TEAM : doreamon sinchan

Indian Institute of Technology Guwahati

Subhashree Sahoo

Arjun Verma (L)

Neha Sellapan Devadas

Objective: The objective of this problem is to enhance the personalization of marketing communications for a bank by predicting the optimal time slots for engaging with customers via email. Due to regulatory and cost constraints, the bank can only send a limited number of communications within a specified timeframe. The challenge is to predict the time slot (a 3-hour window across a week) during which each customer is most likely to open and engage with an email, based on historical interaction data. The solution requires leveraging two datasets—Communication History, which tracks the timing of emails sent and opened, and Customer CDNA, which contains demographic and behavioral attributes of the customers. The final output should rank the 28 predefined time slots in order of engagement likelihood for each customer, and success is measured using the **Mean Average Precision (MAP)** metric. This ensures that the predicted time slots are not only accurate but also ranked correctly to maximize engagement. **NOTE : In the final submission.csv file, we have not been able to predict the list of time slots for a few customer codes as they were not present in any of the training or testing csv files.**

DATA PRE-PROCESSING:

1. Handling Missing Values

Missing data is a common issue in datasets, and addressing it effectively is critical for model reliability and performance. For the Customer CDNA dataset, columns with more than 60% missing values were removed to minimize noise. This decision ensures that features with excessive null values, which provide little to no predictive power, do not skew the model's learning. Columns with fewer missing values that were of numeric data type were **interpolated** using a linear method to preserve patterns in the data, while categorical columns were imputed using the **mode**. These choices ensure that missing values are filled in a meaningful way, preserving dataset integrity while reducing biases caused by null values. For the Communication History dataset, missing timestamps (such as **send_timestamp** or **open_timestamp**) were filled with the mode. This approach ensures that no records are discarded due to null values while maintaining logical consistency in the data.

2. Standardization and Encoding

To prepare the datasets for machine learning models, numerical columns were standardized and categorical columns were encoded. For numeric data, scaling methods were applied to ensure features contributed equally to the model. Integer columns were scaled using **Min-Max Scaling**, which maps values to a range of 0 to 1. This preserves the distribution but at the same time makes the data range uniform. Floating-point columns, on the other hand, were scaled using **Standard Scaling** to normalize data around a **mean of zero** and a **standard deviation of one**. This prevents features with large ranges from dominating model learning. Categorical features,

such as demographic data or engagement preferences, were **label-encoded**. This process assigns numerical values to categorical variables, making them interpretable by machine learning algorithms. For example, inconsistent categories like "UNMARRIED," "Single," and "SINGLE" were standardized to "Unmarried,". Additionally, rare categories were replaced with "Others" to prevent the model from overfitting to sparsely represented data. By simplifying the categorical space, this step improves the interpretability of features and enhances model robustness.

3. Feature Selection and Reduction

Efficient feature selection is essential for reducing computational overhead and improving the interpretability of models. In the preprocessing steps, irrelevant columns were systematically removed based on domain understanding and exploratory data analysis. For instance, in the Customer CDNA dataset, some columns were eliminated after consolidating their information into a single column. Also there were some columns which were essentially dealing with almost the same information such as the ‘location’ of the customer, all of them were merged into a single column(by replacing the null values in that column with the corresponding value from a similar feature column of the same row) to ease out the task of handling such a large dataset. Similarly, in the Communication History dataset, **batch_id** was removed since it did not add any predictive value for the task. Some feature vectors were dropped either because there wasn’t much variation in their values after merging the rows in the CDNA dataset that corresponded to multiple entries of a single ‘CUSTOMER_CODE’ to capture the most recent information or simply because they were not adding much value to the overall predictive power. By reducing the dimensionality of the datasets, this step ensures that models focus only on the most relevant features, improving both computational efficiency and predictive performance. At the same time, this prevents overfitting by removing redundant or noisy information.

4. Time Slot Mapping

A key aspect of solving the problem was transforming the timestamps into predefined 3-hour time slots. This step involved dividing the day into intervals (e.g., 9:00 AM–12:00 PM as slot_1) and mapping **send_timestamp** and **open_timestamp** accordingly. A custom function was created to parse the timestamps, extract the day of the week, and assign each communication event to one of the 28 time slots (spanning the entire week). This transformation is essential for aligning the dataset with the prediction task as the goal is to predict the best time slot for engagement. Filling missing time slots with the mode ensures consistency and avoids introducing errors into the data. This mapping bridges the raw data and the final task, providing a structured way to evaluate engagement.

6. Data Standardization Across Time

To ensure temporal consistency between the datasets, the **batch_date** field in Customer CDNA was used to align customer records with the communication history. Since the CDNA dataset captures customer behavior over time, it was crucial to match records to the most recent date preceding the communication event. This ensures that features like demographics or engagement history reflect the customer's state at the time of the communication, improving the relevance of the data. The records were sorted by **CUSTOMER_CODE** and **batch_date**, and **forward and backward filling techniques** were applied to impute missing data while preserving temporal

dependencies. By doing so, this step guarantees that features are accurate and contextually relevant to each communication.

7. Feature Engineering

New features were engineered to capture engagement behavior. For instance, **open_timestamp** was linked with **send_timestamp** to compute engagement times, enabling the model to identify patterns in customer preferences. By incorporating derived features, this step enhances the predictive power of the dataset, providing the model with additional context to learn from.

MODEL DESCRIPTION:

Model Architecture:

The model deployed for achieving this task is a **Random Forest-based MultiOutputClassifier**, designed to handle the multi-label classification task of predicting customer engagement across 28 predefined time slots. By using Random Forest Classifiers as the base learners, the model independently predicts engagement for each time slot. The hyperparameters include:

- **300 estimators (trees)**: Ensures robustness and generalization by leveraging multiple decision trees.
- **Maximum depth of 20**: Prevents overfitting by limiting the depth of trees while maintaining flexibility for learning complex patterns.
- **Minimum sample split size of 5**: Ensures meaningful splits in the data, avoiding overly narrow or shallow trees.

These hyperparameters strike a balance between model complexity and generalization, ensuring robust performance across varying customer behavior patterns.

The key idea behind designing a model that can predict the best time slots for customer engagement is to create a multi-label classification setup wherein each of the 28 time slots is treated as a binary label indicating whether the customer is likely to engage during that slot.

The **prepare_training_data** function consolidates and processes the input data to form:

1. **Features (X)**: Customer-level attributes that provide insights into their preferences and behavior.
2. **Target Labels (y)**: A binary vector of size 28 for each customer, where each element indicates engagement (1) or non-engagement (0) in the corresponding time slot.

This enables the model to learn patterns from the customer data and historical engagement behavior, making it capable of predicting the likelihood of engagement across all 28 slots for new customers. The ultimate goal is to rank the time slots in descending order of engagement probability, optimizing email communication for maximum impact.

Data Splitting and Training:

To evaluate the model's effectiveness, the dataset is split into an **80-20 ratio** for training and testing respectively. This ensures the model is validated on unseen data, mimicking real-world scenarios. The training process involves learning from customer features to predict engagement probabilities for each of the 28 slots. By using a MultiOutputClassifier, the model predicts all slots simultaneously, enabling efficient and comprehensive learning.

Prediction and Slot Ranking:

During inference, the model produces a **probability distribution** for each time slot, representing the likelihood of customer engagement. These probabilities are used to rank the slots in descending order, prioritizing the most likely slots for engagement. This ranking process ensures that the model meets the requirements of providing a sorted list of time-slots for each customer.The ranked slot predictions are formatted into a submission file with each row containing the customer code and the predicted order of time slots. This structure aligns directly with the evaluation metric, **Mean Average Precision (MAP)**, which emphasizes the correctness and ranking quality of the predicted slots.

Performance Evaluation:

The model's performance is evaluated using **precision scores** for each time slot, which measure the proportion of correctly identified engaging slots among the predicted positives. These scores indicate the model's ability to identify true positives effectively while minimizing the false positives. While precision scores vary slightly across slots, the model demonstrates consistent and reliable performance overall.

Model Deployment and Fine-Tuning:

The model is designed for flexibility and adaptability. After training, it is saved which preserves the trained model, scaler, and customer codes for future use. This allows for easy reloading without retraining, saving computational resources and time.

Additionally, the model supports **fine-tuning** with new test data. This feature enables the model to adapt to evolving customer behaviors or changes in engagement patterns, ensuring continued relevance and effectiveness. Fine-tuning involves updating the estimators of the Random Forest to incorporate new information while retaining previously learned patterns.