SWE041 - MLOps
Homework I
E-commerce Behavior Report

Egecan Evgin
190701157

Please download the Ecommerce Behavior dataset from the link: https://www.kaggle.com/datasets/mkechinov/ecommerce-behavior-data-from-multi-category-store?resource=download

Columns:

- 'event_time': When the event happened (UTC). (i.e. 2019-11-01 00:00:00 UTC)
- 'event_type': One of [view, cart, remove_from_cart, purchase]. (i.e. view)
- 'product_id': Product ID. (i.e. 1003461)
- 'category_id': Product's category ID. (i.e. 2053013555631882655)
- 'category_code': Meaningful category name. (i.e. electronics.smartphone)
- 'brand': Name of the brand in lower case. (i.e. xiaomi)
- 'price': Numeric price value. (i.e. 489.07)
- 'user_id': Permanent user ID. (i.e. 520088904)
- 'user_session': User's session ID. (i.e. 755422e7-9040-477b-9bd2-6a6e8fd97387)

Question I: Identify an appropriate problem for the data and design according to your problem definition using appropriate data representation patterns.

Data design patterns:

- *Data Representation*; input selection, feature engineering and feature extraction.
- *Hashed Features*; transforming high dimensional categorical data to lower dimensional numerical data, also fixes missing words.
- *Embeddings*, learning a low dimensional representation of high dimensional data while preserving relevance using similarity.
- *Feature Cross*, involves creating new features by multiplying existing features.
- *Multimodal Input*, input data comes from different sources such as text, image and audio.

Feature engineering should be done, working with the date column and also cleaning the data.

Date column should be partitioned and hour, day features should be used separately.

My main goal is to predict whether a user is likely to make a purchase based on their past behavior.

Such as the products they have viewed, added to cart or removed from the cart.

This could be a binary classification problem and the target variable would be whether a purchase event occurs or not.

Question II: Design according to your problem definition using the appropriate problem representation patterns.

Problem representation patterns:

- *Reframing* converts a regression problem to a classification problem.
- *Multilabel*, handles the case that training examples can belong to more than one class.
- *Cascade* addresses situations where a problem can be broken into a series of problems.
- *Ensemble* solves a problem by taking multiple models and aggregating results to make predictions.
- *Neutral Class*, creating a class that doesn't have any functionality of its own, it serves as a base or parent class.
- *Rebalancing*, addressing class imbalance in a dataset.

Ensemble design pattern seems like a good idea to use here, because of its robustness and good classification performance.

Boosting could work fine here, Adaptive Boosting, Gradient Boosting all should be experimented.

With the following training pattern, this problem representation will be explained better.

Question III: Design according to your problem definition using appropriate model training patterns.

Model training design patterns:

- *Useful Overfitting*, can work well where a complex model is necessary like distilling ANNs, overfitting batches, simulations.
- *Checkpoints*, used to save middle versions of a model during training and it can help with early stopping and fine tuning.
- *Transfer Learning* involves leveraging pre-trained models for new tasks.
- *Distribution Strategy*, the training loop is carried out at scale over multiple workers, often with caching.
- *Hyperparameter Tuning*, the training loop is inserted into an optimization method to find the optimal set of parameters.

Even though transfer learning is so useful, hyperparameter tuning should be selected here.

Because the subject is pretty specific and there are not so many pretrained behavior predictors.

Random search optimization should work fine with this data and ensemble model.

With a relatively optimized searching method, best hyperparameters such as depth of trees, minimum samples, maximum conditions will be found.

Question IV: Design according to your problem description using appropriate resilient serving patterns.

Resilient serving patterns:

- *Stateless Serving Function*, handles a lot of prediction requests per second synchronously without storing the previous.
- *Batch Serving*, processes large amounts of data at once using batches, instead of handling individual requests asynchronously.
    - *Lambda architecture* is a design pattern which supports both online serving and batch serving.
- *Continued Model Evaluation*, evaluating a model's performance over time as new data gets available, monitoring and retraining.
- *Two-Phase Predictions*, divides prediction into two phases:
    - A fast, approximate phase.
    - A slower, more accurate refinement phase in the cloud.

- *Keyed Predictions*, assigning a unique key to each prediction request and using this key to store and retrieve the prediction.

Continued model evaluation would be a good choice on this occasion.

This machine learning model will be uploaded into a cloud environment and its microservices will start working online.

In online serving, the data is dynamic and history is static, so data could be drifted as the time passes.

User's choices might change and even the data distribution can change with time.

So continued model evaluation would work perfectly here, with doing the necessary retrainings with schedule.

Egecan Celik Evgin
190701157
Istinye University
Software Engineering