Non-Technical Report on Event Recommendation Engine Project

Project Overview

This project focused on building an **Event Recommendation Engine** designed to predict the likelihood of users' interest in various events based on their past behavior, demographic information, and event interactions. The engine is intended to help users, particularly medical professionals, discover relevant events such as webinars, conferences, and clinical gatherings. The goal was to develop a system that ranks events by their predicted relevance for each user, ensuring that users receive highly personalized event suggestions.

Objectives

The key objectives of this project were:

1. **To build a recommendation model** capable of predicting which events users would be most interested in.
2. **To rank events** based on the likelihood of user interest, prioritizing those with the highest chances of engagement.
3. **To optimize the model** to achieve a high score using the **Mean Average Precision at 200 (MAP@200)** metric, which measures how well the model ranks the top 200 event recommendations.

Approach

I used machine learning techniques to accomplish these objectives, specifically focusing on the **XGBoost** algorithm for classification tasks. The project relied on six datasets: train.csv, test.csv, users.csv, user_friends.csv, events.csv, and event_attendees.csv. These datasets contained valuable information on users, their friends, event details, and attendance patterns. After preprocessing the data, I built a model using **XGBoost** and fine-tuned it with **Optuna** to optimize the hyperparameters.

Steps Taken:

1. **Data Preprocessing**: I merged the datasets to create a unified view of user interactions, addressing missing data by filling in default values where necessary. Categorical variables like user location and event details were converted into numerical features that the machine learning model could interpret.
2. **Model Building**: I used the XGBoost algorithm for its ability to handle large datasets efficiently and its robust handling of missing data and categorical variables. I then optimized the model’s hyperparameters (such as learning rate,

tree depth, and number of estimators) using **Optuna**, which helped fine-tune the model for better performance.

3. **Evaluation**: I evaluated the model using the **MAP@200** metric, which assesses how well the system ranks the top 200 events for each user. The model achieved a MAP@200 score of **0.6014549699206235**, indicating a high level of accuracy in recommending relevant events to users.

Challenges and Solutions

During the project, I faced several challenges, which required modifications to the initial approach:

1. **Data Type Inconsistencies**: Some columns, such as those containing dates and categorical information (e.g., birthyear, location), had data type issues. These columns needed proper formatting and encoding before the model could process them.

Solution: I applied transformations to convert these columns into numeric values and used **one-hot encoding** for categorical data. For date columns, I split them into separate components for year, month, and day.

2. **Missing Columns**: Initially, some essential columns, like event_id, were missing during the data merging process, which caused issues during model training.

Solution: I revised the merging logic to ensure all necessary columns were included and handled missing values where appropriate to integrate the data smoothly.

3. **Feature Mismatches in Prediction**: When making predictions on the test data, I encountered issues with missing or extra features in the test set compared to the training set, leading to prediction errors.

Solution: I aligned the columns in the test set with those in the training set by reindexing the test data, ensuring that both had the same feature set before making predictions.

Suggestions for Improvement

While the model performed well, there are several areas where I could improve the recommendation engine further:

1. **Further Hyperparameter Tuning**: Although I used **Optuna** to tune a few key hyperparameters, expanding the search space to include additional parameters like min_child_weight and alpha could enhance model performance. Additionally, using **Bayesian optimization** could lead to more efficient tuning.

2. **Feature Engineering**: I focused mainly on basic demographic and event history features, but more advanced feature engineering could improve the model.

Creating **interaction-based features** such as engagement frequency or clustering users with similar interests could lead to better recommendations.

3. **Ensemble Models**: Instead of relying solely on XGBoost, I could explore using an **ensemble of models**, such as combining XGBoost with random forests or neural networks. This might capture different aspects of the data and improve prediction accuracy.
4. **Cross-Validation**: Implementing **k-fold cross-validation** would help ensure that the model generalizes well. By splitting the data into multiple subsets for training and validation, I could gain a better understanding of how well the model performs across different portions of the dataset.

Conclusion

The Event Recommendation Engine project successfully achieved its goal of predicting and ranking relevant events for users, with a solid MAP@200 score of **0.601**. By addressing challenges like data inconsistencies, missing columns, and feature mismatches, I was able to refine the model and improve its performance. There is potential for further improvement through enhanced feature engineering, deeper hyperparameter tuning, and incorporating ensemble models, which could lead to even more accurate recommendations.