Measuring the role of visual features and marketing bias in product recommendations

Written by
Shubhanshu Gupta*, Kaushik Nishtala†, Kathan Patel‡
DS 5230 Project Report
Northeastern University

Abstract

Recommender systems play a major role in allowing users to discover items "similar" to their personal preference amongst huge corpora of products in a commercial setting. In this study, we use several approaches (Product Similarity, Matrix Factorization) to build a Product recommendation system based on data from Amazon, and experiment with Neural Networks to measure the effect of the product images on user ratings. Our best result was obtained with a hybrid neural network model that uses both textual and image data, with a resulting RMSE score of 0.803. In addition, we also discuss the dynamics of consumer-product interactions and how they can often be influenced by the marketing bias in the data.

1. Introduction

With the rise of Amazon, Netflix, Youtube and many other such web platforms, recommender systems are prevalent everywhere in our daily online journeys - from e-commerce to online advertisements. In the interest of our project, Amazon e-commerce website has a massive database of product and consumer information that is being churned out in billions each second - consumers can search for products, rate them, buy them and share their reviews etc. More specifically, they can rate products they have purchased on a scale from 1 to 5, and discover similar products to their liking.

Our goal is to build an optimal product recommender system using methods that rely on content-based filtering, where in we group products together based on the similarity of their attributes, as well as collaborative filtering which allows us to cluster similar users to predict their product ratings and preferences. Having built such a modeling framework, we then channel our focus towards the impact of product images on the model's efficacy to produce better recommendations using Neural Networks.

The input to our models ranges from product profile features (title, description, category, brand, color etc., but also the product image) to previous ratings from the consumer (user-profile). We use different methods (Item Affinity, Matrix Factorization, Neural Networks) to output predicted ratings such that for any given Amazon consumer, we are able to predict future ratings on products they have not seen yet. We then apply an appropriate evaluation criteria such as RMSE to rate and compare the performance of our models in order to analyze the effect of adding images to our recommender model.

Additionally, the other part of our project deals with the dynamics of consumer-product interactions. However, these consumer preference patterns can often be influenced by an inherent bias in the way the product is marketed, for instance due to the selection of a particular human model in a product image. This bias effects the model by surfacing irrelevant recommendations to users underrepresented in the interaction data. As an example, when a particular gym equipment is promoted using a stereotypically 'male' image, a female consumer who would potentially be interested in it may be less likely to interact with it. To that end, we verify this interesting phenomenon by implementing common collaborative filtering algorithms to identify the bias by studying any notable deviations of the resulting model outputs/errors from the interaction data.

Organization. In Section 1, we discuss results of some closely related body of work. In Section 3, we mention the description of the dataset and the theoretical framework underlying our recommender models. In Section 4, we provide the formal detail of our methodology and algorithms used. In Section 5, we include the analysis of our experiment results and the comparison of different models. In Section 6, we uncover and identify market bias in the consumer product interactions and discuss how it affects the fairness of the recommendation.

2. Related Work

A plethora of recommender models rely on content-based and collaborative filtering to predict ratings and mutual interest. Matrix Factorization, as popularized by Koren et al . - winners of Netflix Prize competition with a RMSE score of 0.8712 for predicting movie ratings, demonstrate that the importance of implicit latent features in the model which makes it possible to improve the performance of the model as compared to clustering or k-nearest neighbors. This work is partly inspired by some of the techniques used by the

*gupta.shubh@northeastern.edu
†nishtala.k@northeastern.edu
‡patel.kat@northeastern.edu

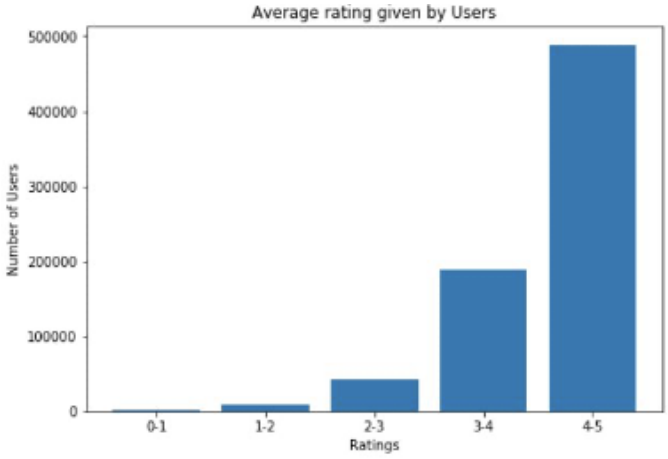

Figure 1: Average rating by users

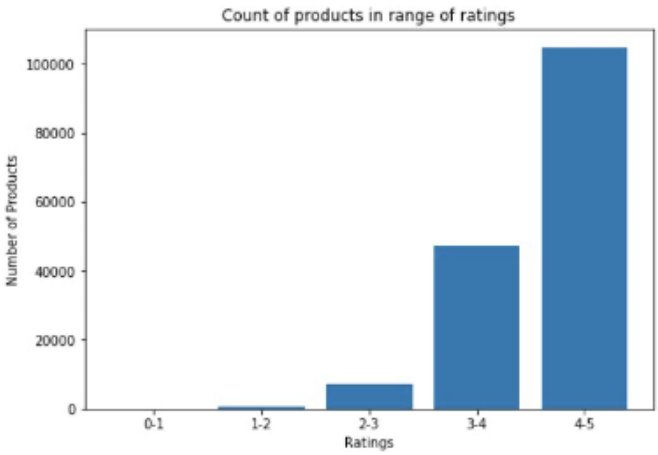

Figure 2: Product count in range of ratings

aforementioned study.

Moreover, Neural Networks excel in obtaining superior results as compared to simple regression models in the setting of large feature set. Inspired by the work of Ahmed et al who improved the performance of his Neural Network by 99% with the addition of images to his text-only model, we are particularly interested in the impact of product images on the model's performance. Additionally, some methods of this project are inspired by the work of Logé et al. on GoodReads book recommendation system.

Finally, the later half of this project is partially motivated by the 'self-congruity' theory in marketing research, which is defined as the congruence between the product image and the consumer's true identity and the perception about oneself. Our project is closely related to previous studies which examines particular types of biases in real-world interactions and their influence on the results of recommender systems , including the popularity effect, the bias regarding the book author gender for book recommenders, and marketing bias in ModCloth recommendations.

3. Background

3.1 Dataset and feature set

Dataset 1: We make use of two separate Amazon datasets in our study. First Dataset[] contains details for over 7.5 million products and around 50 million ratings from users. This dataset is separated into two files: the first file contains the metadata of the products and the second file contains the ratings given by users to products as shown in Figures 1 and 2. Our main interest lies in leveraging features such as user reviews, ratings etc. for the text-based collaborative recommendation system. Furthermore, we utilize features related to product attributes such as price, brand, size, specifications etc., and the images of the product for the content-based recommendation system.

As the data is feature rich ranging from textual data and numerical data to images, we use several pre-processing techniques to clean up the data (removing outliers and duplicates, merging different datasets together etc). For textual data, we use techniques such as stemming, lemmatization, text normalization etc. to extract critical information and to transform them into model-ready format. Additionally, we scale all images to the same size before feeding them as input to our neural network. We also perform other pre-processing techniques such as taking care of missing data, scaling of features, dimensionality reduction, merging datasets etc., as and when required.

amazon_image_final.csv, amazon_image_user_final.csv, amazon_product_final.csv, amazon_user_final.csv: Contains various combinations of product ID (ASIN) x metadata with the following fields: title, description, brand, category, color, image URL etc. We let P be the total number of products (160, 052).

Electronics_5.csv: consumer_id x product ID (ASIN) x rating. (a total of 671, 674 ratings (score from 1 to 5) from 23, 653 consumers). We let C be the total number of consumers (23, 653)

Product images: We use a subset of the products due to computational constraints, we have access to Product images in color and of size 50x74 which we shrink to a 32x32 resolution and flatten them before feeding to the neural network.

Dataset 2: On the other hand, the second dataset includes additional details about user genders and identities that are critical for our study to identify marketing bias. We performed preprocessing, extracted product images and used the face detection API by Face++ to extract the gender with high precision of 96% and low recall of 53%. To improve the relevance, we only kept products where human models are detected and treated them as 'Male' , 'Female' or 'Male and Female' if both of them are detected (not necessarily in the same image).

market_bias_electronics.csv: Product ID x Consumer ID x rating with the following fields: item_id, user_id, rating, model_attr, user_attr, category, brand, year. (containing around 1.3M rating scores across 9,560 electronics products from 1.1M users)

3.2 Feature Engineering

Given the huge amount of data with dataset ranging more than 10 GB for each file, we used PySpark to load the data, clean it and convert it into manageable format. We kept around 50000 thousand products from top 5 electronics category and around 18 million reviews for those products.

We combined product description and features and performed several cleaning steps on textual data like removing contractions and punctuations, special characters, measuring units and converting digits to their string counterpart (For eg., 1 to 'one'). We also saw that a lot of relevant english words are joined together ex. 'themoney', so divided it into two words 'the' and 'money'. Apart from that we performed cleaning on image URLs by removing the thumbnail part in the URL to get complete image of the product and by taking only first image of each product for visual feature visualization.

3.3 Evaluation Metrics

The goal of the recommender system is to:

- Predict ratings of a user for a particular item
- Surface a ranked list of top k products related to a particular product

We used two metrics to achieve these goals:

1. **Top N Similarity matrix :** We chose to work with Top-N similarity metrics, where we evaluate the cosine distance between the original product and similar products and rank the cosine distances in decreasing order to get the top N products.
2. **RMSE :** We use root mean square distance to calculate the average error in ratings predicted by the users.

3.4 Libraries/Tools:

We implemented our entire project in Python.
Main Python libraries used are:

- PythonDataStack(numpy, pandas, scipy, matplotlib etc)
- collections
- Sklearn
- random, requests, BeautifulSoup
- NLTK package
- Surprise Package
- Tensorflow/Keras

Tools:

- Jupyter Notebook
- Google Colab
- Spyder

4. Project Description

In recommender systems, the underlying algorithms aim to suggest relevant items to users. The most common approaches to build a recommender system are content based and collaborative filtering approaches. We now present several methods and algorithms used to predict product ratings of consumers.

4.1 Popularity-based model

Popularity-based model is a straight-forward approach that ranks products by their popularity among users (in our case, the number of ratings the books generated) and suggesting the top k products to all consumers - excluding the ones they have already purchased. More specifically, it ranks the products based on the number of ratings the product generated. Moreover, the popular vote (gross average ratings per product) is equivalent to the predicted ratings for all users.

4.2 Content-based recommendation models

Content-based models make recommendations based on similarity of item features. Popular techniques in content-based filtering include the term-frequency/inverse-document-frequency (TD-IDF) weighting technique from information retrieval and Word2Vec in natural language processing. Content-based filtering has the advantage of being able to solve the cold start problem when there hasn't been enough consumers or when the contents haven't been rated. However, it is limited to features that are explicitly associated with the products and requires extensive amount of data. It is very useful in scenarios where we show similar products based on product-features similar to how Amazon recommends very similar products under the "Products Related to this item" segment on the product page.

4.2.1 Methodology: We take two approaches using content based recommender system:

Approach 1: We take text corpus consisting of features and description of products and try to recommend similar items to the selected item based on feature similarity. We use cosine similarity as a distance measure to compare the results between two approaches.

We used four models using text based features i.e. bag of words, Tf-Idf, average word2vec and topic modelling using LDA as shown in Figure 3.

1. **Bag of words and TF-IDF:** We utilized the Bag of Words to extract numerical features from the raw comment texts. The feature extraction process consists of tokenizing, counting, and normalizing, with the scikit-learn package. First, we tokenized the strings and assigned an integer id to each possible token, with white-spaces as token separators. We combined 1-gram and 2-gram word tokens. Since our corpus is enormous, we selected the top 20000 word features as our final features in both bag of words and TF-IDF approaches. For TF-IDF approach, after getting the word counts by Bag Of Words, we also applied inverse document frequency (IDF) step which is defined as: $idf(t, D) = logN|\{d \in D : t \in d\}|$
2. **Word2Vec:** The Word2Vec algorithm uses a neural network model to learn word associations from a large corpus of text. We used Gensim library to implement Word2Vec and used transfer learning to train pre-trained google model on our corpus. Word2Vec is used to get the semantics of the sentence by looking at the neighborhood of words for context. We took a window of 5 words to look at the context.

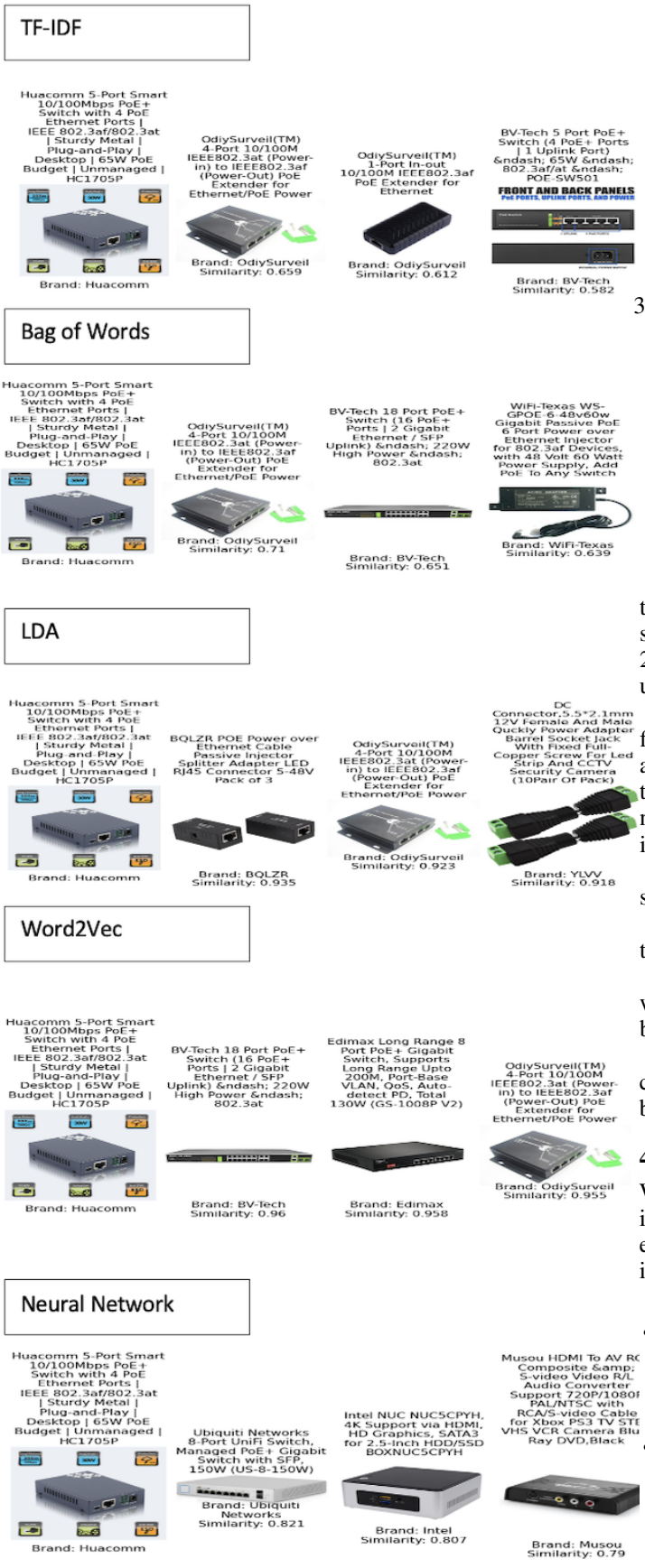

Figure 3: Recommendation results for various models

$$p_{u,i} = \frac{\sum_{j \in N} s(i,j) r_{u,j}}{\sum_{j \in N} |s(i,j)|}$$

Figure 4: Weighted average of k most similar items to i

3. **Topic Modelling (Latent Dirichlet Allocation):** It is an unsupervised generative static model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. In the case of recommendations, we hypothesize that similar products should have similar features and descriptions and so topic modeling using Latent Dirichlet Allocation can help us find similar items with great efficiency because in general, the query product and products similar to it will have similar topics.

Approach 2: For our second approach, we leveraged attributes from items that user has interacted to recommend similar items. We divided our data into 80% training and 20% testing sets by using stratified sampling so that each user has some records in both training and testing set.

We created user profile by taking users that have at least five ratings to overcome the cold start problem. As, there are a lot of products in our system, we only took products from top 5 categories and then filtered out the products which do not have relevant images as we wanted to perform visual inspection as well as text based recommendations.

For evaluation of a recommender system, we used RMSE score to compare different recommenders.

We used cosine similarities to compute similarities between item profiles.

To estimate the rating of item i by user u, we took weighted average of k most similar items to i which are rated by user u as shown in Figure 4.

We then performed this step for each user in test set and calculated a global RMSE value by calculating difference between original and predicted ratings.

4.3 Neural Networks

We made use of Neural Networks to predict the average rating of a product: title, description, color, brand, category, etc. We also investigate the influence of adding the product image into the feature set.

We used three different network architectures:

- A Multi Layer Perceptron for the numerical/textual data only: We used a configuration of 2 hidden layers with 8 and 4 hidden units respectively + 1 output neuron with linear activation.
- Convolutional Neural network using Transfer Learning: We explored the effect of visual features on recommendations by using Transfer Learning on images of products to see how visual features affect recommendations. We used pre-trained architecture of VGG16 which was

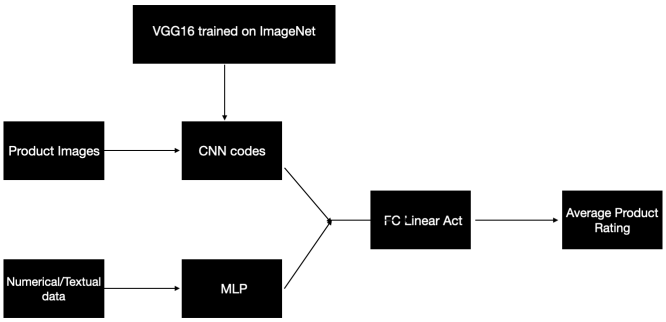

Figure 5: Neural Network architecture

trained on ImageNet dataset to produce the d-dimensional CNN codes (bottleneck features) for all product images in order to find similar images.

- A Mixed model combining both textual/numerical as well as image data as shown in Figure 5.

We then obtained the results of the predictions from each model (RMSE) as a function of size of data where we observed that as expected, the CNN model based on product images only performs poorly as compared to the other two networks.

4.3 Collaborative-filtering based methods

Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between consumers and products in order to produce new recommendations.

Collaborative methods include:

- **Memory based approaches:** It directly works with values of recorded interactions, and are essentially based on nearest neighbor's search.
- **User-user based CF**
- **Item-item based CF**

Using the concept of Matrix Factorization, we are able to assume the existence of d latent features such that our C x P rating matrix R can be represented as the product of two lower-dimension matrices: N of size C x d and M of size d x P. More specifically, the predicted rating from Consumer c on Product p can be represented as: $r_{cp} = n_c^T m_p$

The main objective behind Matrix Factorization comes down to estimating the optimal number of latent features.

We have compared 11 different algorithms for predicting the rating of products by users. From the RMSE score and time complexity we found that Singular value decomposition algorithm is the most feasible approach to implement on whole dataset. We implemented both User-user and Item-item approach using python surprise package and GridSearchCV function.

Some of the models are as follows:

- **SVD:** SVD is a matrix factorization technique that is used to reduce the number of features of a data set by reducing space dimensions from N to K where $K < N$. For our

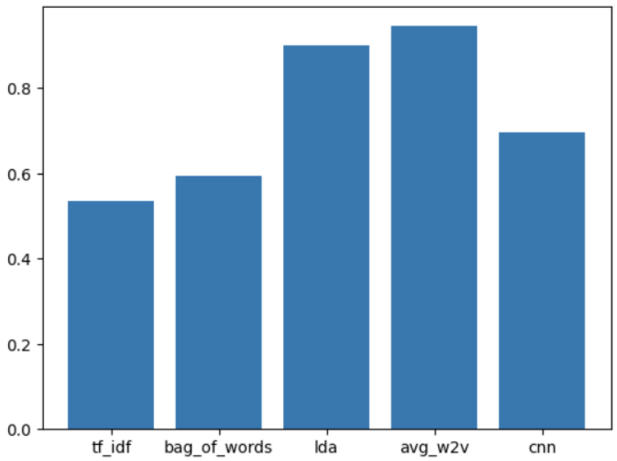

Figure 6: Bar plot indicating RMSEs of various models

recommendation system we are only interested in the matrix factorization part keeping same dimensionality. We did matrix factorization on the user-item ratings matrix.

- **KNN:** KNN is a well known machine learning algorithm where we classify an item based on it's nearest neighbours. kNN for our recommender system is used to find clusters of similar users based on common product ratings, and make predictions using the average rating of top-k nearest neighbors.
- **NMF:** We treated our users as columns and products as rows with values as ratings to create a matrix V. The goal with Non-negative matrix factorization is to approximate this matrix by the dot product of two arrays W and H. Dimensions of the arrays are defined by dimensions of V and number of components we set to the algorithm. If V has n rows and m columns and we want to decompose it to k components, then W has n rows, and k columns and H has k rows and m columns.

We then experimented with various parameters via Cross Validation with the goal of finding the optimal model with respect to RMSE:

- Varying similarity metrics : Cosine and MSD
- Varying number of latent features: we obtained d = 3 as the optimal number of latent features. We observed that the model does not generalize well beyond d = 3 as it was overfitting the train set.
- min_support: [3, 4, 5]
- Number of epochs: 5 - 100
- Learning rate : 0.002 - 0.005
- Regularization: 0.4 - 0.6

5. Empirical results

The results from the Figure 6 and Figure 7 confirms our hypothesis that visual features have an impact on recommend-

Method	RMSE score
TF-IDF	3.221
Bag Of Words	2.601
Average Word2vec	1.121
Visual Features (Using Transfer Learning)	1.507
TF-IDF + Visual Features (Using Transfer Learning) (class weights are 0.6 and 0.4)	1.101
Word2vec + Visual Features (Using Transfer Learning) (class weights are 0.6 and 0.4)	0.803

Figure 7: RMSE comparison of various content-based models

Algorithm	test_rmse	fit_time	test_time
SVDpp	1.081491	3.266554	0.102890
BaselineOnly	1.081865	0.145634	0.168137
SVD	1.082293	1.542222	0.115838
KNNBaseline	1.083377	9.487720	0.176845
KNNBasic	1.100555	9.471471	0.150602
KNNWithZScore	1.107142	13.452598	0.268510
SlopeOne	1.108329	0.295514	0.078623
KNNWithMeans	1.108397	9.793786	0.086872
CoClustering	1.109817	5.667947	0.104899
NMF	1.118467	4.216863	0.061743
NormalPredictor	1.389798	0.061005	0.174467

Figure 8: RMSE results of various collaborative models

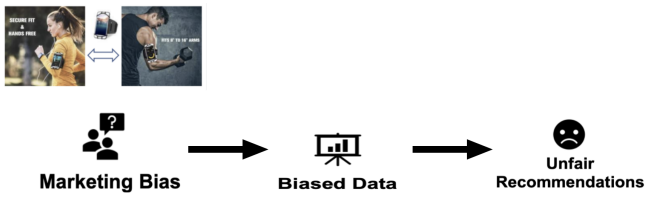

Figure 9: Marketing bias flowchart

ing similar items as RMSE decreases by adding visual features to textual based models.

But we can also see that just by using visual features, we are performing worse than our best textual based models. Thus, just using visual features is not an alternative to text based recommenders.

Additionally, we have calculated RMSE score, fit time and test time of all 11 different algorithm. As we can see from Figure 8, Singular Value Decomposition is giving the most optimal result. From prediction of rating by user we can put a threshold that if the predicted rating is above that threshold then that particular item is recommended to that particular user, otherwise not. We can improve the prediction by including the text review of the users given to the product.

6. Fairness of our recommendations

Another major direction we explored is analyzing the market bias in the product recommendations. As a recap, the consumer-product interactions can be biased by how the product is marketed, for example due to the selection of a particular human model in a product image. This causes unfair (or irrelevant) recommendations to users or items underrepresented in the input data.

We use the following methodology to identify and analyze how the marketing bias affects product recommendations.

Initially, we measure consumer's product preference in two ways:

- the user's preference in terms of willingness to purchase a product; and
- the user's satisfaction feedback (e.g. ratings) on the product.

6.1 Product Selection vs Marketing bias

We analyze the association between product image and user identity in observed data with respect to interaction frequency using Pearson's ChiSquared Test Statistic which is given in Figure 10. where m and n denotes a user group and a product image group respectively, $f_{m,n}$ is the observed number of interactions in the market segment (m, n) and $Ef_{m,n}$ denotes its expectation. The results are as follows: We assume that our null hypothesis is that the product image and user identity are statistically independent. From the results given in Figure 11, we observe a pronounced 'self-congruity' phenomenon. That is, we see more interactions on the consumer-product segments where users' identities match the product images , whereas several segments are

$$\chi^2 = \sum_{m,n} \frac{(f_{m,n} - \mathbb{E}f_{m,n})^2}{\mathbb{E}f_{m,n}},$$

Figure 10: ChiSquared test statistic

user_attr	Female	Male	All
model_attr			
Female	34259	31587	65846
Female&Male	26478	24930	51408
Male	25963	30907	56870
All	86700	87424	174124

chi2 581.849 p-value 0.0

real market size - expected market size

user_attr	Female	Male
model_attr		
Female	1472.89	-1472.89
Female&Male	880.88	-880.88
Male	-2353.77	2353.77

Figure 11: Contingency table

underrepresented in the data. For instance, 'Female' user, 'Male' product have smaller market sizes compared to other market segments.

6.2 Consumer Satisfaction vs. Marketing Bias

Next we analyze consumer ratings as a function of product image and user identity using two-way analysis of variance (ANOVA).

consumer satisfaction $\sim$ product + user + product x user

where we assume the null hypotheses to be such that the average consumer ratings is equal across different product image groups, different consumer identity groups and there is no interaction between different groups.

From the results given in Figure 12, we see that the interaction between product model 'gender' and user gender is insignificant with respect to users' rating scores.

Therefore, we have confirmed from our analysis that the consumer-product interaction data is correlated to market-

	F-stat	p-value
product	62.581	0.000
user	3.508	0.061
product:user	0.907	0.404

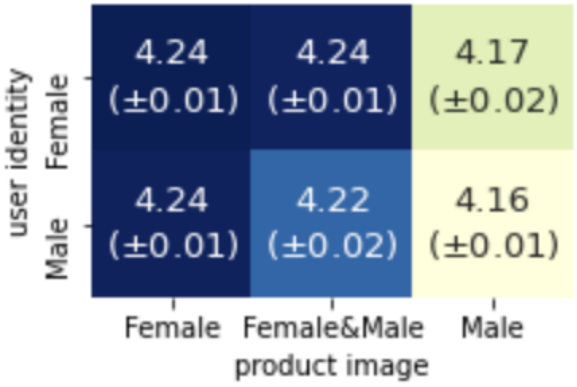

Figure 12: Heatmap of sample means of ratings within market segments

ing strategies used by product sellers on Amazon. So, we can now implement the recommendation models on this data to see how the marketing bias affects the consumer-product interactions and the recommendation results and what kind of methods to employ in order to address this bias. Extending to this use-case requires considerable work in this direction which is not feasible to implement given the time constraints, so this will be considered as an interesting challenge to work on in the future.

7. Conclusions/Future Directions

Our initial aim for the project was to gauge the effect of visual features in recommendations and compare them to other older ways of recommendation like using TF-IDF or Word2vec. We were able to prove our hypothesis that visual features do have an impact on the recommendations but we could also see from rather high RMSE values for our best models that the impact is not so much as can be seen in some other related fields like book recommendations or movie recommendation.

For electronics data, since people tend not to buy very similar electronics products again and again, so it is very difficult to provide product recommendations just on the basis of previous purchases of user. So, in future we would like to improve on this approach by creating a hybrid recommendation system combining both content based and collaborative features for providing recommendations.

7.1 Future directions

- We'd like to make use of other evaluation metrics such as nDCG (Normalized Discounted Cumulative Gain).
- Add a Diversity score metric (defined as the average proportion of products in the Top k recommendations coming from the long tail of the dataset (less popular/less known products).
- Experiment with more optimized Matrix Factorization ALS model and building a hybrid model from different methods to merge each model's effectiveness into making a more efficient and robust recommender system.
- Analyze and address how the marketing bias affects the recommendation results.

7.2 Contributions

Shubhanshu, Kaushik and Kathan all contributed equally to this project. All results were achieved jointly by three project partners.

Link to Project Code: github.com/Kau5h1K/ds5230-amazonrec

8. References

Academic Paper

Mengting Wan and Jianmo Ni and Rishabh Misra and Julian McAuley. Addressing Marketing Bias in Product Recommendations 2019. *Corr* 1912.01799.

Websites

https://en.wikipedia.org/wiki/Recommender_system
https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
https://realpython.com/build-recommendation-engine-collaborative-filtering/
https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0
https://medium.com/towards-artificial-intelligence/recommendation-system-in-depth-tutorial-with-python-for-netflix-using-collaborative-filtering-533ff8a0e444