**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**School of Information and Communications Technology**

**PROJECT REPORT**

**SESSION-BASED RECOMMENDER SYSTEM FOR RETAIL PRODUCTS**

**Course**: Web Mining - Fall 2024

**Instructor**: Ph.D. Nguyen Kiem Hieu

**Group 1**

Đoàn Ngọc Cường - 20210141

Võ Đình Đạt - 20214890

Lê Trung Kiên - 20214907

Phạm Quang Trung - 20214935

Đoàn Thế Vinh - 20210940

*Hanoi, 2024*

# **1. PROBLEM INTRODUCTION**

## 1.1. Background & Motivation

Digitalization is on the rise for all aspects of everyday life, and people’s shopping habits are no exception. Nowadays, more and more brands of various industries, along with supermarkets, convenience stores, and retail chains, offer an online alternative to shopping, in place of face-to-face on-premise interactions, leading to a dramatic shift in consumer behavior. E-commerce platforms have become integral parts of daily life, and this development offers an immense wealth of valuable data on user interactions and behaviors within a digital system, since each action, such as product clicks, additions to cart, or purchases, taken by a user can be tracked and logged onto a structured database for further analysis. In return, marketing teams or sales managers can make use of this information as a guide towards strategies and policies the entice customers into spending more, thereby helping to increase profit.



*Figure 1.1: A few of the various e-commerce platforms in existence nowadays*

It goes without saying that analyzing such data presents an opportunity for businesses to gain profound insights into customer preferences and decision-making processes. By leveraging these insights, companies can develop personalized marketing strategies, optimize product recommendations, and improve the overall shopping experience. A core component of these strategies is the use of recommender systems, which help match users with items they are most likely to engage with or purchase.

Among different approaches to recommending, **session-based recommender systems (SBRS)** are particularly valuable in contexts where user identification is not persistent, such as guest sessions or short-term interactions. Unlike traditional user-based recommenders that rely on historical data tied to individual users, SBRS focuses on understanding the intent and preferences expressed within a single session. This makes them ideal for environments with high user churn, where data privacy concerns or regulatory constraints may limit the use of long-term user profiles.

Among the existent datasets available to the public, the **OTTO Recommender Systems Challenge** on Kaggle centers on this critical topic, tasking participants with developing innovative algorithms to predict the next actions (e.g., clicks, add-to-cart, or purchase) within a session. The challenge reflects real-world scenarios faced by online retailers and demonstrates the importance of SBRS in driving customer engagement and boosting sales.

**Motivation**

In the process towards deciding a topic for this Capstone project, the problem of Session-based recommendation systems align closely with topics covered in the course, and the group wants to use this opportunity to bridge the gap between the theoretical teachings and the implementation of an actual system utilizing such principles. In addition, in an era of fierce competition among e-commerce platforms, recommendation systems have become a cornerstone of business strategies. Success stories from companies like Amazon and Netflix highlight the potential of such systems to enhance user experience and drive revenue growth. With this in mind, the group feels that this project can serve as a chance to enhance the industry-wise applicability of what is learned during the lectures. Finally, this topic also requires us to make use of our analytics skill in order to understand this data, in contrast to simpler and more popular datasets. These reasons warrant the choice of this topic.

## 1.2. Problem Description

The challenge involves predicting user actions within an ongoing session, based on sequential event data. Each session is represented as a sequence of events, where each event includes information such as *product ID*, *timestamp*, and *action type* (e.g., click, add-to-cart, purchase). Given the partial sequence of events, the goal is to **accurately predict the remaining actions in a deliberately truncated session**, a typical setup of a Session-based Recommender System.

Specifically, the Otto dataset is provided in chronological order for each session, simulating the temporal nature of user interactions. In terms of the outcome to be achieved, the challenge involves three prediction tasks: clicks, add-to-cart actions, and purchases. The complexity of the problem is heightened due to the fact that the dataset is extremely large-scale, necessitating efficient algorithms capable of handling millions of sessions and events revolving almost two million products. Details regarding the dataset

# **2. DATA DESCRIPTION**

## 2.1. Overview

The **OTTO Recommender Systems Dataset** is a comprehensive collection of anonymized user behavior logs from the OTTO e-commerce platform, designed to serve as a benchmark dataset for research in session-based recommendation systems.

The dataset comprises approximately **12 million real-world user sessions**, with any identifiable details of users removed. Each session represents a sequence of user interactions on the OTTO retail site within a specific timeframe - all sessions are from 2022 and are of varying length. The OTTO group collects such information, from which data analytics and machine learning models can be implemented to derive insights into user navigation patterns and decision-making process when they shop online. It is also confirmed that each session is of a unique user, so there are correspondingly 12 million users appearing in the dataset; however, this will not matter much since the data itself is inherently anonymized.

Across sessions, the dataset encompasses around **220 million events**, with loggings of three primary types of actions that users of the site can take. Details of these actions are elaborated on as belows.

| **Action** | **Description** |
| --- | --- |
| Clicks | Short for “clicking a product”. These are events where a user clicks on a specific product, from which detailed information are presented to them. |
| Carts | Short for “adding to cart”. These are events where a user adds an item onto their shopping cart, usually indicating a higher level of intent to purchase. Most e-commerce sites offer a virtual cart - users can add multiple items before heading to check-out for payment. |
| Order | Short for “ordering a product”. These are complete purchases or full transactions. Business terminology consider this a successful conversion. |

In addition to events within a session, a crucial piece of information is the unique articles (or products/items). The dataset includes approximately **1.8 million articles**, a diverse catalog offered on the inventory of the OTTO platform. It is also notable that aside from article IDs, we are offered no extra information on the inherent nature of the products - no names, descriptions, or categories. All articles are nameless and stateless entities, only known for their unique distinction from one another. The detailed statistics regarding the dataset are as follows.

| **Set (Training/Test)** | **Data** | **Size** |
| --- | --- | --- |
| Training | Sessions | 12,899,779 |
| Articles | 1,855,603 |
| Events | 216,716,096  + *Clicks*: 194,720,954  + *Carts*: 16,896,191  + *Orders*: 5,098,951 |
| Test | Sessions | 1,671,803 |
| Articles | 1,019,357 |
| Events | 13,851,293  + *Clicks*: 12,340,303  + *Carts*: 1,155,698  + *Orders*: 355,292 |

## 2.2. Data Format

The dataset is structured in JSON Lines (.jsonl) format, where each line represents a single session with the following structure:

| {  "session": 42,  "events": [  {  "aid": 0,  "ts": 1661200010000,  "type": "clicks"  },  {  "aid": 1,  "ts": 1661200020000,  "type": "clicks"  },  {  "aid": 2,  "ts": 1661200030000,  "type": "clicks"  },  {  "aid": 2,  "ts": 1661200040000,  "type": "carts"  },  {  "aid": 3,  "ts": 1661200050000,  "type": "clicks"  },  {  "aid": 3,  "ts": 1661200060000,  "type": "carts"  },  {  "aid": 4,  "ts": 1661200070000,  "type": "clicks"  },  {  "aid": 2,  "ts": 1661200080000,  "type": "orders"  },  {  "aid": 3,  "ts": 1661200080000,  "type": "orders"  }  ]  } |
| --- |

*Note: This sample is not part of the dataset; rather, this is an example of a possible session.*

In this sample, ***session*** is a unique identifier for the user session, while ***events*** is a list of events that occurred during the session, each containing: ***aid***, the article (product) identifier; ***ts***, the timestamp of the event in Unix milliseconds, and ***type***, the type of interaction (one among *clicks*, *carts*, or *orders*)

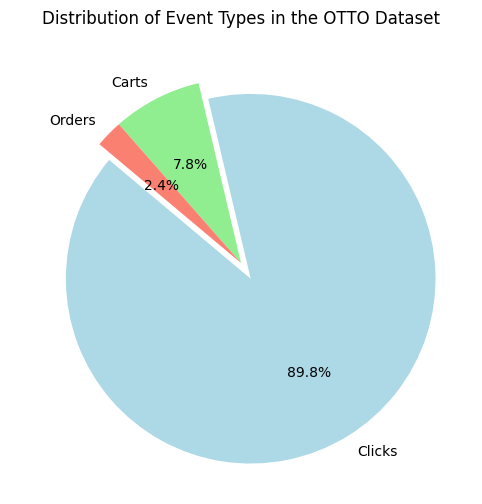
## 2.3. Exploratory Data Analysis

In this section, we delve deeper into the nature of the dataset and offer some insight, via notable statistics, into the size and scale of the dataset.

The dataset exhibits a wide range of events per session, with a mean of **16.80** and a standard deviation of **33.58** in the training set. The median number of events per session is **6**, indicating a right-skewed distribution where a significant number of sessions have relatively few interactions, while a smaller proportion includes a high volume of events.

The average number of events per item is approximately **116.79**, with a substantial standard deviation of **728.85**, suggesting that while some items receive consistent attention, others experience sporadic or minimal interactions. The median number of events per item is 20, further highlighting the skewness in item interaction distribution.

Constituting the majority of events, clicks account for about **89.8%** of the total interactions, reflecting users' initial engagement with products. Representing approximately **7.8%** of events, cart additions indicate a higher level of purchase intent. Making up around **2.4%** of events, orders signify completed transactions and are the primary metric for conversion.



*Figure 2.1. The distribution of Event Types in the OTTO Dataset. It is highly clear that product clicks are the most common, followed by cart additions then complete orders, which follows the intuitive understanding that only a small proportion of considered items are purchased by users*

With a density of **0.0005%** (calculated as the ratio of the number of observed interactions between distinct user-product pairs to the total number of possible interactions) , the dataset is highly sparse, indicating that only a small fraction of possible user-item interactions are observed. This sparsity presents challenges for modeling and necessitates robust techniques to effectively capture user preferences. Overall, these findings will serve as natural guidance towards the modeling process, in order to shape practical solutions that can manage the dataset’s unique characteristics and challenges.

# **3. METHOD DETAILS**

## 3.1. Method 1

The method follows the classical Retrieve (Candidate Generation), Rerank and get the top 20 items for click, add to cart or buy for the session. Approximately 56 candidates are retrieved per session using a combination of session-based heuristics, co-visitation statistics, and Word2Vec embeddings, followed by ranking and selecting the top 20 items for clicks, carts, and orders.

### 3.1.1. Candidate generation

The retrieval process incorporates multiple sources to maximize diversity and relevance:

* **Re-visitation matrix**: Add all previous items from the session to the pool of candidates.
* **Co-visitation matrix**: Add items frequently visited together. Then we add the 20 most frequent items per each previous item. We utilized several interactions between items listed below to create the co-visitation matrix.
  + Click-to-click: items are clicked together in the span of 12 hours.
  + Click-to-cart-or-buy: one item is clicked and then the other is added to cart or ordered in the span of 24 hours.
  + Cart-to-cart: items are added to cart together in 24 hours.
  + Cart-to-buy: one is added to cart and then the other is ordered in 24 hours.
  + Buy-to-buy: items are bought together in a 24-hour interval.
* **Similar item matrix:** Add similar items in terms of Word2Vec embeddings.We add at most 20 similar items per each previous item. Embeddings of items are created based on Word2Vec embeddings from the sequence of all items in the session and from sequence of only cart & buy items in the session.
* **Popular items in the same cluster of sessions:** Compute sessions embeddings based on Word2Vec embeddings of items in the session (a weighted average by type and time). Find clusters of sessions using KMeans clustering (~50 clusters found). The top 20 most popular items are added from the same cluster.

The maximum recall@20 possible for top K retrieved candidates if ranked ideally is shown below.

| **type** | **recall@20-top20** | **recall@20-top100** | **recall@20-top200** | **recall@20-topall** |
| --- | --- | --- | --- | --- |
| clicks | 0.196203 | 0.5307 | 0.560093 | 0.569288 |
| carts | 0.152458 | 0.424199 | 0.467714 | 0.50739 |
| orders | 0.16003 | 0.481797 | 0.584761 | 0.713684 |
| total | 0.161375 | 0.469408 | 0.54718 | 0.637356 |

### 3.1.2. Feature Engineering

To enhance ranking, feature engineering is conducted. Approximately 100 features are generated for each candidate, focusing on both interaction history and contextual information.

* Session specific features:
  + Interaction counts by type (click, cart, order) and their ranks.
  + Time-based features, such as time since the last interaction of a specific type.
  + Relative position and rank of items within the session.
* Co-occurrence and Similarity Features:
  + Co-occurrence counts and ranks (e.g., frequency of items being clicked or carted together).
  + Word2Vec similarity metrics (Euclidean distance and ranks).
  + Similarity between session embeddings and item embeddings (cosine similarity, Euclidean distance).
* Popularity Features: Item popularity rank within the same session cluster.

### 3.1.3. Re-ranker Model Training

The ranking phase utilized **LightGBM** with a LambdaRank objective to predict item relevance for clicks, carts, and orders. Separate models were trained for each target type using the following strategy:

* Target: a retrieved item is marked with 1 if item was clicked, carted, ordered; and 0 if not.
* Removed sessions without positive samples. This decreased the volume of rows to ~13% of the original dataset for clicks, ~3.5% for carts, and ~2.5% for orders.
* A maximum of 100 negative samples per session were included, maintaining a positive-to-negative ratio of 1:40.

The datasets had the following:

| **Type** | **Avg positive / session** | **Avg negative / session** | **Total rows** | **Sessions** |
| --- | --- | --- | --- | --- |
| Clicks | 1 | 41 | 40M | 1M |
| Carts | 1.3 | 50 | 11M | 220K |
| Orders | 1.7 | 57 | 7.5M | 130K |

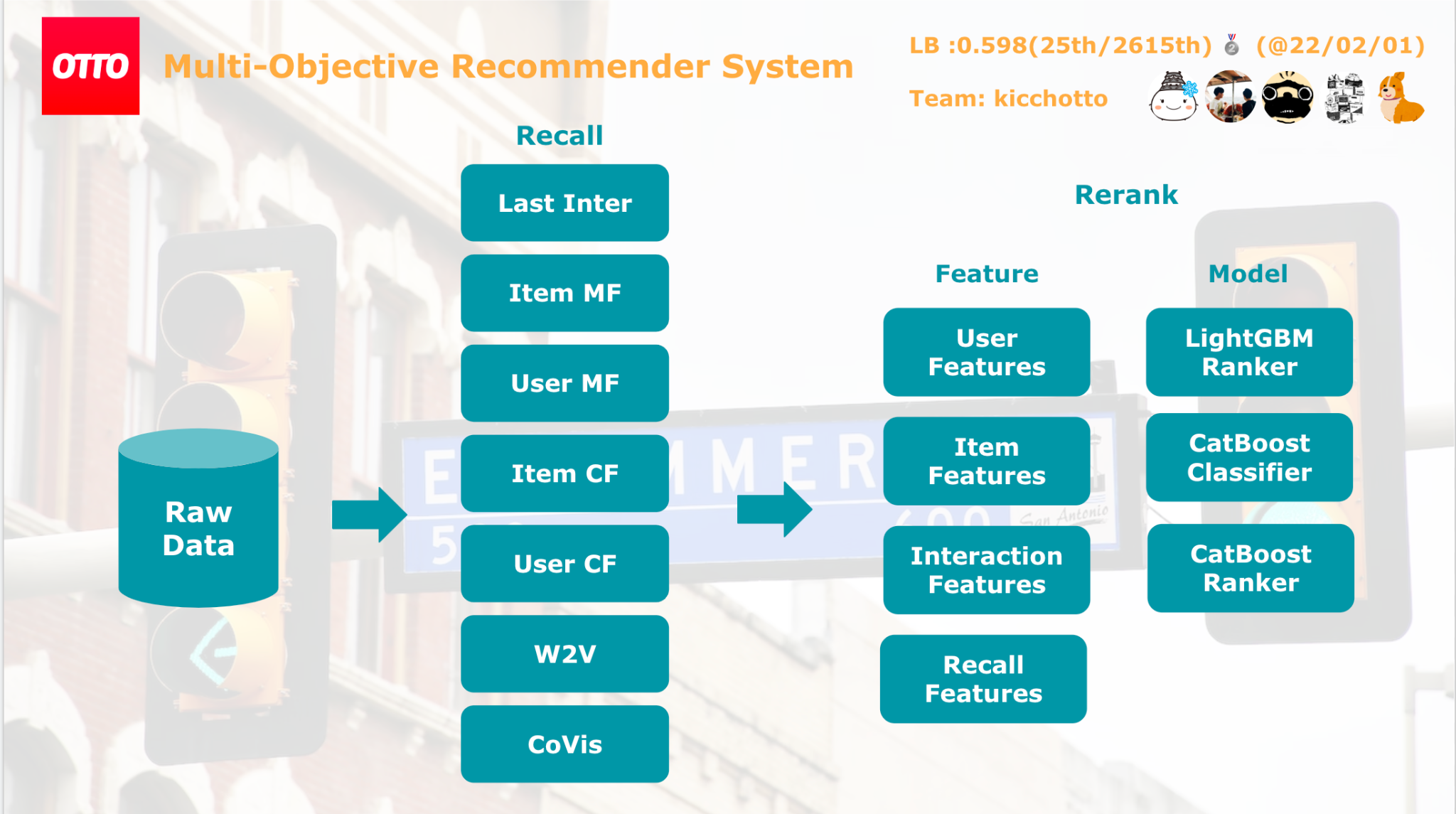
We train 3 LightGBM models for each target with the same parameters as below:

| PARAMS\_LGBM = {  'objective': 'lambdarank',  'boosting\_type': 'gbdt',  'metric': 'ndcg',  'n\_estimators': 150,  'learning\_rate': 0.25,  'max\_depth': 4,  'num\_leaves': 15,  'colsample\_bytree': 0.25,  'subsample': 0.50,  'min\_child\_samples': 20,  'importance\_type': 'gain',  'seed': 42,  } |
| --- |

No hyperparameter tuning was performed due to time constraints, and parameters were adjusted based on initial experiments.

## 3.2. Method 2

The classical pipeline is still applied in this method. The summary of this method can be seen in the image below



*Figure 3.1: Illustration of the second methods workflow*

### 3.2.1. Candidate generation

As in the image, the candidates are generated from:

* Recently interacted items
* **Word2Vec based:** A Word2Vec model was trained on item sequences (aids) to learn dense embeddings for each item. These embeddings capture semantic relationships between items based on their co-occurrence patterns. Top-K candidates were retrieved by finding the nearest neighbors in the embedding space, accelerated using FAISS-GPU for efficient similarity search.
* **Co-Visitation Matrix**: A co-visitation matrix, inspired by Chris’s methodology, was used to identify candidates based on items frequently appearing together in user sessions. This approach leverages historical interaction data to surface co-related items.
* **Item Matrix Factorization (Item MF)**: An Item Matrix Factorization model was implemented with item embeddings learned using BPR loss. This encourages items co-occurring in sessions to have similar embeddings. To address biases, the loss was scaled inversely by item popularity (item\_size) and temporal difference (ts\_diff), ensuring embeddings prioritize temporally proximate co-occurrences while reducing the impact of popular items dominating the representation.
* **User Matrix Factorization (User MF)**: A User Matrix Factorization model was built, learning session embeddings and item embeddings jointly through BPR loss. This model ensures session-specific preferences are captured effectively. Similar to Item MF, the loss was adjusted inversely by item\_size and ts\_diff, encouraging embeddings to focus on temporal proximity and mitigate popularity bias.
* **Item-Based Collaborative Filtering (Item CF)**: Item-based collaborative filtering was implemented using Polars to compute similarity weights between item pairs. Candidate items were generated based on the most similar items, determined by aggregating weights using sum, min, max, or mean. Adjustments to weights included scaling inversely by item\_size, inversely by ts\_diff, and multiplying by a trend coefficient that emphasizes more recent interactions.
* **User-Based Collaborative Filtering (User CF)**: User-based collaborative filtering was similarly implemented, focusing on session-level interactions to identify candidates. Adjustments similar to Item CF were applied to mitigate biases and emphasize temporal relevance.

### 3.2.2. Feature Engineering

We created approximately 200 features to capture diverse session, item, and interaction-level characteristics.

**Session-Level Features**:  
Features were aggregated over sessions to capture temporal and behavioral patterns, including:

* **Timestamp Statistics**: Aggregated metrics such as mean, min, max, and standard deviation of interaction timestamps.
* **Interaction Frequency**: Count and rank of candidate selection events within sessions.

**Item-Level Features**:  
Features describing individual item interactions and properties, such as:

* **Candidate Characteristics**: Metrics like candidate score, rank, and whether the item was selected.
* **Interaction Time Distribution**: Mean hour of interactions (e.g., clicks, carts, orders) aggregated over items.

**Interaction Statistics**:

* Aggregated counts of interaction types (e.g., clicks, carts, orders) over various time windows.
* Multi-action probabilities for each aid, capturing the likelihood of specific actions .

**Temporal Dynamics**:  
Temporal features ensured the model could account for recency effects and patterns over different periods.

### 3.2.3. Ranker training

For the rankers, we experimented with 3 models

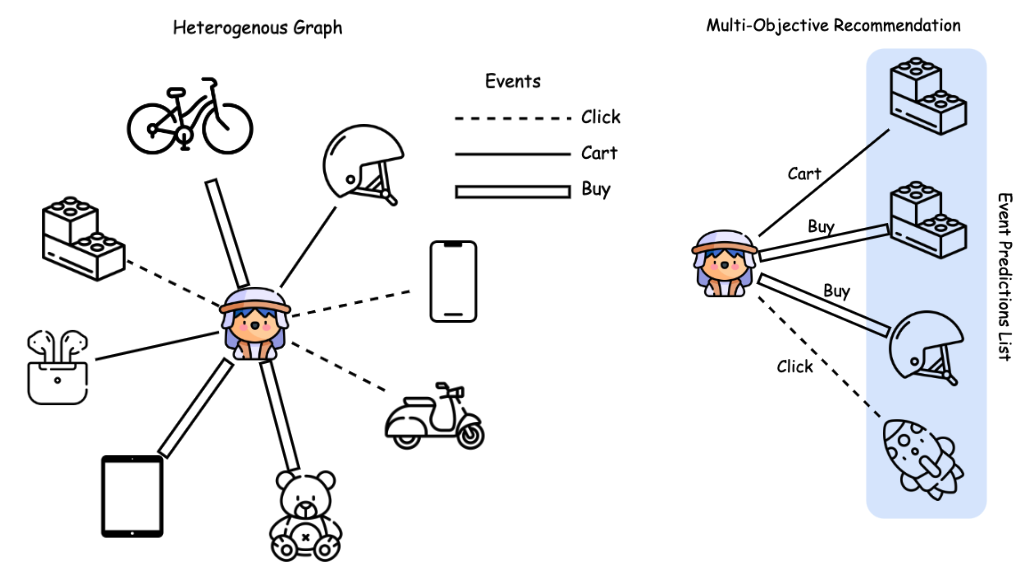
**LightGBM Ranker (LambdaRank)**:  
This model was used to directly optimize a ranking objective. LambdaRank adapts gradient boosting for ranking tasks by incorporating pairwise comparisons and position-based weighting, making it well-suited for improving metrics like NDCG and Recall.

**CatBoost Classifier (Logloss)**:  
A classification approach was applied using CatBoost, optimizing for log-loss. This model predicts the likelihood of an item being interacted with based on the feature set. The predictions were then used to rank items by their probabilities.

**CatBoost Ranker (YetiRank)**:  
The YetiRank algorithm in CatBoost was used for direct ranking optimization. YetiRank incorporates pairwise and listwise ranking methods, considering user preferences and engagement levels to produce highly relevant rankings.

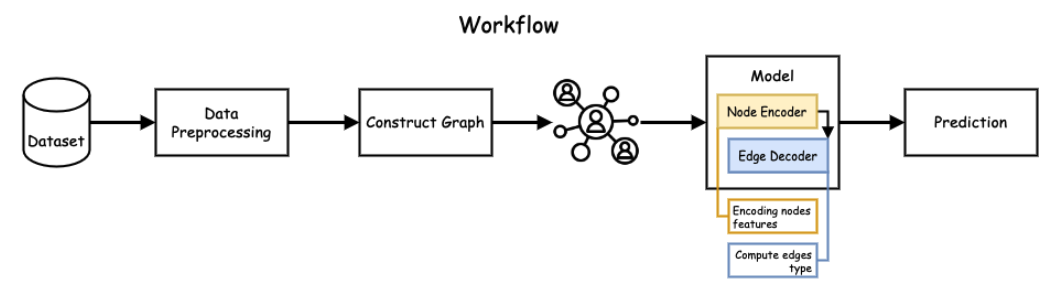
## 3.3. Method 3

Another way of approaching the session based recommendation system problem beside the classical architecture is using Graph Neural Network, that is we formulate the problem into a link prediction problem. Heterogeneous graph (also called a **heterogeneous information network**) is a type of graph where nodes belong to different types and edges represent different relationships. This is particularly useful for our problem as sessions (or users) can click, add to cart or order an item that can be of many types (different aid). The objective is to build a multi-objective link prediction on a large-scale e-commerce heterogeneous graph. An illustration of this method is shown below.



*Figure 3.2: An illustration of the heterogeneous graph (left) which consists of multiple node and edge types and multi-objective recommendations (right) which takes prediction on multiple event types and items.*

First, we will preprocess the data into nodes and edges. For the nodes, we will have 2 types of nodes: the first represents the users (sessions) and the other represents the items. The latter will also encapsulate its embeddings (randomly initialized) along with its id. We connect these 2 nodes with the ‘event’ edge. The edges will contain their source and destination id along with its label (click, cart or order). The edges are undirected. We build a model that contains a node encoder and edge encoder and train it with weighted mean squared error. The overall workflow can be seen below:



*Figure 3.3: An illustration of the overall workflow of the third method.*

# 

# **4. EVALUATION METRICS**

For the evaluation metrics, we will use Recall@20 provided by the competition on each action type. The three recall values are weight-averaged:

where is defined as

and is the total number of sessions in the test set, and **predicted aids** are the predictions for each session-type (e.g., each row in the submission file) truncated after the first 20 predictions.

For each session in the test data, the task is to predict the aid values for each type that occur after the last timestamp is the test session. In other words, the test data contains sessions truncated by timestamp, and weare to predict what occurs after the point of truncation.

For clicks there is only a single ground truth value for each session, which is the next aid clicked during the session. The ground truth for carts and orders contains all aid values that were added to a cart and ordered respectively during the session.

# **5. DISCUSSION**

# **6. CONCLUSION & FUTURE WORKS**

# **7. REFERENCES**