.  
✅ Handle categorical variables properly.  
✅ Train an XGBoost model for multi-class classification.  
✅ Avoid data leakage by excluding inappropriate features.

**By completing this assignment, you’ll gain practical experience in feature engineering, ML model training, and evaluation!** 🚀