Applied ML Final Project Report – Team 36

Hongtao Jiang, Arvind Kanesan Rathna, Zeyu Jin, Mohit Medini Singh, Ling Sun

I. Background & Context

H&M Group is a Swedish multinational clothing company known for its fast-fashion apparel and accessories. It is the second-largest global clothing retailer, with nearly 5,000 stores globally. It is present in 53 online markets, where it utilizes product recommendations to help its customers navigate through its extensive catalog. Our task is to apply machine learning techniques on purchase history of customers across time and supporting metadata to predict twelve articles of clothing a customer would purchase in the 7-day period following the end of the training data.

II. Exploratory Data Analysis (Dataset Information)

The dataset has been provided by Kaggle here. There are ten broad clothing types which encompass 131 underlying product types sold by H&M. Ladieswear makes up most SKUs, followed by clothing for children; menswear pales in comparison. Most of the clothing sold is black, and trousers, dresses, and sweaters are the most frequently sold articles of clothing. The average H&M customer age is around 37, with outliers being those above the age of 82. Most customers are active Club Members (93.1%), but most customers (64.72%) do not subscribe to H&M fashion news. The transaction dataset was quite large, requiring 6.28 GB of memory.

We performed numerous memory reduction techniques to transform the data without losing any information and reduce total memory usage to .92 GB. There were no missing values in the transaction dataset, and it was impossible to impute values on articles of clothing dataset. The customer dataset had some missing values associated with club membership, which we were able to replace with binary values. We also imputed the mean of the data for missing values in the age column. Lastly, we performed one-hot-encoding on membership status and ordinal encoding on fashion news frequency.

III. Recommender Algorithms

Evaluation Metric

To build recommender algorithms on the dataset, six techniques are experimented with, and models are trained on pre-processed data. The evaluation metric chosen is Mean Average Precision @ 12 (MAP@12) to measure predictions on a balanced basis.

$$MAP@12 = \frac{1}{U}\sum_{u=1}^{U}\frac{1}{\min(m, 12)}\sum_{k=1}^{\min(n,12)} P(k) \times rel(k)$$

where U is the number of customers, $P(k)$ is the precision at cutoff k, n is the number predictions per customer, m is the number of ground truth values per customer, and $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant (correct) label, zero otherwise.

Baseline: Top-12 Frequent Recommender

We defined our baseline as a recommender that outputs the customer's most frequently bought products within a certain time interval. If the customer has purchased fewer than 12 different products within the given time interval, the predictor fills in with the most popular products across all customers. We experimented with two different time intervals to examine the relevance of older purchases on future ones: 3 weeks & 6 months. For the 3-week-period the recommender scored 0.0204, while for the 6-month-period it scored 0.0068. From this we concluded that recency effects of purchasing history were quite strong, in line with our intuition that consumer shopping habits potentially change over time/season.

Modified Baseline: Time-Decaying Popularity

Applying what we learned from our baseline model, we engineered a new metric, *pop_factor*, that is simply the inverse of the number of days since 9/23/2020, one of the most recent purchases in the dataset. We used this metric to weigh each transaction before inputting it into our top-12 calculations for each customer. Additionally, we chose to further narrow down the time frame over which to look for customer transactions to two weeks. This yielded significantly better results than either of our baselines, with a score of 0.0216.

Regression-based Recommender: Logistic & Random Forest

Before applying complex recommender algorithms, we experimented with two common regression-based ML models. Besides aiming for a better performance, the goal was to fetch feature importance from the weights of the regression-based models.

Based on **Figure 1**, both the random forest predictor and the logistic regression model give high importance to *last_time_the _item* and *total_sales_the_items*. This once again confirms what we saw in our baselines and time-decay model that customers tend to purchase articles that they have purchased recently. We also saw from this that pricing also has a significant impact on the prediction outcomes.

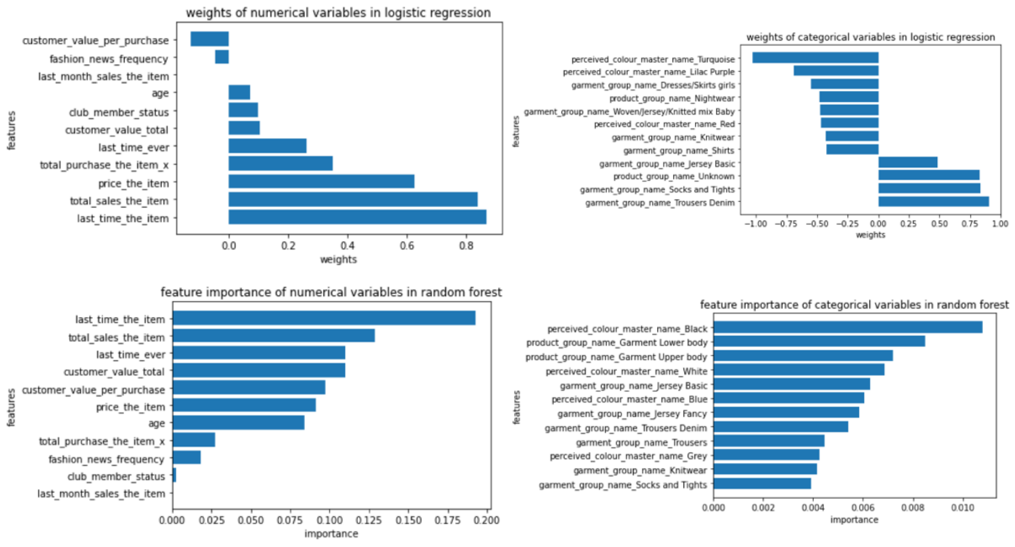

Figure 1: Feature importance for Logistic Regression and Random Forest

In terms of model performance, both regression-based recommenders did relatively poorly; the logistic regression has a MAP@12 of 0.0066 while the random forest scored 0.0065. The potential drawbacks of regression-based models are that one has limited control over the model fitting process, with heavy dependance on proper feature engineering. Hence, we decided to apply more complex recommender algorithms.

LightGBM Ranker

LightGBM is an improved version of traditional Gradient Boosting Machine (GBM). GBM is an ensemble model of decision trees trained in sequence for the algorithm to learn from residual errors; LightGBM focuses more on the grow on leaves while traditional GBM focuses more on the grow on trees per level, that is, the leaf-wise-growth in **Figure 2**.

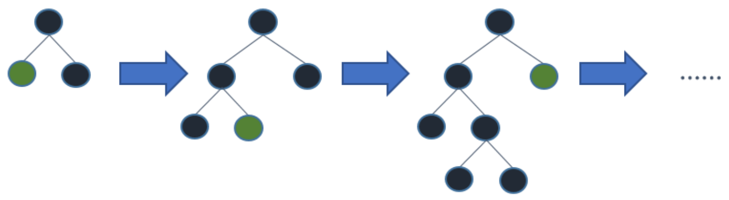

Figure 2: Leaf-wise-growth

Based on the results from data exploration, we noticed that people tend to buy more popular items from last week, where the popular item (Best Sellers) are defined by order counts on a weekly basis. Therefore, it is reasonable to add a variable to indicate the ranks for best sellers of the respective last week on each order.

The hypothesis is made that customers tend to purchase articles that were trending; and if they do not purchase any, then the respective article is not in the customer's favor. The labeling process is to label articles that customer purchased before as positive, and articles that belongs to the best sellers of last week, but customer did not purchase as negative. According to the feature importance plot (**Figure 3**), the best seller rank variable (BSR) significantly dominates the model fitting process, which supports the hypothesis.

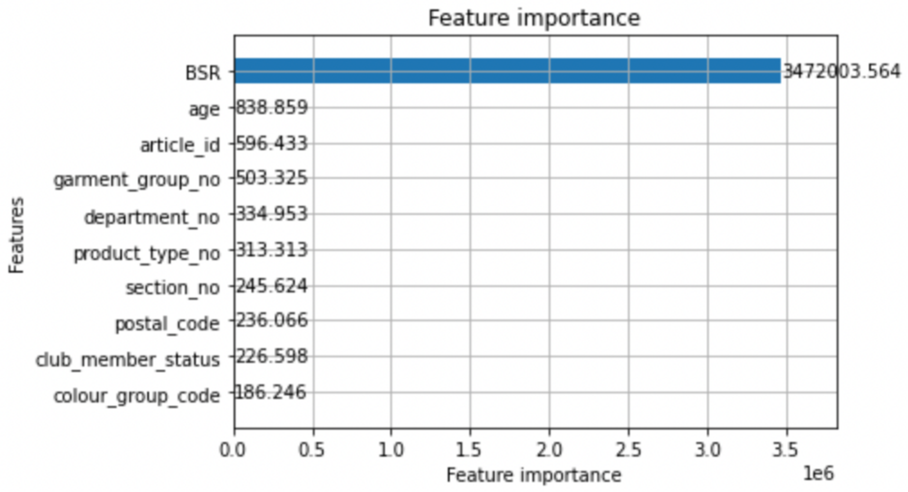

Figure 3: Feature importance for lightGBM Ranker

The performance of the model is optimized such that it acquired a MAP@12 of 0.0208, which is slightly better than the 3-weeks baseline and much better than the 6-months baseline model.

Alternating Least Square (ALS)

Alternating Least Square (ALS) is an algorithm to factorize matrices. More specifically, the goal is to factorize user-item interaction matrix with collaborative filtering for implicit feedback datasets. The interaction matrix in the project is a sparse matrix generated by the coo matrix technique, where each matrix value is $C_{ui} = 1 + \alpha r_{ui}$, and r is the interaction and scales by α. The matrix value also supports $P_{ui} = \{1\ if\ R > 0\ else\ 0\}, R = r_{ui}$. P_{ui} represents preference of each customer-article pair while C_{ui} represents confidence one has over the interaction pair. The cost function of locating the matrix is as below.

$$\min_{y^*,y^*} \sum_{u,i} (C_{ui}P_{ui} - X_u{}^T Y^i)^2 + \lambda(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

where λ is the regularization factor, X is the customer matrix, and Y is the article matrix.

The ALS locates a preference score by first computing $customer(x)$ and $article(y)$ vector differentiating the above cost function by x, y respectively; and then compute the preference score by $p_{ui} = x^T y_i$. The algorithm would recommend top 12 pairs with largest preference score p_{ui} to each user.

The optimized MAP@12 of the ALS is 0.0183, which is slightly worse than the 3-weeks baseline model but much better than the 6-months baseline model.

Neural Network

A PyTorch deep learning model is trained and fitted. For the preprocessing part, only articles that are purchased in the past 2 years are considered. Each transaction from the dataset defines a single training sample. For each training sample, the target is the article that was purchased, and the of input to the model is the list articles purchased up to k weeks before the current purchased article by the transaction's customer.

A NN model using 1-D convolution is constructed, with Leaky ReLu as activation function for middle layers along with batch normalization to avoid potential overfit. Adam optimizer is defined while the learning rate schedule is also set up based on epochs. Dice loss and BCE (Binary Cross-entropy) loss are defined to be the loss functions where $dice = \frac{2TP}{2TP+FP+FN}$. The output of the model is a vector representing the probability of any article from the dataset being purchase. The algorithm selects top 12 result based on the output probabilities. The training loss and validation loss is shown in **Figure 4**.

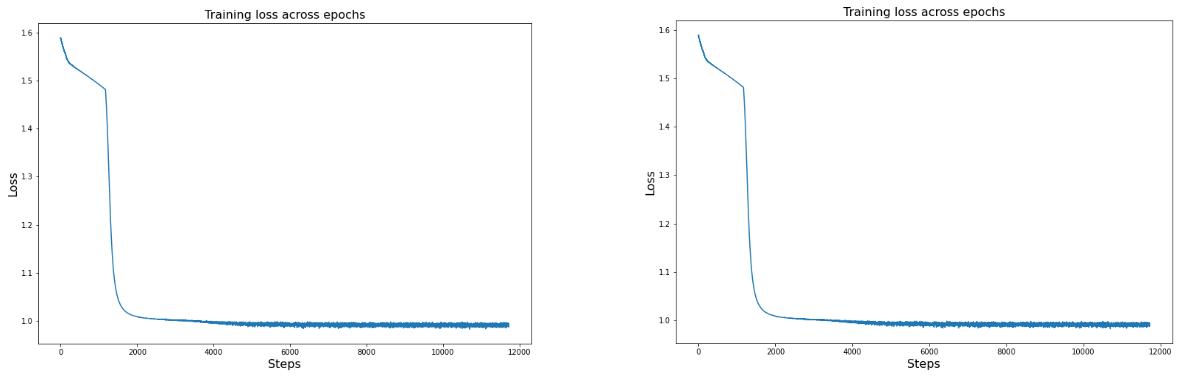

Figure 4: Loss Plot over Epochs

The MAP@12 of the NN model is 0.0198, which is slightly worse than the 3-weeks baseline model but much better than the 6-months baseline model.

IV. Summary of Results

The goal of the project was to make predictions for customers' next 12 purchases of articles. After performing pre-processing and exploratory data analysis, six recommending strategies are applied and tested. The baseline predictor of predicting top-12 popular articles of clothing has a good performance on 3-weeks' time interval but poorly on a 6-months' time interval. The regression-based predictors are outperformed by the baseline as one has limited control of the fitting process and the model heavily depends on data pre-processing. The time Decaying Popularity model obtains the best performance with a MAP@12 of 0.0216. The lightGBM Ranker, Alternating Least Squares, and Neural Network algorithms have similar performance comparing to the 3-weeks baseline model. The submission of the best model on Kaggle ranks 1118 out of 2395 submissions.