CSCI567 Final Project Report: Team 46

Akshit Keoliya, Peng He, Yi-Chien Lin
{keoliya,penghe,yichienl}@usc.edu

1 Approach

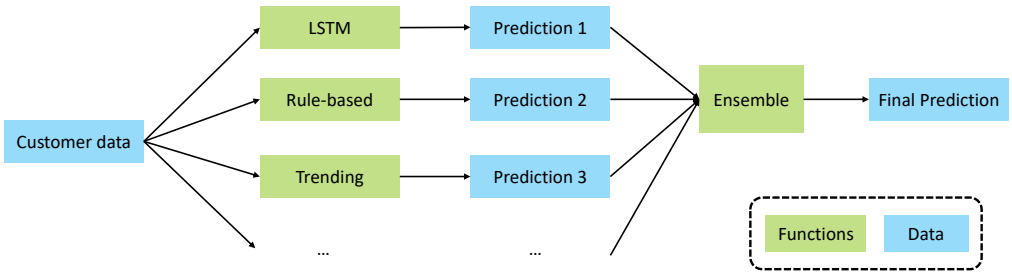

Figure 1: Framework overview

The general idea of our approach is shown in Figure 1. We utilize client purchase data as input and run several models to make predictions. Then, to gather the prediction results and make a final prediction, we use our customized ensemble approach (Section 3), this allows us to take use of diverse models and deliver a robust prediction. Choosing different models aimlessly, on the other hand, rarely improves the final prediction. If two models are quite similar, for example, choosing both will result in a similar final result. In the following section, we'll go through how we process the data, the models we choose and how they differ from one another, our customized ensemble method, and finally our experimental results.

2 Data

2.1 Data Analysis

We were provided with three databases in the form of CSV files:

- **Articles Database:** It contains the metadata for each Article ID. This database consists of 105,542 articles, each with a feature dimension of 25 (14 text columns and 11 numeric columns).
- **Customers Database:** It contains the metadata for each Customer ID. There are 1,371,980 customers in total, each with a feature dimension of 7 (4 text columns and 3 numeric columns).
- **Transactions Database:** It contains purchases of each customer for each date, along with additional information. It consists of 31,788,324 transactions in total, each transaction has a feature dimension equal to 5 (2 text columns and 3 numeric columns).

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2.2 Data Preprocessing

Direct use of the data provided may lead to some issues since there are some missing values, also, some features are not useful for prediction and should be eliminated. We summarize the data preprocessing done in the following subsections.

2.2.1 Data Augmentation

We use the last four weeks of transactions, as transaction before do not provide any valuable insights. We only consider users that have bought more than 40 items and items that have been bought more than 40 times. Also, if there are no transactions in last week, we consider transactions from the previous week. If there are no transactions in last 4 weeks, the most popular items from last week are considered.

2.2.2 Dealing with Null values

The customers database has 15,861 null values for age feature, which comprise of 0.01% of the entire database and hence it is safe to drop these entries altogether. Other null values for features such as "club_member_status", "Active", "FN" were substituted with 0.

2.2.3 Converting Data Format

We use one-hot encoding to convert the categorical features such "fashion_news_frequency" and "club_member_status" into numerical values. Note that the encoding also takes into account the null values that were substituted to 0.

2.2.4 Dropping Unnecessary Columns

Though there are a lot of features in the database, some are not useful for our prediction. We remove closely related features such as keeping only one of "product_type_no" and "product_type_name" features. We also drop the features such as "postal_code" and "sales_channel_id" which we consider ineffective for prediction.

2.2.5 Normalizing Values

We normalize the feature values using Scikit-Learn library's StandardScaler. We calculate new value (z) using the formula below:

$$z = (x - u)/s \tag{1}$$

where u is the mean and s is the standard deviation of the feature values.

2.2.6 Feature Engineering

We create new features which can be used by one of our models; this include features such as "growth_rate" for each article and "mean_price" for customer's purchase per week.

3 Ensemble Method

This method has two important components: how to choose weights for different models and how to rank different recommended products. Our initial thought was to utilize the LB score of each model as the model weight and the reciprocal of the item's sequence order as the rank weight for each object. Later, we attempted to assign similar weight to models using similar methods, as well as using an exponential function as the denominator to provide penalty for low-ranked items. We found out that giving low-ranked items a higher penalty can help increase the score, so we use a piece-wise function to combine these two ranking algorithms as

$$Rank(n) = \begin{cases} \frac{1}{n} & \text{if } n \leq p \\ \frac{1}{p+1} \cdot (\frac{1}{base})^n & \text{otherwise} \end{cases} \tag{2}$$

Customer ID	Model 1	Model 2	...
0	A, B, C, ...	B, C, E, ...	
1	...	...	
...	(Weight 0.6)	(Weight 0.4)	

Table 1: An example of Ensemble

Table 3 shows an example to further explain our idea. For each customer, each model produces a set of prediction. Each model is also assigned a weight. When ensemble the result for customer 0, item A has the total score weight/rank = $0.6/1$; item B is predicted by both model, so the total score is $0.6/2 + 0.4/1 = 0.7$, and therefore is consider a more promising prediction over item A since it has higher total score. Let say we set the p in Eq. 2 as 2; this means that we downplay the importance of item with rank lower than 2. So, item C's total score is $1/4 \times 0.5^3 + 0.4/2 = 0.23$.

3.1 Models used for Ensemble

In order to improve prediction accuracy and robustness, we ensemble different models to produce the final result. One important factor to consider in terms of picking models is diversity. Picking similar models would make similar predictions which would hardly bring any improvement to the final prediction. Thus, we pick 9 different models, each has its own design philosophy. We describe the general idea of the models in the following subsections.

3.1.1 LSTM Model

Long Short-Term Memory (LSTM) is known for processing sequential data and make predictions, which suits this project ideally. This model first creates item features using the GRU4REC model and then starts training with the produced features. The limitation of this method is that it fails to make prediction for user who has bought very few items (less than 40 in heuristic); thus, for these cases, the model simply uses frequently bought items as a default recommendation.

3.1.2 Heuristic-based Model

First, identify the most bought items from the past few days or weeks and assign them a score. The score is evaluated by (number of times bought / time penalty), which favor recently bought items, e.g., if an item is bought yesterday, it is more likely to be bought again than items bought last week. The model makes prediction based on item score. The effectiveness of this model can be validated by splitting the dataset by time. For example, the dataset provides a purchase record from January to May, then we can use data in January to predict purchase in February, and use data in February to predict purchase in March, so on and so forth.

3.1.3 Singular Value Decomposition (SVD)

Singular Value Decomposition is a popular technique used is Recommendation Systems. The basic idea is to decompose the user-item matrix (which is the transaction record given to us) into user and item matrix, and we can make predictions using the decomposed matrices. Given that the user-item matrix is sparse and large, this model approximates the solution of SVD using stochastic gradient descent. The data is split into validation sets using the method mentioned above (Heuristic-based).

3.1.4 Exploratory Data Analysis (EDA)

This model employs an EDA approach. First, it uses another model to generate some predictions. Then, it analyses the similarity between each age range (Shown in Figure 2). Using the similarity matrix, the model can produce prediction for each age range. One drawback of this approach is its granularity, as it only produces results for each age range instead of each customer unlike other models. Still, it provides some insight and can be used in ensemble.

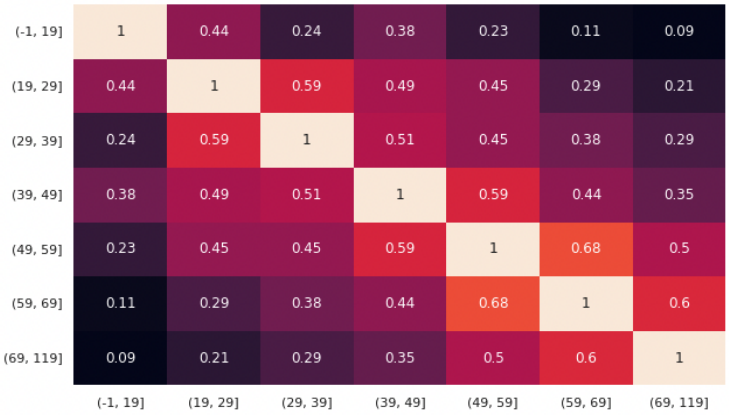

Figure 2: Similarity matrix of the EDA model.

3.1.5 K-Means Model

The K-mean approach can be view as an advanced version of the EDA model above. Instead of using the age feature only, the K-mean approach groups the customers into 12 clusters using various information like age, member status, new fashion frequency, etc. Then, it computes the analysis matrix and make prediction for each cluster. Like the EDA model, it only generates prediction for each cluster instead of individual customer, but it consider more factors when grouping the customers.

3.1.6 Principal Component Analysis (PCA)

This method creates a user feature matrix using the transaction record, and item feature matrix using the image dataset. Then, runs PCA on both datasets, and measures similarity by taking the product of two PCA matrices. By choosing the top 12 results, we obtain the prediction for each user.

3.1.7 Item Trends

This approach assume this week's sale is similar to last week, and the trend will be similar for the next week. It calculates the sales rate adjustment from last week to this week, and then applies the adjustment rate to predict next week's sale. Finally, by looking at the prediction sale rate, we pick the top 12 products for each customer.

3.1.8 Frequently Bought Together

The model first constructs a dictionary, for each item, which records the items that were usually bought together. Then, we apply a score evaluation function for the co-purchase items. Finally, based on the transaction record, we predict the customers' purchase using the co-purchase items with the highest score. The drawback of this method is that for some customers with less record, it is hard to make a robust prediction, thus, ensemble this method with other rule-based or model-based approach is essential.

3.1.9 Partitioned Validation

The model first creates different partitions of data based on "Last Purchased Item", "Colors of Purchased Item" "Popular Items for Group" and then ensembles predictions from each partition. The predictions are evaluated using one-week hold out validation.

4 Results and Analysis

We tried two different approaches: (1) one-level ensemble and (2) two-level ensemble. The one ensemble uses 8 models, and it is a straight-forward implementation as described in Section 3. For the second approach, we first ensemble 8 models, produce an intermediate prediction, and then run

the second level ensemble with the ninth and first model which treats the intermediate prediction as one model. We show some of the experimental results of approach 1 in Table 2, and results of approach 2 in Table 3. The two-level ensemble boost the performance from 0.0239 to 0.0242. The advantage of the second approach is that it add diversity to the way predictions are ensemble, which makes the result more robust.

The results show that the hybrid ranking approach can improve the LB score with little reliance on the base and p value used. This approach is shown to be quite resilient and requires little parameter tuning. Our second finding is that the final weight has little relevance with their LB scores. An intuitive explanation is that, some models with high LB score use similar approach, assigning higher weights to similar approaches decrease diversity and robustness. Thus, setting weights should not use LB score as the only factor, but should also take diversity into account.

Table 2: Experiment results of one-level ensemble

Parameters										
Weight for different models									penalty coefficient	avg MAP
0.45	0.88	0.72	0.80	0.88	0.70	0.92	0.92	1.22	3	0.0239
1.05	0.78	0.86	0.85	0.68	0.64	0.70	0.24	1.01	4	0.0239
1.05	0.78	0.86	0.85	0.68	0.64	0.70	0.24	1.01	5	0.0239
1.05	0.78	0.86	0.85	0.68	0.64	0.70	0.24	1.01	6	0.0238

Table 3: Experiment results of two-level ensemble

Parameters												
Weights in layer 1								Weights in layer 2			p	avg MAP
1.05	0.78	0.86	0.85	0.68	0.64	0.70	0.24	1.20	0.85	0.000	12	0.0240
1.05	0.78	0.86	0.85	0.68	0.64	0.70	0.24	1.20	0.85	0.020	7	0.0241
1.05	0.78	0.86	0.85	0.68	0.64	0.70	0.24	1.20	0.75	0.018	4	0.0242
1.05	0.78	0.86	0.85	0.68	0.64	0.70	0.24	1.00	0.75	0.030	12	0.0242

5 How to run our code

5.1 Library Dependency

Our method is based on public Python libraries: os, numpy, pandas, gc

5.2 Ensemble model location

Our method uses 9 different public results, here is the list of their corresponding url for the reader's convenience. In actual use, the version provided shall prevail:

- Model 1: [LB: 0.0231] `https://tinyurl.com/j8jv9vbz`
- Model 2: [LB: 0.0223] `https://tinyurl.com/254bmx82`
- Model 3: [LB: 0.0217] `https://tinyurl.com/26p6kdw8`
- Model 4: [LB: 0.0221] `https://tinyurl.com/bdhff2jk`
- Model 5: [LB: 0.0224] `https://tinyurl.com/yz97mnup`
- Model 6: [LB: 0.0204] `https://tinyurl.com/3a9mewdu`
- Model 7: [LB: 0.0227] `https://tinyurl.com/bp5b2mn5`
- Model 8: [LB: 0.0226] `https://tinyurl.com/mr3nt87t`
- Model 9: [LB: 0.0232] `https://tinyurl.com/yc7rd4hc`

NOTE: Due to D2L platform's 2GB file upload limitation, we have uploaded the respective submission files on a Google Drive folder. The folder can be found at `https://tinyurl.com/4nkhj4hb`

5.3 Running Suggestions

One submission file for each model above is saved in the custom_predictions folder. After running the kaggle.py in the parent folder, the final submission file will be saved in the same directory as the python file.

6 Future Work

To improve the performance of the ensemble method, we can focus on the following directions:

1. **More Robust Models:** We can use more algorithms to bring diversity to our ensemble algorithm, like the following:
   - **LGBM Ranker:** The LGBM algorithm is based on a gradient-based learning model that uses an ensemble model of decision trees. It has fast training speed and low memory usage.
   - **LightSANs:** is a self-attention model which allows long term semantics but uses an attention mechanism to make predictions based on relatively few actions. It scales linearly with respect to the user's historical sequence length in terms of time and space and is more resilient to over parameterization.
2. **Collaborative Filtering:** Recommend items from users with similar interests as the target users. Similarity of users can be computed using cosine similarity. Also computing interest level of each user for each item would help form suitable recommendations for new users.
3. **Additional Feature Engineering:** Other insightful features can be created such as "article_rebuy_rate", "days_after_buy" and "buy_recent_geographical".

7 Conclusion

For the "H&M Personalized Fashion Recommendations" competition, we developed an ensemble method with hybrid ranking function to recommend products based on data from previous transactions, as well as from customer and product meta data. Our method achieved Mean Average Precision score of 0.0242.