Amazon Data Analysis

Advisor

Professor Julian McAuley

Team Members

Nirmal Budhathoki, Nolan Thomas, Chris Chen, Toby Moreno

0. Abstract

There are two major approaches in building recommender systems: content-based filtering methods and collaborative filtering methods. Content-based systems recommend items with similar properties of items users have liked in the past. Collaborative filtering systems recommend items based on user-items similarities. In this paper, we discuss building a recommender system using Bayesian Personalized Ranking (BPR) methods on amazon reviews data. This paper focuses on reviews of Women's Clothing and Electronics' Computers and Accessories and their corresponding product metadata. We evaluate model performance using Area Under Curve (AUC) metric with both sparse and dense user-item interactions to simulate the effects of cold-start problems. In our experimental results, we see how certain features improve the AUC in cold-start situations for both Women's Clothing and Electronics. We also see that feature combinations can provide an additional boost to the AUC.

1. Introduction

To increase sales, retailers know that they need to provide personalized shopping experiences for each of their customers. This is not feasible for the traditional brick and mortar stores due to the constraints of the store sizes and shelve spaces. In contrast e-commerce can practically make anything and everything available to its customers. The challenge is how to use data science to provide a personalized shopping experience and drive higher sales.

1.1 Recommender system

E-commerce recommender systems can provide product recommendations to their customers and suggest what they might like to buy based on their past history of purchases, reviews, and/or product searches [1]. There are two basic architectures for a recommender system:

A. Content-based filtering systems:

The main idea of these algorithms is to recommend items that are similar to those that a customer rated highly in the past. User profiles and item profiles are created to capture unique characteristics used by the recommender system. We then use user and item profiles to predict the heuristics by computing the similarity scores between the user's and item's vectors.

B. Collaborative filtering systems:

This approach relies on past user behavior - for example, previous products reviews or ratings to infer new user-items associations [2]. The key advantage of this approach is that it does not require user profiles or item profiles, which can be challenging to build. The drawback of this approach is that it is suffers from what is known as the *cold start* problem. The model performances heavily depend on the "density" of user-item interaction. This means that when a new user or a new item comes into the system the model's predictions can deteriorate substantially.

There are two major methods in collaborative filtering: *neighborhood methods* and *latent factor models*:

a. **Neighborhood methods**: These techniques perform recommendation in terms of user/user and item/item similarity. User similarity can be measured in terms of the items they purchased. Item similarity can be measure in terms of the users who make the purchase. Similarity

measures such as cosine similarity, Jaccard similarity, and Pearson correlation can be used to gauge the similarity between users and items.

b. **Latent factor models**: These methods try to explain user behavior by inferring the "factors" based upon user and item interactions. One way to implement this is by using *matrix factorization* which maps users and items into a K-dimension space. The resulting matrix can be estimated by the dot product of user vectors and item vectors shown below.

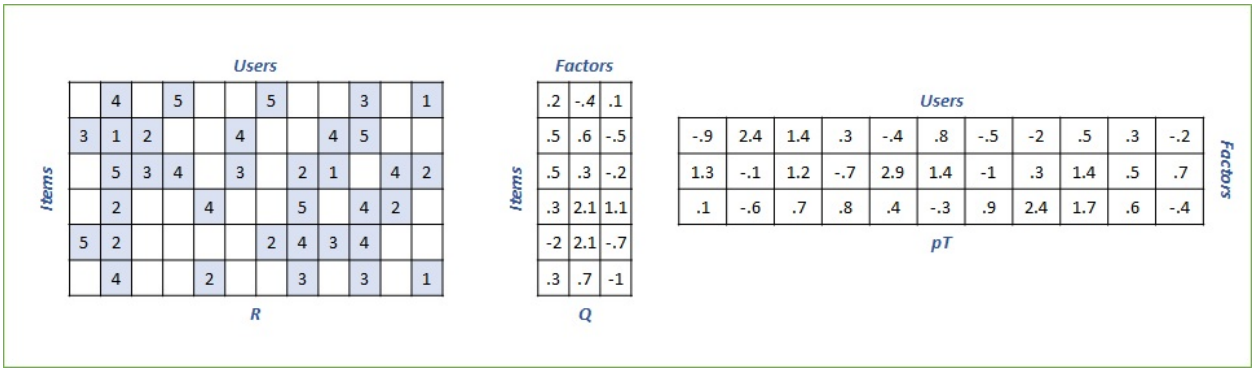


Figure 1: Latent Factor Models

1.2 Related Work

Professor Julian Mcauley has gathered 142.8 million product reviews and meta data from Amazon spanning May 1996 to July 2014 [3, 4] to answer many of his research questions in recommender systems. Team Sharknado from Cohort 2 looked into combining visual features [6] with non-visual features and evaluating the performance of this hybrid model.

1.3 Question Formulation

To build a recommender system that is capable of capturing customers' preferences by:

- ❏ Exploring relevant product features in addition to the latent factors
- ❏ Analyzing how price sensitivity may affect the user's purchasing behavior
- ❏ Investigating the effect of user-item sparsity related to model performance

We built two separate recommender systems in two product categories: Women's Clothing and Electronics (Computers and Accessories). We will use these two prototype recommender systems in answering the questions listed above.

3. Team Roles and Responsibilities

Nirmal Budhathoki: Nirmal is currently working as a Cyber Security Analyst at SPAWAR Systems Center Pacific (SSC PAC), holding MS in Information Systems from Marist College, and MBA from Brandman University. In this project he worked on data pre-processing, some feature engineering, modeling, generating final scoring for recommendation list, report writing, outlining presentations, and creating poster.

Chris Chen: Chris is currently a principal investigator at SPAWAR Systems Center Pacific (SSC PAC). He holds an MS in Electrical Engineering and two BS in Electrical Engineering and Computer Engineering from UC Irvine. He served as the project coordinator and Amazon Web Services (AWS) treasurer for the team. Chris was involved in feature engineering, architecting end-to-end ML pipeline in the AWS and ensure reproducibility of the project.

Christopher "Toby" Moreno: Toby currently works as a senior software engineer at SPAWAR Systems Center Pacific (SSC PAC) with a bachelor degree in Computer Science. He worked on the web app's front-end using AngularJS. Dabbled with the middle-tier using Flask framework including the REST API services. And wrote a thin data abstraction layer for the back-end PostgreSQL database.

Nolan Thomas: Nolan is an ETL Database Engineer at Slacker Radio with a BS in Applied Mathematics from Florida Institute of Technology. He worked on the data pipeline ~ JSON ingestion into PostgreSQL, transformations, extractions for modeling and web application database. Involved in feature engineering, scripting jobs and capturing model outcomes.

4. Data Acquisition

4.1 Data Sources and Data Collection

We used Julian McAuley 's Amazon dataset which contains 142.8 million reviews spanning May 1996 - July 2014 and corresponding product metadata. We made use of

the enhanced Women's Clothing and Jewelry 5 Core dataset that Cohort 2's Sharknado team produced. Via screen scraping Sharknado obtained a higher fill rate for key attributes such as brand and price in the Women's Clothing and Jewelry category There are total of 9.4 M records on product details (metadata). Within Clothing, Shoes and Jewelry category, there are 279K reviews. There are 1.7M reviews in electronics.

4.2 Data Pipeline

Understanding an end-to-end data pipeline workflow is very crucial when it comes to data science and problem solving. Figure 2 shows the journey of data from collection to results as interpreted by the machine learning model. The end goal of our project is to generate a recommended list of products for users. The final output from the model is a list of recommended products sorted by priority of user's preference. The model's output is the final stage of the data. Let's step thru the earlier phases that the data goes through to produce this recommendation list.

The first stage of data pipeline is to collect and assemble data. We have used GDrive and S3 buckets for storing the data. The next stage is pre-processing which is also called data cleaning. This phase of the pipeline takes most time and effort because the results and output of your machine learning model are only as good as what you put into it. Basically, the principle of garbage in garbage out.

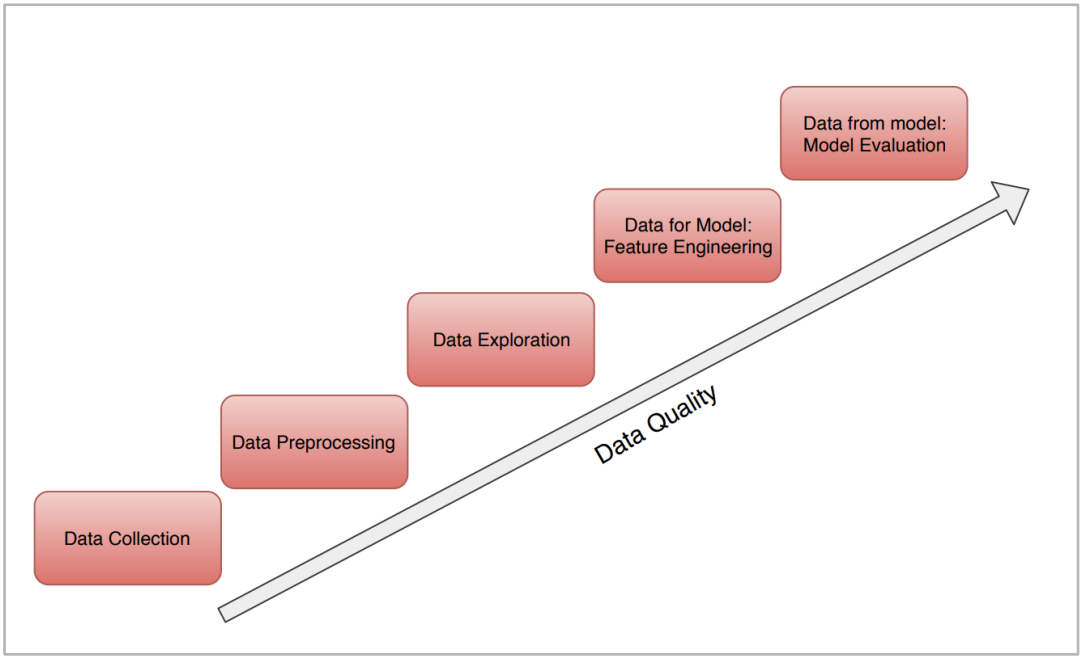


Figure 2. Journey of Data from collection to Modeling

The next stage is exploration and visualization. During the exploration phase we try to understand what patterns and values our data has. We have used different types of visualizations and hypothesis tests to back up our findings and derive hidden meanings behind our data through various plots.

Data-for-model phase of data is all about feature engineering and passing on the features to the model and test which of them produces higher accuracy. In our case, Area Under the Curve (AUC) is used. More explanation on AUC is discussed in Section 8.

The final stage of data is model evaluation and interpretation. Once the desired AUC (or best AUC) is achieved we then present and communicate the results back to the users. We have achieved this by developing a webtool with a dashboard to present the results to the user. More details about this is explained in Section 7.

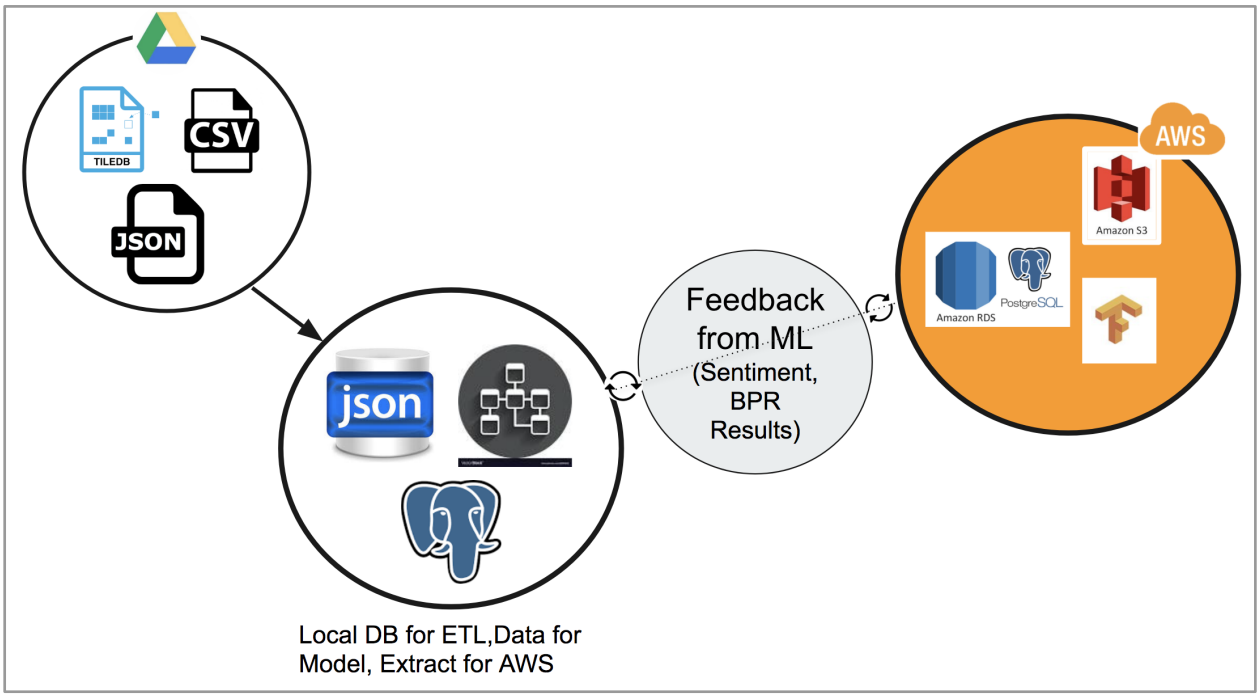


Figure 3: Database Environment (Local and Cloud)

One of the relevant works was from Cohort's 2 Team Sharknado. We inherited their scrapped version of data from their Gdrive. The data collected from online google storage were in json/ csv formats. Some of the features like original price bucketing, one-hot encoded brand names, bi-grams (two consecutive words from product description- meta data) were already generated and stored.

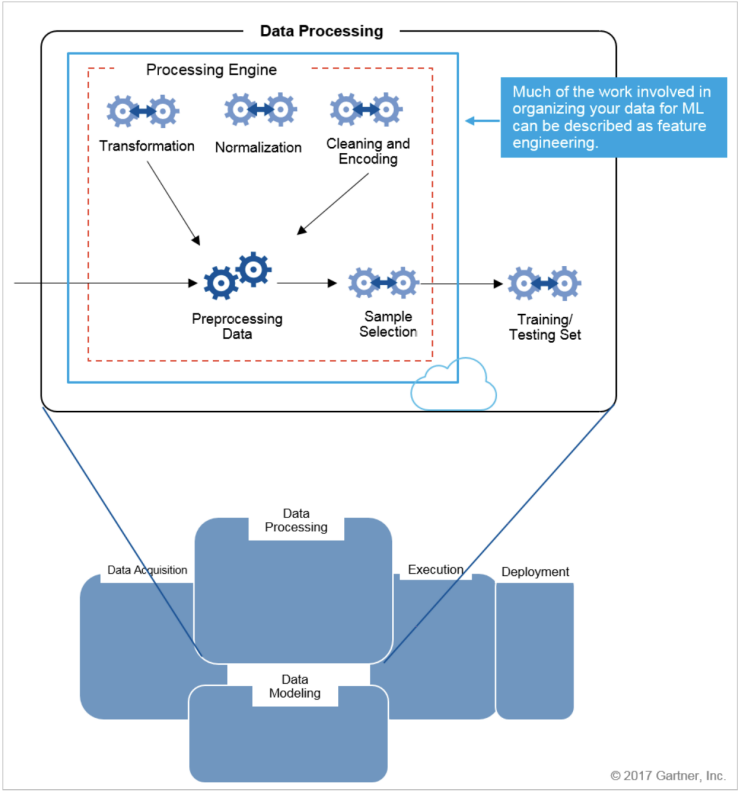
The local database environment is set up in PostgreSQL database for ETL. Additional feature engineering was performed in PostgreSQL. The PostgreSQL database environment is also set up in the cloud using a relational database model on AWS. The local and cloud database has a feedback loop to accommodate the results and feedbacks from model output.

5. Data Preparation

We made use of the enhanced Women's Clothing and Jewelry 5 Core dataset that Cohort 2's Sharknado team [5] produced. Via screen scraping they obtained a higher fill rate for key attributes such as brand and price

- We found prices at the high- and low-end range to be suspect and incorrect so these were filtered out
- Only records with pricing data were retained

We used Julian McAuley 's Amazon dataset which contains 142.8 million reviews spanning May 1996 - July 2014 and corresponding product metadata. The raw JSON files on product meta data and the 5 Core data on- I) Clothing Shoes and Jewelry and II) Electronics where loaded into PostgreSQL database. Cohort 2's scrapped price and brand for Women's Clothing Shoes and Jewelry were merged. PostgreSQL database was used to generate feature engineered extract files for modeling and EDA.



Source: Gartner (January 2017)

Figure 4: Data Processing Architecture (Reference: Gartner Report 2017)

The above figure shows how data processing is aligned with data modeling. The raw data had two different sources: User Reviews and Product Metadata. Both tables are structured data, and user_id or reviewer_id is used as the 'key' to join the tables. PostgreSQL is used to transform and integrate the data sources.

A python function is written to load the integrated data and generate the dictionaries: user_ratings and item_ratings. User ratings is the dictionary where key is the user_id and values are the list of items being rated. All items being rated are assumed to be purchased. On the other hand, item ratings is the dictionary where the key is the item_id and values are the list of users who rated the item. These dictionaries are very important when it comes to generating the sample dataset for our recommendation engine. The sample data set for the Bayesian Personalized Ranking is triplets of users, item_purchased (i), item_not_purchased (j), or in other words a tuple of (User, Item 'i', Item 'j'). These tuples are uniformly sampled into training data, and test data in batches to train and evaluate the model.

Some of the pre-processing methods used are: Data Cleaning, Data Integration, Data Transformation, and Data Reduction.

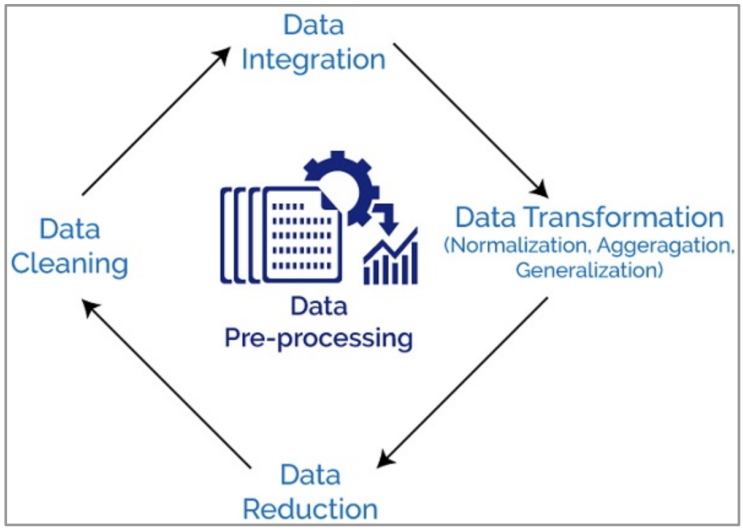


Figure 5: Data Preprocessing [8]

Few of the challenges we faced during data cleansing are missing values, some brand names as comma separated values (csv read error) which was corrected by using tab delimited format, erroneously imputed values (example: range value of price 0-3000 imputed as mean value 1500).

Using a relational database tool like PostgreSQL database for integrating the structured data from various data sources proved to be very significant. Some of the feature engineering is also performed at the query level.

One of the important attributes of our analysis was the price feature. The price data distribution was right-skewed or positively skewed meaning most of the products fall under small price ranges as shown by figure below. The data transformation step helped us to apply log transformation, which helped us to achieve the somewhat normal distribution as shown by the second figure. While the transformation did not significantly improve actual test samples accuracy it did improve the cold start accuracy.

Figure 6: Price Distribution of Women's Clothing data

Figure 7: Log Transformed Price Distribution of Women's Clothing data

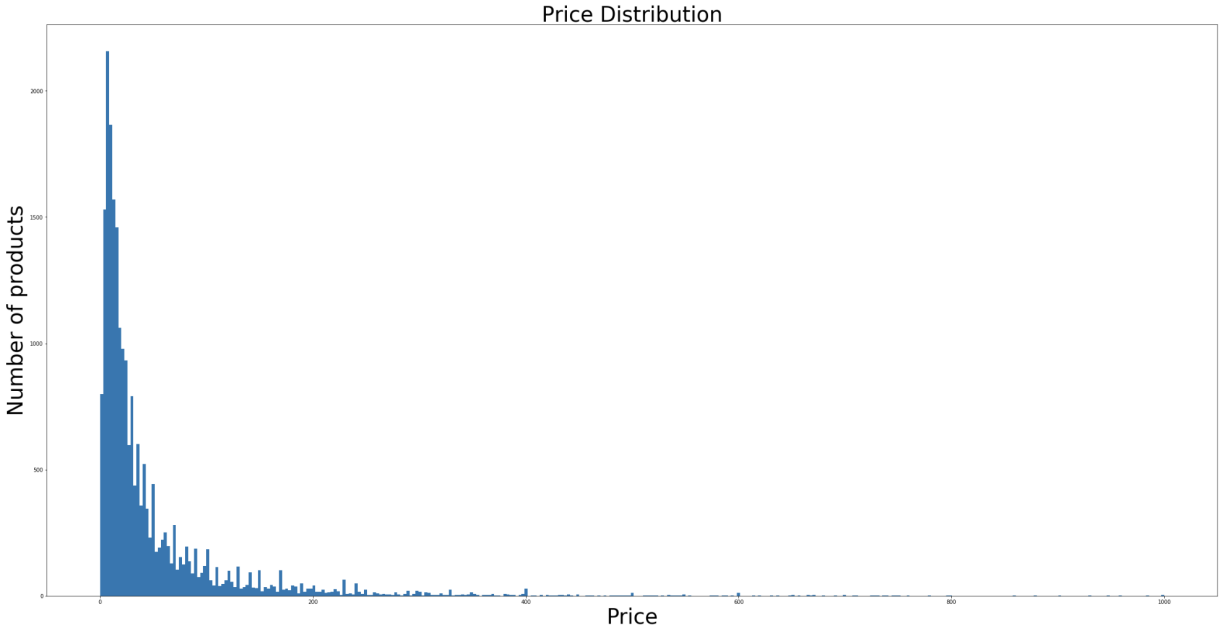


Figure 8: Price Distribution of Electronics Data

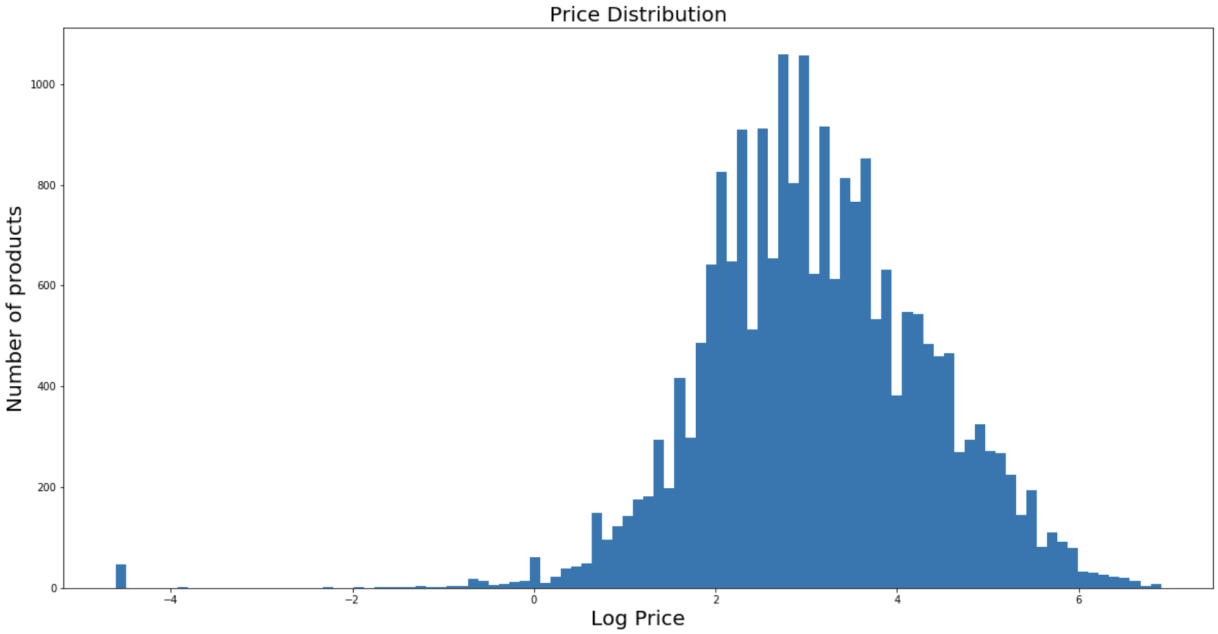


Figure 9: Log Price Transformation of Electronics Data

Data filtering was another significant pre-processing step we followed. Most of the outlier values, some missing values were removed after doing some statistical analysis (row-wise reduction). All the column features are still preserved.

During the EDA we identified some features to augment the BPR model with. BPR requires binary attributes. One-hot encoding was performed on categorical features and quantitative attributes were binned so that they could be one-hot encoded.

The baseline BPR model is the matrix factorization of users and items in some latent space without passing along any external item features. The model finds the latent factors (hidden) that groups the users/items together on some common features. These features are inherently learned by model. However one of our objectives was to find relevant features to aid baseline model and improve accuracy. One of the important feature we have identified with significant improvement in accuracy is category tags.

Category tag as feature: The hierarchical classification of amazon products are flattened, dropped duplicates, and used on more than one reviewed items in that category. The first two high level categories were ignored as it applies to all. After that 697 unique category tags in Women's Clothing and Jewelry were identified. We then one-hot encoded the items based on those categories.

Figure 10: Example to explain category feature

In this example, TOMS women classic canvas shoes belong to fashion sneakers under shoes. However this item also belongs to other categories as shown below:

#2 in Clothing, Shoes & Jewelry > Women > Shoes > **Flats**
#3 in Clothing, Shoes & Jewelry > Women > Shoes > **Loafers & Slip-Ons**
#15 in Clothing, Shoes & Jewelry > Women > Shoes > **Fashion Sneakers**

This shows that category tags for this item has more than one group for positive one-hot encoding. This makes sense because Amazon has categorized products based on customer's search interest and this is one reason that this feature performs best in improving our AUC BPR.

Some of the other features we have tried and implemented are listed below:

Subcategory (Level 4)	70 sub-categories in clothing and jewelry	Beneficial
Sentiment/Polarity	Using TextBlob sentiment polarity	No value add

Seasonal	Meteorological seasons	No value add
Brand	1769 brands	Beneficial
Overall Rating	Simple encoding	No value add
Price	Log transformation	Beneficial
Bi-gram words	Most frequently used 4525 bi-grams from product title	Beneficial

6. Analysis Methods

To build a recommender engine for a E-commerce giant like Amazon, there are many practical factors to consider:

- *Generalization*: With the diverse categories of items that are available for sale the recommender engine needs to be able to recommend items across different categories.
- *Cold start/sparsity problems*: How does the models handle new users and new items that do not exist in the current system? Moreover, what can you do to improve the model performance when the user and/or item have limited feedback information?
- *Scalability*: When new data is available, how do you retrain the model? What is the computation cost in training the models?

The preliminary analysis of the review data and product meta data indicate that we don't have enough user relevant data to generate user profiles More importantly, the fact that we are not able to collect more Amazon user related data lead us to the conclusion that the only feasible way forward is to build the recommender engine with collaborative filtering systems using latent factor models. The recommender engine architecture decision will shape our effort in focusing on incorporating relevant product features to alleviate the model performance degradation of cold start/sparsity issues.

One of the classical approaches in building recommender engine is described in a paper called Bayesian Personalized Ranking (BPR) from Implicit Feedback [6] by Steffen Rendle. BPR is a state-of-art implicit feedback pairwise ranking model. The goal is to estimate a personalized ranking function ($>_u$) for each user of all items, where user prefer items i over item j (i $>_u$ j). To formalize it, let *Ds: U x I x I* be all training samples:

$$D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\}$$

A generic optimization criterion called BPR-OPT is defined as:

$$\text{BPR-OPT} := \sum_{(u,i,j) \epsilon Ds} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2$$

$$\hat{x}_{uij}(\Theta) := \hat{x}_{ui} - \hat{x}_{uj}$$

λ_Θ is a model specific regularization parameter and $\sigma$is a logistic function. $\hat{x}_{uij}(\Theta)$ is an arbitrary real-valued function of the model parameter vector Θ which captures the relationship between (user u, item i and item j). Common collaborative filtering techniques such as adaptive K-nearest-neighbor or matrix factorization are used for training the models. The models can be optimized via stochastic gradient descent with bootstrap sampling over (user, item i, item j) shown below:

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \Theta \right)$$

α is the learning rate, Θ is the parameter vector of an arbitrary model class.

To train our models we implemented the BPR algorithm in TensorFlow, an open-source machine learning framework from Google. The key advantage in implementing the BPR algorithm in TensorFlow is that it abstracts the optimization methods so we do not need to derive the closed form solutions of gradient descent by hand. In the past ML researchers often struggle to derive the correct closed form solutions and implement it in code.

In building our recommender engine we designed a robust end-to-end machine learning workflow. After downloading the data it was ingested into a local PostgreSQL database. Exploratory data analysis is done in jupyter notebooks to understand the requisite data processing. Extract, Transform, and Load (ETL) steps are performed in the PostgreSQL database. Since training recommender engines takes a lot of computing resources we used AWS Elastic Computing Cloud (EC2) P3 instances to train our models.

The figure below shows the detail steps of our analytical processes and the associated tools used for each step. The modeling phase is captured as an entity of four different sub-processes: building model, optimizing model, evaluating model and finalizing model. Tensorflow in python is the analytic software package used for modeling. MF (Matrix Factorization), BPR (Bayesian Personalized Ranking), SGD (Stochastic Gradient Descent) and AUC (Area Under Curve) are four major machine learning concepts involved in our modeling.

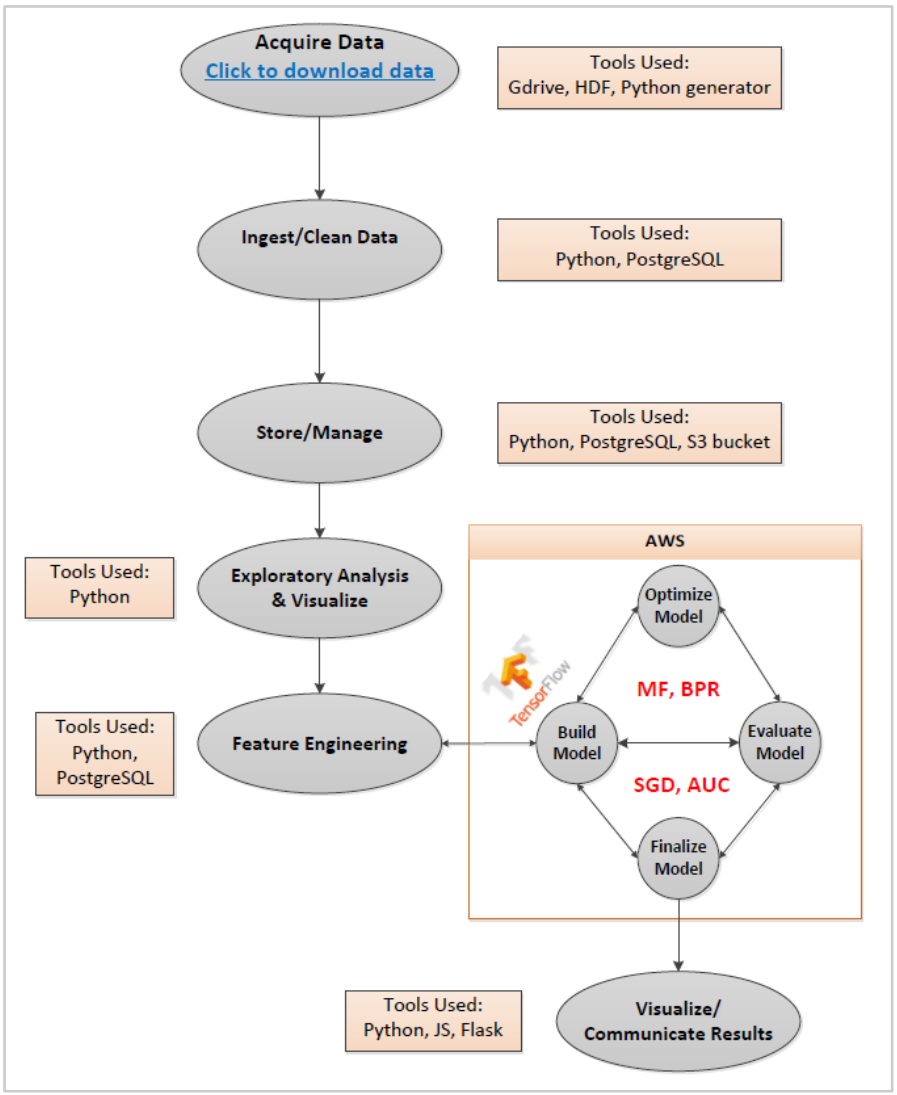


Figure 11: Analytical Process workflow from Data Acquire to Communicate Results

7. Findings and Reporting

7.1 Exploratory Data Analysis (EDA) and Feature Engineering

Ratings and sentiment analysis for Women's Clothing category show data are heavily skewed to high ratings and positive polarity. Various feature transformations techniques were used; however, these didn't show significant improvement in AUC to consider exploring further. In addition, we established a null hypothesis: Women's tend to give higher average ratings for clothing category, and after some tests, we could not reject the null hypothesis.

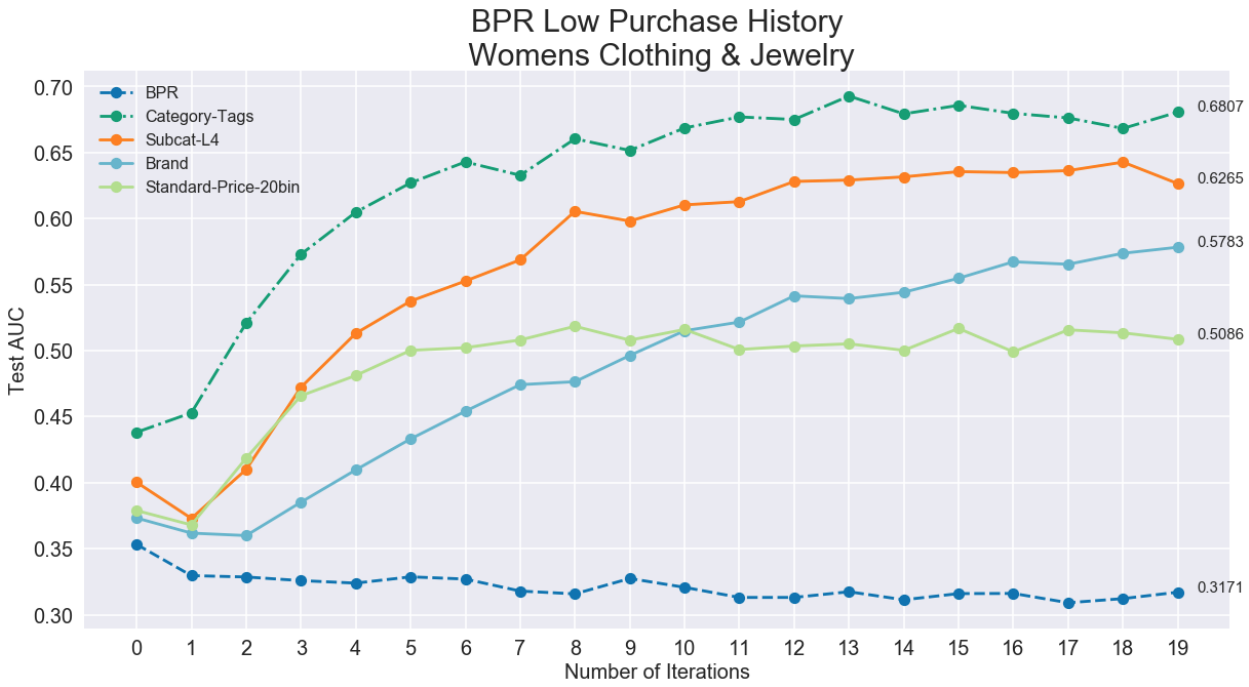
Price attribute is skewed as most of the products fall in lower price ranges. We applied log transformation to the features to expand the prices in the lower price level and shrink it for the higher price.

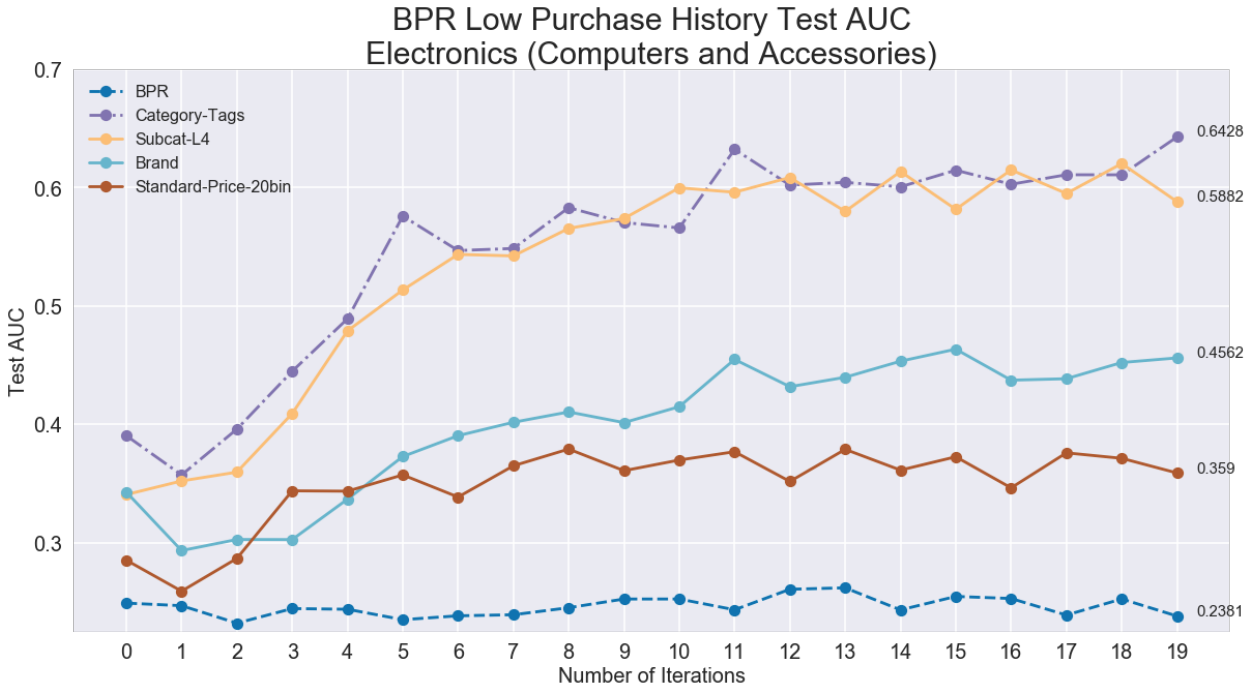
7.2 What we learned from Analysis

We performed analysis in both Women Clothing and Computers and Accessories categories. By including two categories in analysis we can better generalize our findings and validate our hypotheses. For BPR models, metrics evaluation are based on Precision Recall Area under Curve (AUC). The maximum value is 1 for this metric.

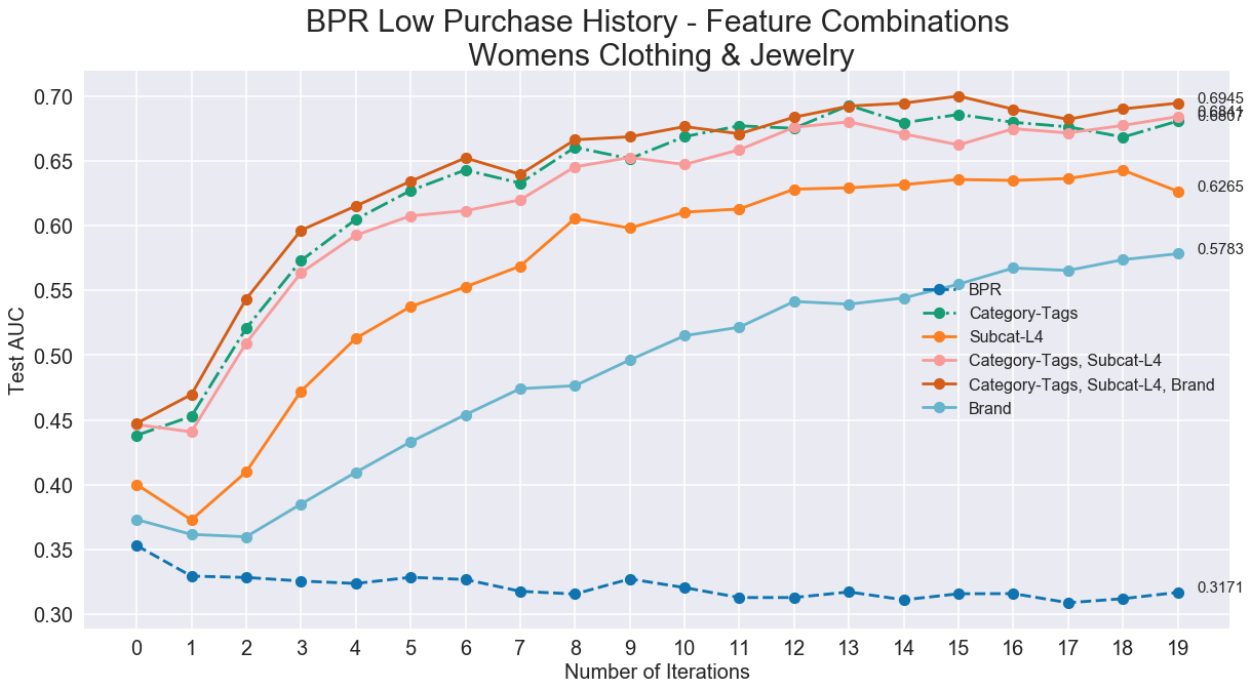
7.2.1 Low user-item interactions

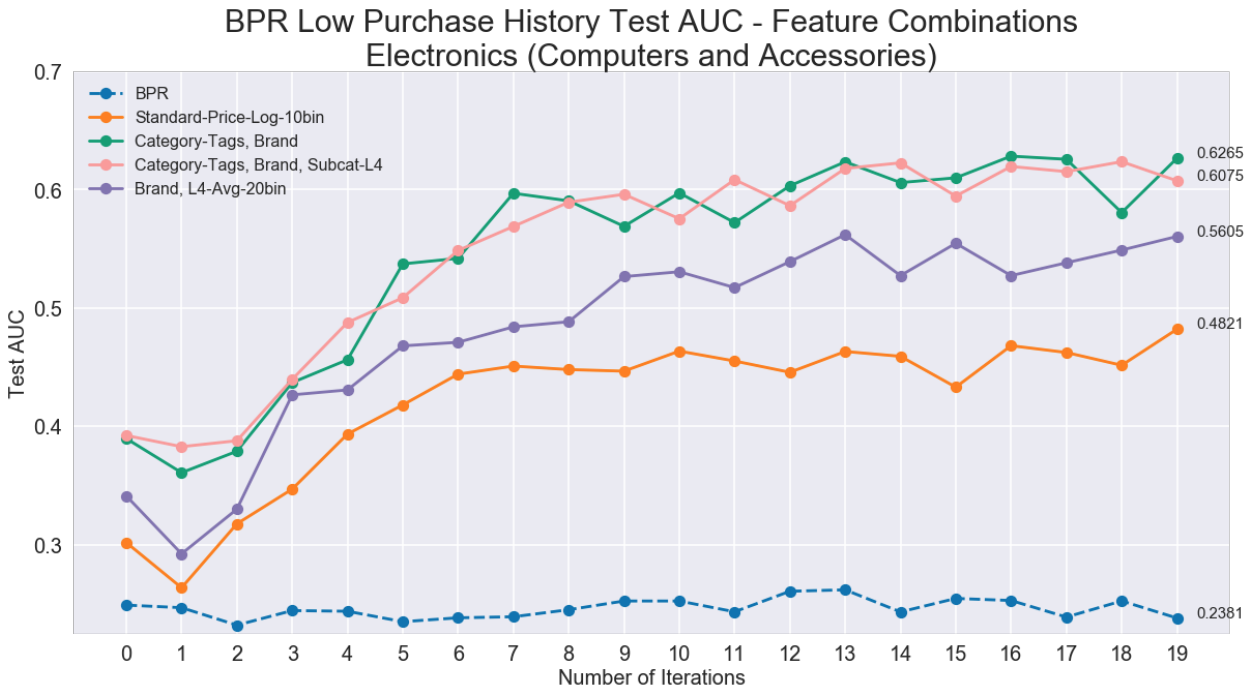
By removing users with less than 10 purchasing items and items with less than 10 purchasers from the dataset we can simulate the "cold-start" effects. We first evaluate individual features with respect to model performance. As we can see on both charts below Women Clothing category and Electronics category are showing the same order of feature importance. The order of feature importance ranking is: category-tags, subcat-L4, Brand and price.





We also ran experiments by combining multiple relevant features together and determined whether adding more relevant features together can further improve model performance. As you can see on the pairs of the figures below adding more relevant features can further enhance the model performance.

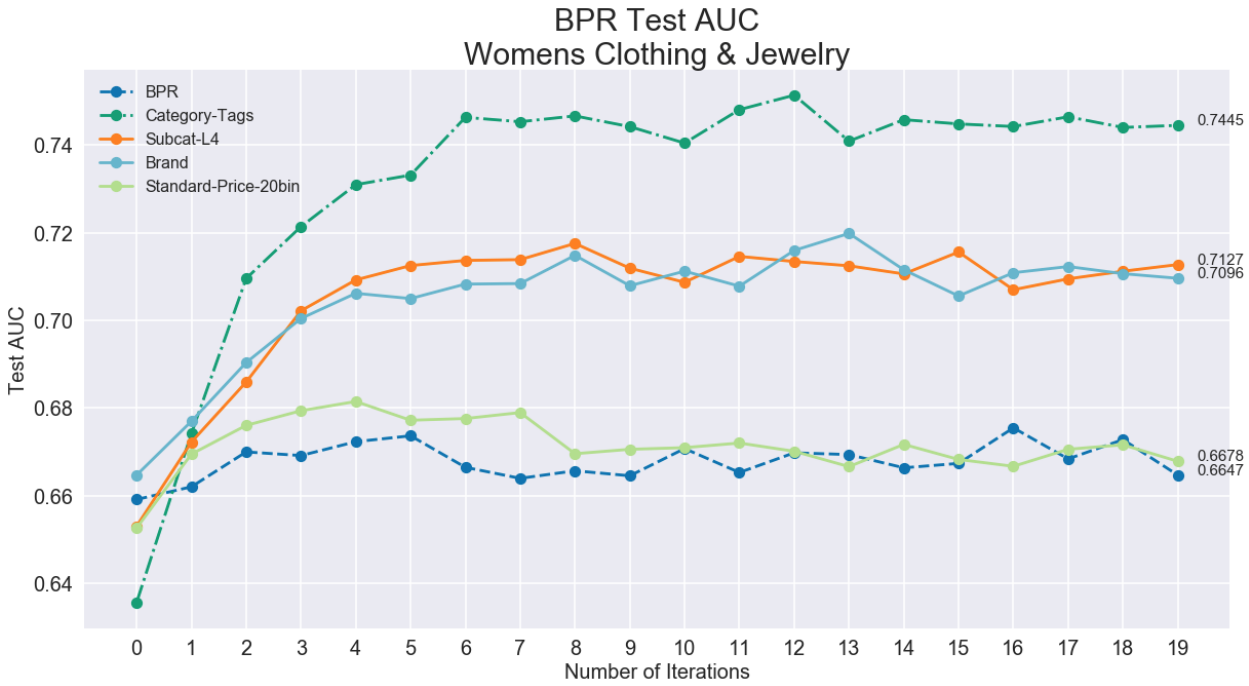




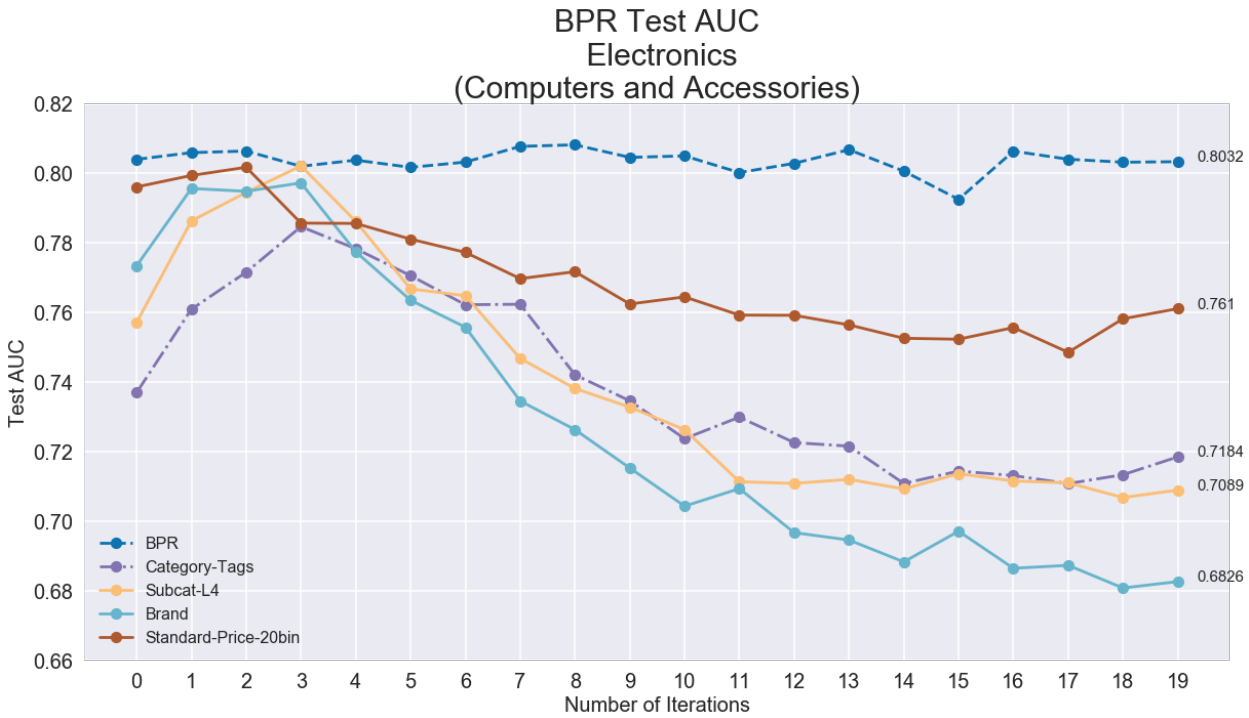
7.2.2 High user-item interactions

For these set of experiments we remove users who have less than 5 purchases and items who have less than 5 purchasers from the training dataset. We want to determine whether we shall still incorporate items features into our training models when you have dense user-item interactions. For Women's Clothing category knowing the category which the product falls under is still the most important feature. Brands and

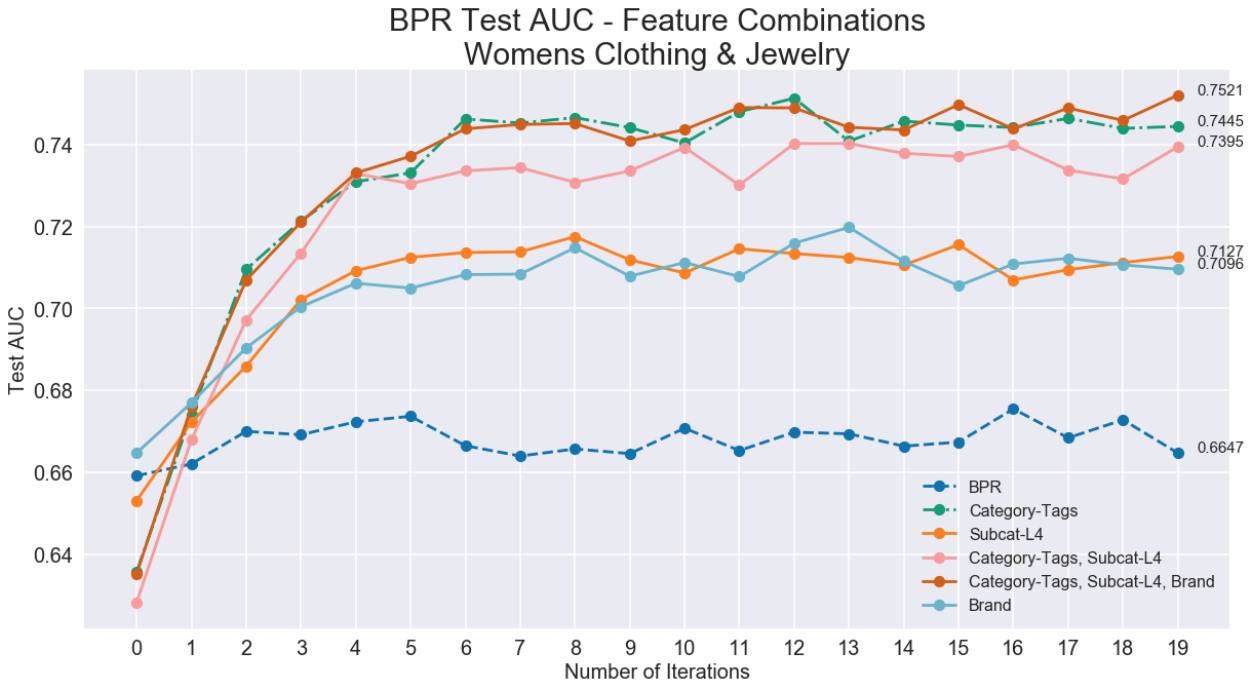
subcategory L4 perform equally well, a notch below. However, by incorporating the price feature when you have dense user-item interactions did not help model performance.

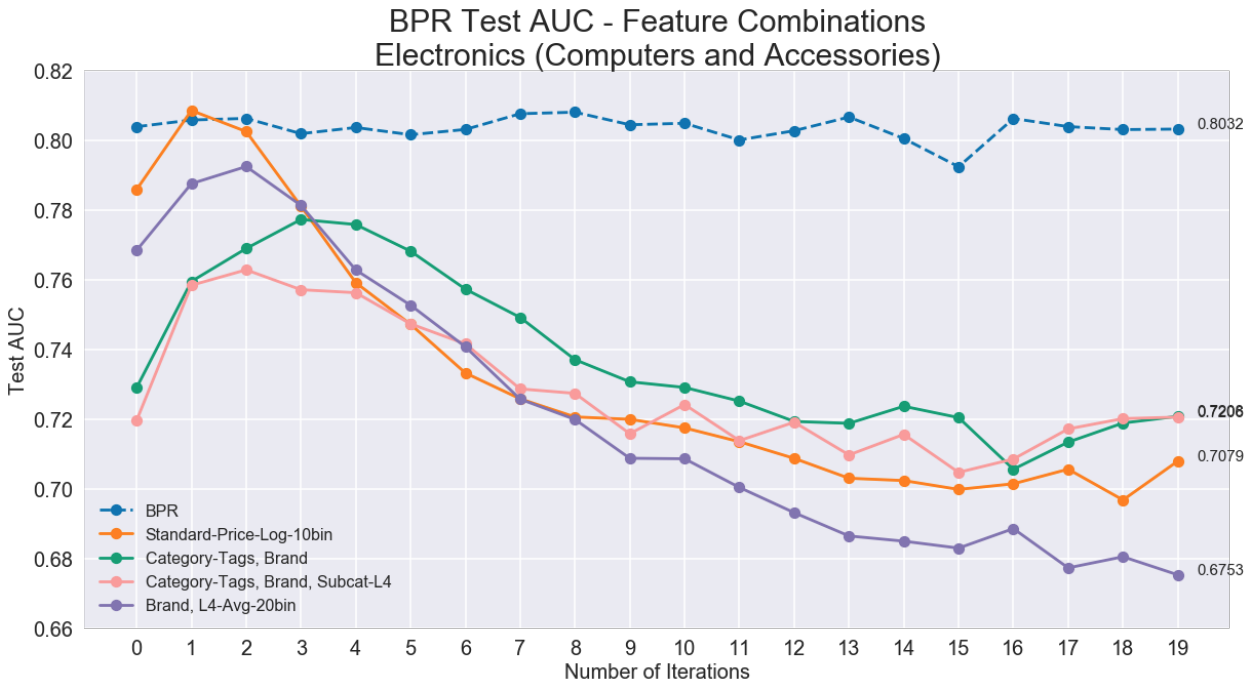


In the Computers and Accessories category incorporated features seem to degrade model performance the more epochs you train the model. This might be the case that BPR's latent factors already captured this information. Consequently, adding these features didn't have a positive impact on model performance.



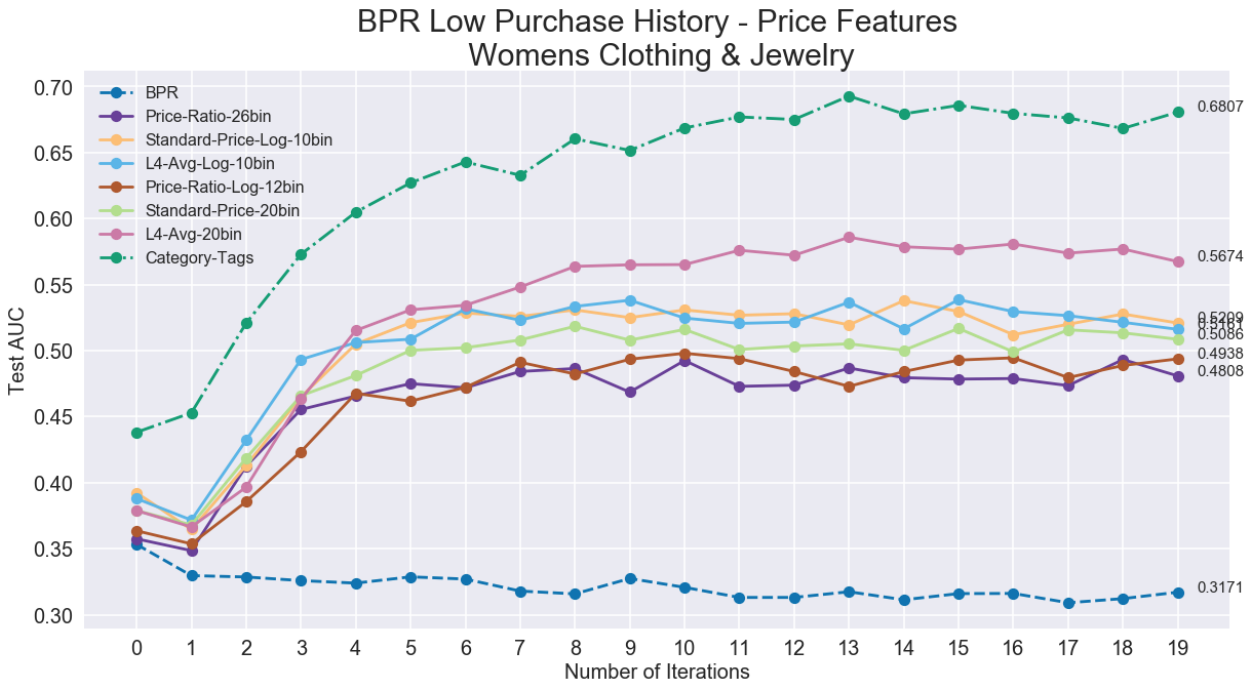
We also ran experiments by combining multiple relevant features together to determine whether adding more relevant features together can further improve model performance. For Women's Clothing category, it improves model performance. However, in the Electronics category it performs worse than model with no additional features.

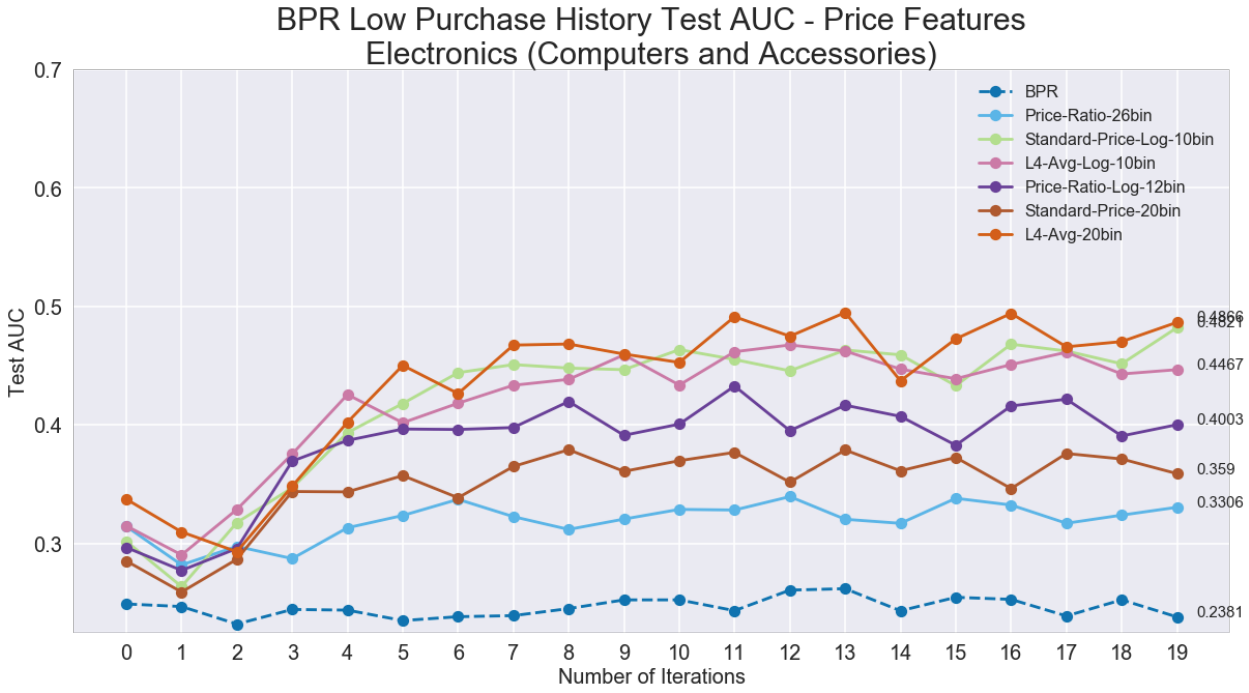




7.2.3 Price

In this project, we try to generate features that we can use to test our hypotheses. One of these features that we create is called price ratio, which we define as price over the average price of items in the same level 4 category. Our assumption is that there are customers who might prefer to purchase higher price items over lower price items or vice versa given they are similar items. We test our hypothesis using price ratio feature encoding, and the models perform worse than simple price encoding for both Women's Clothing category and Electronic category.





7.3 Tools for Communicating our Results

One of our many tools for communicating results is jupyter notebook. Notebooks allow us to show calculated results through python charts and exploring 2-dimensional view of results data in pandas dataframe. PgAdmin web console workbench is also another

tool we used for querying sql resultset. And finally we took advantage of the collaboration platform that Google provides by leveraging google chats for exchanging code snippets/images and connecting online using google talk / hangouts for collaborative discussions of results.

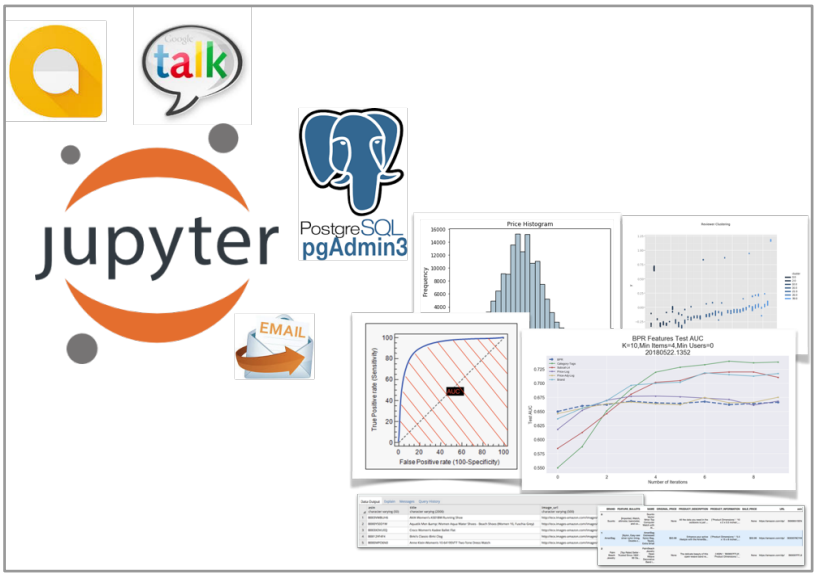


Figure 12: Sample charts and other visualization idioms

The other main tool we used is the actual side-by-side visual comparison of image of product items to analyze the results of the BPR model. We use a visualization tool to see if there's a natural relationship of recommended items unseen by the user versus what's purchased. From the initial early stages, we noticed that there were no natural correlation despite having achieved a decent AUC score. This anomaly was reported back to the optimization team, and it was later uncovered that the algorithm for extracting the BPR matrix had a flaw. This process continued throughout the development, evaluation, and reporting cycle forming a continuous feedback loop. Based on this exchange, we have decided to store our BPR ranking based on item score directly to the user-item matrix instead of extracting the learned weights in Tensorflow.

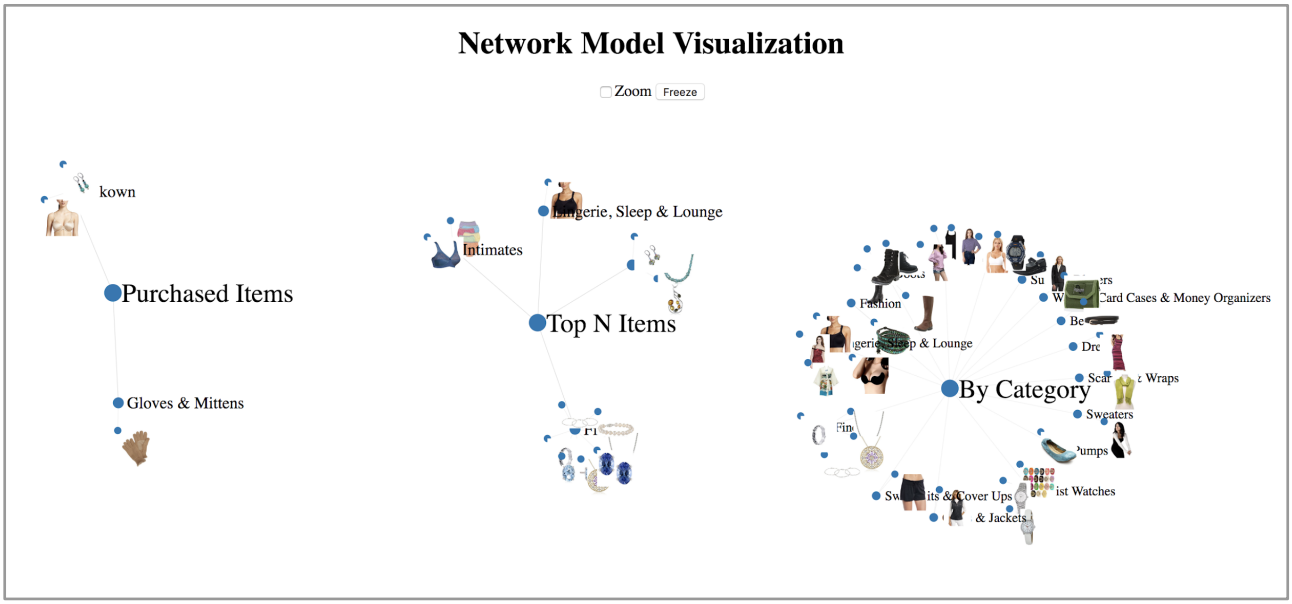


Figure 13: Visual Recommendation Comparison between user's purchase history Vs. BPR's top recommendation items v.s User Recommendation By Category.

7.4 Visualizations / Reporting Dashboard

To achieve the maximum effect user experience when presented with recommended items, we designed our visualization and reporting dashboard as a simulated online shopping application mimicking the Amazon web site's look-and-feel where we copied various Amazon's web page layout and strategic placing of recommended product items designed to maximize sales.

3-Tier Web App Architecture

Backend Layer:

The web app's product and user data used specifically for visualization reside in Postgres DBMS and the stored BPR score residing in S3 repository. The back end layer provides a data abstraction layer library to expose the data from the middle layer to ensure data persistence sources remains flexible and agnostic from the middle-tier.

Middle-Tier (REST API):

After exposing PostgreSQL data through the data abstraction layer the data flows

from the backend layer to the front-end layer and vice-versa through the middle-tier using mostly REpresentational State Transfer (REST) operations of GET and POST action methods.

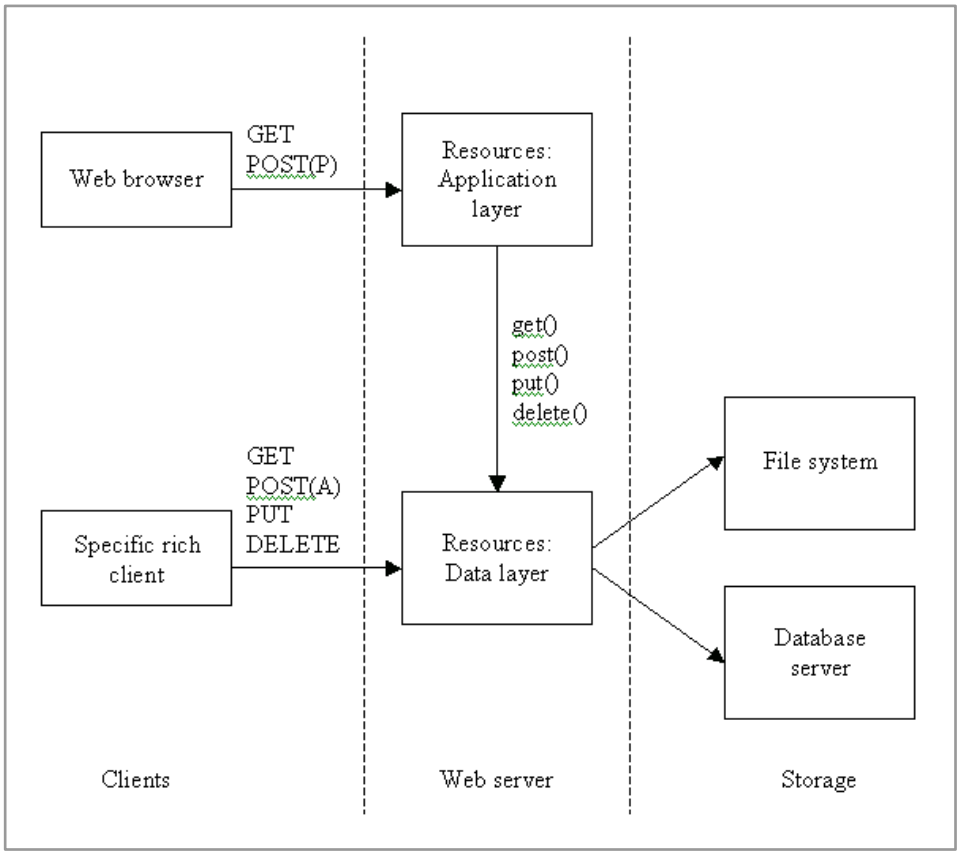


Figure 14: Three- tier web visualization tool architecture

And finally, the front-end layer is made up of user interface components rendered natively in HTML5, javascript and CSS. Our design theme is to mimic the look and feel of Amazon's shopping site so we used styling layout of BOOTSTRAP mobile-first UI layout technology, and utilized jQuery library to simplify AJAX calls and DOM manipulation. Our client-side app will be using JS LINQ for data transformation (selecting projecting, filtering, and sorting) of JSON objects and will ultimately be binded to HTML5 elements and rendered in accordance to expressiveness and effectiveness visualization principles. These principles maximize sales and help create a personalized shopping experiences that tailored for each individual customer.

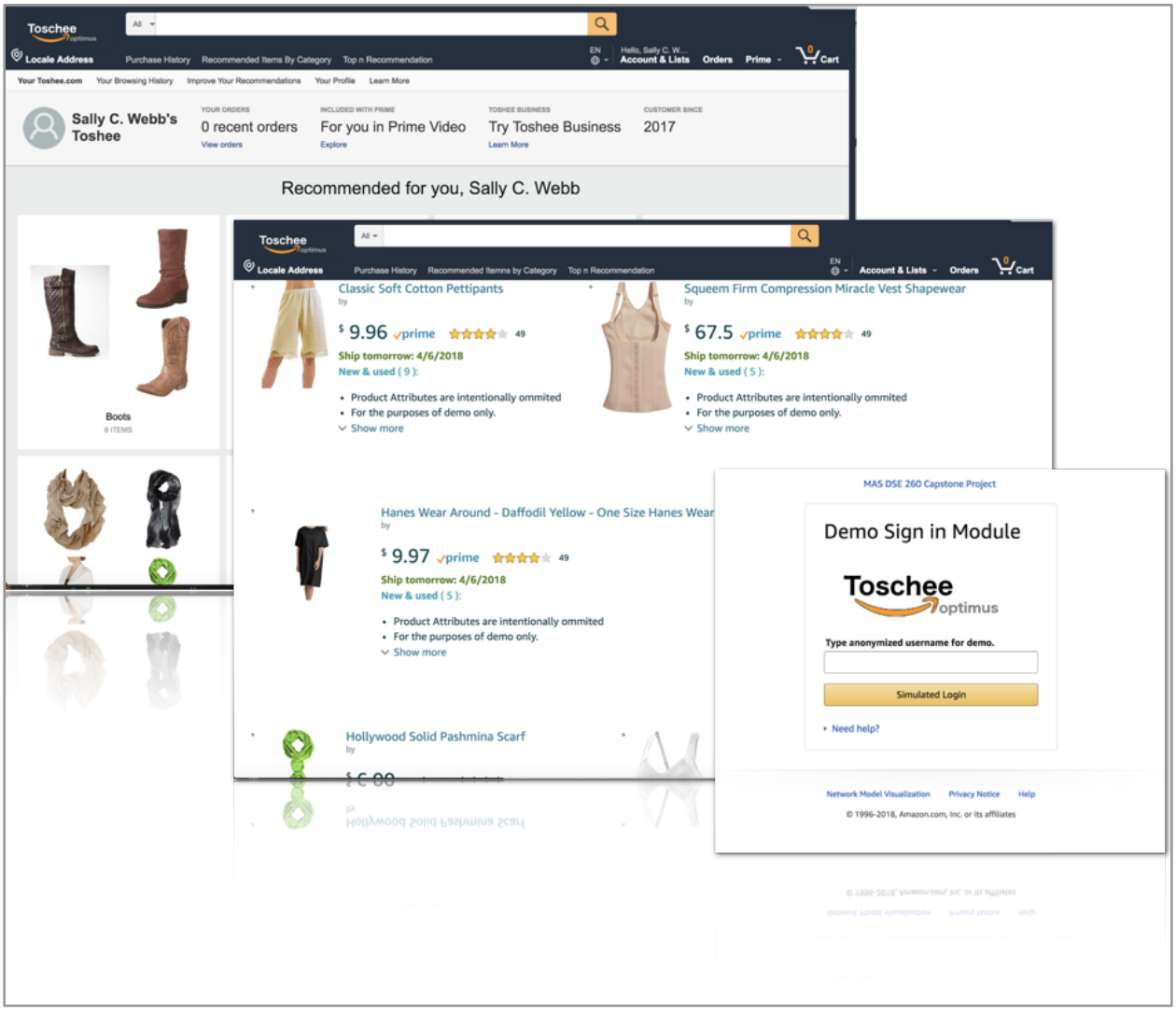


Figure 15: Few screenshots captured from web-based visualization tool

8. Solution Architecture, Performance and Evaluation

8.1 Performance Measurement

We used AWS EC2 instances to train the models. We trained the model using P3.2xLarge and P3.8xLarge instances on Women clothing category with no feature and a single Brand feature. The training time using multiple GPU P3.8xLarge instance is slightly better than training time for P3.2XLarge instance. Future work would be to optimize tensorflow code to utilize multiple GPUs that are available in GPU P3.8XLarge instances. This would improve scalability and performance.

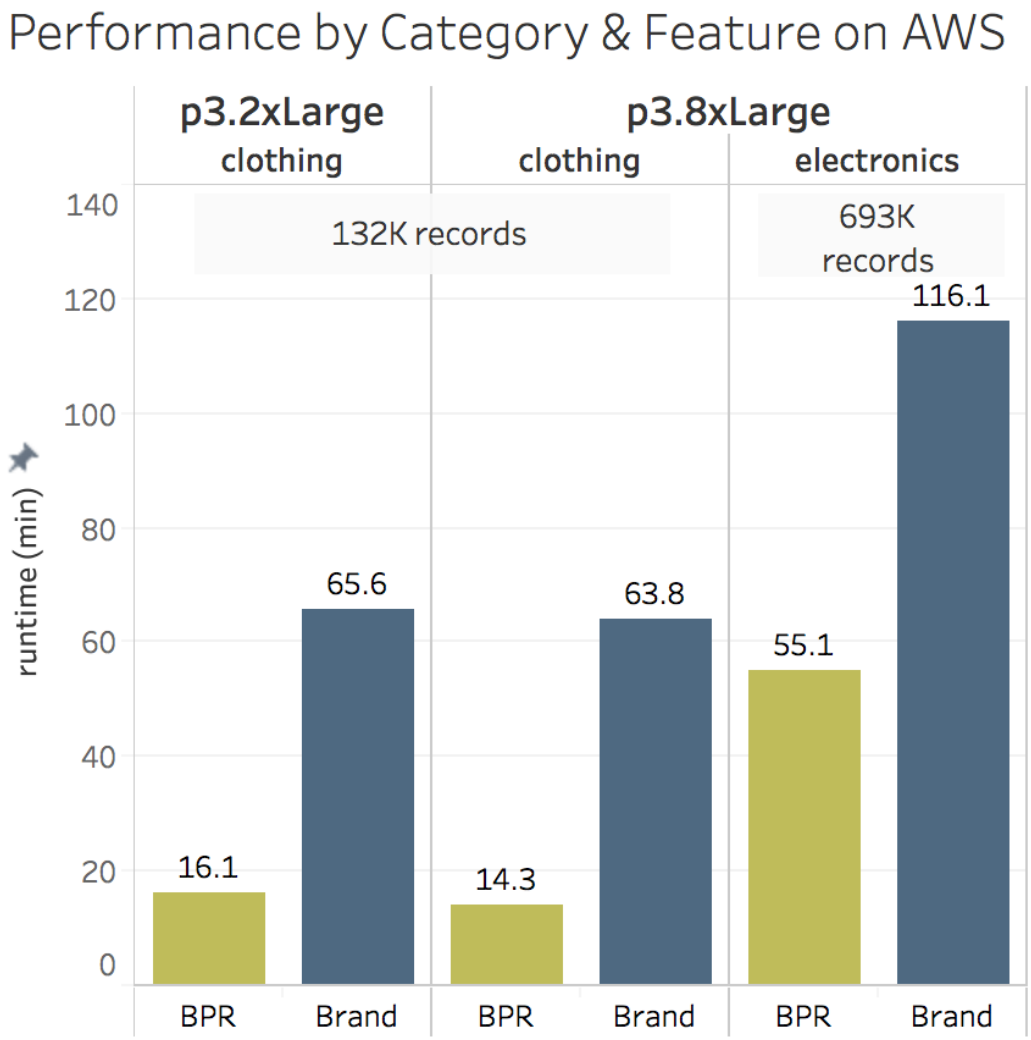


Figure 15: EC2 performance on training models

8.2 How did you scale and evaluate your models?

Even training on small subsets (5-core) of data, training time for single feature models or combination of features models can take up to 70+ minutes . On-demand AWS EC2 instances enable the scalability to experiment with different hyperparameters tuning in training the models. The models parameters and results are saved in pickle formatted files that can be load in the future to evaluate model performance.

8.3 Budget Management

The team was allocated of $2000 in AWS credits for this project. From the beginning, we know majority of the budget will be spent on training models on EC2 instances. We tried to conserve AWS credits by performing EDA and feature engineering on our laptops. Hosting the PostgreSQL database on a t2.micro also saved substantial amount of AWS credits for training models. We use multiple P3 instances to train the recommender systems, which are powered by the latest-generation NVIDIA Tesla V100 GPUs. On-demand P3 instances cost between $3.06 an hour to $24.48 an hour on-demand instances.

9. Conclusions

In this paper, we evaluate the effectiveness of BPR on Amazon reviews dataset. In cold-start like situation, incorporating relevant features can improve model performances. However; when the user-item interactions are dense, incorporating the same features into models not necessary will give you better performance. This might be because the latent factors have already capture information the features that you already incorporate it in the models. Due to the constraint of time and AWS budget, we are only able to experiment with two categories. We believe it will be prudent to test our conclusions in more categories of the AWS dataset.

References

1. J. Leskovec, A. Rajaraman, and J.D. Ullman, “Mining of massive datasets”, (2nd Edition). Cambridge University Press, 2014.
2. Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer (Long. Beach. Calif).*, 2009.
3. R. He, J. McAuley. Modeling the visual evolution of fashion trends with one-class collaborative filtering. WWW, 2016
4. J. McAuley, C. Targett, J. Shi, A. van den Hengel. Image-based recommendations on styles and substitutes. SIGIR, 2015
5. A. Egg, P. Hesami, D. Nagaraj, J. Remigio, AdvancedBPR, https://github.com/DSE-capstone-sharknado
6. S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “BPR: Bayesian Personalized Ranking from Implicit Feedback,” 2012.
7. G. Linden, B. Smith, and J. York, “Amazon.com Recommendations Item-to-Item Collaborative Filtering, ” IEEE Internet Computing, 2003.
8. “What do you mean by data preprocessing and why it is needed?”, 2017. *Electronic references* [Online]. Available:
9. https://www.electronicsmedia.info/2017/12/20/what-is-data-preprocessing/

Appendices

A. DSE MAS Knowledge Applied to the Project

The capstone project requires us to utilize a comprehensive set of skills that we learned through out in the program.

- Exploratory Data Analysis (EDA) using Python
- Relational (PostgreSQL database) / Beyond Relational (JSON)
- Machine Learning
- Principal Component Analysis (PCA)/ T-distributed Stochastic Neighbor Embedding (t-SNE) for data dimensionality reduction for displaying and processing high-dimensional data
- Stochastic Gradient Descent (SGD) / Loss Functions
- Bayesian Probability
- Data visualization, Color Application, and Visual encoding idioms to maximize channel effectiveness and expressiveness

B. Data and Software Archive for Reproducibility

Project Links

GitHub Repository (https://github.com/mas-dse-csc006/dse_capstone)

Please refers to our GitHub repository for datasets, reports, notebooks, and all other project material to access. We also created notebooks with detailed instructions in setting up AWS pipeline and PostgreSQL database.