Stacking Ensemble for Fashion Recommendations

Carlos Osorio Vargas
University of Southern California
osoriova@usc.edu

Miguel Tay-Lee Macias
University of Southern California
tayleema@usc.edu

Anthony Guerra
University of Southern California
ag_394@usc.edu

1 Introduction

A recommendation system suggests items that are likely to be preferred by a user. Examples of recommendation system problems include Netflix providing relevant movies that a user would likely enjoy or YouTube providing suggestions on videos a user would potentially like. In the case of the H&M competition, the challenge is to develop twelve product recommendations based on customer and product data, as well as previous transaction history. The available meta data spans from simple data, such as garment type and customer age, to text data and images for different articles.

This paper first describes some exploratory data analysis on the data. It then goes on to describe the methods attempted to tackle this challenge as well as the reasoning behind those methods. Furthermore, the final results for these methods are shown. Finally, the paper concludes with thoughts on the problem as well as further work that might improve what was done.

2 Exploratory Data Analysis

2.1 Dataset Description

The data given in the competition consists of three files. These files describe the customer data, the product/article data, and the transactions when a customer bought a specific article. Additionally a folder of images is included which correspond to an article in the product data.

2.2 Exploration

Customer transactions were explored and sorted by different age groups to try to understand a correlation between different features. When noting the number of articles purchased by age group in Figure 1, one could see that younger people bought a much higher number of items. In contrast, when comparing average price of items by age group the graph seems uniform. Also, when looking at the distribution of top articles sold it seems as though articles start off very high and drop off dramatically. For the articles, there are a total of 45875 unique product names, 132 unique product types, and 19 unique product groups. Total number of unique product appearances of a product is 30. It starts with upper body garments being the most popular followed by lower body garments and accessories. When looking at most popular product group it falls in line with what you would think is popular.

3 Feature Engineering

The initial dataset contained different features pertaining to the customers, articles, and transaction history. In order to be able to represent the data in the different recommendation models, the different

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

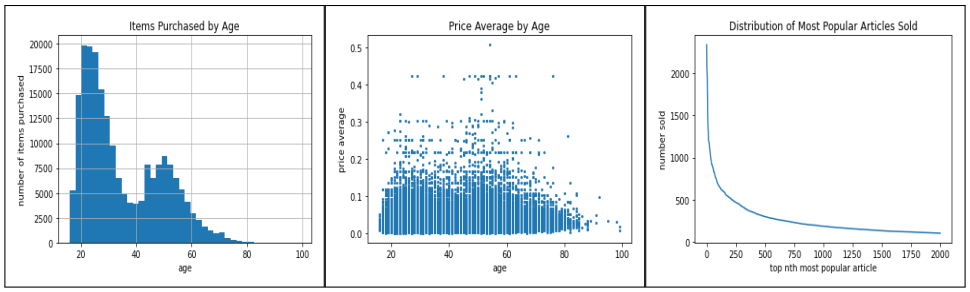

Figure 1: (a) Customer age histogram (b) Average price vs customer age (c) Distribution of most popular articles sold

features were tokenized or engineered using available packages such as the ones from scikit-learn and pandas. To start with, repeated columns and . Thus, only the string columns were kept, and the categorical features such as article color, product type, section name, among others were represented with a one-hot encoding, using the pandas getDummies method, or the sklearn OneHotEncoder package.

The article detail description was also encoded and used to identify groups of similar articles. In order to do this, the description column was transformed into a Term Frequency Inverse Document Frequency (TF-IDF) numerical feature set using the tfidfVectorizer from scikit-learn. Likewise, the customer features such were tokenized to the cluster different target groups, and although the customer location (postal code) would be useful to group nearby customers, it was ultimately dropped since there were about a million categorical codes, and computational resources were limited.

4 Recommendation Models

4.1 Collaborative Filtering

One of the initial approaches for the recommender system was to use collaborative filtering. This type of system focuses on identifying customers with similar interests to make the recommendations based on the items they have bought. The algorithm applied to support the collaborative filtering was matrix factorization, specifically the singular value decompostion (svd) of a pivot table of user-article interactions. The decomposition would allow to identify principal components in the article catalog and allow to rank items for each customer. This approach was advantageous for its simplicity and efficiency. However, due to its memory-intensive nature and the limited computational resources, significant results were not able to be obtained for the large customer dataset. Recommendation results could have only been obtained for a very small sample of the customer database.

4.2 Clustering

Clustering is a common data analysis technique used to get an intuition about the structure of the data. The purpose is to identify subgroups within the data such that data points that are clustered together are alike while different clusters are different. For the challenge a couple of different clustering methods were used to cluster customers together. After the customers are clustered together a count of the most popular articles of clothing from that specific cluster is created and the top twelve articles are used as the prediction for a customer if the customer came from that cluster.

Originally the k-means algorithm was used on the customer data concatenated with the mean price of everything the customer has bought. This proved to not have the best results so the model distributed by the TA was improved upon to make a model where customers were manually clustered based on age ranges. This improved score results by a little. This second model of manual age range clusters was then used in an ensemble of models which improved scores even further to a score of 0.02239. This goes to show a common theme throughout the paper that ensembling our models tend to show an increase in scores. In further work, it might be beneficial to experiment with gaussian mixture models seeing as though we might benefit from the soft-assignments given to customers.

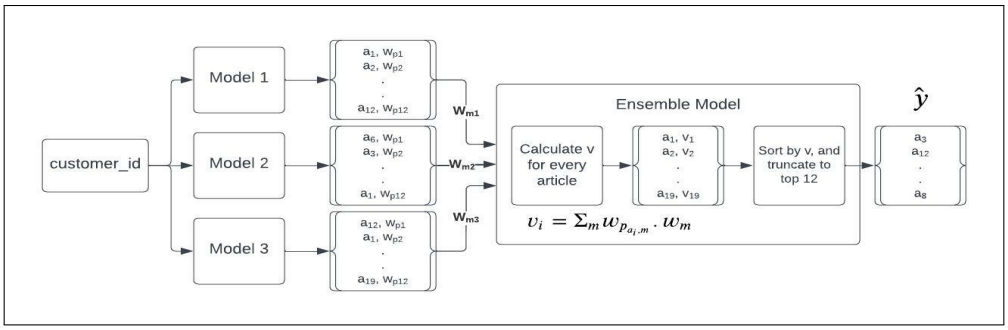

Figure 2: Diagram of the level-0 and level-1 ensemble prediction process. Here a customer ID is passed to the base models, which then provide a prediction of 12 articles each. A weight (w_{p_i}) is assigned to each article depending on its position inside each model's prediction list, and a weight (w_m) is assigned to each model. The ensemble model then returns the final prediction ($\hat{y}$).

4.3 Cosine Similarity

The cosine similarity measures the similarity of two vectors in an N-dimensional feature space. Recommendation systems often employ this metric to identify similar products to the ones a customer has previously bought. The approach for this model was to retrieve the articles a customer has recently bought and consolidate their features in a customer query. The query would then be compared to trending articles in the last few weeks by calculating the cosine similarity. The top articles were obtained by filtering popular article product names, as the same product could have multiple IDs, relating to different colors or versions.

Finally, each customer query was used to calculate the cosine similarity with the top articles, an obtain the top 12 results. The prediction with this model resulted in a MAP score of 0.0079, which is fairly good considering the large number of customers. Moreover, upon close examination of a few recommendations, the recommended articles were actually similar and related to what the customer had bought. However, obtaining an actual prediction of what items would be purchased turned out to be a more challenging task because of the sheer size of the H&M catalog.

4.4 Further approaches

In this section, typical models used for the H&M competition fashion recommendations are described. These approaches were adopted and modified in order to leverage them in the project's final stacking ensemble, and obtain the most optimal prediction.

Firstly, sequential models were some of the most popular alternatives used. A most commonly used library for this is Recbole, since it contains a lot of pre-defined models for context aware recommendations as well as sequential ones. The Recbole model utilized in the ensemble was a GRU4Rec, which uses a recurrent neural network architecture for session based recommendations. Likewise, another approach consisted on recommending articles that are frequently bought after a customer makes a purchase. It is a kind of sequential model, as it identifies what products the majority of customers will likely purchase based on their previously purchased product. Lastly, other approaches focused on recommending products that were trending during the time prior to the target week. These type of models obtained popular articles based on some metric, such as an exponential decay weighting associated with the recentness of the purchase.

5 Combining Model Results

5.1 Ensemble Learning

Ensemble learning is a commonly used approach in machine learning to help boost the predictive capabilities of diverse models. It mainly consists on combining the results of different individual models to build new predictions with a higher performance. While there are various different methods for constructing ensemble systems, such as bagging and boosting, the most appropriate for the

recommendation model being built was chosen to be stacking. This ensemble technique considers multiple base learners in parallel, where each provides its specific prediction, and the ensemble model is built to learn how to aggregate the results in a manner that it improves the accuracy of each prediction. Additionally, stacking can have different levels of hierarchy, where level-0 models are constituted as the weak base learners and level-1 is the model in charge of learning how to combine the level-0 predictions. There can also be additional levels in stacking systems which combine the results of level-1 learners and so forth.

For the purpose of the given task, level-1 and level-2 stacking models using a weighted average approach were built and experimented on to improve the score of the base models mentioned previously. As seen in Figure 2, after each level-0 model makes a prediction, the results are passed to the ensemble learner, which calculates a value (v_i) for each article (a_i) based on its positional weight ($w_{p_{i,m}}$) on each model and the respective model weight (w_m), as expressed in Equation 1

$$v_i = \Sigma_m w_{p_{a_{i,m}}}.w_m \tag{1}$$

Further on, the ensemble model sort the articles by their values (v_i) and truncates the result to return the top 12 predicted articles.

5.2 Training Model Weights

One of the main tasks in building the ensemble learner was to implement an algorithm to train the model weights. The first step in doing so, however, was to define the training and validation data. Since the task is to predict the articles most likely to be bought in the next 7 days, we divided the data as following. First, each base model was trained with data up to 09/08/2020 (14 days before the cutoff date). Second, a training set was constructed with transactions from 09/09/2020 up to 09/16/2020. This training set would be used to train the ensemble based on the predictions of the base models as specified above. Finally, a validation set was made from all transactions after 09/16/2020 (the last week of data).

Now, to train the model weights, stochastic gradient descent based on logistic loss was used. That is, for each training epoch, the ensemble model would iterate randomly through customers on the ground truth training-set, and compare each article found there to the level-0 predictions on that customer. Each model weight would then be updated using Equation 2 if the article was predicted by the model, or by Equation 3 if not predicted.

$$w_m := w_m - \eta(\sigma(v) - 1.5)w_{p_v} - \lambda w_m \tag{2}$$

$$w_m := w_m + \eta(\sigma(v) - 1.5)\delta - \lambda w_m \tag{3}$$

Here, η (as the learning rate), λ (as the regularization constant), and δ (the "weight" of an article not found on the list) are all hyper-parameters that will be tuned, whereas w_{p_v} is the positional weight of the article on the final predicted list and σ represents the sigmoid function. Also note that the -1.5 was included here to have the desired effect of reducing the model weight by more if the article is not found and v is large, as well as increasing it the most when the article is found on the model, and v is small. Finally, after each epoch, the mean average precision (MAP) (as defined by **??**) was used, along with the validation data-set, to estimate the precision of the current ensemble and plot its learning curve.[1]

5.3 Tuning Hyper-Parameters

One of the important aspects in building a learning model is tuning any hyper-parameters the model may include. In this case, there are three hyper-parameters that need to be tuned; η, λ, and δ. To do so, for each hyper-parameter an initial (low) value was chosen, and the model would be trained using that hyper-parameter for 15 epochs. Once the model was trained, its MAP score was recorded, and the hyper-parameter was multiplied by 3. This process would then repeat for 10 iterations, finally yielding the MAP scores for each parameter used, and plotted to decide on the most optimal value. The results for the level-1 ensemble tuning can be seen in Figure 3.

[1]The function used to calculate the mean average precision was taken from **??**

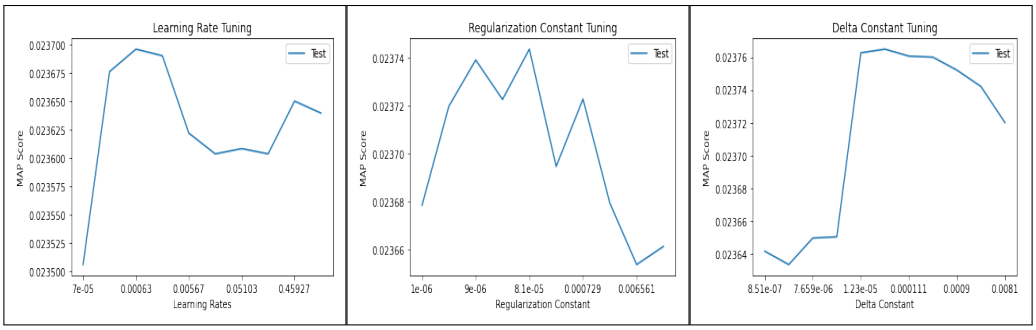

Figure 3: Hyper-parameter tuning for (a) Learning rate (b) Regularization constant (c) Delta constant.

6 Final Results

From the hyper-parameter plots in Figure 3, it becomes evident that there is "sweet spot" for each hyper-parameter to give out the best result, as the MAP score starts increasing and then decreasing in all of them. After getting these results, the following values were chosen for the hyper-parameters:

$$\eta := 0.0001, \quad \lambda := 0.00007, \quad \delta := 0.0004 \tag{4}$$

The optimal hyper-parameters were then used to ensemble different public and custom edited notebooks and obtain a final prediction. After trying various combinations of base models, the highest score obtained was 0.0240. Nonetheless, this score could potentially be surpassed by stacking more predictions, but because of memory constraints the variety of base models that could be used was limited.

7 Conclusion and recommendations

The H&M Fashion recommendations obtained in this project indicate that ensembling different model results is always a good approach to obtain a better final result. For this problem, obtaining accurate fashion recommendations can add a lot of value to the H&M company, since recommending adequate products to the customers would result in an engaging online shopping experience, lower return costs, increased customer satisfaction, and a better understanding of each customer profile and target groups. Machine learning algorithms have proven to be a powerful tool in recommendation systems, as they provide a means to understand features related to customers and products, and identify what a customer will buy next through sampling techniques, clustering algorithms, similarity metrics, and sequential models, among others.

For the cosine similarity model, improved results could be obtained by using more accurate techniques when selecting a subset of popular articles, for instance, by selecting the most popular ones through an exponential decay weighting. With a smaller subset of the trending articles and narrowing down the customers recently purchased articles, a more accurate prediction could be obtained with the cosine similarity metric.

On the early stages of building the ensemble learning model, a list of both public and custom models were used (list the model references here), in hopes of increasing the variety of models used and achieving a better result. However, due to the public models not being trained with the same sample data from the custom ones, the model's weights became biased towards the models that were trained with the entire data. As such, when using the learned weights in an ensemble with these models, the score of the ensemble was not able to surpass the score of the top model used.

In order to alleviate the bias, it was decided to utilize models where it was possible to train them using the same level-0 training set described in subsection 5.2. The results for these ensemble models gave a much higher score than that of its constituents, as evidenced by plot (show here the score of both our models). This, in turn, shows the benefits that can come from using a stacking ensemble on base learners.

Table 1: Sample table title

Level-1		Level-2	
Models	Ensemble Score	Models	Ensemble Score
SVD_ReRanking	0.0225	model1	0.02
Weekly Quotient Mixture	0.0226	model1	0.02

One final thing that was tried was to use a level-2 ensemble, in which the resulting level-1 ensemble model was then used as a level-0 model with other base and ensemble models. This resulted in much higher local MAP scores, however, when uploaded to Kaggle, the final score did not improve as much. The reasoning behind it is believed to be that by choosing high score ensembles based on the validation data-set, and then selecting a level-2 ensemble of these with a high score on the same validation data could have produced the final ensemble to over-fit the training and validation set. In the future, it might be worth separating the training and validation sets even further to avoid over-fitting.

Some other main challenges that were faced during the project were issues with memory and CPU usage, which affected the amount of data that could be processed by the custom base models, as well as the number of models that could be used in an ensemble together. Additionally, having to modify all the public notebooks used to be trained on the same level-0 training set took considerable time, and reduced the team's efforts in other areas.

8 References

[1] Philipe, H. (2022). *H&M: Faster Trending Products Weekly.* Kaggle. Retrieved from https://www.kaggle.com/code/hervind/h-m-faster-trending-products-weekly

[2] Bhattacharya, M. (2022). *SVD Model ReRanking: Implicit to Explicit Feedback.* Kaggle. Retrieved from https://www.kaggle.com/code/mayukh18/svd-model-reranking-implicit-to-explicit-feedback

[3] Tarick, M. (2022). *H&M Exponential Decay with Alternate Items.* Kaggle. Retrieved from https://www.kaggle.com/tarique7/hnm-exponential-decay-with-alternate-items/notebook

[4] Zheng, H. (2022). *Time is our best friend, version 2.* Kaggle. Retrieved from https://www.kaggle.com/code/hengzheng/time-is-our-best-friend-v2/notebook

[5] Tuananh, N. (2022). *Recbole:LSTM/sequential for Recomendation Tutorial.* Kaggle. Retrieved from https://www.kaggle.com/astrung/lstm-sequential-modelwith-item-features-tutorial

[6] Tuananh, N. (2022). *LSTM model with item infor-fix missing last item.* Kaggle. Retrieved from https://www.kaggle.com/code/astrung/lstm-model-with-item-infor-fix-missing-last-item/notebook

[7] Lunana (2022). *H&M Byfone's speed up.* Kaggle. Retrieved from https://www.kaggle.com/lunapandachan/h-m-trending-products-weekly-add-test/notebook

[8] Abdelnaeem, A.H. (2022). *Trending.* Kaggle. Retrieved from https://www.kaggle.com/code/ebn7amdi/trending/notebook?scriptVersionId=90980162

[9] Hecht, J.P. (2022). *H&M EDA and Rule Base by Customer Age* . Kaggle. Retrieved from https://www.kaggle.com/code/hechtjp/h-m-eda-rule-base-by-customer-age

[10] Tarick, M. (2022). *H&M Ensemble Magic - Multi Blend* . Kaggle. Retrieved from https://www.kaggle.com/code/tarique7/lb-0-0240-h-m-ensemble-magic-multi-blend

[11] Hoizzto (2022). *H&M Ensembling - How to Get Bronze.* Kaggle. Retrieved from https://www.kaggle.com/code/chaudhariharsh/lb-0-0238-h-m-ensembling-how-to-get-bronze

[12] Hiro (2022). *H&M EDA and Customer Clustering by K-Means.* Kaggle. Retrieved from https://www.kaggle.com/code/hirotakanogami/h-m-eda-customer-clustering-by-kmeans