NEURAL COLLABORATIVE FILTERING BASED RECOMMENDATION SYSTEM FOR PURCHASED PRODUCT RECOMMENDATION

Dushyantha Widanagamage

209395L

MSc in Computer Science Specialising in Data Science Engineering and Analytics

Department of Computer Science and Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

November 2022

NEURAL COLLABORATIVE FILTERING BASED RECOMMENDATION SYSTEM FOR PURCHASED PRODUCT RECOMMENDATION

Dushyantha Widanagamage

209395L

Thesis submitted in partial fulfilment of the requirements for the degree MSc in Computer Science Specialising in Data Science Engineering and Analytics

Department of Computer Science and Engineering

Faculty of Engineering

University of Moratuwa
Sri Lanka

November 2022

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The above candidate has carried out research for the Masters Dissertation under my supervision.

Name of the supervisor:

Signature of the supervisor: Date:

ACKNOWLEDGEMENTS

Throughout the writing of this dissertation I have received a great deal of support and assistance. I would first like to thank Dr. Sapumal Ahangama, for the invaluable help, guidance and feedback provided to me throughout this research. His expertise and guidance was the key to complete this research project successfully.

Furthermore, I would like to express my gratitude to my friends, who had helped me to find invaluable resources and sharing their invaluable thoughts. I also, like to express my gratitude to my Parents, for the loving and kind help they provided and the encouragements given throughout my life.

ABSTRACT

In order to validate that the problem exists, I followed the procedure as explained below.

First I grouped the data set with user id and the product. Then, for each user and item, I have derived the number of views, transactions and add to cart events. Then, I have created 10 new data sets. For the first five data sets, I have assigned different weights based on the event type (i.e. view, purchase or transaction). As for the second five data sets, they were created with different volumes of view, transaction and purchased events. Then I have verified that, with the presence of outliers (view events), the purchased products are not recommended to the user. To verify this behaviour I have used Bayesian Personalized Ranking, Neural Collaborative Filtering, Generalized Matrix Factorization, Most Pop, Item KNN adjusted and Multi-Layer Perceptron models.

Thereafter, I have removed view data from the data set and grouped data records based on the product and user. Next I have used a weighting scheme combined with binning to derive a rating score.

Next, I have used four models to verify my solution. These includes, Bayesian Personalized Ranking, Neural Collaborative Filtering, Item KNN adjusted, Generalized Matrix Factorization and Multi-Layer Perceptron. I have used fivefold cross validation to train the models and used a separate data set for validation. The results were promising. I received a Hit ratio 0.275 for HR@10. This was a major improvement as, before this the Hit ratio was near to 0.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
Pop	Item popularity based, non-personalized baseline method.
LSTM	Long short-term memory
RNN	Recurrent Neural Networks.
CF	Collaborative Filtering
OCCF	One Class Collaborative Filtering
RMSE	Root Mean Square Error
NDCG	Normalized Discounted Cumulative Gain
MPR	Mean Reciprocal Rank
HR	Hit Ratio
Caser	Convolutional Sequence Embedding Recommendation Model
CNN	Convolutional Neural Networks
AUC	Area under curve
ALS	Alternating Least Square
ARP	Average Recommendation Popularity
APLT	Average Percentage of Long Tail Items
ARP	Average Recommendation Popularity
ACLT	Average Coverage of Long Tail items
MF	Matrix Factorization

1 INTRODUCTION

Recommendation systems helps their users discover new contents. Most e commerce websites today incorporate recommendation systems into their applications in order to help consumers discover good products speedily. For example, internet web sites that sells movies may use a recommendation system to suggest movies according to their preferences. In a shopping site, user may be suggested to purchase goods based on their goods purchase sequence. These suggestions are found to be very helpful to consumers as product recommendations may help them find products very quickly. From a business stand point recommendation system may help business help to increase their sales. The use of recommendation systems is beyond e commerce. They are used by organizations to find and recruit employees based on their project needs. They are used in marketing to send users emails suggesting new products in the market. In the context of education, recommendation systems can be used to suggest Students' new Courses and materials.

There are three main types of recommendation systems. They are collaborative filtering, content based filtering and hybrid methods. Examples of collaborative filtering methods includes, user based methods, and item based methods. Apart from these, deep learning techniques are used in collaborative filtering domain as well. Furthermore, matrix factorization methods such as principal component analysis, singular value decompositions and sparse linear methods are used to implement collaborative filtering. Content based methods are based mainly on deep learning based methods as well as in most cases k nearest methods. Finally, hybrid methods combine both collaborative filtering and content based methods to increase their performance.

Finally, recommendation systems need to be evaluated for their performance. Root mean square error and mean average square are earlier used to measure performance of recommendation systems. Low values for these indicates that the recommendation system is well trained. However, nowadays, most recommendation systems are evaluated using Hit Ratio and Normalized Discounted Cumulative Gain. Normally top 10 recommended items are considered for evaluation purposes. In order to train models, five-fold cross validation is used and a separate validation data set is used for evaluation purposes. The quality metrics of recommendation systems are classified based on three main categories. They include, Qualitative, ranking and user satisfaction metrics.

1.1 RESEARCH PROBLEM

The objective function in a recommendation system can be biased towards more abundant signals over signals that are indicators of stronger preferences, which are very less in volume. As a result of this data imbalance, the recommendation system will not recommend users items that have a strong preference. Here, due to the high amount of low quality data in the data set, when the models are trained with such data, the model effectiveness will be reduced dramatically, in terms of HR@10.

Problem classification: Outliers/Anomalies

Research Importance

Quality data need to be used to train a recommendation system because [Table 1.1], only then they will produce more accurate results. In this context, products with only views are considered as data points that has a lesser probability in getting sold. However, purchased items can be considered as products that have a higher tendency to getting sold out. If data points with views were used to train the recommendation system, then the model will output products that has a lesser tendency in getting sold and as a result, organization may lose its profit.

Research Gap

Real data contains products that has been purchased, added to cart and viewed by the users. (Outliers/ Anomalies).

However, in the existing research benchmark data sets (MovieLens, Amazon, Neflix, RecSys, Youtube OTT, Pinterest , MSD) used to evaluate the models, do not contain large number of outliers and anomalies. The benchmark data sets used to evaluate models are sparse. However, benchmark data sets, do not reflect real world product purchase patterns. For example, rating of a product may not be a good indicator for purchasing a product, as it is not proportional to purchase event counts. Furthermore, the benchmark data sets do not reflect real world user behaviors.

Hence, in the existing research, a real-world data set that contains view, addToCart and purchase events were not used to come up with a solution that will help recommendation systems recommend purchased products to users and increase their effectiveness in terms of Hit Rate.

Table 1.1: Data Quality Dimensions Describing Data (Measurements)

Completeness	correct.
Uniqueness	correct.
Timeliness	Correct
Validity	Correct
Accuracy	Only items with purchased events and add To cart best describes purchasing products. View events do not contribute to purchasing a product.

1.2 OBJECTIVES

1. Improve recommendation system model performance so that the recommendation system will recommend purchased products to users. I will be using, HR@10 (Hit Ratio) as an evaluation metric to measure recommendation system performance.

1.3 REASEARCH CONTRIBUTION

I have selected an event data set which contains event data from an e commerce data set. This data set contains purchase, addToCart and view events. Here view data was considered as outliers.

Research Contribution

In the first part of the research, I have followed the following procedure to verify, that in the present of view events, the recommendation system is not recommending products that are purchased by the users (Detail can be found in section 3.1).

a. I have used a real-world data set containing purchase, addToCart and view events to train five different recommendation system models. The comparison used to compare e commerce data set with the ones used in the existing research can be found in section 4.7.2.

b. I have used five models, namely Bayesian Personalized Ranking, Neural Collaborative Filtering, Generalized Matrix Factorization, Multi Layer Perceptron, MostPop and ItemKNN to train, test and to report HR@10. To train the models, I have used the best parameters reported in the literature. The parameters used to initialize models can be found in Table 3.3. I have included, train test data creation and model training procedure in Section 3.1.

c. To verify the effect of view events on Hit rate, I have carried out the following procedure. Here, I have created 10 data sets. For the first five data sets, I have included user item interactions containing purchase, addToCart and view events. Then for the first 3 data sets I have assigned more weights to purchase and addToCart events. For the next 2 data sets, I have assigned more weight to view events. Please refer section 3.1 for data set creation and table 3.1 for weight assignments.

d. For the second 5 data sets, I have varied the volume of data points containing view, addToCart and purchase events. For the first 3 data sets, I have included more purchase and addToCart events while for the rest of the 2 data sets, I have included more data points containing view events. Please refer section 3.1 for data set creation and table 3.2 for how volumes varied in the data sets.

e. Next, I have used above mentioned 10 data sets to train recommendation system models and reported the HR@10. I have trained each model using 5-fold cross and

used test data set for evaluation. I have used 80% of the data set for training and 20% for testing and used a random split to derive train test data sets. Then, I have reported the NDCG@10 and HR@10. Please refer section 3.1 for model training procedures.

f. The result discussion can be found in section 4.1.

In the next phase of the research, I have removed view events from the data set, and used those data set to train the same above models. Then, I have compared Hit Rate obtained, to see which model best suits for recommending purchased products to users. Please refer section 3.2 for train test data derivation, model training and for evaluation detail.

g. For the first experiment, I have varied the learning rate and trained four models namely, Bayesian Personalized Ranking, Neural Collaborative Filtering, Generalized Matrix Factorization and for Multi-Layer Perceptron and reported the HR@10. Please refer section 3.2 for more detail about first experiment details and section 4.2 for discussion.

h. As for the second experiment, I have varied the predictive factors and trained 2 models, namely, Generalized Matrix Factorization and Neural Collaborative Filtering and reported the HR@10. Please refer section 3.2 for more detail about second experiment details and section 4.3 for discussion.

i. For the third experiment, I have trained the models, and reported Hit rate for top k results where k varied from 1 to 10. I have used best parameters reported in the literature to train the models. Please refer section 3.2 for more detail about third experiment details and section 4.4 for discussion.

j. Next, I have conducted an experiment, to verify effect of pretraining on HR@10. To verify this behavior, I have used a neural collaborative filtering with pretraining and a model without pretraining. I have used Adam as the optimizer and reported HR@10 varying predictive factors to 8, 16, 32 and for 64. Please refer section 3.2 for more detail about 4^{th} experiment details and section 4.5 for discussion.

k. Finally, I have varied the number of MLP layers to verify whether the deep learning could help neural collaborative filtering improve HR@10. Please refer section 3.2 for more detail about 5^{th} experiment details and section 4.6 for discussion.

l. Finally, in section 4.7, I have compared steps I have performed with the steps performed in the existing research.

2 LITRATURE REVIEW

The problem of recommendation can be viewed as follows. That is to predict users' next preferences based on his or her past behavior [77].

Application of recommendation systems are as follows.

1. Some web sites may recommend user sports news articles because he had viewed or he had spent more time reading those articles.
2. An internet web sites that sells goods may suggest users' food products based on the consumer purchase history.
3. YouTube suggests which video you want to watch next based on similar videos you watched.
4. Google apps store application recommendation.

In general, following activities had been carried out, during building recommendation system. The first step in building a recommendation system is to define data for training and evaluation.

Some common data split protocols used includes [78].

1. Random split.
2. Chronological split.
3. Data Split in Scale.

Furthermore, users provide feedback on different items mainly via explicit and implicit feedback. Then, the data gathered from different sources are transformed into counts and averages using different techniques.

There are three types of Recommendation Systems. These include:

1. Collaborative Filtering.
2. Content-based Filtering.
3. Hybrid Methods.

Here a suitable technique is required to be selected, based on the problem. Furthermore, once the relevant items are returned to the user, by the model, items need to be ordered, so that the result can be displayed based on user interests, so that the user experience can be improved. Finally, one additional re ranking criteria need to be applied to remove incorrect candidates from the list. In order to evaluate recommendation systems, rating ranking and classification metrics are used. Two widely used evaluation metrics are, Root Mean square error and precision and recall.

Once the recommendation system has been successfully trained, they need to be deployed. One popular choice for deploying recommendation systems is the cloud and offering them

as a service. At the time of the deployment, it is decided, whether to go with platform as service or Infrastructure as a service.

Finally, before moving the recommendation systems to production, response time of the model need to be tested. Response time is critical, as users expect quick responses from the system. Additionally, a load test need to be performed to find out whether the system can handle the required number of requests. Moreover, these results need to be accurate as well. Otherwise, users will leave the e-commerce website without remaining in the website for long.

2.1 Training and Test Data Splits

Random split is a method which can be used to divide the given data set to training, testing and validation data sets. This method assumes that, all the data points are independent of each other. The advantage of this method, is we can create as many as training, testing and validation data and can be used to train the model, so that it can help improve machine learning evaluation metric which is used to measure model performance, significantly.

On the other hand, the data can be ordered in chronological order as well. One such example is click streams. To handle these types of data we, simply split the data, based on a given time stamp, so that the percentage of the data set will be assigned to training, testing and validation datasets accordingly. To ensure training, testing and for validation will have data with correct ratings, user ratings more than a predefined threshold is selected.

Chronological splits can be also combined with stratified splitting techniques to ensure that the data points are available in training as well as in the test sets. In this way it is guaranteed that the training and test data set has the same distribution.

For the larger data sets, data splits are performed using apache spark clusters.

2.2 Feature Engineering in Recommendation Systems

The first step in the feature engineering phase is the data exploration. In this phase, attributes of the data set are well understood [76]. It will be beneficial to identify selected data date range and to identify how a particular attribute were measured. Then target variables and their classes are identified. In the data exploration phase, the data shape is understood and their distribution is understood as percentages (Examples: How many users had bought books in a given set of data records.). To identify the sparsity, we can use the frequency of occurrences grouped by user or item, in the data. This helps understand sparsity of the data and identify patterns in it. We can also use products, users had frequently bought, to identify purchase patterns. In summary, in feature engineering phase, data is explored extensively to identify frequent patterns in the data set.

In the feature engineering phase, we first check for missing data. For missing data, we can either remove records or can use a Laplace correction (For the categorical data). If the missing data is numerical, we can use mean or median to replace missing values. Furthermore, to create more associations between predicted variable and target variable, we can combine them together and create new categorical variables. We can use a decision tree to visualize our new feature engineered variables, to see whether any gain is available by combining variables. We can use factorization methods to convert long ranges of product ids or names to discrete number of items ids. Furthermore, we can put products into different categories, based on their occurrences to further reduce the product dimensions.

Target encoding is another method which is used in feature engineering. Here, the information is extracted from data, that can be fed into any model. We can simply calculate the probability of item or feature appearing in the data set to find this. Smoothing is an another improved feature engineering technique used in target encoding. Here we use the average rating, if the occurrences of the item are very high in the data set. Otherwise we use the global rating average of the item. We can further improve the target encoding with the k fold. In this case we calculate the global and average means using the all the folds in the data set excluding the current one.

Another encoding technique used is count encoding. Here based on product occurrences, we tell the model whether it is frequent or not. We can combine features to create count encodings as well. If we have both validation and test data sets, we simply merge values calculated in training data set to test and validation data sets.

Another, technique we use in feature engineering is binning. Here, we take the support of an expert or use predefined interval to map values to the relevant bins. This help reduce model over fits. We can also use normalization for feature engineering as well. The main

reason for using this is to prevent both training and test data sets having attributes with different scales and statistics.

GaussRank is an another technique that can be used for feature engineering. Here, we sort the data according to their rank or value. Then we scale the value between -1 to +1. Finally, we can use inverse error function to create the Gaussian distribution.

We can further use events with timestamp generated in system logs and aggregate them using window functions to create values, which can be used to feed into recommendation systems. We can use shift functions to create data with tabular formats. Furthermore, we can use the difference between time stamps, to create how a particular product related attribute had changed with time. This can be very useful if we are using tree based methods or deep learning techniques to create recommendation systems.

2.3 Collaborative Filtering

2.3.1 Matrix Factorization Algorithm

Matrix factorization is the most basic algorithm used in recommendation systems. Here, in this method users and items are represented in a two dimensional matrix. The values of the matrix consist of user feedback.

A feedback can be in two forms.

Explicit: User specifies the preferences of an item by rating it using a number range from 1 to 5 or using binary value 0 or 1.

Another form of feedback is implicit feedback where user behavior related information is used to derive the ratings. For example, a purchase event may be a good indication of user's preference of a product.

In a typical matrix factorization problem, a list of items and a set of users will be provided.

Then, both user and item features are learnt in a form of latent variables. Here, user embedding is learnt in such a way that the product of the user embedding and item is a good approximation of a feedback matrix.

User embedding matrix: $U \in R^{m \times d}$

Item embedding matrix: $V \in R^{n \times d}$

The embedding is learnt in such way, that the UV^T can be seen as a better estimate to the user product preference matrix. To learn this U and V matrices we can use an objective

function. In order to minimize the squared error, we can use the objective function such as one specified in figure 2.1.

$$\min_{U \in \mathbb{R}^{m \times d},\, V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{obs}} (A_{ij} - \langle U_i, V_j \rangle)^2.$$

Figure 2.1: Objective function for Matrix Factorization

The main problem with this approach is that the model over fits resulting in poor generalization. To handle this problem, we can follow the following procedure. That, is for the product that do not has a rating from the user can be treated as 0. In this way, we can reduce the frobenius distance between the original utility matrix and approximated matrix multiplication.

$$\min_{U \in \mathbb{R}^{m \times d},\, V \in \mathbb{R}^{n \times d}} \|A - UV^T\|_F^2.$$

Figure 2.2: Frobenius distance between two matrixes

Furthermore, we can use singular value decomposition to reduce the model overfitting and make model more generalized. However, this is not practical, because in reality, feedback matrix A can be very sparse. To solve this problem, Weighted Matrix Factorization can be used. Here, the new objective function can be defined as (Figure 2.3),

$$\min_{U \in \mathbb{R}^{m \times d},\, V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{obs}} (A_{ij} - \langle U_i, V_j \rangle)^2 + w_0 \sum_{(i,j) \notin \text{obs}} (\langle U_i, V_j \rangle)^2.$$

Figure 2.3: Weighted Objective function used in recommendation systems

In order to optimize, hyper parameters in objective function defined in figure 2.3, following techniques are used.

1. Stochastic Gradient based methods
2. ALS or Alternating Least Square which are based on weights.

Table 2.1: Comparison of Stochastic Gradient Descent and Weighted Alternating Least Squares

Optimization Technique	Advantages	Disadvantages
Stochastic Gradient Descent	Easy to use with loss functions.	Takes more time to train.
	Can be parallelized.	Harder to handle unobserved entries
Weighted Alternating Least Squares	Can be parallelized.	Can be used with root mean square error only.
	Can be trained quickly compared to Gradient based methods	
	Can easily handle missing values.	
matrix factorization	No domain knowledge requires, as the model is capable of learning automatically.	Collaborative filtering is not suitable to handle cold-start problems.
	These models can help users discover new interests.	It is difficult to incorporate new features into the query.

Next we will look at collaborative filtering methods.

Alternating least square is a matrix factorization method which can be used to solve personalized recommendations [2]. This model proofed to be well performing over their nearest neighbor techniques on both explicit as well as implicit data. In order to produce good accuracy, it is essential to optimize the objective function. However, with the amount of the data this will become extremely inefficient. One way to solve this is to have a tradeoff between computational efficiency vs accuracy. This is to use a good sample of

data to which then can be used to train the model and produce good result. The main contribution of this paper is to improve the objective function which is used for ranking without the support of sampling.

In order to evaluate the solution two data sets were used. The first data set, which is Y!Music, that contains over 6.2 million user ratings of nearly 250000 users and 30000 products. The other data set which is the Netflix data set contains nearly 500000 users and ratings of 18000 items. For the both data sets training and tests were derived although it is not specified in the paper how this was done.

In order to compare model performance, Pop, ImplicitALS [4], RankSGD [3], RankSGD2 [3] and RankALS was used. Error rate, ARP and Recall was used as the metrics. The first experiment was done to find out the effect of model size on the accuracy. Here, in order to change the model size, the researchers have trained the models with the different feature numbers. RankALS has reported lower error rate and ARP on Y!Music data. Considering Error rate, Pop had reported a steady value of 0.4 and for ARP this was 0.11. For all the other models the error rate and ARP has started with a higher value and ended up in a value of 0.1 and 0.03 respectively. For recall RankALS had outperformed all other models, and reported a highest value of 0.42. Pop model has reported a lowest value of 0.05. Again for all the models, as the number of features increases, the recall had increased.

Same pattern can be observed on the Netflix data set as well. However, considering ARP, for features between 40 to 50, RankedSGD and RankedSGD2 had performed well over RankALS. As for the recall, again for features between 20 to 50, ImplictALS had outperformed other models. However, the performance of RankALS was nearly same as the ImplictALS.

In the next experiment ErrorRate change was analyzed with the increase of itterations. To compare the performace, RankSGD, RankSGD2 and Implict ALS was used. It is found that for the low number of iterations, both models have reported a higher error rate. However, as the number of iterations increased, the error rate got exponentially reduced. Finally, in all iterations, RankALS had outperformed the other models.

In conclusion, RankedALS had managed to show better performance, with a lower training time. The model performance can be further can be improved using more context aware information or by using time related information or incorporating meta data to the objective function.

SLI-rec[6] is a recommendation system which tries to improve the model performance by combining users' short term and long term preferences. One major contribution of this paper is to use time and context information to change RNN model state transitions, which in return helped improve recommendation system performance. Furthermore, the

researchers have introduced a new attention based mechanism to add both short term and long term preferences together.

Two data sets were used to evaluate this new solution. One date set, which contains customer behaviors found while shopping in Movies, TV, CD and Vinyl. Here the test set was created by selecting his or her next selected T number of interactions. After test set was created, 50 percent of the data set was used for hyper parameter tuning. The other data set which is collected from MSN data set, which contains click stream data. These clicks are generated based on user clicks on MSN web site advertisements. Test data set generation was similar to the Amazon data set. In order to compare new model performance with others ASVD [7], A^2SVD, DIN [8], LSTM, LSTM++, NARM [9], RRN_{day}, RRN_{week}, [10] CARNN [11], T-LSTM [12], and DIEN model was used. Area under curve and F1-score was used as the evaluation metrics. In the first experiment model performance was compared with other models.

For the entire, electric and for movie data sets, the new model had outperformed the other models. Considering F1-score, all models had reported a value of 0.7. However, the new model had reported a F1-score above 0.77 with entire, electric and for movie data set. For the CD data set, this was 0.85 and for other models it was around 0.83. As for the Ads data set, the new model had reported a value of 0.3367 and this value is closely followed by the other models as well. Another importance finding was that, A^2SVD and DIN had reported better results proving that assigning importance score to sequential models can help improve performance. Furthermore, LSTM and LSTM++ had reported higher performance values proving that adding short term preferences can help models improve their performance. Furthermore, since we have used contextual information to train our model, CARNN, T-LSTM and DIEN outperformed LSTM and RNN models.

Next, in order to verify effectiveness of introducing a short term component to SLI-Rec, LSTM model was compared with SLI-Rec with time aware information and SLI-Rec models created with combining different contextual information. What found was that adding time aware information helped models improve their performance significantly. Furthermore, combining contextual information had a slight effect on improving model performance as well.

To verify effectiveness of including an attention based mechanism researches have selected categories that has more than 100 products. Then the products were ordered by their importance score and for each category mean and variance were found. Hence it is found that adding a discriminative function had helped improve recommend products based on user long term preferences.

One main problem in recommendation systems is that, they become inaccurate, for short lived sessions of user interactions as well as for user interactions that has a long history. This problem can be resolved by using item to item recommender system approaches.

However, in this context, treating all user short lived sessions as a one whole session can be very useful in providing users with better recommendations [20]. Hence one key contribution of this paper is to introduce a new method to model user interactions in order to improve performance over matrix factorization and item to item recommender system models.

Two data sets were used to evaluate the GRURec model. The first one is click stream data set obtained from e commerce purchased events. This was obtained from Rec sys 2015 data set. The next data set was collected from a video service platform. As for the click stream data set, researches have removed sessions with length 1. They have also removed clicks where clicks are not available in the training data set. The, preprocessing step for video streaming data set was similar to the purchased events.

To compare the effectiveness of the solution four models were chosen namely, POP, S-POP, Item-KNN and BPR-MF [15]. Recall@20 and MRR@20 was used as the evaluation metrics. In order to initialized the models, researches have used the best parameters reported in the literature.

Here, before starting the experiment, for each baselines best set of parameters were found for each data set and these were used to initialize the models. In order to tune session based recommender system, 100 experiments were ran, with each data set with different loss functions and then best parameters were selected with the relevant loss function. Furthermore, it is found that the model performs better with GRU layers than RNN with LSTM layers.

The optimized version of the GRURec was compared with the Item-KNN with both data sets. The results indicate that GRU4Rec has outperformed Item-KNN. Another finding was that with the increase of number of GRU units the pair wise loss had increased while cross entropy loss had gone down. Furthermore, increasing the number of units had resulted in increasing the training time as well. As a further research it is proposed to do more research on the proposed neural network, and to modify the network to train on video and text data.

Caser[16] is a recommendation system that is based on convolutional neural networks. The idea is to predict user next top N prediction based on user past record history.

Here in this research, one limitation of the existing work was identified. This is in Markov chain based model, given a set of actions previously performed by the user, the predicted actions depend only on one action in the previously performed actions. Furthermore, for a given sequence of actions, each action has a strong association with the next action. Hence, in Markov chain models, the predictions are based on previous action. They do not allow to predict actions only by skipping previous action.

The contribution of this research is to use of convolutional neural networks to predict next sequence of actions given a set of actions of a user. Hence, it is claimed that the model has successfully managed to solve the limitations aforementioned using convolutional neural networks.

In order to evaluate the solution four data sets has been used. These are MovieLens, Gowalla, FourSqauare and Tmall. In order to evaluate models, precision, recall and mean average precision was used. Four models were used to compare new model performance. They are Pop, Baysian Personalized Ranking, Factorized machines, Markov based chain models and GRU4Rec model. Before training the models, grid search method was used to find best hyper parameters. Hence these para meters were found using different latent space sizes and learning rates.

The first experiment was to compare the model performance with other models. Other than MovieLens, for all other data sets, Caser had outperformed other models. On movielens data set, GRU4Rec had better performed over Caser. Next it is looked on effect of dimensionality on the Caser. It is observed that, dimensionality required to be chosen correctly. For higher dimensions, the model got over fit. For the sparser data sets, Caser had managed to show a good performance, with smaller latent dimensions.

Next model performance was studied over markov order L and number of predictions T. It is found that for the dense data set Caser with higher number of targets performs better, proving the improvements of side effects. However, for the sparser data sets, Caser with two targets had performed better and the performance of Caser with 3 targets and 1 was not consistent. This is because the sparser data set contains more noise. Finally, it is studied the effect of introducing of convolutional filters for the model. Clearly Caser models with horizontal filters and models with horizontal and vertical filters had outperformed Caser models with only vertical filters. This shows that the introduction of convolutional filters had helped improve the model performance.

Next Item Recommendation model is an another variant of a convolutional neural network, which is been used to predict next items user will interact for a give session. Here the concept is simple. Order items user had interacted in a given session to create a two dimensional latent space as an embedding. Again this new matrix is considered as an image. Then convolutional operators were applied on the image. The new model combines generative models and convolutional operations to improve the model performance. Furthermore, the researches have used residual block structures to improve model performance.

This new model tries to solve following limitations of caser model [19]. The first problem is in real world sessions can be in variable sizes. Secondly small filters cannot effectively learn features from the given input. Caser model model uses large filters and use max

pooling to solve variable size input problem. However, this technique is not effective in learning important features.

In order to evaluate the new model, two data sets were used namely, purchase data set containing RecSys challenge 2015. Next data set contains, music names and how many times they have listened by the users. As a preprocessing step for the first data set, they have remove sessions with interactions lesser than 10. For the music data set a predefined session length were defined. This is because model need to be tested against, short and long range sessions. These lengths were 5, 10, 20, 50 and 100 respectively.

In order to evaluate the models, NextItNet model required to be trained. In order to do this, learning rate of 0.001 and batch size of 32 was selected. Embedding size was fixed to 64 and soft max and residual blocks were used to report the results. In order to evaluate the model, NDCG@N and HR@N was used.

Before starting the experiment, from the Yoo data set researches have removed session length lesser that 3. For the music data set, first three data sets were created, randomly taking 20,000 to 200000 songs from the data set. Then using music data set that has large amount of songs 5 data sets were created based on the session length 5, 10, 20, 50 and 100. Training, test and validation data set was created by splitting data randomly. Out of whole data set, 50% was used for training, 45% for testing and 5% for validation. Caser and GRU4Rec performance degrades when the session length gets more and more longer. To avoid this problem, longer sessions were divided into sub sessions and those were used to train models. Finally, the new model NextItNet was trained using learning rate 0.001, batch size of 32 and embedding size was set to 64 and used a full soft max layer.

When comparing with other models, neural network models such as GRU4Rec, Caser and NextItNet report very good accuracy results. The results were 120 times better than Mostpop model when considering metric MRR@5. NextItNet had reported a MRR@20 of 0.3233 on music data set Music_M5. Furthermore, for all data sets NextItNet had reported best results compared to GRU4Rec and Caser. One reason for this is NextItNet performs better on sequential data sets compared to other models. Furthermore, since NextItNet do not use pooling layers, it keeps whole information in the model without any information loss. Finally, since NextItNet uses residual blocks it can better learn complex relationships in sequence data compared to other models. Finally, it is observed that, it takes only less time to train NextItNet compared to other neural network models because it uses full sequential information to converge to optimal solution. Furthermore, the new model is fully based on convolutional neural networks.

One limitation of the work is that, in the evaluation, contextual information is not incorporated to the model, although model is flexible to such data.

Bayesian personalized ranking [21] is a model that is been introduced to recommend products based on user behaviors such as using click events data. There are KNN based

methods available, however, one contribution of this paper is that, they have introduced a Bayesian based method for optimization. In order to evaluate the solution two data sets were selected. One containing consumer purchase history from an online shop. Next, DVD rental history of consumers from Netflix. In order to evaluate performance mean area under curve was used. In order to compare the solution, Matrix factorization based SVD as well KNN based models were used. Additionally, matrix factorization and KNN based model was trained by applying Bayesian optimization.

For the both data sets for each model, the average AUC was found with the increase of feature dimension. Here the first observation was that, SVD-MF although it uses matrix factorization internally, it displayed poor performance. WR-MF on the other hand shows improvement and kept growing as the amount of features increases. BPR matrix factorization on the other hand outperformed all other models for all the feature dimensions. Furthermore, compared to non-personalized ranking, Bayesian personalized ranking had performed better in terms of AUC.

Hence, with this result it is clear that, for personalized ranking Baysian optimization performs better.

One class collaborative filtering [23] is a method which is mostly suitable for binary class classifications. The problem, which is trying to solve is that, normally one class problems contains data which is very sparse. That is, the data set contains one class of data much higher compared to other. Here, a model was introduced where its performance outweighs solutions based on negative sampling and weighted ranking techniques.

In order to evaluate the solution, two data sets were used. One containing events data collected from yahoo news. Preprocessing was done to ensure that given the news item, it gets same article id. In order to evaluate models, mean average precision, Half-life utility was used. In order to compare model performance, ALS and SVD were used where these models were trained by applying non negative sampling and applying weighted ranking techniques.

The first experiment performed was to find effect of dimension SVD and ALS based models. For SVD as the number of dimensions increased the performance increased although after a certain threshold it started to drop. On the other hand, for ALS, the performance had increased with the increase of feature dimension. The maximum accuracy had converged when the number of features size was 50.

Next, the model performance was compared based on sampling and weighting methods. Hence, it is found that weighting based on users are best suited and item based weighting is not suitable. Next, with negative sampling it is found that considering missing value as negative samples, can produce good results. Motivated by these results, the researches had treated all missing values as negatives and corrected the bias with negative sampling

techniques such as weighing and using examples. The solution had produced better results outperforming ALS and KNN based methods as well as singular value decomposition.

One limitation of the work was that, no test been conducted on the effect of positive to negative weight ratio effect on weighting and sampling schemes.

Variation Auto encoders [30] can be used for collaborative filtering. This is mainly because they can well model nonlinear relationships. One of the key contribution of this paper is use of multi nominal distribution to model non linearity. Furthermore, in order to improve the model performance regularization method was used as well.

In order to evaluate the solution, MovieLens, Netflix prize and Million Song data set was used. Recall@R and NCDG@R was used as the evaluation metrics. Weighted matrix factorization (wmf) [31], Slim [32], Collaborative denoising autoencoder (Mult-vae^{pr}) [33] and Neural collaborative filtering [34] were used as the base lines for comparisons.

First for each data set model performance was compared. The first observation was that, Mult-vae^{pr} and Multi-DAE had significantly outperformed the other models. Furthermore, Mult-vae^{pr} had performed significantly well on movie lens 20M data set. In most cases Mult-vae^{pr} , Multi-DAE, CAE had reported good results.

Then, NCF was compared with Multi-DAE with 0 hidden layers with MLP generative model. The results indicate that Multi-DAE performs well for all data sets. Next an experiment was performed to check effectiveness of introducing multi nominal distribution to the Multi-DAE. To compare the effectiveness Gaussian likelihood, log loss was used. Hence, for all the data sets, introducing multinomial had proven to be producing good results.

As for the future work, there need to be investigation need to be done to find out why introducing additional regularization parameter had helped improve model performance. Furthermore, model performance can be improved by adding more side information.

Neural Collaborative Filtering [34] tries to solve recommendation system problem using deep learning, where data contains implicit feed backs. The model performance was measured using MovieLens and Pinterest data sets and Hit Ratio and Normalized Discounted Cumulative Gain was used as the evaluation metrics. ItemPop, ItemKNN [35], BPR [15] and eALS [36] were used as the baselines for comparisons.

The first experiment was performed for both data sets, with increasing number of factors. As for the ItemKNN number of neighbors was changed to report the results. The results indicate that the neural collaborative filtering had outperformed BPR and eALS by 4.5% and 4.9% respectively. The model managed to produce good results even the number of factors was 8. Hence, this shows that neural collaborative filtering can learns nonlinear functions better compared to GMF and MLP. Moreover, it is shown that GMF outperform MLP and for the larger data sets and for the large number of factors the model becomes

over fit to the data sets. Furthermore, considering top k recommendations where k ranges from 1 to 10 NueMF outperforms all other models. Furthermore, GMF had outperformed BPR consistently for all k values.

Considering pre training, NueMF had managed to gain better performance over without pre training. However, when number of factors were around 8, the model performance was slightly lower compared to without pretraining. NCF solves one class problem as a binary class classification. Here as the number of training iterations increases, NCF performance improves. For 10 iterations, good training loss reported, although the model got over fitted. Moreover, when point wise log loss used, with the increase of negative sampling, model performance had improved. Finally, as the number of MLP layers got increased the model performance got increased, showing how important to use deep learning in recommendation tasks.

The future work includes use of user reviews, knowledge bases and temporal signals to improve recommendation tasks. Furthermore, to extend the recommendation task for group of users. Use of multimedia information to build multi modal recommendation is an another research area. Finally use of hashing and recurrent neural networks to do recommendations.

Finally, restricted Boltzmann machines [37] were introduced to handle recommendation systems problems that contains large number of data sets. Hence, this model was evaluated using Netflix prize data and root mean error square was used as the evaluation metrics to evaluate the solution. The model performance was compared with Singular Value Decomposition. First, the model performance was studied with the increase of epoch. As the number of epoch got close to 0, the RMSE had reached a value near to 1. However, as the number of epochs got increased, the RMSE had dropped and steadied at a value of 0.9. Similar, pattern was observed for RBM with hidden Gaussian units and conditional RBMs.

2.4 Content Based Recommender Systems

Content-based filtering uses previous history to recommend products to users. These can be user previous actions or likes on a product.

Advantages of Content Based Recommender Systems

1. Models are easily scalable as they only require particular user data only.
2. These can be used to capture interests of users and recommendations.
3. Can provide recommendations to all the users as well.

.

Disadvantages of Content Based Recommender Systems

1. Require good domain knowledge as the features are engineered manually by the experts.

2. The models can suggest only few items to users as only limited data is available specific to a user to train.

Deep Knowledge-Aware [38] recommendation systems are originally introduced in order to improve news recommendations to users. This model uses knowledge graphs to train itself. A knowledge graph can be created using entities. As a result, this model has the ability to learn complex patterns in the news articles. The model also has a convolutional layer, where it can take news title and generate embedding from it. In order to aggregate users' click events and news title an attention based mechanism was used. In order to evaluate the model, data was collected from BING web site logs where the data contains user information news titles, url and time they are accessed as a timestamp. In order to compare the model results LibFM [39], KPCNN [40], DSSM [41], DeepWide [42], DeepFM [43], YoutubeNet [44] and DMF [45] was used.

The first observation in the experiment was that entity embedding had helped improve model performance. For KPCNN, DeepWide and YoutubeNet, the improvement was around 1%. However, for the FM based models, introducing entity based embedding do not help improve the performance. Amongst the models, DMF performance was worse, because it does not perform well when time cycles were less. All deep learning models outperformed LibFM, except for DMF, in terms of F1 score and AUC, showing how effective deep learning models are to learn nonlinear functions. Although architecture of DSSM is similar to DeepWide and YoutubeNet, it outperformed these models because it internally uses raw text with hashing internally. KPCNN had outperformed all models because it uses CNN architecture internally. DKN had outperformed KP CNN mainly due to the fact that, it internally uses word entity based learning internally and uses attention based model to learn user interests.

Next an experiment was performed to find how effective was to introduce new components for the DKN models. Introduction of word and contextual level embedding had helped improve AUC of 1.3% and 0.7% respectively. DKN+TransD had performed well over all models because this model use word and sentence level embedding as well as entity knowledge graph information. Finally, introduction of attention networks brings AUC and F1 gain of 0.9% and 1.7% gain. The studies on word embedding size indicates that with the increase of dimension, the model performance improves. However, further increase of dimension decreases performance as the noise introduced to the data.

Furthermore, for KPCNN and DKN, as the number of filters increases, the model performance improves. However, for larger filters the model displays low performance.

Gradient boosting methods are widely used in machine learning because their reputation to scale well as well as produce good results on data sets. However, these models do not perform well when the dimension of the feature are much higher. In order to resolve this problem, LibFM[46] was introduced. The main idea of this model to remove large amount of the data set that has less effect on gradients.

In order to evaluate this new model five data sets were used. First data set contains search queries which is obtained from Microsoft ranking data set. Two data sets contains Insurance claims and flight delay information. Other two data sets were obtained from KDD cup. Here the researches have used features used by winning solution to better test their solution. Area under curve and NDCG was used to evaluate the models.

In order to compare model performance baselines created from XGBoost and LightGBM was used. For XGBoost models, two new models were created xgb_exa and EFB created from lgb_baselines. For the xgb based on histograms and lgb base lines and xgb exa model layer based strategy for growth was used. The first observation indicates that the LightGBM could converge faster to solutions compared to other models. However, its accuracy was similar to its baselines. Compared to the LightGBM model without gradient based one sided sampling the models with GOSS strategy had two-time training time improvement. Furthermore, exclusive feature bundling had helped improve training time on larger data sets which are sparser in nature. The further improvement included in this model to improve feature bundling strategy to deal with sparser and larger data sets.

Recommendation systems are widely used for news recommendation. LUTER[47] is a model, that can learn user representations regardless of the fact that it is long or short term. Inside this model, an encoder is used to learn news representations using different news titles. User encoding are learnt using the user ID.

In order to evaluate the model, a data set created from MSN website log was used. To compare the accuracy of the model, LibFM [39], DeepFM [43], Wide and Deep [48], DSSM [41], CNN [49], DKN [38] and GRU [50] models were used.

The first observation was that, deep learning models, such as CNN, DKN and LSTUR had outperformed models that uses manual feature engineering such as LibFM, DeepFM, Wide and DSSM. Moreover, LSTUR had managed to outperform CNN, GRU and DKN as the model is capable of better capturing short term and long term interests of the user. In order to find out effectiveness of using LSTER model long and short term user representation, the model performance was compared with models that uses long term and short term user preferences separately. This includes STUR model. The results indicate that the LTUR and STUR performs better and STUR had outperformed LTUR. Combining these two models together could improve model performance as well.

Next it is explored the effectiveness of using GRU model inside the STUR model with three encoders. These includes, average of all news browsing of the user, addition of all news representations weighted by their attentions and replacing GRU with LSTM. The sequential models had outperformed the averaged based models. Furthermore, GRU model had reported better results over the LSTM models. Next, news encoder was integrated with CNN and LSTM models and their performance was measured. Hence a performance improvement was achieved as well as CNN had outperformed LSTM. Similarly introducing topic and sub topic encoders had helped improve model performance as well. Moreover, randomly masking with probability for long term user representation had helped models improve their performance as well.

Neural News Recommendation with Attentive Multi-View Learning [51]. is a model, that tries to learn a good user and new representation from the given dataset. Compared to traditional models which only use title to learn a better representation of news, this model uses news body, topics and using different categories. Similar to LUTER model this model also uses two encoders to effectively learn better user and news embedding. This model has a user and news encoder. The news encoder has body encoder, vert encoder and a subvert encoder. The title encoder and body encoder are CNN based and learn news title and body representations using semantic information in the word. After learning these embeddings, an attention network is used to aggregate those vectors.

To evaluate the models a real world data set was used which consists of different users, news and user impressions. The data set was collected from one month starting from December 13, 2018. To compare the model performance baselines such as LibFM [39], DeepFM [43], Wide and Deep [48], DSSM [41], CNN [49], DKN [38] and GRU [50] models used. During the experiments, the size of the embedding was set between 300 and 1000. The number of CNN filters used was 400 and window size was set to 3. The negative sampling rate was set to 4 and 20% of dropout was set. The batch sizes were set to 100 and hyper parameters were selected using the validation data set.

First the model performance was compared with the other models. It is observed that the new model which is based on deep learning had outperformed traditional models which is based on matrix factorization. Moreover, models that used negative sampling had helped models improve performance as DSSM and NAML had reported better results over CNN, DFM and DKM. Finally, the new model had out performed all other base line models. Furthermore, the new model uses attention based mechanism to recommend news to users. It is found that, using news body is very useful in modeling news topics compared to using category and titles. Hence, combining all three had helped models improve their performance a lot.

Neural News Recommendation with Personalized Attention [52] which is capable of learning representations of users and news by using user viewed events and using news titles. NPA uses CNN to learn hidden features in news articles. Users are learnt based on

different articles users had clicked on. Moreover, word level news level personalized attention is used to capture different information useful to users.

In order to evaluate the model, data extracted from MSN logs were used. While this contained one-month data, the last week of data was used as test data while the others were used for model training. To compare the model performance baselines such as LibFM [39], DeepFM [43], Wide and Deep [48], DSSM [41], CNN [49], DKN [38] and GRU [50] models were used. Mean reciprocal rank, AUC, nDCG@5 and 10 (Normalized Discounted Cumulative Gain) was used as the evaluation metrics.

The first observation was that, CNN, DSSM and NPA which is based on neural networks had outperformed matrix factorization methods. Second, the models that uses negative sampling such as DSSM and NPA had outperformed models that do not use negative sampling such as CNN, DFM and DKN. Third, deep learning models that uses attention mechanisms such as DFM, DKN and NPA had outperformed ones that is not such as CNN, Wide and Deep and DeepFM. Finally, the new model had outperformed DSSM because it uses contextual information more effectively and use more negative sampling. It outperforms DFM because, new model uses context information, and it can better select important words than DKN which uses attention mechanism to select important words.

In terms of the computational cost, comparing the new model with feature based methods, the computational time is low, if the number of training samples are low. Comparing the new model training cost with CNN, DFM and DKN, the training cost was reported by dividing by a constant K as the model uses negative sampling. More over in the testing phase, the new model cost was lower than DKN. This is because DKN uses news level attention and scores were predicted by encoding all news clicked by a particular user. DFM had recorded a higher computational cost, as it requires a sparser input. Next NPA performance effectiveness was evaluated based on days. The observation was that, on the first day the model performance was at highest, however as the time progresses, the model performance has gone down. The reason is that; user preference is based on neighbor days. Finally, the effectiveness of model personalize attention mechanism was evaluated. To do this NPA with attention mechanism had compared with the vanilla version of the NPA. Hence, the NPA with attention had outperformed all other models, because attention mechanism can select words with importance. Moreover, the personalized attention mechanism could has proved to be effective as it could model in formativeness of the words based on the user. Similarly, using personalized attention on word level and news level had helped models improve their performance. In order to validate the effectiveness of using negative sampling, NPA model that uses negative sampling was compared with a one that uses negative sampling. It is observed that negative sampling had helped new model improve its performance. Furthermore, if the amount of negative sampling was too

high or low, the model performance was decreasing. Hence the suboptimal negative sampling found to be 4.

2.5 Hybrid Recommender Systems

Data sets used to train recommendation systems are very sparse in nature. This is because, given a user there will be many products in the utility matrix that has 0 interactions. Many collaborative filtering methods generates latent variables from this utility matrix and in return this encoding to predict user item preferences. However, this data sparseness has negative effect on the model performance. That is model will not be able to predict rating for a product which is unseen in the training data set resulting cold start problem. One way to solve this problem is to use deep learning to effectively learn latent factors [54]. Here the concept is simple. Use side information to learn embedding for the users and products and using collaborative filtering. These information is used for recommending items to users.

To evaluate the method movielens and Book Crossing data set was used. RMSE and recall@k was used as the evaluation metrics. Probabilistic Matrix Factorization [55], Collective Matrix Factorization [56] and Collaborative Deep Learning [57] was used to compare the new model performance. Movie lens 100k, 1M and Book data set was used to train models. In order to train the models, for each model, only 60%, 80% and 95% of data was used. The training data set was selected randomly while the rest was set as the test set. The experiment was running for few iterations and the average performance was reported. Furthermore, masking was used to get raw input from the corrupted data sets. The number of deep layers was selected as 4 and latent factor was set to 64.

The comparison of models indicates that hybrid models such as CMF, CDL, DCF and the new hybrid model had achieved better performance. Moreover, CDL, DCF and the new hybrid model had outperformed PMF and CMF models. Hence, this suggests that the deep learning models can learn better quality features. Considering recall, PMF had recorded worst performance due to lack of side information. Compared to the new hybrid model, the performance of CDL and DCF was low as due to same lack of side information problem. The reason for higher performance in hybrid model was due to the use of aSDAE model using internally. This model can learn better latent factors from much sparser user item matrix. The future work of this includes replacing aSDAE with deep learning models such as recurrent and convolutional neural networks.

Wide and Deep Learning [58] is another hybrid recommendation system model, which is capable of accepting sparse inputs as well learning generalized functions. This new model had two advantages over others. That is, it can learn generalized functions better as well as it can remember frequently access products or items. This new model has two distinct

features. One is that, neural network with feed forward ability was trained to accept sparse inputs. The model was evaluated data obtained from google play store and it was implemented using tensorflow.

In order to train the model following procedure was applied. Given the input data the model generates features which are sparse and dense in nature. The model has a deep learning component that accepts input and learn an embedding size of 32 for each given category. Then these categories are combined together to formulate an embedding vector size of 1200. This vector is the given as an input to 3 Relu layers and for one logistic unit. In order to avoid model retraining time every time a new data set comes the new models were initialized using previously trained model weights and embedding.

In order to evaluate the models, both online and offline methods were used. For the online method, A/B testing framework was applied for a 3 weeks. For the 1/% of the users, the produced recommendations were provided. The model managed to successfully improve app acquisition +3.9 on app store. Considering offline experiments, wide and deep had reported a higher AUC as well.

2.6 Outlier Detection

Anomaly detection is formally defined as finding data objects where their behavior is different from the expected behavior. Outlier detection can be extremely important in the context of credit card fraud detection, medical applications and in intruder detection. Outliers can be detected using statistical methods, proximity based and clustering based methods. Types of outliers includes, Global, contextual and collective. Global outliers deviate significantly from the rest of the data points and measurements of deviations are used to find these. Another form of outlier is called contextual outliers where, these types of outliers defined as objects where their values changes significantly based on the context. These kind out outliers are found using contextual and behavioral attributes. Collective attributes are defined as group of points where their behavior significantly change from the others.

The challenges of outlier detection as follows. The first thing is modeling normal objects. This is challenging as it is difficult to model normal behaviors of the application. The next one is defining outliers based on the application. However, defining global model for finding outliers based on the application can be challenging because based on the application, detecting the outlier can be different. Another problem is how to remove noise from the data set. Outliers are different from the noise. Noisy data points which are low in quality and they may be amongst the outliers making outliers detection inefficient. Another requirement is understandability where, users may need to know, why data was classified as outliers.

There are many outlier detection algorithms. They are based on supervised, unsupervised and based on semi supervised methods. In supervised learning domain expert define labels where which defines what is anomaly and what is not. In most cases, outlier data are much lesser leading to class imbalance problem. In unsupervised methods, there are no data points classified as normal and erroneous. Here, it is assumed that once the classification is applied, the normal and error data points will be group separately autonomously. One problem in this method is that, for the large amount of data points, this method can become very inefficient. In semi supervised learning, there will be some amount of data that is classified as outliers. Here if the data points which are closer to the normal label data points are considered as normal data. However, if the label data points contain outliers, then it will be trickier as number of outliers are very smaller.

Anomaly detection in multi variate time series is a very important problem in data mining. One of a major drawback in the existing methods is not capturing relationships between different time series. As a result, number of false alarms raised by the recommender system will be much higher. As a solution, a model based on unsupervised learning was introduced [80] where, univariate time series was considered as a single feature. Moreover, two graph based attention layers were used to learn relationships between two multi variate time series in both temporal and feature dimension.

The contributions of this research as follows. A novel framework was introduced to solve multi variate anomaly detection with F1 performance improvement over 9%. Moreover, two parallel graph attention layers was used to capture relationships between different time series. A joint model of forecasting and reconstruction based models were used to find anomaly in the data set and the network had displayed a good interpretability. That is time series that learnt by graph attention layers are well understood by humans.

In order to evaluate this new solution, 3 data sets were used. They include Soil Moisture Active Passive Satellite, Mars Science Library rover, and Time series anomaly detection data set. Precision, recall, F1-score and AUC was used to evaluate the model performance. Five models were used, namely OmniAnomly [81], LSTM-NDT [82], KitNet [83], DAGMM [84], GAN-Li [85], MAD-GAN [86] and LSTM-VAE [87] for the result comparison purposes. The results indicate that, the new model had achieved a performance improvement over 1%, 1% and 9% improvement in SMAP, MSL and TSL data sets respectively. The reason for outperforming OmniAnomly model is because the new model is using graph attention layers. The model had managed to outperform DAGMM because it uses GRU to capture temporal dependencies. The LSTM-NDT perform data that have time dependent information where they perform poorly on MSL and TSA data sets. However, models such as OmniAnomly better perform on this data implying that forecasting and reconstruction methods can help models improve performance. The anomaly occurs in continuous data streams. To find how effective the new model can detect anomaly, it is compared with OmniAnomly with different delays.

The new model had achieved best performance when delay was at 10 and the F1 score had improved dramatically when the delay increased from 5 to 10.

To find effectiveness of using a graph attention layers feature and time oriented layers were disabled one at a time. Then model parameters were adjusted to remove the complexity. The result was when feature oriented layer was disabled the F1 score dropped by 3.2% and for time oriented layer this was 2.5%. For the TSA data set this was around 4.8%. Then it was studied the effectiveness of using joint optimization techniques. It is found that, reconstruction model is effective at capturing the global performance but not effective to disturbances in the time series. However, in this case forecasting model had managed to find anomaly in the data set.

The future work in this model includes using prior knowledge to find correlation of the features and test the model on more complicated test data sets.

Furthermore, services were created by cloud services to monitor time series continuously and detect incidents on time [88]. In this case new pipe line was created where the major modules include data ingestion, experimentation and online compute. To detect anomalies algorithm based on Spectral Residual and Convolutional neural networks were used.

Three data sets were used to evaluate the model. KPI data set which contains KPI curves with data with anomaly labels, and yahoo and Microsoft data set which is used to detect anomaly detection. To find the accuracy of the model, F1 score was used while for efficiency time taken to execute the pipe line was used. To evaluate whether the model generalize better, yahoo data set was divided into three major classes and tested. These classes include seasonal, stable and unstable. The first experiment was carried out to find usefulness of using SR-CNN module. Hence the model was compared with Fast Fourier transform [89], Twitter-AD [90] , Luminol [91], DONUT [92] and DSPOT [93]. To evaluate the solution, F1-score, precision, recall and CPU execution time was used. The SR-CNN model reported an improvement of 36.1% F-score improvement on KPI data set and 68.8% and 21.2 on yahoo and Microsoft data sets respectively. Furthermore, in terms of efficiency and generalization the SR-CNN had outperformed the other models. Then the same experiment was carried out with DNN a supervised based method to see whether the solution outperform the other models. L1 and L2 regularization was set to 0.0001 and dropout for 0.5. The number of layers was 2 and as for the train and test split, the data set split was performed so that, train and test data set has 50% each data points and they have equal number of anomaly data points. Again the results indicates that the new solution had outperformed the other models.

2.7 Evaluation Methods

There are many evaluation methods available to measure performance of recommendation systems [59]. These can be categorized into 3 main types. They include, rating metrics, ranking methods and classification metrics. Rating metrics includes, Root mean square error, R squared, mean absolute error and explained variance. These all methods are a measure of, error in predicted ratings.

Nowadays popular ranking methods such as Normalized Discounted Cumulative Gain, Hit Ratio, novelty, and diversity is used. Hit ratio method measure, out of recommended items how many have user have actually rated as a hit. Normalized Discounted Cumulative Gain measure how well predicted items for a user ranked based on relevance and mean average precision used to average out precision for a series of performance measures.

Classification metrics includes, Area under curve and Logistic Loss. These metrics are used mostly when data set includes binary classification data or when data set is highly imbalanced.

Recommendation systems can be evaluated in offline and online settings. In online settings, the model performance is measured based on their predictions in real production environments. This is called A/B testing. We can do a simple A/B testing by recommending products to real users using our recommendation system model and then ask them to rate recommendations. In production environments, users' intents may change more frequently, introducing a bias into the system. To overcome this problem, online evaluation settings can be used.

The offline testing is basically to compare models with our model and to select the best one based on the performance. Furthermore, offline testing is less expensive to perform. This is because we start evaluating algorithms online we may have to be ready to address production issues that might come online, allocate necessary hardware and other resources to train model online which will be quite expensive. There are disadvantages as well. In offline settings we need to ensure that our test and training data set has the same distribution. Furthermore, our test data set must be similar to the distribution we may get in real practical settings. This is because, our recommendation system will become successful only if it produces best out comes in production settings.

2.8 Hyper Parameter Optimization

Hyperband [61] is a novel framework, that is been especially been introduced to improve performance of machine learning models. Recent approaches use Bayesian optimization to adaptively select configurations. This technique is based on random search and early stopping. It is very important to find hyper parameters as they have a profound impact on model accuracy as well as on generalization. However, it will become difficult to find better hyper parameters with the growing number of parameters as the optimization will take more time to execute. Hyperband is a configuration based method which is much more efficient than random search.

There are many advantages of using HyperBand. First, it depends minimally on previous configurations. As the procedure depends on early stopping, it can evaluate more configuration compare to Bayesian optimization.

Furthermore, the performance of the hyperband depends on the prior knowledge of the task, as this will be helpful for early stopping. Finally, considering resource dependent hyper parameters, the setting for a given hyperparameter should depend on the allocated resource. This procedure can reshape itself to unknown convergence as well as it is much faster than Bayesian methods.

In most optimization techniques, early stopping is applied in a way, that once the model start reporting low values on a subset of the validation data set that start reporting a poor performance. However, the hyperband work differently. It trains models on sub set of the training data and evaluate on full data set. The aforementioned techniques are based on strong assumptions or heuristics which is a disadvantage.

In order to evaluate this new solution MINST and CIFAR images data sets were used. The results indicate that the hyperband converges to solution much faster for the data sets with high dimensional spaces.

Furthermore, hyper parameter optimization techniques were introduced, particularly only to neural networks and for deep belief networks [62]. The more traditional methods, can be used for fewer hyper parameters because they require human intervention. The new method uses random search and greedy methods to search for better configurations as it helps to improve the performance of the models. Formally, the configuration set can be represented in a graph structure. Then different configurations are selected and based on these, loss function is selected based on the optimal performance. However, due to limitations, the configuration space is restricted for a tree space.

This solution has two key significant contributions. That is to use random search over manual methods making models use more configurations to search better parameters. Furthermore, this new method had reported better results over the manual techniques. In

order to improve the optimization criterion, the model internally uses Gaussian and Parzon methods based on tree structures.

In order to evaluate the result, the researches have compared the random search with sequential methods. In the experimentations, they have used ZCA preprocessing, used layers with the different sizes and transformed continuous data to Bernoulli values. For the deep belief networks, the manual search had outperformed this new method. However, for MINST data set, the new solution had outperformed the manual search.

2.9 Deploying Recommendation Systems

For the e learning platforms recommendation systems are used to recommend courses to users. In this context, it is ideal to use cloud resources to implement recommendation systems [63].

Here we will discuss about a recommendation system architecture implemented in cloud with scalability and better response time and performance.

First we will discuss about the service layer of the model. This layer is responsible for accepting user requests and recommend learning products to users. This layers stores user consumed learning resources as well as their present state. For this a meta data driven approach has been used. A separate layer called client is responsible for updating users' current state in the service layer. The service layer includes a separate function, which can be used to create a user profile which consists of user present state, learning resource utilization as well as current state given the user, these learning resources.

Another key feature of the service layer is that it has a recommendation system cluster. There is a benefit in using cluster approach. The first one is that, recommendation system is easy to scale and this is essential because, course recommendation task is considered to be very time consuming task. Internally, this cluster uses, user based collaborative filtering which once the recommendation task is completed, similar users with same resource usage patterns are selected and courses are recommended accordingly. The similarity function of two users are defined as learning material access by count when they are at similar states. Once the learning materials that need to be recommended are found then they are sorted based on their relevancy and with respect to the user neighborhood group.

Next we will look at client layer. This layer is responsible for filtering recommended courses for users for adaptation purposes. This component is responsible for sending user identification details as well as any information that will be responsible for changing effective state, to the service layer. Furthermore, this model is responsible for tracking user activity. Client layer calls for service layer when it needs to suggest courses to user when user first login or when user update his or her state when required, where once these actions are performed totally new resources are predicted and recommended to the user.

The result is displayed in the format of title and its description. A typical user may select required resources based on the relevancy,

Here the model was deployed in AWS cloud environment. AWS EC2 instances were used to deploy web applications and Apache Mahout was used to build the recommendation system. Here in order to make recommendation system faster map reduce technique was used where the map is defined as for a given user effective state and the course resources. The client layer was implemented using java script.

Below, is a figure which depicts, the client interface provided to the user.

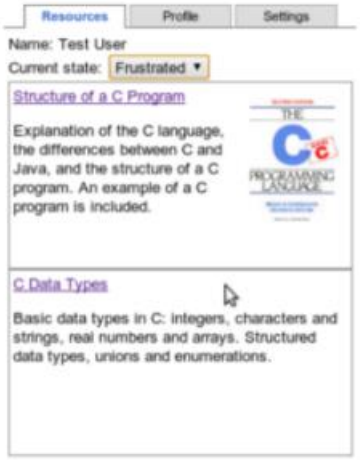


Figure 2.4: User interface of the learning system

One main problem in this architecture was that it cannot cache the results. This is mainly because the model was using a distributed architecture for computation. Another problem is storing user state had raised privacy concerned in the system.

2.10 What is Bias in Recommendation Systems?

Bias can occur in any stage of machine learning development process [64]. Using bias information to train machine learning models may result in models that produces incorrect results. Using such models in applications, and using in real systems may results in unwanted consequences. The problem here is if the data is biased, then the trained model will not be able perform well on unseen data resulting in poor generalization. Data is only a one main source of bias. However, if we think about machine learning pipeline we are required to make a series of decisions. One or combinations of these actions may also lead to unwanted bias as well.

Let us first look at a typical machine learning flow. Here first we may collect data. We may use different sampling techniques based on our problem. For example, if we are dealing with a class imbalance problem, we may sample more positive class. Otherwise, in an ideal case, all classes should have an equal representation in the data set. In the next step we may derive features from the data set. We may do a preprocessing to remove outliers. We use feature engineering to select features that has more relationship with our output variable. Then, we may use fivefold cross validation to train out model and use test data set for validation. We can use F-score for comparing results. Hence for example, not using proper sampling techniques, removing outliers, or proper training or evaluation metrics can lead to bias decision.

2.10.1 Types of Bias and Mitigation Techniques

Next we will discuss about different bias that can occur in system and how to mitigate them. One common source of bias is through historical data. Historical data may become misleading. For example, our data may have many data examples, which depicts that increase of gift card sales of a particular brand due to its low price. However, the real reason could be because, it is an effect of seasonal change.

Furthermore, the bias may have occurred based on the representation. As a simple example our data set may have more males over females. This may be because our data sample may not have generalized with population or may be because selected population itself have more males over females. The result is that our learning model prediction will be more biased towards males over females. In a case of food data sample males may prefer more meat related menus and females may prefer vegetable food. In this case the model may give more preference for meat related foods over vegetable products.

Measurement bias occurs when the features we have selected do not best explains the classes in the data set. This is may be because, there may be a confounding variable which explains the relationship between input and output variable. Outliers can be an another reason for a poor feature selection. Moreover, measurement bias can occur if we do not control our experiment and do not use same techniques to get out measurements. For example, in an experiment where study of worker's error classification, if we monitor errors of a one group more then, this error will be more biased to this group. Furthermore, data quality depends on different groups. For example, mature people may do less errors over the others. Finally, it may be because we have over simplified our classification task. For example, in this case we may erroneously use age as an attribute which explains worker errors.

Aggregation bias is another form of bias which can occur where different groups in the data sets are not correctly represented. This can occur because we have used in correct sampling techniques or may be because in the population groups are not well represented.

Finally, evaluation bias may be a cause because the data set we use for benchmarking purposes do not represent our classes in the dataset correctly. As a result, some models may over fit and may lead to poor generalization.

To mitigate aggregation bias we can train different groups simultaneously and take the average output as the prediction. This is similar to multi task learning. Similarly, to mitigate evaluation bias we can use separate metrics to separate groups.

Apart from bias, mentioned above, bias can occur in following ways [65]. One form of such bias is defined such as omitted bias. In these types of biases, the features that best describes the output variable is left off. Another form of bias is Linking bias. Here for example, let's say a user is categorized into a group based on his or her activity. In this case the attributes used to derive users' behavior many not best describe a particular users' behaviors. Another form of bias which is very interesting is algorithmic bias where bias is introduced because of the choosing of wrong model, wrong hyper parameters or training or test procedures. Furthermore, biases can be introduced into system due to users' behaviors. This is mainly because how the data is shown or presented to the user. Biases can be introduced into data sets because it contains false information, which is then used to increase the popularity. Finally, the bias can occur inside user groups. For example, different users may have different choices based on their background within the groups.

Furthermore, bias such as positional bias can cause problem in selecting or interacting with products. Studies has been carried out to find out the relationship between click through rate and ranking of articles in online digital libraries [73]. The results indicate that users prefer to click more articles that has higher relevance score over the other. Initially, over 1.6 million article sets were available, where each containing 6 recommendations. Then the researchers have delivered over 10 million recommendations to the users and collected over 12000 clicks. Out of 10 million article sets half of sets were shuffled before presenting them to the users. Then 1% of recommendations were chosen for evaluation purposes from the sets randomly. Then a chi square was performed for total average click through rate for shuffled articles and those are not. Hence, this test has revealed that there is a significant relationship between item rank and its click through rate. The values indicate that there is a significant relationship.

Furthermore, in order to deal with popularity bias, regularization based frameworks were introduced [68]. This framework is based on matrix factorization and it is argued that, the objective function will reach to a minimal value when the recommendation is fair.

In order to evaluate the new model, 1 million movie lens data set and consumer movie data sets containing user opinion was used. Then in the movies data set, movies having ratings lesser than 30 were removed from the data set. Then for the both data sets short head items were defined where for the movie lens data set it was movies where their number of ratings was greater than 400 and for the other data set the threshold was 46

rating counts. In order to compare model performance Pop, ALS and ALS with regularization was used. NDCG@10 and APT@10 was used as the evaluation metrics. For all cases ALS with regularization reported best performance followed by ALS and Pop. One key finding was that with the increase of regularization weight, ALS with regularization had recommended products which are in the tail. This was observed for the both data set.

Personalized re ranking is an another model which was introduced to deal better with popularity bias [70]. Here in this model, a preprocessing is applied to make sure items with low ranks will be pushed higher. Here in order to move items with low ratings, a control parameter called lambda was used. In order to evaluate the solution, 1 million movie lens data set and consumer movie data sets containing their opinion was used. As a preprocessing step, the products with rating lesser than 20 was removed from the data set. In order to evaluate the model, ARP, ACLT and APLT metric was used. Five-fold cross validation was used to train the model. Two baseline models were used to evaluate the new solution. These includes, RankALS [68] and LibRec.

Considering consumer opinion data set, as the weight of the items with low ratings increased, APLT@10 and ACLT@10 had increased. This is an indication that, items with low ratings were recommended to users. This was not true for NDCG@10 and ARP@10. As the value of the weight which controls low rating values increased, the values started to go down. Here for APLT@10 and for ACLT@10, the reported values were between 0 to 0.35 while for NDCG@10 and for ARP@10 it is 0.09 to 0.015 and 300 to 550 respectively. Similar pattern was observed for the movie lens data set. These results indicate that, as the weight assigned to low rating values increased, they get more recommended.

Bias exists in the data sets. However, for the collaborative filtering domain, new strategy been introduced to counteract bias [71]. This paper has three significant contributions. First it has introduced a probability likelihood based mechanism to test user had seen a particular item. New framework was introduced to remove bias from recommendation systems. They have also developed a test framework to simulate user product interaction and track its effect using real data set and evaluated using newly introduced debiasing strategy.

In order to evaluate the new solutions movie lens 1m data set was used. Expected novelty, diversity and hit ratio was used as the evaluation metrics. The newly introduced model performance was compared with Matrix Factorization, Propensity-MF [72], Naive Multi-Armed Bandits Strategy + MF (MAB) and Naive Multi-Armed Bandits Strategy + PEAR-MF. Considering expected novelty, the proposed model had out performed all others. Here number of iterations were increased and as the numbers increased the expected novelty was calculated. For all cases, PEAR-MF had reported average value of 0.8. This is followed by 0.7, 0.7 and 0.6 for MF, MAB + PEAR-MF, and MAB+ MF respectively.

Considering diversity, with the increase of iterations, PEAR-MF performance decreased and recorded and average value of 0.15. Same was observed for the other models as well. The performance of MAB + PEAR-MF was similar for all iterations by margin, while matrix factorization and MAB+MF recorded performance which 0.05 lesser that PEAR-MF in most iterations. Furthermore, for all iterations Propensity-MF, had recorded highest diversity values compared to PEAR-MF.

Finally, for the Hit Ratio, PEAR-MF had outperformed all other models. Its Hit ratio had decreased, with the increase of iteration and ended up in a value close to 0.4. Same was observed for the other models as well. The performance of MAB + PEAR-MF was similar for all iterations by margin. In this case propensity MF, it had recorded the lowest Hit ration in this case near for 0 in all iterations. The Hit ratio recorded by MF and MAB+MF and they were between 0.3 and 0.4.

3 METHODOLOGY

3.1 Problem Verification

In order to conduct the research, I have selected e commerce data set which was published in kaggle. The data set had 96% of the view events. 3% of addToCart events and only 1% of purchased data.

The e commerce data set used to conduct the research can be found in following url (Retail Rocket Data Set):

https://www.kaggle.com/retailrocket/ecommerce-dataset (event.csv)

The proposed method consists of two main steps. They are problem verification in the data set and the solution.

Step 1 : Problem verification

The first task I have carried out is to verify that the problem exists in the data set. In order to verify this, I have carried out following tasks.

1. I have first created 10 data sets.
2. The first five data sets considering products which has view, purchased and add to cart events.
3. Then I have grouped based on the product and user.
4. Then I have found out that how many times that a given user had viewed, added to the cart and how many times he had purchased it.
5. Finally, I have added weights based on the fact that whether the item was viewed, purchased or added to the cart. Below table shows weights I have assigned.

Table 3.1: Events and assigned weights

Data set	addToCart (w1)	Transaction(w2)	View(w3)
1	1	1	1
2	2	1	1
3	1	2	1
4	1	1	2
5	3	3	3

Then I have used following equation to derive a score for each user and product records.

Score = w1*(addToCart counts) + w2*(transaction counts) + w3*(view counts)

6. I have created the next five data sets varying the amount of addToCart, transactions and view events in the data sets. Below table shows how I have varied the event amounts. Please note that, the data set 7, 8, 9 and 10 has a large amount of view records.

Table 3.2: Events and added data volumes

Data set	addToCart (w1)	Transaction(w2)	View(w3)
6	Greater than 2	Greater than 1	Greater than 40
7	Greater than 3	Greater than 1	Greater than 15
8	Greater than 2	Greater than 3	Greater than 15
9	Greater than 3	Greater than 3	Greater than 15
10	Greater than 4	Greater than 4	Greater than 10

7. Then for each data set, I have defined the test data set. I have randomly selected 20% of each data set and created this test data set.
8. Next, I have selected 6 models to verify the problem. That is when there are products with high amount of views, products with purchased events are not recommended to users. These models include, Bayesian Personalized Ranking, Neural Collaborative Filtering, Generalized Matrix Factorization, Multi-Layer Perceptron, Most Pop and Item KNN adjusted.
9. I have used five-fold cross validation to train the models.
10. To initialize models, I have used best values reported in the research. Please refer Table 3.3.

Table 3.3: Parameters used initialize the models

Model	Parameters	Evaluation Metrics
Bayesian Personalized Ranking	Number of latent factors: 50, Epoch: 200, Learning Rate : 0.001, Lamda Reg : 0.001	HR@10, NDCG@10
Neural Collaborative Filtering	Embedding size : 8, MLP Layers : [64, 32, 16, 8], Activation function : tanh,	HR@10, NDCG@10

	Optimizer : Adam, Number of Epoch : 10, Bach size : 256, Learning rate : 0.001, Number of negatives : 50, Seed : 123	
Generalized Matrix Factorization	Embedding size : 8, Number of epoch : 10, Learner : Adam, Batch size : 256, Learning rate : 0.001, Number of negative : 50, Seed : 123	HR@10, NDCG@10
Multi-Layer Perceptron	MLP Layers : [64, 32, 16, 8], Activation function : tanh, Optimizer ; Adam, Number of epoch : 10, Batch size : 256, Learning rate : 0.001, Number of negative : 50, Seed : 123	HR@10, NDCG@10
Most Pop	None	HR@10, NDCG@10
Item KNN	Number of nearest neighbors : 50, Similarity : cosine, User item rating metric centered : true,	HR@10, NDCG@10

11. Finally, once the models were trained, I have recorded HR@10 (Hit Ratio) and NDCG@10 (Normalized Discounted Cumulative Gain).

    The values indicate that in the present of view events, products with purchases were not recommended to the users.

3.2 Experiment Detail And Solution

1. Next I have removed data records containing view events from the data set.
2. Next I have grouped the data set by user id and product id and found number of occurrences of addToCart and purchase events.
3. Then I have used weights mentioned in Table 3.4, so that the products that has more purchases will get a higher score.

Table 3.4: Events and assigned scores

Event	Score
View	0
AddToCart	1
Purchase	2

Then I have used below equation to derive the score.

Score = 2*(purchase events count) + (addToCart event count)

Then I have used binning to bring the score in between value 1 and 5.

4. Next I have created the test data set by randomly selecting 20% of the data set .
5. Finally, I have trained five models using five-fold cross validation and used test data set to measure the Hit Ratio and Normalized Discounted Cumulative Gain.

6. Table 3.5 – 3.12 shows parameters used to initialize the models, changed parameters and results obtained.

For the experiment 1, I have recorded how model performance changed in terms on HR@10 with the learning rate. In order to initialize the models, I have used parameters specified in Table 3.3. The table 3.5 shows, how learning rate was varied for each model and how HR@10 was recorded.

Table 3.5: Model Learning rate change and recorded HR@10 and NDCG@10

Model	Learning Rate	HR@10	NDCG@10
Bayesian Personalized Ranking	0.1	0.335	0.022
	0.01	0.98	0.0166
	0.001	0	0.0178
	0.0001	1	0.0175
Neural Collaborative Filtering	0.1	0.28	0.013
	0.01	0.21	0.0123
	0.001	0.32	0.0168
	0.0001	1	0.0176
Generalized Matrix Factorization	0.1	0.19	0.0111
	0.01	0.23	0.0129
	0.001	0	0.0176
	0.0001	0	0.0126
Multi-Layer Perceptron	0.1	0.03	0.0086
	0.01	0.255	0.0057
	0.001	1	0.0166
	0.0001	1	0.0172

In the second experiment, I have changed the number predictive factors and reported HR@10 and NDCG@10 for the models. Moreover, I have used parameters specified in Table 3.3 to initialize the models. The table 3.6 shows how models recorded HR@10 in response to the change of number of predictive factors.

Table 3.6: Model Predictive factor change and recorded HR@10 and NDCG@10

Model	Predictive Factors	HR@10	NDCG@10
Neural Collaborative Filtering	8	0.34	0.0156
	16	0.215	0.0186
	32	0.26	0.018
	64	0.23	0.0201
Generalized Matrix Factorization	8	0	0.0182
	16	0	0.0184
	32	0.35	0.0186
	64	0.74	0.0177

In the third experiment, I have changed the top K recommendations where K is the number of recommended items and reported Hit Rate and NDCG for the models. Moreover, I have used parameters specified in Table 3.3 to initialize the models. The table 3.7 shows how models recorded Hit Rate in response to the change of number of K, where K is the number of recommended products.

Table 3.7: Top K recommendation evaluation results and NDCG (When K changed)

Models						
Top K	BPR	NCF	GMF	MLP	ItemKNN	Most Pop
1	0.0111	0.0101	0.0104	0.0109	0.0091	0.0112
2	0.0108	0.0107	0.0109	0.0109	0.0126	0.0112
3	0.012	0.0116	0.0124	0.0125	0.0164	0.0125
4	0.0134	0.0129	0.013	0.0134	0.0172	0.0131
5	0.0146	0.0132	0.0143	0.016	0.0198	0.0144
6	0.0149	0.0166	0.0151	0.0151	0.0222	0.0152
7	0.0156	0.0148	0.0166	0.0159	0.0225	0.0155
8	0.0145	0.015	0.0168	0.0153	0.0241	0.0173
9	0.0163	0.015	0.0172	0.0162	0.024	0.017
10	0.0178	0.0158	0.0183	0.015	0.028	0.0179

Table 3.8: Top K recommendation evaluation results and Hit Rate (When K changed)

Models						
Top K	BPR	NCF	GMF	MLP	ItemKNN	Most Pop
1	0	0.02	0	0	0.02	0
2	0	0.05	0	0	0.02	0
3	0	0.075	0	0	0.025	0
4	0	0.12	0	0	0.09	0
5	0	0.13	0	0	0.115	0
6	0	0.145	0	0	0.15	0
7	0	0.17	0	0	0.105	0
8	0	0.26	0	0.31	0.1	0
9	0	0.23	0	0.89	0.15	0
10	1	0.275	0	1	0.17	1

For the experiment 4, I have checked whether pretraining could help improve neural collaborative filtering performance in terms of HR@10. I have used parameters specified in Table 3.9 to initialize the models. The table 3.10 contains HR@10 values models had reported in response to the change of number of factors in neural collaborative filtering model.

Table 3.9: Model parameters used to initialize pretraining models in experiment 4

Model	Parameters
GMF	Number of factors : 8, epoch : 10, optimizer : Adam, batch size : 256, learning rate : 0.001, number of negatives : 50, seed 123
MLP	Layers : [64, 32, 16, 8], activation function : tanh, learning rate : Adam, number of epoch : 10, batch size : 256, learning rate: 0.001, number of negative: 50, seed 123

Table 3.10: Neural Collaborative Filtering evaluation results when is used Pretraining and Not and reported HR@10 and NDCG@10

Factors	NDCG@10		HR@10	
	Pretraining	Without Pretraining	Pretraining	Without Pretraining
8	0.0144	0.017	0.31	0.29
16	0.0173	0.0168	0.27	0.24
32	0.0172	0.017	0.22	0.255
64	0.0196	0.205	0.24	0.18

In the 5th experiment I have checked whether deep learning could help models improve their performance. For this I have varied the number of MLP layers and reported the HR@10 and NDCG@10 values. I have used parameters specified in Table 3.3 to initialize the models. The table 3.12 contains HR@10 values models had reported in response to the number of MLP layers changed in neural collaborative filtering model.

Table 3.11: Neural Collaborative Filtering evaluation results and MLP Layers. Models evaluated in NDCG@10

Factors	MLP0	MLP1	MLP2	MLP3	MLP4
8	0.0161	0.0164	0.0154	0.0156	0.0168
16	0.0166	0.0209	0.0198	0.0194	0.0176
32	0.0179	0.0178	0.0225	0.0231	0.0173
64	0.0211	0.0244	0.0265	0.0285	0.0218

Table 3.12: Neural Collaborative Filtering evaluation results and MLP Layers. Models evaluated in HR@10

Factors	MLP0	MLP1	MLP2	MLP3	MLP4
8	0.325	0.445	0.0154	0.265	0.295
16	0.26	0.26	0.24	0.28	0.26
32	0.265	0.25	0.195	0.255	0.245
64	0.23	0.18	0.32	0.25	0.24

In the next section I will present results obtained and discuss about it.

4 RESULTS

4.1 Problem Verification Discussion

Below bar charts demonstrates NCDG@10 and HR@10 values reported for Bayesian Personalized Ranking, Neural Collaborative Filtering, Generalized Matrix Factorization, Multi-Layer Perceptron, Most Pop and Item KNN adjusted. The data set 4 and 5 can be considered as the sets where more weight has been given to the view events. Data set 9 and 10 can be considered as data sets with high amounts of views.

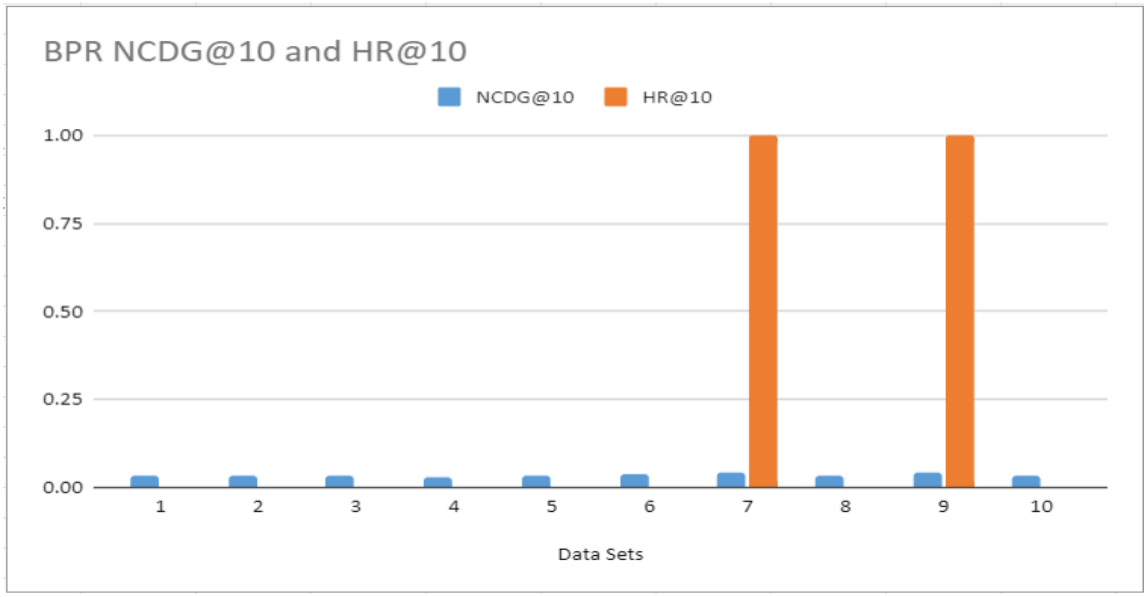


Figure 4.1: BPR - NDCG@10 and HR@10 performance for different weights and volumes for view, purchase and addToCart events.

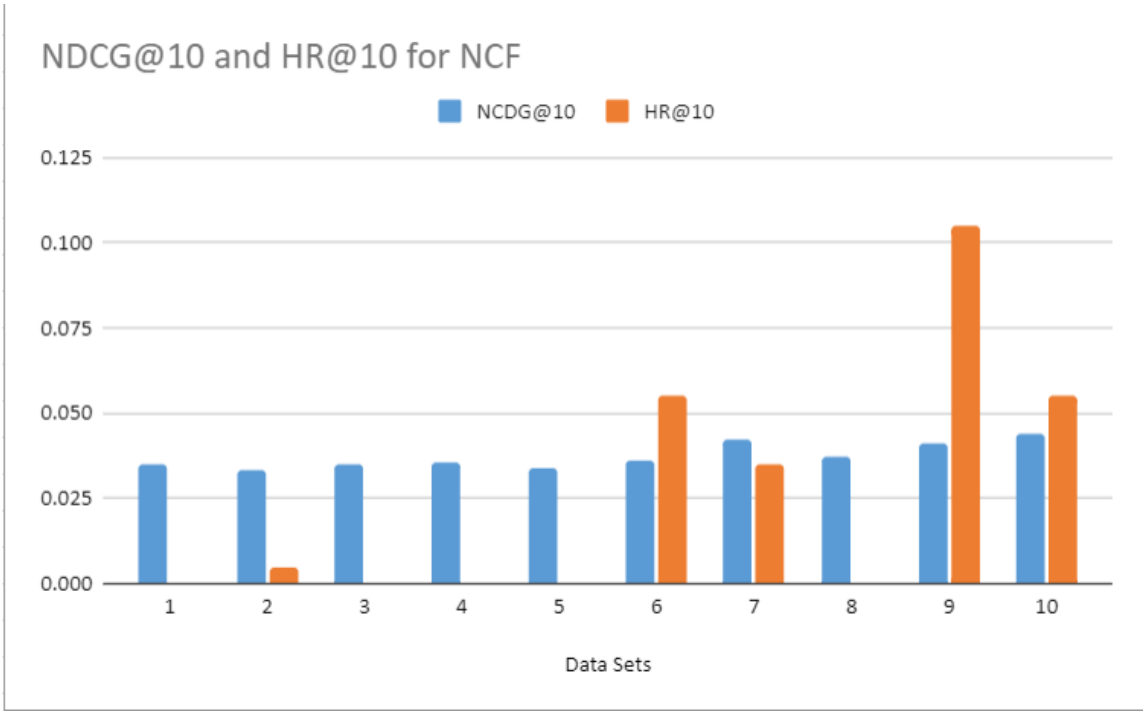


Figure 4.2: NCF NDCG@10 and HR@10 performance with different weights and volumes for view, purchase and addToCart events.

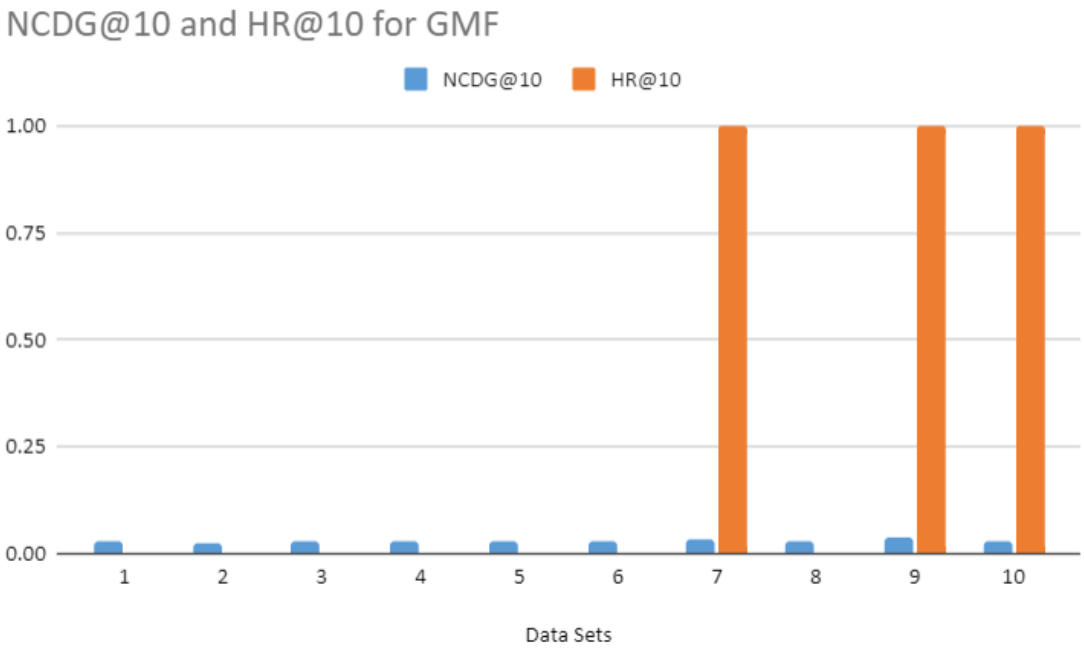


Figure 4.3: GMF - NDCG@10 and HR@10 performance for different weights and volumes for view, purchase and addToCart events.

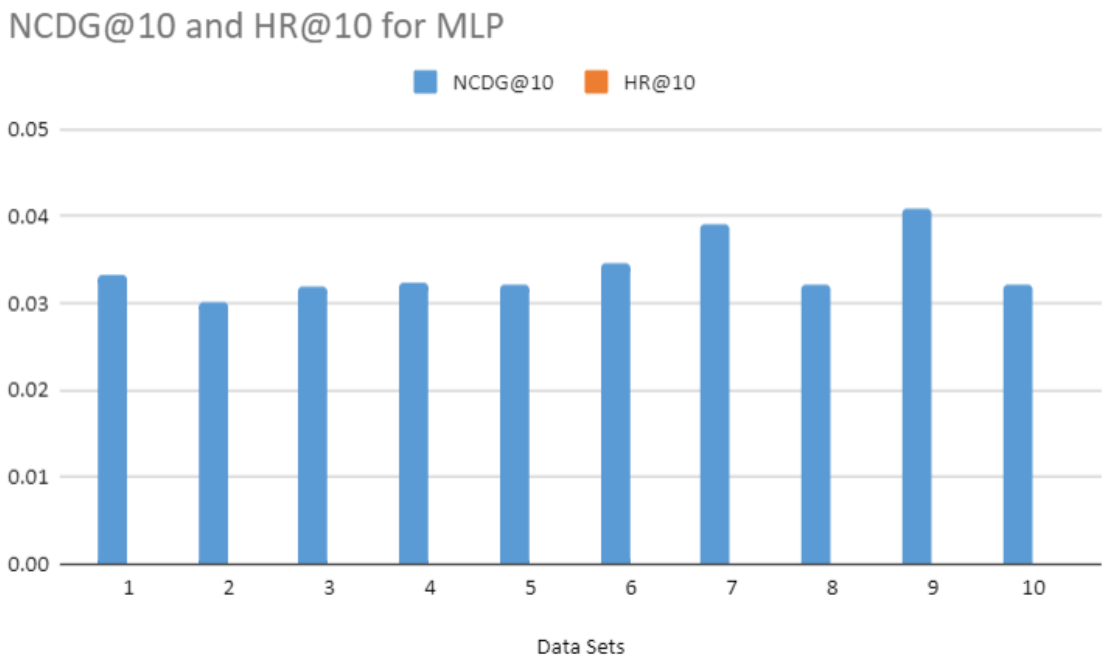


Figure 4.4: MLP - NDCG@10 and HR@10 performance for different weights and volumes for view, purchase and addToCart events.

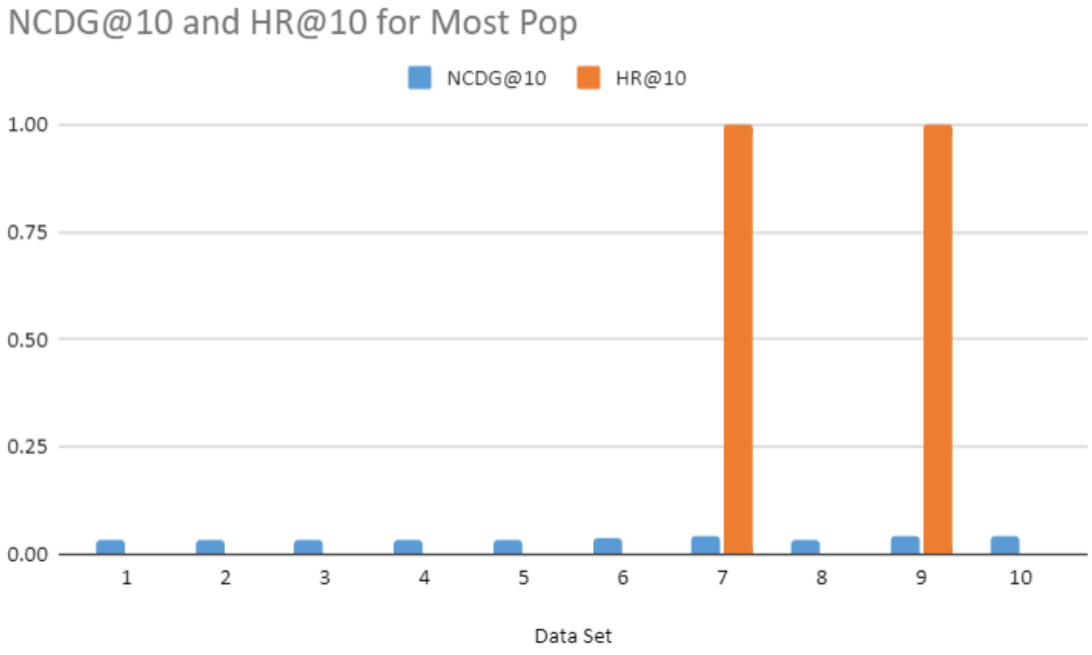


Figure 4.5: Most Pop - NDCG@10 and HR@10 performance with different volumes and weights of view, purchase and addToCart events.

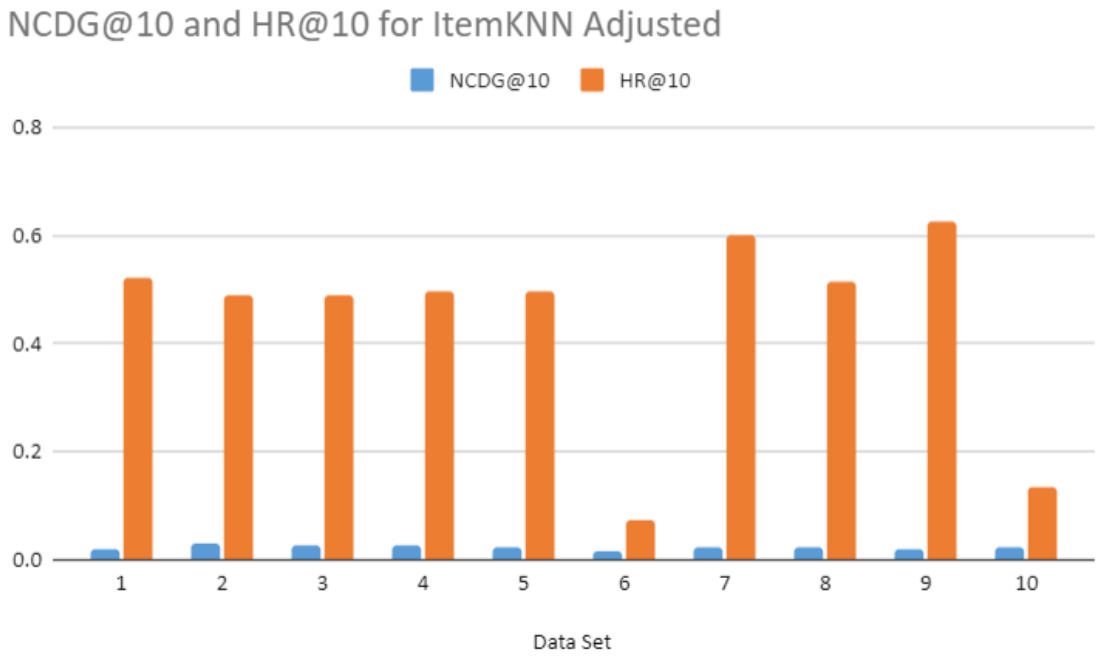


Figure 4.6: ItemKNN NDCG@10 and HR@10 performance with different volumes and weights of view, purchase and addToCart events.

In this section, I will discuss about, obtained results.

Next for each model, I will discuss results obtained for data set 1 – 5. This data set contains, views, addToCart and purchase events. Considering Bayesian Personalized Ranking for all 5 data sets a Hit Rate value of 0 was reported. This was true for neural collaborative filtering, generalized matrix factorization, multi-layer perceptron and Most Pop. This is because to create data sets, I have used user item interactions where they contain at least one purchase, view and addToCart events. However, data is organized in such a way, that, there are very large amount user item interaction where their event types are viewed, are positioned very closer to user item interactions where their event type is purchased or added to cart. Due to this reason, although I have increased the weight of purchase or add to cart events, it did no helped recommendation system recommend purchased products to users.

On the other hand, ItemKNN had reported HR@10 value near to 0.5 for all the data sets. This is because, for a given user and a set of items he had interacted with, there are many other products which are similar and purchased by the other users. In fact I have set the K, the number of nearest neighbors to be the value of 50. Because of this high K value, Item KNN had recorded HR@10 value near to 0.5.

Considering data set 6 – 10, Bayesian Personalized ranking had reported a HR@10 value of 0 to data set 6, 8 and 10. For the data set 7 and 9, it reported a HR@10 value of 1. This was observed for MostPop and generalized matrix factorization. Similar pattern was observed for neural collaborative filtering as well. However, the reported HR@10 values were near to 0. In the case of data set 9 it had reported a HR@10 value of 0.1. For the data set 7, a HR@10 value of 0.035 was reported.

Considering data set 6 and 8, they have more user item interaction where these interactions contains add To Cart and purchases. This value is close to 30000. However, in the data set, still the amount of view events are larger and it is close to 500000 records. However, purchase pattern in addToCart and purchase data records are not similar to purchase patterns that are learnt from data records containing view events. Furthermore, function learnt based on view events do not have a positive effect on function learnt based on addToCart and purchase events. Considering other way, function learnt based on addToCart and purchase events do not have a positive effect on function learnt based on view events. As a result, all 4 models had failed to learn a function to recommend purchased products and reported a HR@10 value near to 0.

As a result for data set 6 and 8, Bayesian Personalized ranking, MostPop, generalized matrix factorization and neural collaborative filtering recorded a HR@10 value close to 0.

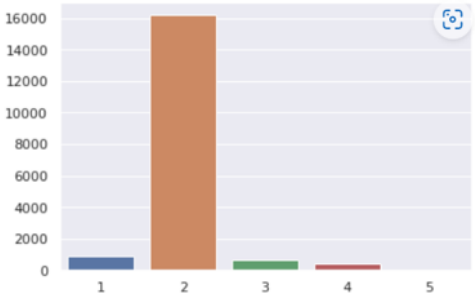
As for the data set 10, the amount of purchase and add To Cart events are very low. Moreover, it has a very large amount of user interactions which are viewed and as a result, and this data set do not exhibit any purchase patterns. As a result, all 4 models had reported a HR@10 value near to 0.

Finally, for data set 7 and 9, they have a relatively small amount of purchase and add to cart events where the total volume is closer to 10000. However, the data set contains view event records close to 500000. In this case, all four models had learnt purchased item patterns from the data records containing view events. As a result, for data set 7 and 9 all 4 models had reported a high HR@10 values.

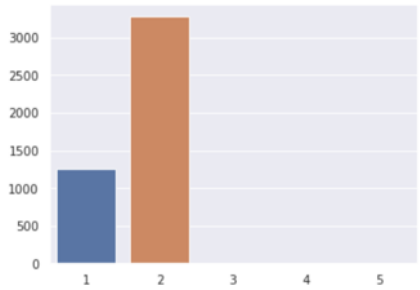
For all 5 data sets, MLP had failed to learn function, that could be used to recommend purchased items to users. This is because, view events are not a stronger indicator of purchasing a product. However, all data sets, 6 – 10 has a considerable amount of view events. Due to this reason, it has recorded a HR@10 value close to 0.

Finally, Item KNN had recorded a HR@10 value close to 0.5 for all five data sets from 6 – 10. This is because, for a given user and a set of items he had interacted with, there are many other products which are similar and purchased by the other users. In fact we have set the K, the number of nearest neighbors to be the value of 50.

Next I have analyzed, the data distributions of the data used to train the models, and the output in terms of the score I have assigned. For this comparison, I have used NCF, MostPop and ItemKNN.

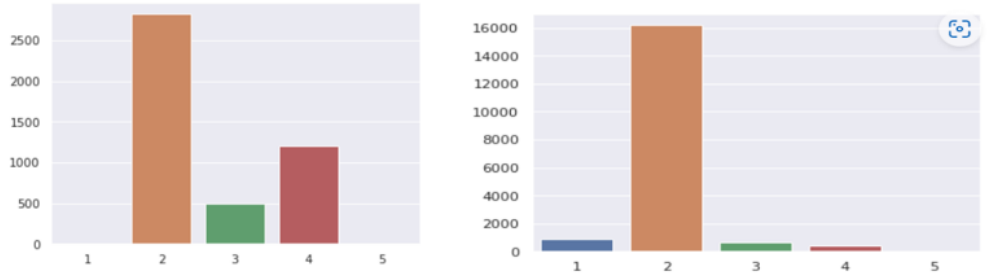


Training data score distribution for data set 9.



Model result score distribution for data set 9

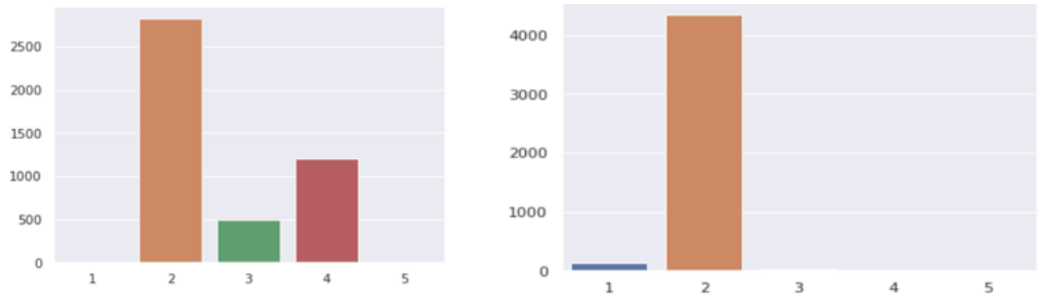
Figure 4.7: Comparison of NCF model output score distribution with training score distribution for the data set 9. Left training score distribution and right model output score distribution.



Training data score distribution for data set 9.

Model result score distribution for data set 9

Figure 4.8: Comparison of MostPop model output score distribution with training score distribution for the data set 9. Left training score distribution and right model output score distribution.



Training data score distribution for data set 9.

Model result score distribution for data set 9

Figure 4.9: Comparison of ItemKNN model output score distribution with training score distribution for the data set 9. Left training score distribution and right model output score distribution.

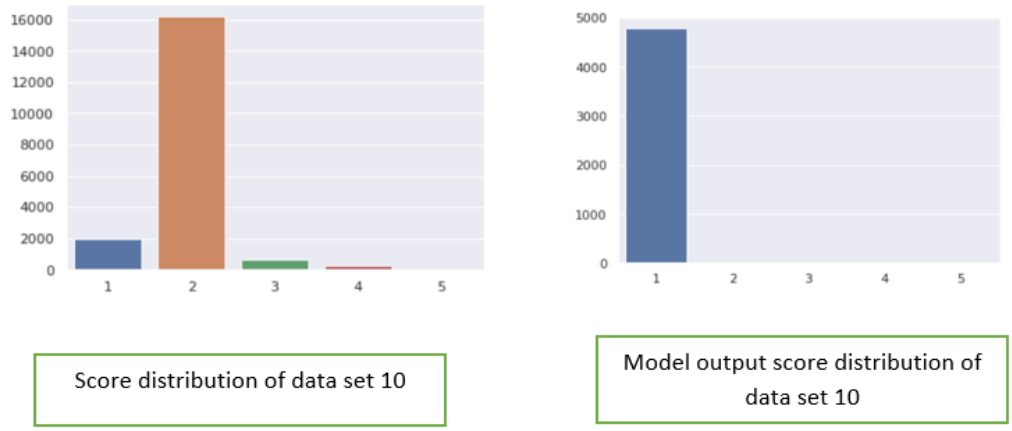


Figure 4.10: Comparison of NCF model output score distribution with training score distribution for the data set 10. Left training score distribution and right model output score distribution.

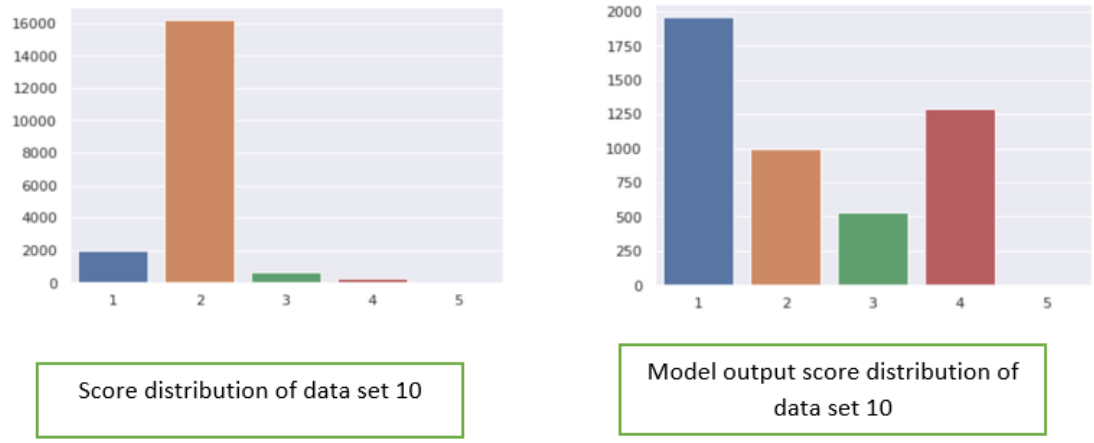


Figure 4.11: Comparison of MostPop model output score distribution with training score distribution for the data set 10. Left training score distribution and right model output score distribution.

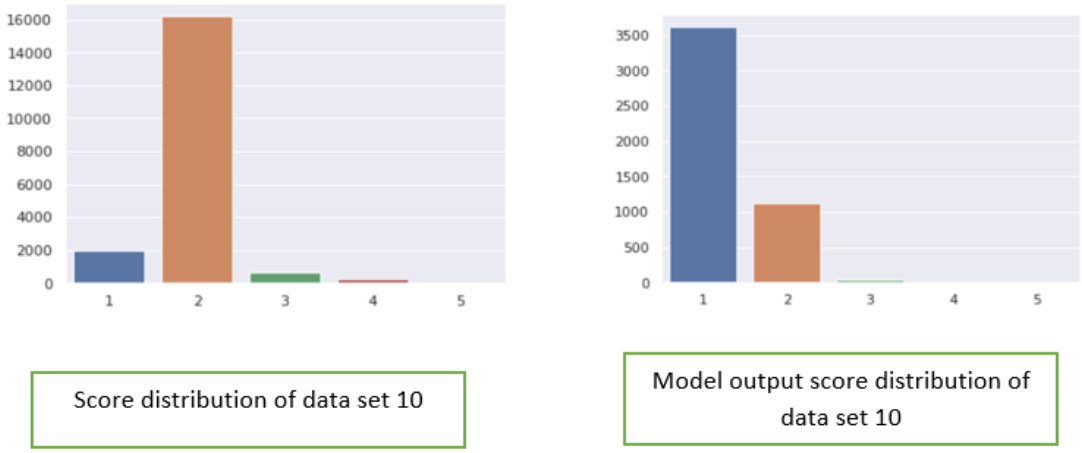


Figure 4.12: Comparison of ItemKNN model output score distribution with training score distribution for the data set 10. Left training score distribution and right model output score distribution.

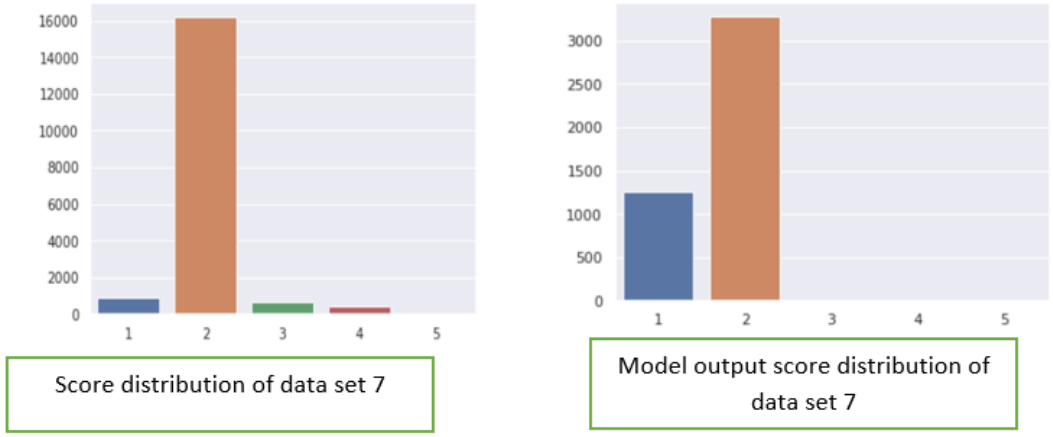


Figure 4.13: Comparison of NCF model output score distribution with training score distribution for the data set 7. Left training score distribution and right model output score distribution.

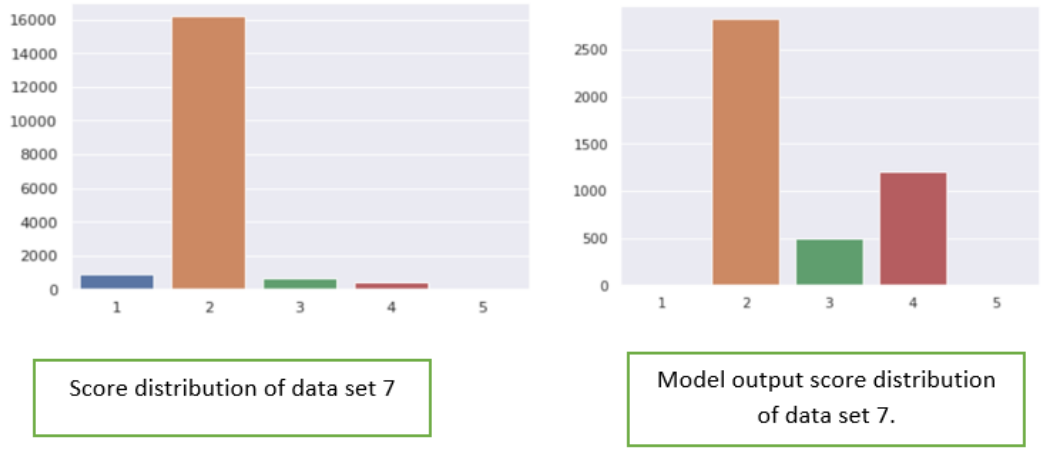


Figure 4.14: Comparison of MostPop model output score distribution with training score distribution for the data set 7. Left training score distribution and right model output score distribution.

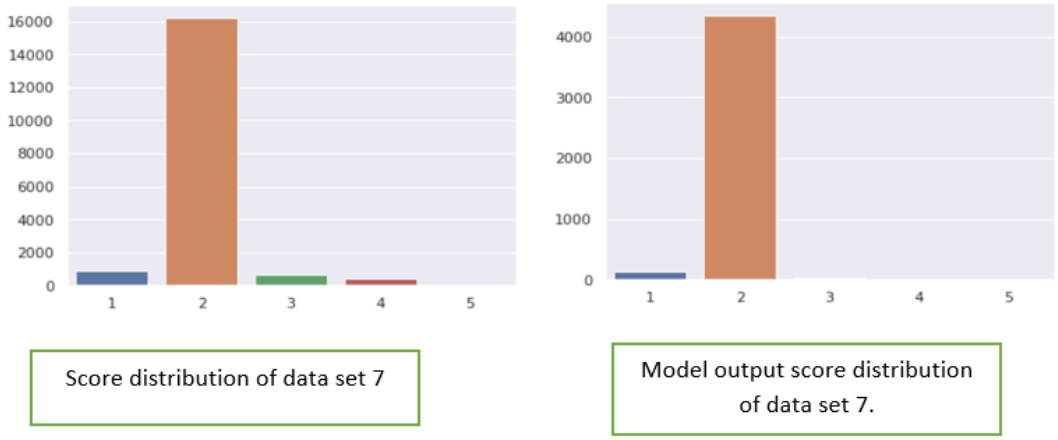


Figure 4.15: Comparison of ItemKNN model output score distribution with training score distribution for the data set 7. Left training score distribution and right model output score distribution.

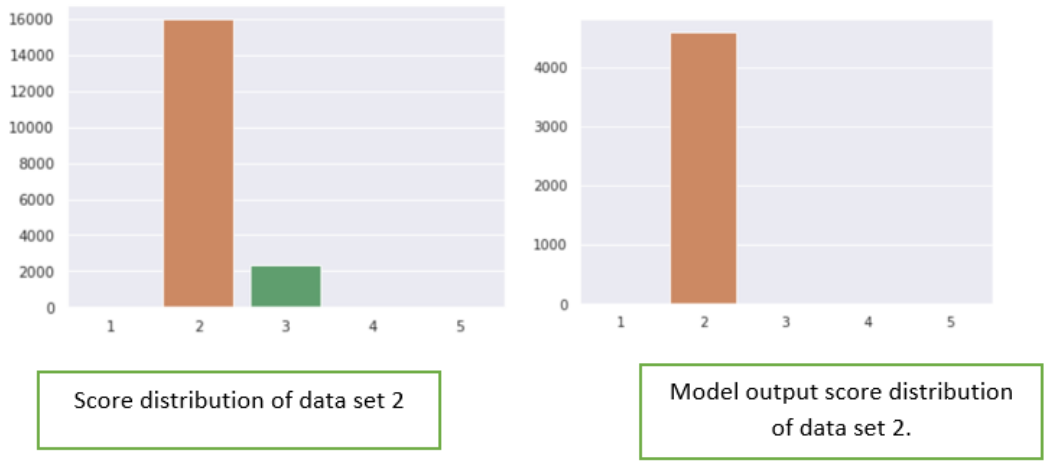


Figure 4.16: Comparison of NCF model output score distribution with training score distribution for the data set 2. Left training score distribution and right model output score distribution.

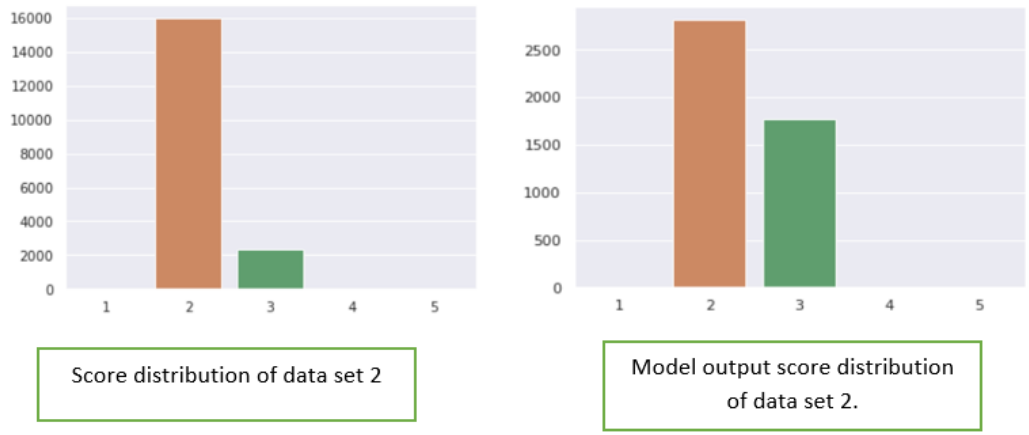


Figure 4.17: Comparison of Most Pop model output score distribution with training score distribution for the data set 2. Left training score distribution and right model output score distribution.

Figure 4.18: Comparison of Item KNN model output score distribution with training score distribution for the data set 2. Left training score distribution and right model output score distribution.

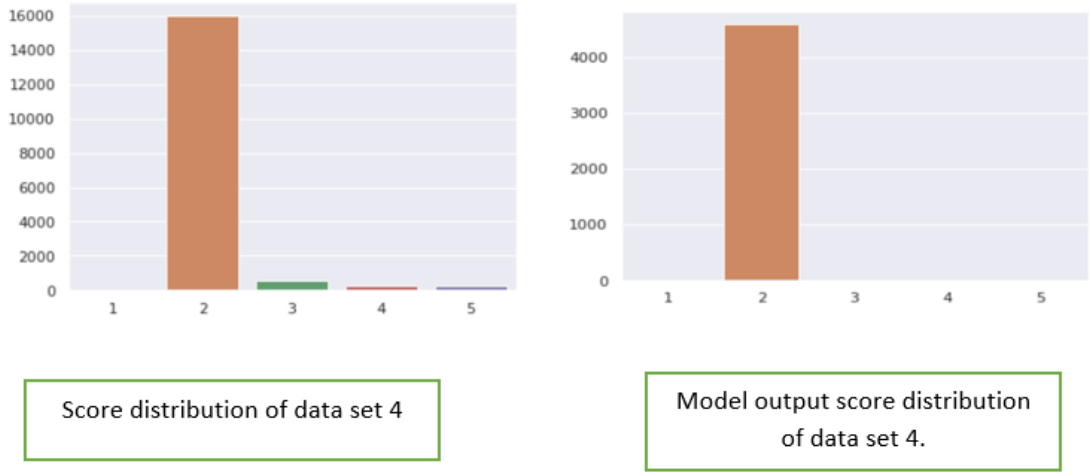


Figure 4.19: Comparison of NCF model output score distribution with training score distribution for the data set 4. Left training score distribution and right model output score distribution.

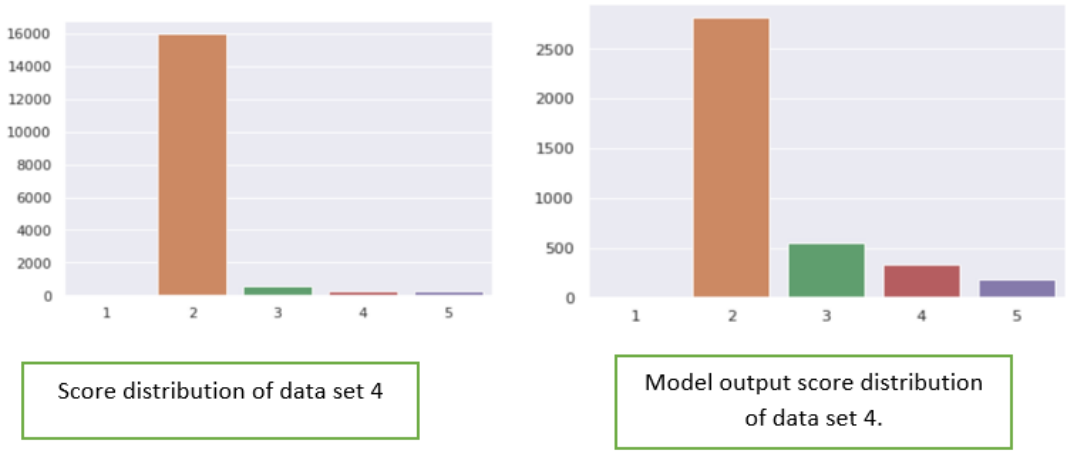


Figure 4.20: Comparison of Most Pop model output score distribution with training score distribution for the data set 4. Left training score distribution and right model output score distribution.

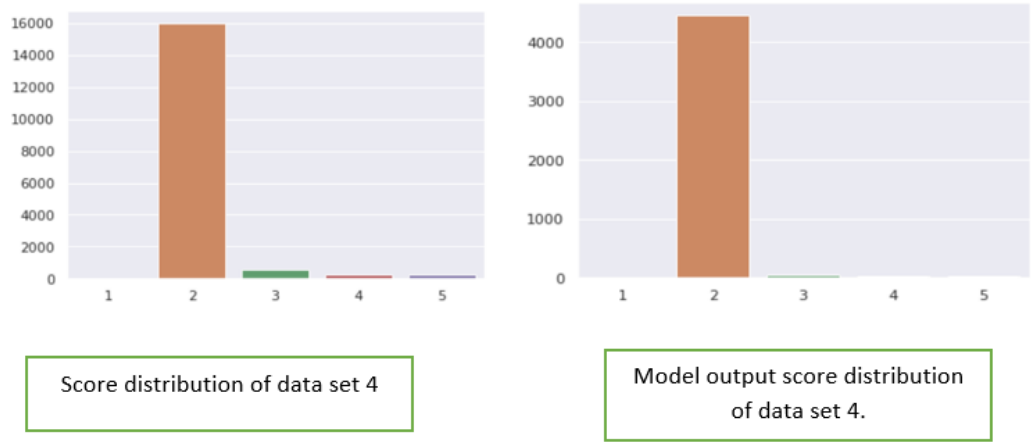


Figure 4.21: Comparison of ItemKNN model output score distribution with training score distribution for the data set 4. Left training score distribution and right model output score distribution.

Next, I will discuss about the results I have obtained. First, I will consider Neural Collaborative filtering. The model had reported a high HR@10 values for the data set 9 and 10. For the both data sets, the distribution of the training data set and the distribution of the model output is different [Figure 4.16 and 4.19]. However, compared to the data set 10, neural collaborative filtering had managed to learn a better function for the data set 9. Furthermore, considering data set 9, the NCF had managed to predict scores with values 2 and 1. For the data set 10 model had

managed to predict scores with value 1 better. Considering data set 7, NCF had managed to learn a distribution similar to the training data set [Figure 4.13]. Furthermore, the model had predicted event counts with value 2 better. Similar observations were made for data set 2 [Figure 4.16 and 4.19] and 4 as well. Hence, looking at the HR@10 values, the NCF had reported a value of 0.005, 0.055, 0.105 and 0.055 for the data set 2, 7, 9 and 10 respectively.

The first observation is that, in most occasions the model had learnt scores with high frequencies.

Considering this fact, it is clear that, the model has predicted scores with higher frequencies. However, data set 9 and 10 has a very low amount of purchased events although products that are viewed more than 15 times are associated with purchasing a product. For the data set 2, 7 and 10, because the frequency of the scores associated with purchases were low, a low value of HR@10 value was reported.

As for the Most Pop it had managed to learn training data set pattern only for data set 4 [Figure 4.20]. For the other training data sets the model had failed to learn the distribution [Figure 4.8, 4.11, 4.14 and 4.17]. However, like NCF, it had managed to predict scores with high frequencies. As a result, a high HR@10 values were reported to data set 7 and 9.

As for the ItemKNN, other than for the data set 9 [Figure 4.9] for data set 2,5 and for 10, the model had learnt training model distribution [Figure 4.12, 4.15, 4.18, 4.21]. Compared to NCF and MostPop, ItemKNN has managed to predict scores with low frequencies. These scores are associated with purchased events. As a result, it had reported a HR@10 value over 0.5 for the all the data sets.

4.2 Effect of Learning Rate on Model Performance

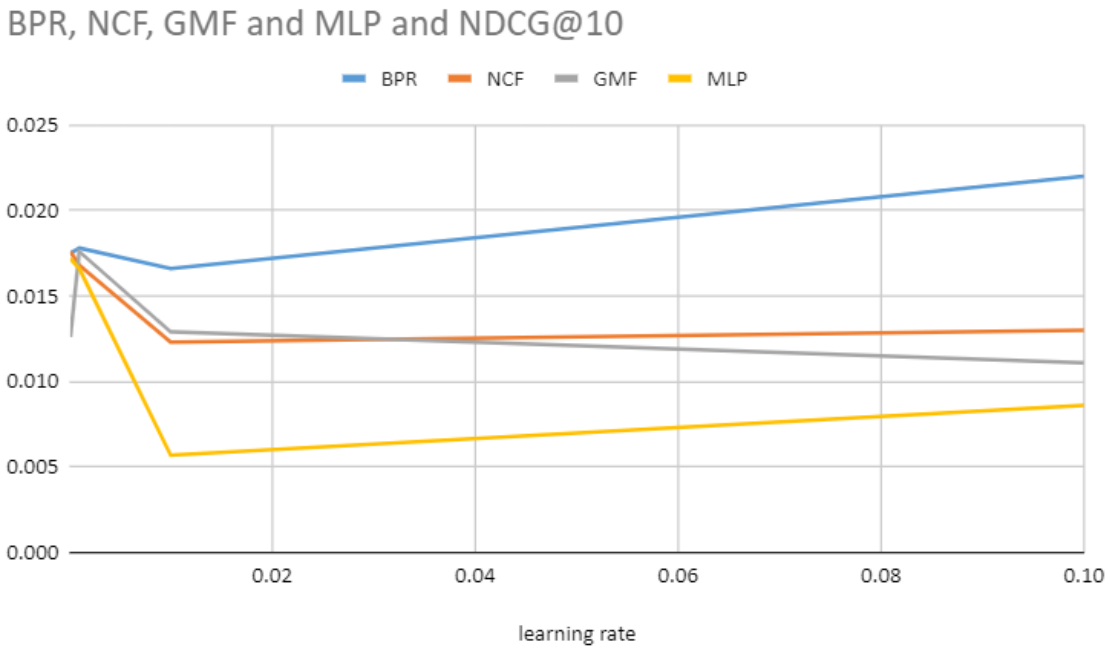


Figure 4.22: Learning rates and model NDCG@10 performance.

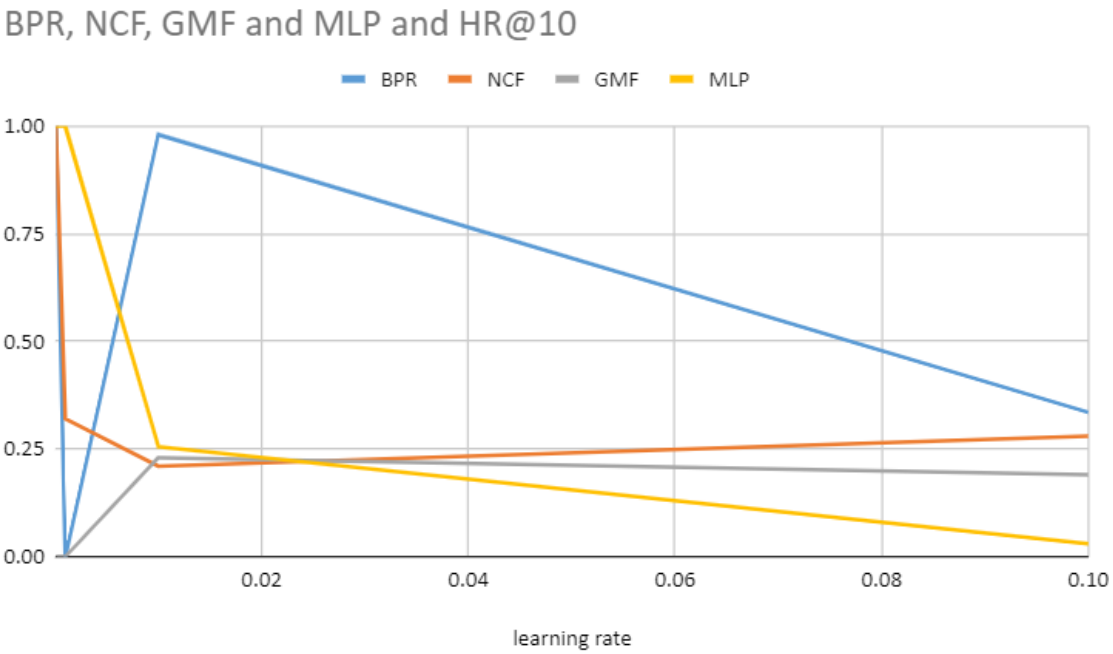


Figure 4.23: Learning rates and model HR@10 performance.

Considering HR@10 [Figure 4.23], for BPR, GMF, MLP and for NCF, for low learning rate, higher hit rate has been reported. In other words, the smaller updates had helped models improve their performance whereas as the amount of updates got higher the model HR@10 value had started to go down.

However, as the amounts of updates got higher, the NDCG@10 started to go down for all the models as the learning rate goes down [Figure 4.22]. The reason for low NDCG@10 values is because we did not perform any hyper parameter search to find best parameters suited to improve NDCG@10.

4.3 Effect of Predictive Factor size on Model Performance

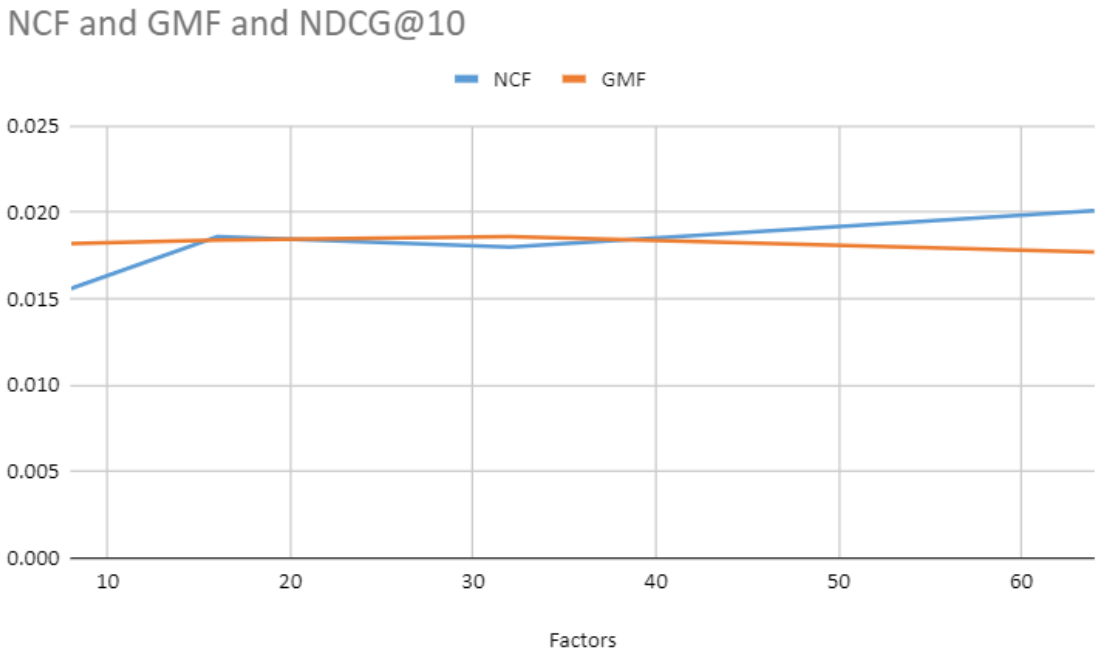


Figure 4.24: Embedding size and model NDCG@10 performance.

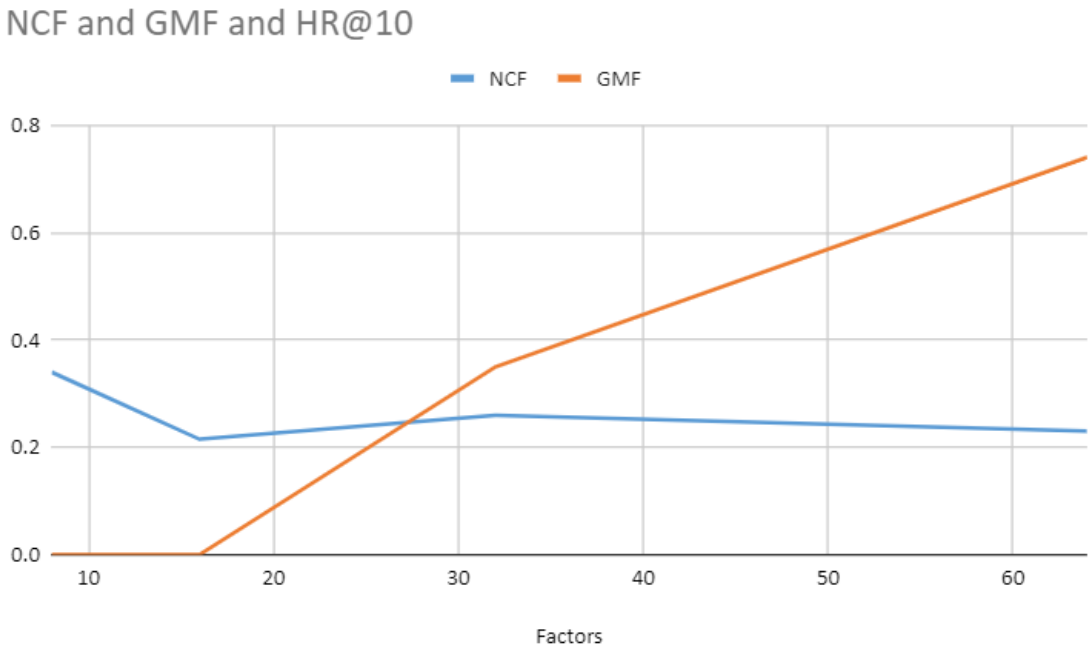


Figure 4.25: Embedding size and model HR@10 performance.

Considering HR@10 [Figure 4.25] we can see that for the predictive factors lesser than 25 NCF had outperformed the GMF. However, as the number of factors started to go higher, GMF had outperformed NCF in a large margin. However, since NCF is a combination of GMF and deep neural networks and as a result it can better learn latent factors compared to GMF. In this case, for large factors, GMF had outperformed NCF due to lack of purchase data resulting less features to learn better latent factors from the data set.

Considering NDCG@10, [Figure 4.24] for almost all predictive factors, the values were same for both GMF and NCF. Again, the reason for low NDCG@10 value is because, we did not perform any hyper parameter search to find best parameters suited to improve NDCG@10. However, for the small predictive factors GMF had marginally outperformed NCF while it was the opposite for the large number of predictive factors.

Next, we will look at, how hit rate and NDCG varies with the top k.

4.4 Top K recommendation and model performance discussion

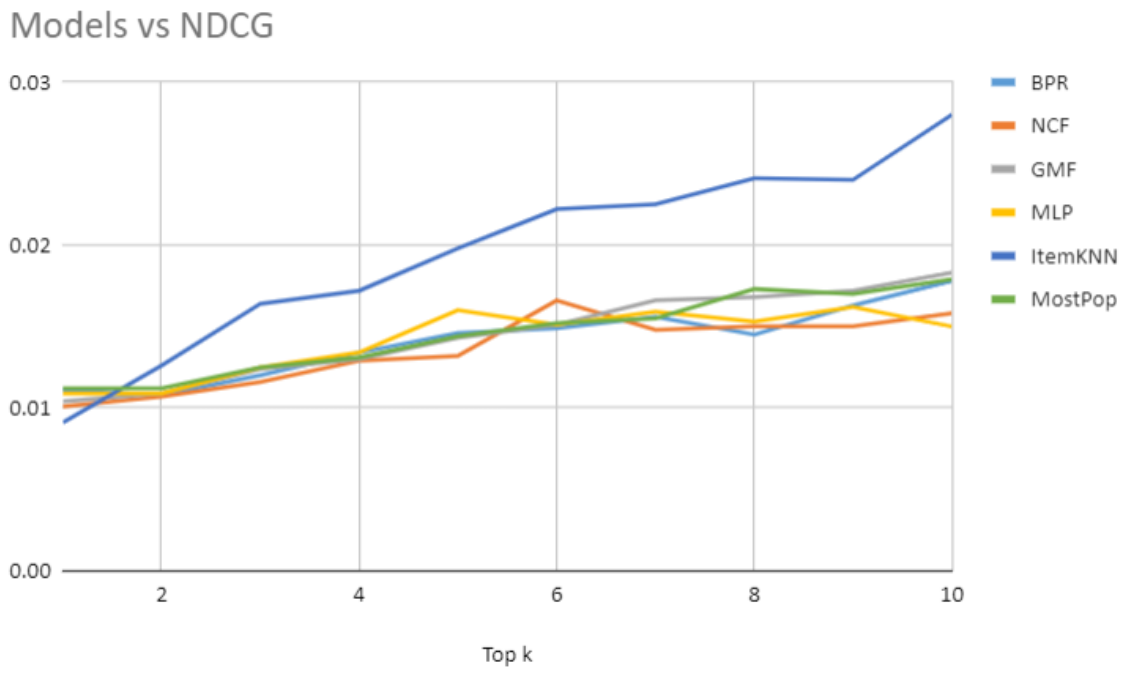


Figure 4.26: Top k and model NDCG@10 performance.

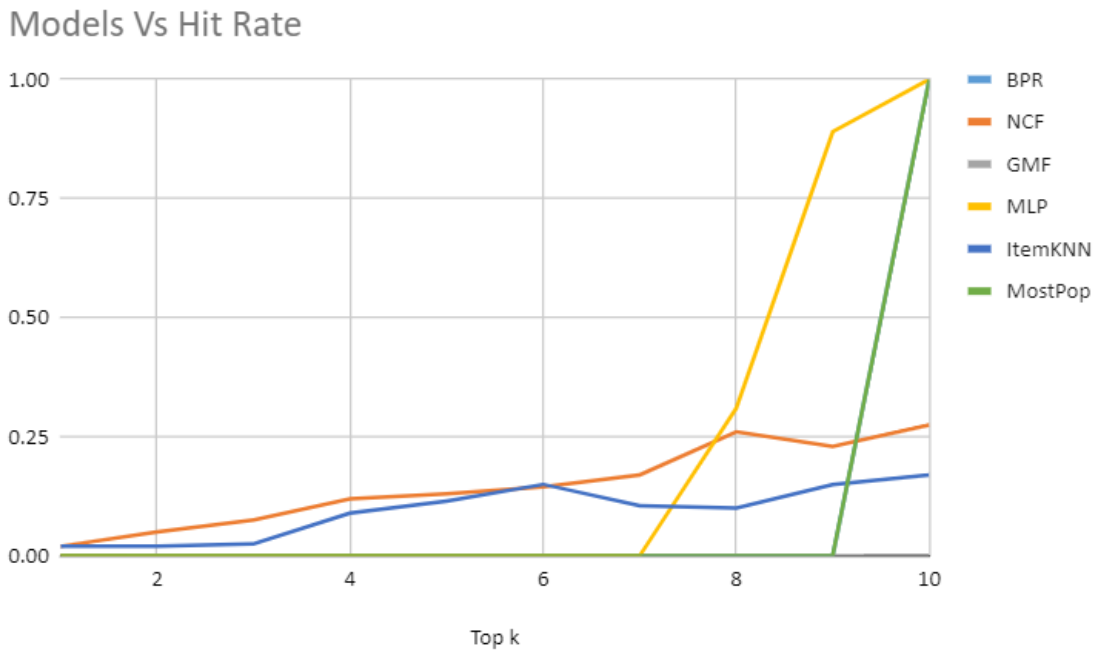


Figure 4.27: Top k and model HR@10 performance.

Considering NDCG [Figure 4.26], for all top k, BPR had displayed the best performance. The performance of other models was similar in most cases and as the k value increases, the NDCG value had increased as well.

Considering the Hit rate [Figure 4.27], for k values lesser than 8, NCF had displayed a better performance. Above k values over 8, BPR and Most Pop had displayed a higher hit ratio closer to 1. Except for Item Pop and BPR, as the k values increases, the hit rate of all the models had increased by significant values. Furthermore, if the data set contains sufficient number of purchase events for user item interaction, it is expected that NCF will outperform Most Pop and BPR for HR@9 and HR@10.

4.5 Effect of Pretraining on model performance discussion

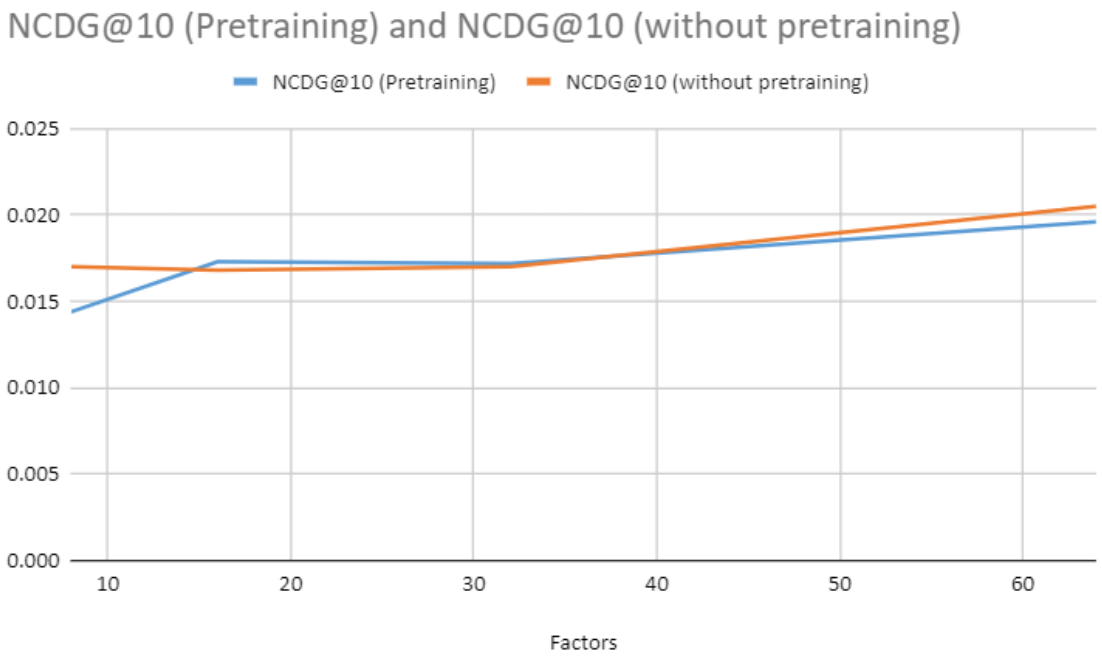


Figure 4.28: Pre trained models and model NDCG@10 performance with embedding size.

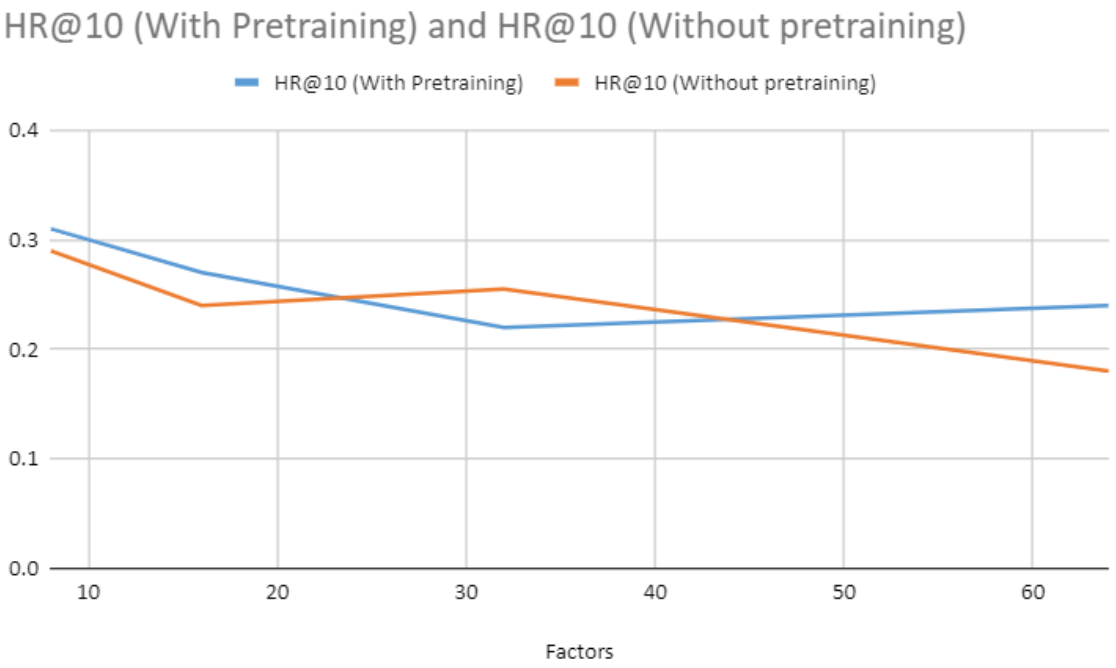


Figure 4.29: Without pre training models and model NDCG@10 performance with embedding size.

Considering NCDG [Figure 4.28], pre training had marginally helped increase the performance. It is observed that, as the number of factors increases, the model performance had increased as well.

In the case of Hit Ratio [Figure 4.29], again in most cases models with pre training had out performed, models without pertaining in most cases. We observe that as the number of factors increases the Hit rate had dropped for both pre training and for models without pre training. This is because of the lack of purchase events. Due to the lack of purchase events for lots of user product interactions, the model was unable to learn latent factors effectively.

4.6 Effect of change of MLP layers on model performance

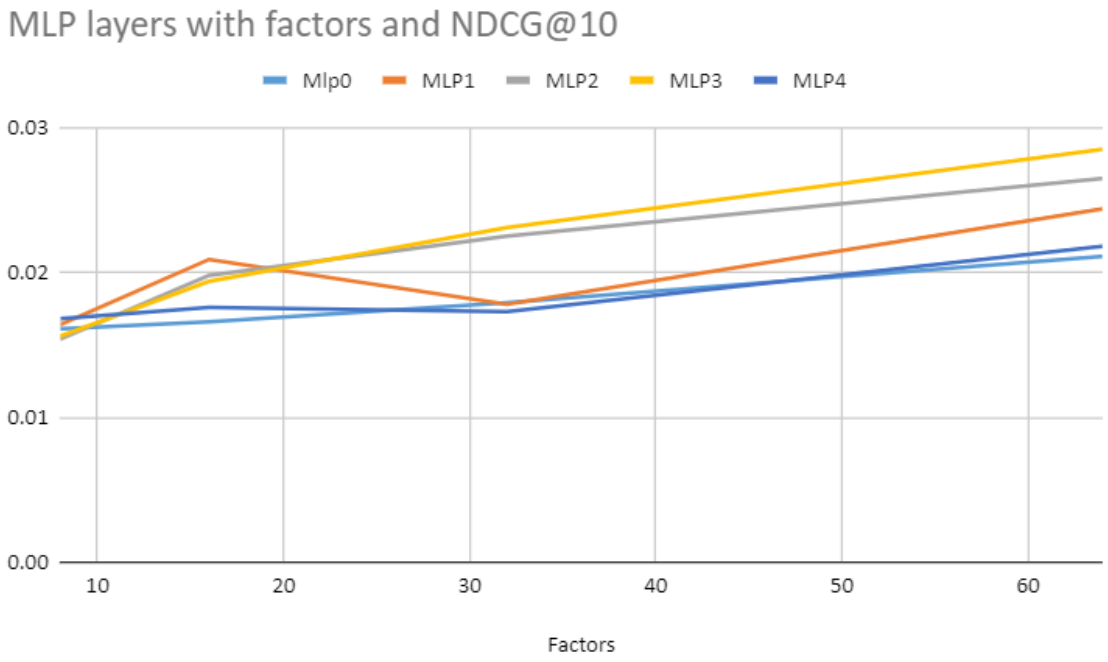


Figure 4.30: Increase MLP layers, and embedding size with model NDCG@10 performance.

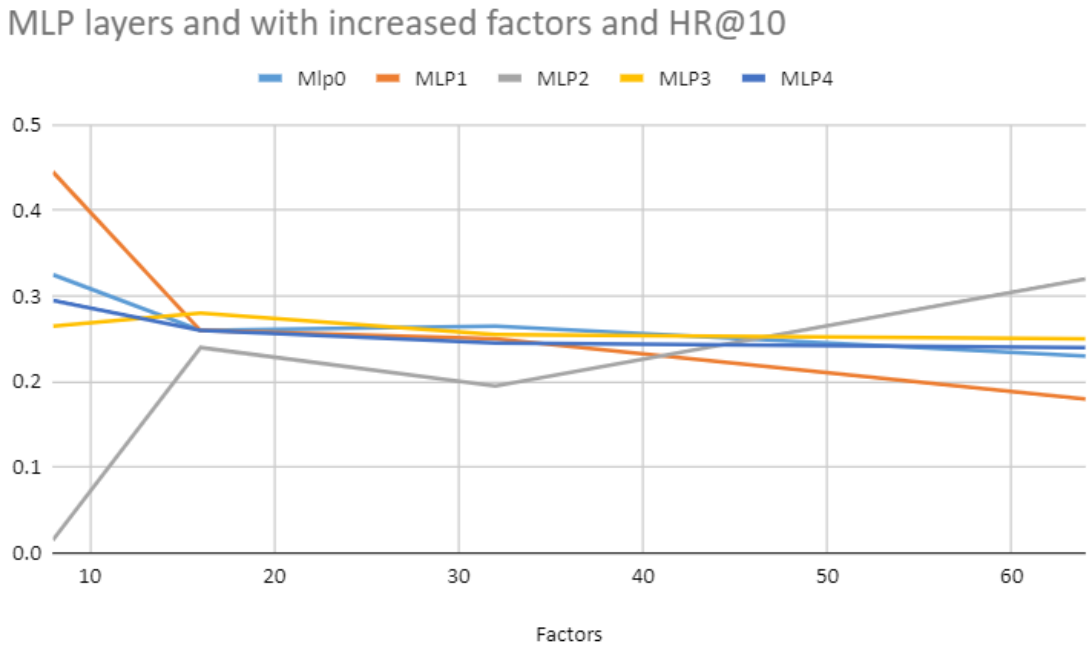


Figure 4.31: Increase MLP layers, and embedding size with model HR@10 performance.

Finally, we analyzed whether the deep learning could help recommendation system recommend more purchase products. For NDCG [Figure 4.30] models with 3 MLP layers had recorded best performance followed by MLP-2 and MLP-1. The performance of MLP-0 and MLP-4 was lowest.

Considering Hit Rate [Figure 4.31], model with 3 MLP layers had reported best result. This is followed by MLP-2 and MLP-1 for factors between 20 to 50. For factors greater than 40, MLP-2 had reported the best performance, while for factors lesser than 20 it was model with 1 MLP layer. Adding more MLP layers in general had helped NCF improve its performance considering factors between 20 – 40 and factors more than 40. However, the results had become inconsistent because due to less amount of user product purchase interactions resulting models cannot learn good latent factors.

4.7 Comparison with the existing research

4.7.1 Comparison of bench mark data set with Retail Rocket Data Set

In this section, I will be comparing my research with the existing work. First I will compare the selected data set with the benchmark data sets used in the literature.

Table 4.1: Existing Benchmark Data Set Statistic Comparison

The Model	Data Set	Description	Key Statistics
Alternating Least Squares	Y!Music	Music and their ratings (KDD Cup 2011)	No of ratings : 62,551,438 (0 - 100). No of users : 249012 No of Items : 296111

	Netflix	Music and their ratings	No of ratings : 100,480,507 (1 – 5) No of users : 480189 No of Items : 17770
SLi_Rec : Adaptive User Modeling with Long and Short-Term Preferences for Personalized Recommendation	Amazon data set	Product reviews and metadata. This reviews belongs to electronics, movies, TV, CDs and Vinyl). Data set is sequential in nature)	
	Industrial data set	Native advertising data set. Data set sequential in nature. Collected the data set for 3 months starting from November 2018.	
GRU4Rec (SESSION-BASED RECOMMENDATIONS WITH RECURRENT NEURAL NETWORKS)	RecSys Challenge 2015	Click stream events containing purchase events.	No of sessions : 7966257 No of clicks : 31637239 No of items : 37483
	Youtube OTT data set	Events containing watching a video. The data was collected for two months.	
Caser	Movie Lens	Movie ratings	Number of users : 6k

			Number of items : 3.4k Average actions per user : 165 Sparsity : 95%
	Gowalla	Implicit feedback data on venue check in.	Number of users : 13.1k Number of items : 14k Average actions per user : 40 Sparsity : 99.71%
	Foursquare	Implicit feedback data on venue check in.	Number of users : 10.1k Number of items : 23.4k Average actions per user : 30.16 Sparsity : 99.87%
	Tmall	User purchased events.	Number of users : 23.8k Number of items : 12.2k Average actions per user : 13

			Sparsity : 99.89%
NextItem	Yoochoose	Recsys 2015 purchase events	
	Last.fm	Music data set	
Multi-Interest-Aware Sequential User Modeling (SUM)	Display Ads Data set	Two weeks click data collected from Bing advertising logs.	
	Taobao Data set	E commerce data set which contains purchase events.	
Baysian Personalized Ranking	Rosmann data set	Purchase history of the users	Total number of purchased data : 426612 No of users : 10000 No of items : 4000
	DVD rental data set	Explicit ratings of songs	
BiVAE (Bi variational auto encoder)	MovieLens data set	Movie songs	No of users : 145000 No of items : 30000 No of feedback : 21000000
	Amazon data set (Product categories Office, clothing and sports.		Clothing : No of users : 39145 No of items : 22275

			No of feedback : 272765 Sports : No of users : 72464 No of items : 26640 No of feedback : 439650 Epinions: No of users: 23247 No of items : 59451 No of feedback : 553354
Variational Auto encoder	Movie Lens 20M	User movie data set with rating	
	Netflix data set	User movie rating with binary rating	
	Million song data set	User song play counts	
Neural Collaborative Filtering (NCF)	Movie Lens data set	User and item ratings	No of users : 6040 Item : 3706 Interaction : 1000000

			Sparsity : 95.53%
	Pinterest data set		No of users : 55187 No of item : 9916 Interaction : 1500809 Sparsity : 99.73%
RBM (Restricted Boltzman Machine)	Netflix Data Set		No of ratings : 100480507 No of users : 480189 No of movies : 17770
Wide and Deep Learning	Used A/B testing		

Considering our data set it has 1400000 number of users and 467000 number of items in the data set. Furthermore 97% of the data set contains the view events while, 3% are add to cart events and 1 % of the data set contains purchased events. The data set was initially sparse as the number of items viewed or addToCart or purchased by a user was less than 90.

Considering the variations of click events starting from 2015-05-01 to 2015-09-15, the number of views had varied significantly, while their counts were much higher compared to purchased and addToCart events. Over the time period the purchased and add to cart events remained much lesser and on daily basis they were nearly equal and close to 1.

Similar pattern was observed, daily basis as well. For a given a day, for every hour view events were much higher compared to add to cart and purchased events. Looking at each hour, the number of purchase and addTo cart events were closer to 1. In the 10^{th} hour on every day, we see a large drop of view events in the data set. However, one interesting

observation in the data set is that, for a given day, user viewing and purchasing of items are similar. Moreover, over 50 of items were bought in the first sight while 25% on after viewing once and rest after a 2-5 viewings.

Furthermore, if we consider a scatter plot number of purchases vs number of views, 99% of data points gets clustered near to the x and y axis. Hence, we do not observe any linear or any other type of relationship between the number of purchasing and view event counts. Same, pattern was observed for addToCart and view event counts. However, we noticed a linear relationship between purchase and addToCart event counts. Moreover, there are five items that have been viewed more than 5000 times and 5 items that has over a 1500 times addToCart events and 5 items that has over 100 times of purchase events. However, these items are not similar.

If we consider the daily sessions it had varied from 6000 to 20000. On average there was 7500 sessions reported on a given day.

Next I will look at different preprocessing techniques applied on the data set and the volume resulted after applying the steps.

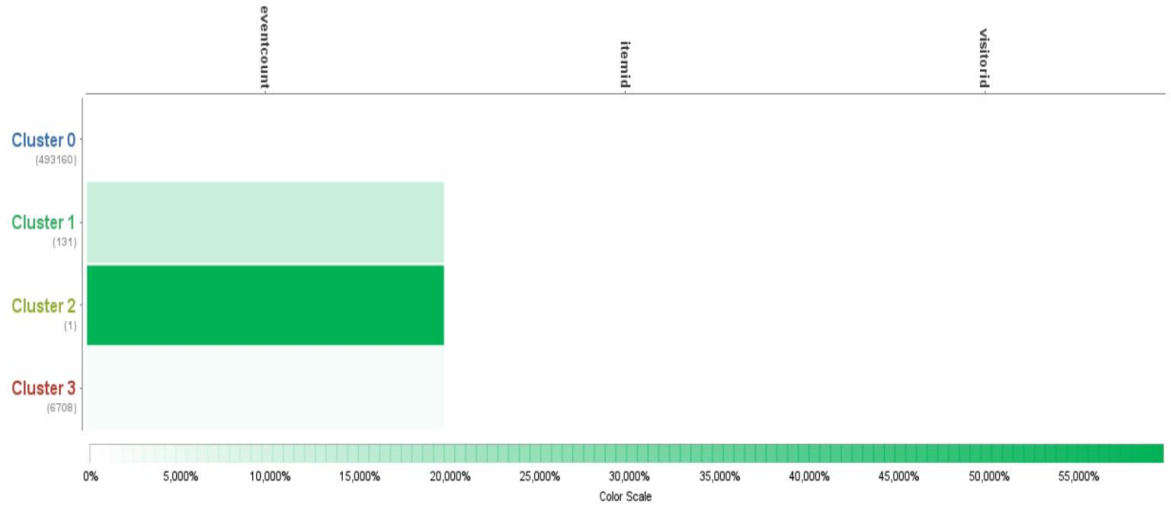


Figure 4.32: Clustering for the data set with item, visitor and event counts (Heat Map)

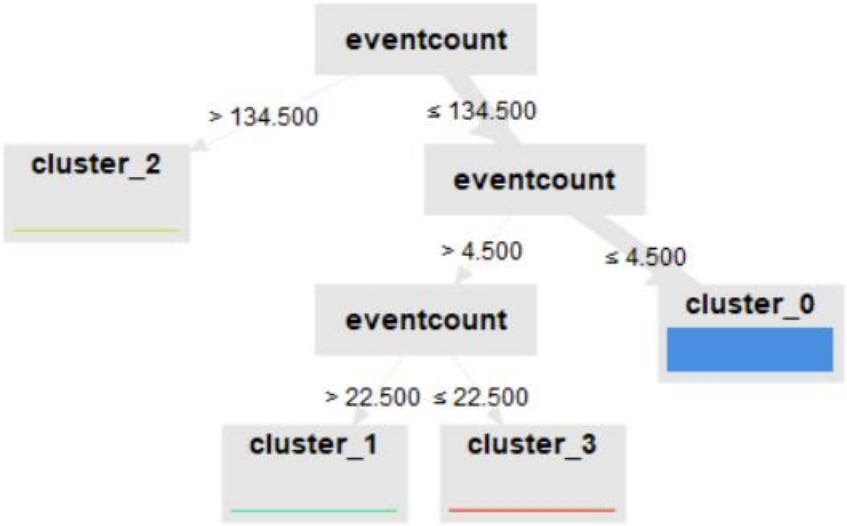


Figure 4.33: Cluster tree created for the data set with item ,visitors and event counts

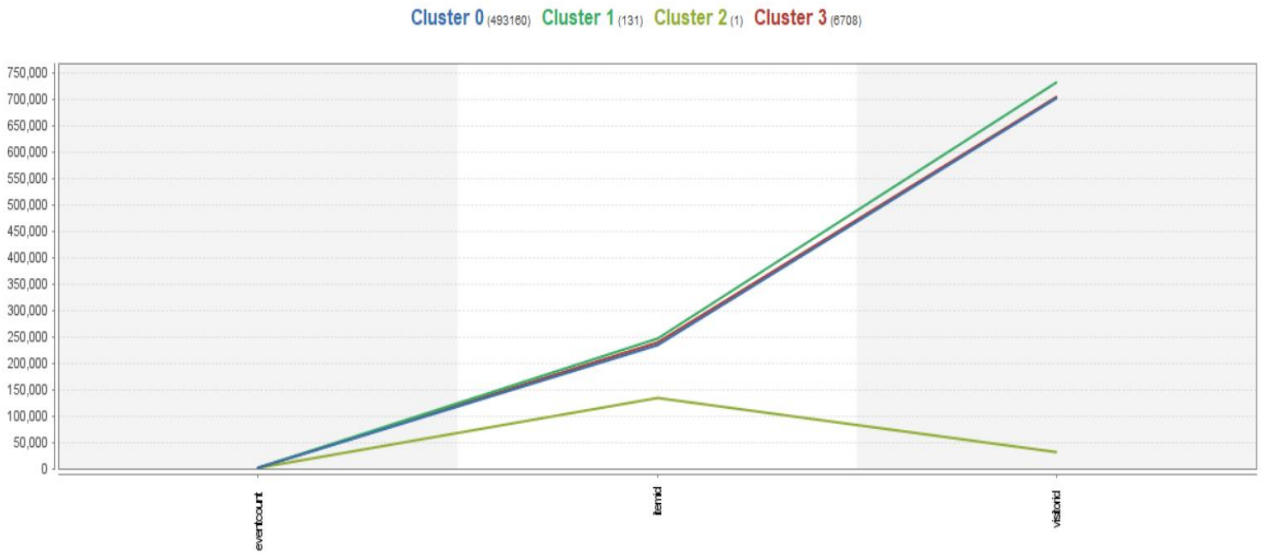


Figure 4.34: Centroid Chart for the item id, visitor id and event counts

Cluster	eventcount	itemid	visitorid
Cluster 0	1.196	234416.571	702730.014
Cluster 1	37.288	246094.314	731287.720
Cluster 2	170	133138.000	32252.000
Cluster 3	6.971	238611.521	704828.420

Figure 4.35: Cluster table item id, visitor id and event count

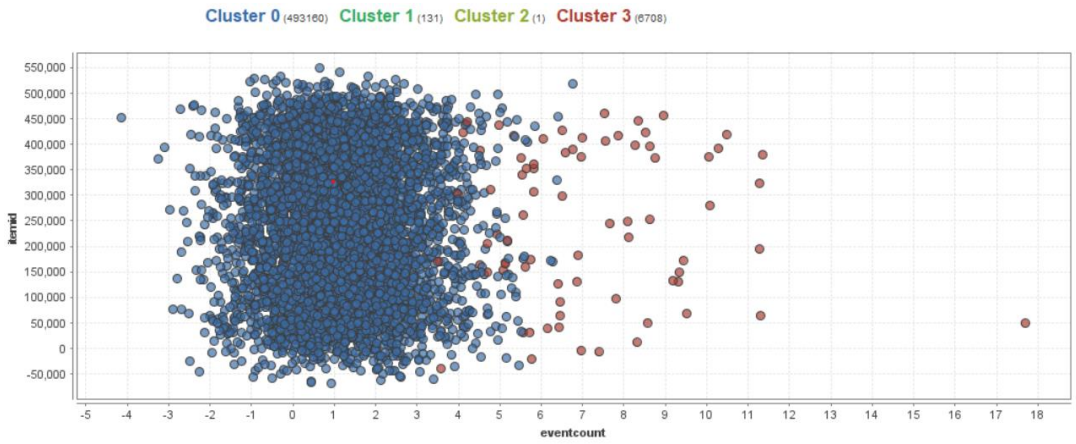


Figure 4.36: Scatter plot view for the cluster 0 and 1

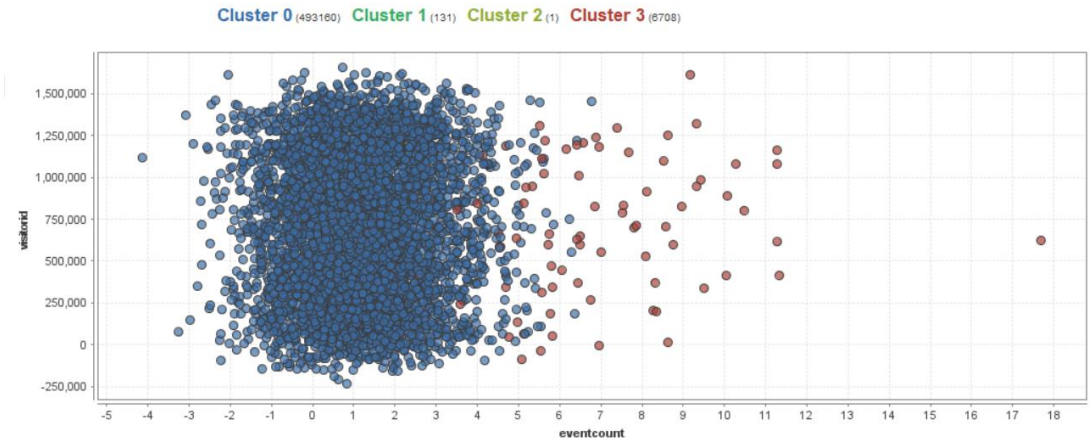


Figure 4.37: Scatter plot view for the second cluster

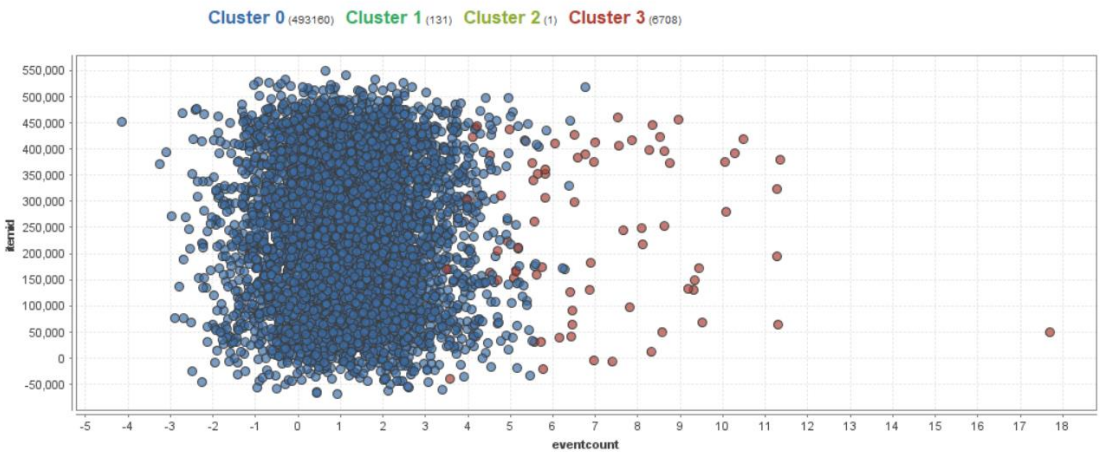


Figure 4.38: Scatter plot view for the cluster 3

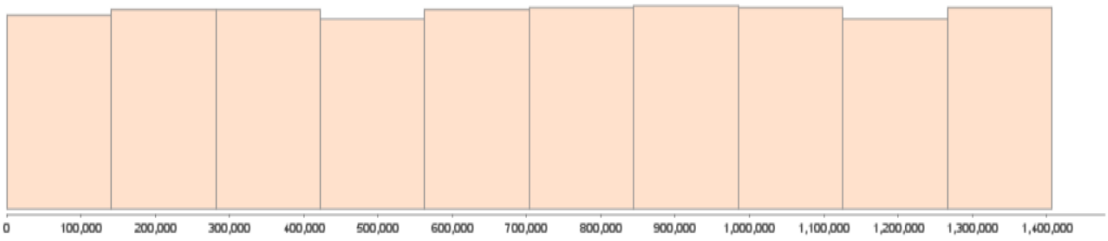


Figure 4.39: Bar chart for the visitor distribution

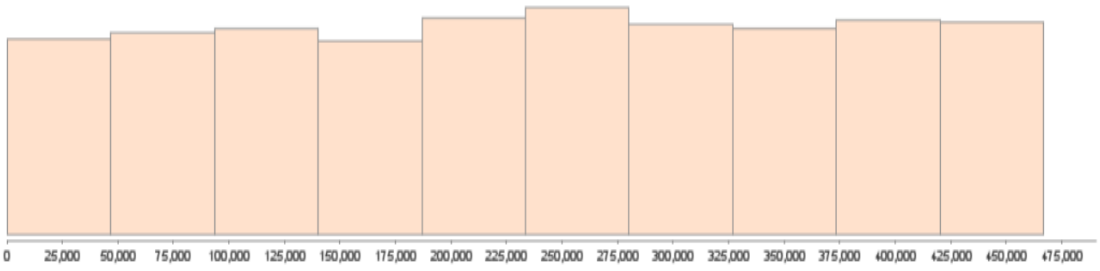


Figure 4.40: Item id distribution (Bar Chart)

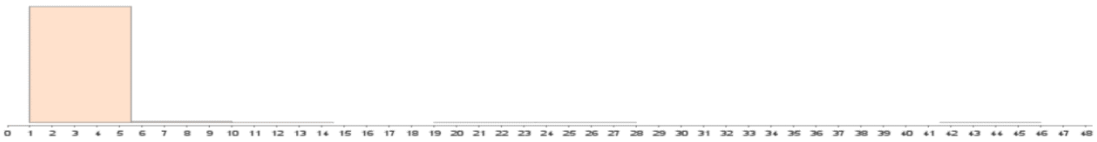


Figure 4.41: Event count distribution (Bar Chart)

4.7.2 Comparison of Preprocessing techniques used in existing research and Retail Rocket Data Set

Table 4.2: Compare preprocessing techniques applied in different research

Model	Data Set	Preprocessing Steps	How train test data sets were created	Data Volume after applying the step
Alternating Least Squares	Y!Music	Remove items that has less amount of ratings than 80.	Not specified	Training: 22699314 Test set : 303516
	Netflix	No	Not clearly specified	Training : 22783659 Test : 384573
SLi_Rec	Amazon data set	No	Data was collected for 10 days. First 7 day data used as the training and other 3 days as test data. Moreover, 50% of data was used to hyper parameter tuning.	No
	Netflix data set	No	No	No

GRU4Rec	Recsys challenge 2015	Removed session length equal to 1.	No	Purchase events (test set) : 71222 Training set : 31637239
	Youtube OTT data set	Filter out very long and short sessions.	No	Training : 13 million view events 180000 view events
Caser	MovieLens, Gowalla, Foursquare, Tmall	Removed all the user item interactions where their value was lesser than predefined number of feedback.	First 70% of data as training and rest as test.	MovieLens Users : 6k Items : 3.4k Gowalla Users : 3.1k Items : 14k Foursquare Users : 10.1k Items : 23,4k Tmall Users : 28.8k Items : 12.2k
NextItem	YOO data set	Remove data where session length is shorter than 3.	No	Training : 0.14M Testing : 0.07M
	LibFM		Two data sets were created. They are MUSIC_M (20000 songs) and	MUSIC_M5 Total : 0.61M

			MUSIC_L (200000 songs). Then new 6 data sets created MUSIC_M5, MUSIC_L5, MUSIC_L10, MUSIC_L20, MUSIC_L50 and MUSIC_L100. Numeric values represent how many time a user listened to the movie. 50% of data used as training, 45% testing and 5% for validation.	MUSIC_L5 Total : 2.14M MUSIC_L10 Total : 1.07M MUSIC_L20 Total : 0.23M MUSIC_L50 Total : 0.2M MUSIC_L100 Total : 0.11M
Multi-Interest-Aware Sequential User Modeling (SUM)	Display Ads Data set	No	No	No
	Taobao Data set	Only selected user sequences greater than 100.	No	No
BPR	Rosmann data set	No	No	No
	Netflix data set	No	No	No
BiVAE auto encoder	Movie lens	Removed users item interactions that do not has at least 20 ratings.	10% test and 80% training And 10% validation.	Total data : ML-100K,ML-1M, ML-20M

	Amazon movie data set	Removed user where they have less than 5 ratings.	10% test and 80% training And 10% validation.	Office – 51,453, Clothing – 272,765, Sports – 439650, Epinions – 553354
Variational Auto encoder	Movie lens 20M data set	Removed users who have watched movies less than 5.	Training / test and validation data set.	No
	Netflix	Removed users who have watched movies less than 5.	Training / test and validation data set.	No
	Million song data set	Removed users that have listened less than 20 movies and removed songs that are listened less than 200 by users	Training / test and validation data set.	No
Neural Collaborative Filtering	MovieLens	Removed users where they have less than 20 ratings. Changed ratings	Training and Test split	Total data points : 1000209 Item : 3706 User : 6040
	Pinterest	Removed users where the number of	Training and Test split	Total data points :

		interactions were lesser than 20.		Item : 9916 User : 55187
Restricted Boltzman Machine (RBM)	Netflix data set	No	Data set was drawn randomly from the total data set.	Total : 2817131 No of users : 480189 No of items : 17770
Deep and Wide	No	No	No	No

Considering, preprocessing our data set contains a large amount of view events, which do not contribute for purchasing a product. Compared to other research, removing view events had left only close to 70000 records for us to train the recommendation system. However, in other research, removing outlier data points had left them still a significant number of data to train and evaluate the models. Moreover, considering the data quality for the visitor id the reported ID ness, stability missing and textness was 6.28%, 0.18, 0 and 0 respectively. For the item id this was 3.14%, 0.14, 0 and 0 respectively. Finally for the event counts the value was 0, 84.42%, 0 and 0 respectively. Furthermore, considering spread of values frequencies, visitor and item ids are equally spread in the value ranges of 0-1,000,000 and 0 – 75,000 respectively [Figure 4.39 and 4.40]. However, for event counts, the value frequencies are only between 1 and 6 [Figure 4.41].

Furthermore, when the data set was clustered using 4 clusters, all data get clustered into a single cluster [Figure 4.32]. Furthermore, the event counts were significant in generating clusters [Figure 4.33]. When distance-based clustering was used, approximately only 2000 records were identified as the outliers. The figure 4.35 shows the cluster centers based on visitor id, item id and event counts and figure 4.34 on how cluster center changes with the visitor id, item id and event counts.

4.7.3 Comparison of feature engineering techniques used in existing research and Retail Rocket Data Set.

Next we will compare different feature engineering techniques used in different research.

Table 4.3: Compare different feature engineering techniques applied in research

Model		Feature Engineering Steps
Alternating Least Squares	Y!Music	The items were arranged according to their taxonomies. The lower level nodes were considered to be tracks and higher level as albums and genres. In these hierarchies the relationship between nodes were not considered.
	Netflix	For each rating that has a value between 1-5 was assigned with 1.
SLi_Rec	Amazon data set	No
	Industrial data set	Embedding were derived from advertisement title and from items.
GRU4Rec	RecSys 2015, Youtube OTT	The state of the session is calculated. If the session contains items, then 1-N encoding was used to calculate the session state. If the sessions contain the events then, weighted sum was used to find the session state. Vectors were normalized as well.
Caser	MovieLens, Gowalla, Foursquare, Tmall	Converted all the ratings to 1.
NexItem	Yoo and LibFM	No

Multi-Interest-Aware User Modeling	Display Ads Data set	Generated 128 long text from the text encoder
	Taobao data set	Items and category id was represented as one hot encoding.
BPR	Rossmann and Netflix data set	No
BiVAE	Movie Lens and Amazon data sets. Training : 80%, Test : 10% and validation : 10%	Embedding size was set to 20.
Variational Auto Encoder	Movie Lens, Netflix and Million song data set	No
Neural Collaborative Filtering (NCF)	Movie Lens and Pinterest	No
Restricted Boltzman machine (RBM)	Netflix data set	No
Wide and deep	No	No

In the research I have used a weighted scoring scheme to derive the scores for user and item interactions. For purchase events I have assigned a weight value of 2 and for addToCart, I have assigned a value of 1. This is because, addToCart events were nearly 2 times of purchase events. Due to this reason, to minimize the effect of more addToCart events, I have assigned a weight value of 2 to purchased events. Then, I have applied binning, to normalize the score between 1 and 5. This is done by analyzing the data set and to ensure high score containing user item interactions will get rating near to 5 and for smaller scores near to 1.

4.7.4 Training Procedures used in Research and one used in the research

Next I will be looking at the different training procedures.

Table 4.4: Compare different training and evaluation procedures used in research

Model	Training procedure	Evaluation procedure
Alternating Least Squares	To train RankSGD and RankSGD2 learning rate was used was 0.008 and 0.016 respectively. The model training was stopped when the test performance started to drop. Model performance was recorded with change of the feature size. Model accuracy was compared with the change of the number of iterations.	Used following models for comparison. Pop, ImplicitALS, RankSGD, RankSGD2, Rank ALS. Considering KDD 2011 data set, all the data was used to measure Error rate, ARP and recall. For Netflix data set, randomly drawn data was used to measure recall@50
SLI-Rec	Model performance was compared with ASVD, DIN, LSTM, NARM, RRN, LSTM++, CA-RNN, T-LSTM and DIEN. Adam was used as the model optimizer. Embedding size used for both RNN and MLP layers was 18. Best hyper parameters for each data set was found using the validation data set. Here the learning rate was found as 0.001, L2 regularization as 0.0001 with no drop outs and batch	AUC and F1 score was used as the evaluation metrics.

	normalization (after concatenating user and item embeddings). To verify the effectiveness of the time and context aware controllers variants of each models were compared.	
GRU4Rec	POP, S-POP and Item-KNN was used for comparison. The models were trained using session parallel mini batches. First a random data points were selected and those were used to find the best parameters for all the model. Then the all the models were re trained using training and validation data set and evaluated with final test set. Experiment was carried out to find whether GRU or LSTM better for the model and how model performance change with different loss functions.	Ranking Loss was used as the evaluation metric.
Caser	Model performance compared with POP, BPR, FMC, FPMC, Fossil and GRU4Rec was used. Grid search was used to find the optimal hyper-parameters with the validation data set. Different feature dimensions	Precision , Recall and MAP was used as the evaluation metrics.

	was used with learning rates ranging from $1 – 10^{-4}$. For Fossil markov order ranging from 1 to 9 used for Fossil, Caser and for GRU4Rec. For Caser horizontal filter size ranged from 1 to L and targets from 1 to 3. Caser performance was measured with identity , sigmoid , tanh and relu. Then model performance was measured using filters ranging from 4 to 64 and vertical filters ranging from 1 – 16.	
NextItem	The model performance was compared with MostPop, GRU4Rec and Caser. For each model, best parameters were found using the validation data set and learning rate and batch size was set manually.	MRR@N, HR@N and NDCG@N was used as the evaluation metrics.
Multi-Interest-Aware User Modeling (SUM)	Model performance was evaluated with AttMerge, GRU, SGRU, HRNN, NTM, RUM, MCPRN, MIMN and HPMN. User, Item and target items were concatenated together and fed into 2 layer fully	Group AUC, LogLoss and NDCG

	connected layer with Relu activation function.	
BPR	The model performance was compared with BPR-MF, BPR-KNN, WR-MF, SVD-MF, Cosine-KNN, and MostPop.	AUC was used with leave one out technique.
BiVAE	The model performance was compared with HPF, Gauss-PF, NeuMF and VAE. For auto variational encoder family MLPs with 0,1,2 and 3 layers was tested with using 20-dimension layer form infer and 0 hidden layer for decoding. Tanh was used as activation function and experimented with learning rates ranging from $1e^{-4}$ to $1e^{-1}$. Batch size was set to 128 and optimizer as Adam.	NDCG and Recall was used as the evaluation metrics.
Variational auto encoder	The model performance was compared with WMF, SLIM, CADE, and NCF. First using the validation data set, best number of model layers was found. Tanh was used as the activation function between layers. The model was trained using Adam optimizer, for the batch size of 500. Number of epochs used to train models was 100 and 200.	NDCG and Recall

Neural Collaborative Filtering (NCF)	The model performance was compared with ItemPop, ItemKNN, BPR and eALS. Validation data set was used to tune hyper parameters. They were found by optimizing log loss. Model parameters were initialized randomly using Gaussian distribution. Adam was used as the optimizer and learning rates were changed from 0.0001 to 0.005. Embedding size was change from 8 to 64. The model performance was recorded using pre training and changing the number of MLP layers.	Hit Rate and NDCG was used as the evaluation metrics.
Restricted Boltzmann Machine	The model was trained with the number of features equal to 100 and mini batches size of 1000. The number of epochs used to train the models was 40 – 50 and in order to update the weights learning rate of 0.01 momentum of 0.9 and weight decay of 0.01 was used.	RMSE
Wide and Deep Networks	Given the input data set, the model dense feature set was generated. Then embedding generated from the products with dense features concatenated into a new vector and then input into 3 Relu layers. To overcome the problem of retraining the model with the new incoming inputs, the models were	A/B Testing was used

	initialized with the previous weights and learnt embedding.	

The research problem contains two parts. That is problem verification and the solution. The problem as follows. Our data set contains large amount of view events and a small amount of purchase and addToCart events. Hence, once the model is trained with this data, then the model will recommend users products that is been viewed by the users. To verify this behavior, I have followed the following procedure.

1. For the first five data set, I have selected user item interactions where they contain at least on purchase, addToCart and view event. Then, I have defined a weight score for each event. This is done in a way for the first 3 data sets, addToCart + purchase events get more weights, while for next 2 data sets, for view events, more weights been added.

2. For the next five data sets, I have varied the amount of addToCart + purchase events and view events. For the first 3 data sets, I have included more addToCart and purchase events, while for other two data sets, I have increased the view events.

3. Please refer section 3.1 for more information.

I have created 10 data sets to study, how volume of view events and weights assigned to them have an influence on recommendation made by the recommendation system. It was found that, in most cases, when the view event volume is very high or when more weights are assigned to them, and used those data to train the recommendation system, the system will recommend products that are been viewed by the users. Please refer section 4.1 for more details.

At the second phase of the research, I have removed the all user item interactions where they contain view events and then trained the model. To generate train test data, I have used a 80/20 random split and, used kfold cross validation to train the model and reported HR@10 and NDCG@10 on test data set. Please refer section 3.2 for more detail.

4.7.5 Compare different results obtained in research with this research

Finally, I will be comparing the model results with the best results reported in the research for different models.

Table 4.5: Compare different results obtained in different research

Model	Data Set	Performance
Alternating Least Squares	Y!Music	Error rate : 0.09 , ARP : 0.02 , Recall : 0.42 (Features = 50) Iteration 10: Error rate : 0.04
	Netflix	Error Rate : 0.2, ARP : 0.07, Recall : 0.18 (Features = 50)
Sli-Rec	Entire, Elec, Movies, CDs, Ads	Entire : 0.8494, Elec : 0.8282, Movies : 0.8769, CDs : 0.9272, Ads : 0.6654 (AUC results)
GRU4Rec	Recsys 2015	Recall@20 : 0.64, MRR@20 : 0.27
	Video	Recall@20 : 0.67, MRR@20 : 0.39
Caser	MovieLens	Prec@5 : 0.21, Prec@10 : 0.19, Recall@5 : 0.06, Recall@10 : 0.11, MAP : 0.15
	Gowalla	Prec@1 : 0.19, Prec@5 : 0.11, Prec@10 : 0.08, Recall@1 : 0.03, Recall@5 : 0.08, Recall@10 : 0.12, MAP : 0.09
	Foursquare	Prec@1 : 0.13, Prec@5 : 0.06, Prec@10 : 0.04, Recall@1 : 0.05, Recall@5 : 0.10, Recall@10 : 0.12, MAP : 0.09
	Tmall	Prec@1 : 0.03, Prec@5 : 0.01, Prec@10 : 0.01, Recall@1 : 0.01, Recall@5 : 0.03, Recall@10 : 0.05, MAP : 0.03

NextItem	YOO	MRR@5 : 0.17, HR@5 : 0.28, NDCG@5 : 0.20
	MUSIC_M5	MRR@5 : 0.31 , HR@5 : 0.37, NDCG@5 : 0.32
	MUSIC_L10	MRR@5 : 0.25, HR@5 : 0.30, NDCG@5 : 0.27
	MUSIC_L100	MRR@5 : 0.25, HR@5 : 0.30, NDCG@5 : 0.27
Multi-Interest-Aware User Modeling (SUM)	Taobao	gAUC : 0.94, LogLoss : 0.18, NDCG : 0.92
	Display Ads	gAUC : 0.83, LogLoss : 0.37, NDCG : 0.74
BPR	Rossmann data set	AUC : 0.89
	DVD Rental data set Netflix	AUC : 0.93
BiVAE	Amazon Office	NDCG : 0.07, Recall : 0.20
	Amazon Clothing	NDCG : 0.02, Recall : 0.07
	Amazon Sports	NDCG : 0.03, Recall: 0.11
	Amazon Epinions	NDCG : 0.04, Recall : 0.12
	ML-100k	NDCG : 0.19, Recall : 0.38
	ML-1M	NDCG : 0.15, Recall : 0.28
	ML-20M	NDCG : 0.16, Recall : 0.31
Variational auto encoder	ML-20M	Recall@20 : 0.39, Recall@50 : 0.53, NDCG@100 : 0.42
	Netflix	Recall@20 : 0.35, Recall@50 : 0.44, NDCG@100 : 0.38
	MSD	Recall@20 : 0.26, Recall@50 : 0.36, NDCG@100 : 0.31
NCF (Neural Collaborative Filtering)	MovieLens	HR@10 : 0.74 , NDCG@10 : 0.43 (Factors 64)
	Pinterest	HR@10 : 0. 89 , NDCG@10 : 0.55 (Factors 16)
Restricted Boltzmann Machine (RBM)	Netflix Data Set	0.93(Factors 10)
Wide and Deep	A/B testing	Offline AUC : 0.728

Finally, I will be comparing the obtained results. First, I will be comparing the obtained results, with the existing work. I will be using the Hit Rate, for comparing the results with other research. In the existing research, almost all the models have used NDCG, AUC, Precision and Recall to evaluate the model performance. Only neural collaborative filtering had used Hit rate to evaluate its model performance. However, neural collaborative filtering had not used the data set I have used, as the data patterns in movie lens and Pinterest data set is different (Refer Section 4.7.1). The NCF had reached a Hit rate near to 1 when learning rate was around 0.001. The model had recorded a HR@10 value of 0.34 when the predictive factors were reduced to 8. Furthermore, the model had reported a hit rate near to 0.25 when the number of predictive factors were increased to 32 and MLP layers were increased from 1 to 4.

5 RELATED WORK

In this section, I will discuss about the related work. In summary, the problem as follows [96]. Objective functions are more bias towards abundant data points. However, in some cases, the abundant data may not reflect a stronger preference of a user over the others. As a result, a model trained based on such data, may not recommend items that has a stronger user preference.

One way to overcome this problem is to normalize by volume. Related to news publishing [97], it is essential to predict number of online comments. However, due to the large amount of the data, it is essential to normalize the them. In this context, log-normal and negative binomial was used to model news comments. These models were used to normalize the raw comment counts as well.

6 CONCLUSION AND RECOMMENDATION

To summarize, for the learning rate value of 0.0001, we have managed to achieve a HR@10 of 1 with neural collaborative filtering. NCF managed to reach a HR@10 value of 0.34 when number of embedding size was reduced to 8. We were able to record a HR@4 value of 0.12 for the neural collaborative filtering. With pre training, we were able to achieve a HR@10 of 0.32. Finally, with 3 MLP layers and with only 8 embedding size, we managed to achieve a HR@10 of 0.295.

Before removing the view events, the best HR@10 we received was 0.1. Hence, it is clear that removing view events and using neural collaborative had helped recommendation systems improve their hit rate. That is, we can recommend more purchased products and improve model effectiveness in terms of HR@10.

In this research I have used neural network learning-based method for recommending purchase products for users. Here, neural collaborative filtering is an extension of Generalized matrix factorization. Generalized matrix factorization outputs two low dimensional vectors where they were been given as an input to a deep neural network.

Here during this research, we had relatively small number of purchase and add to cart events to train our model. To increase the amount of purchase event recommendations and improve model HR@10, it is important to identify other sources of information such as user search history and browsing history to identify, types of products he or she is looking for and use this information to train the model. This can help reduce sparseness of the user item interaction matrix. In this context, using contextual event data and product information could help improve model performance significantly. In this context using sequential recommendation models based on deep learning could help models improve their performance as well. Furthermore, it is worthwhile to investigate whether reinforcement learning can be used to get user feedback, so that, we may have more rich information in the data to train deep learning models. However, these need to be done as a part of future work.

In this work, I have used batch learning. However, it is interesting to investigate whether streaming libraries such as River can be used to recommend purchased products to users [94].

Knowledge graphs [95] are used in e commerce, to recommend purchased products to users. One such application is search and recommendation which uses user shopping basket information to recommend products to users. For example, products can be categorized based on their features and then those features can be used to search products. Then, the characteristics of the filtered products [Figure 6.1] can be used to recommend products belongs to different categories [Figure 6.2].

In a knowledge graph, nodes function as entities. These graphs have the capability to model relationships between users and products more accurately. This in return help models improve their recommendation accuracy and make model more explainable. Resource description format is a widely used standard to represent knowledge graphs. In this representation node can be a tail or a head entity and they are related to each other via a relation. Some of the commonly used knowledge graphs includes YAGO, Freebase, DBpedia, Satori, CN-Dbpedia, NELL, wikidata, Google knowledge graph, Bio2RDF and knowlife.

One possible future direction of knowledge graph is to learn knowledge graphs dynamically because user preferences change with the time rapidly. Multitask learning can be used in knowledge graph creation. This can be useful, in predicting missing relations and entities. When the data set is denser, and interaction between domains can be not equal in many domains. Hence, in this context transfer learning can be used to learn from the source domain and learn about the target domain. Finally, to improve recommendation quality, side information such as demographic information can be incorporated. This information can be used to improve user relationships.

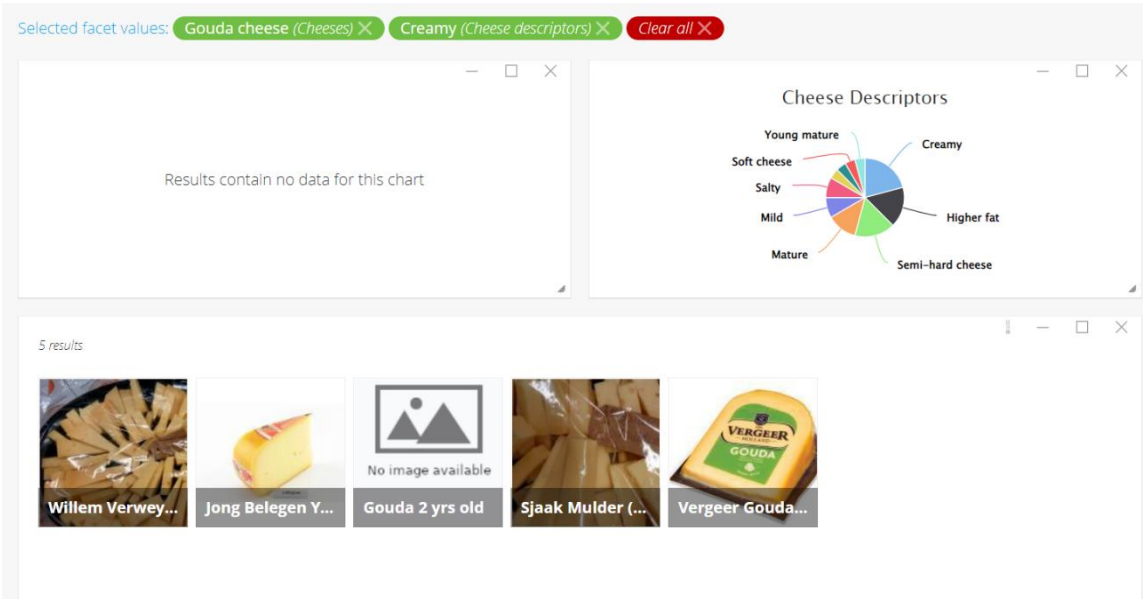


Figure 6.1: Results for the given criteria

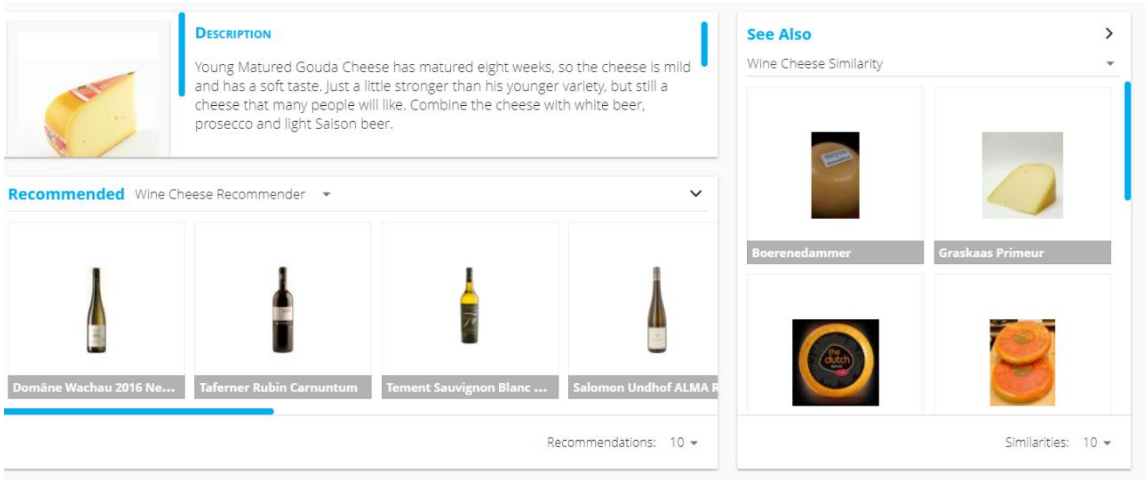


Figure 6.2: Recommended products for a given result

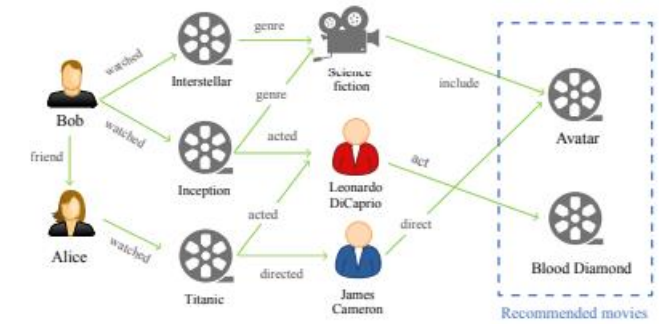


Fig. 1. An illustration of KG-based recommendation.

Figure 6.3: Internal Representation of a knowledge graph

Finally, frequent pattern mining methods such as Apriori, FpGrowth or event ECLAT can be used to recommend purchased products to users based on their history of product purchases.

7 REFERENCES

[1] Y Koren, R Bell, C Volinsky, Matrix factorization techniques for recommender systems. Computer 42 (8), 30-37.

[2] G Takács, D Tikk, Alternating least squares for personalized ranking. Proceedings of the sixth ACM conference on Recommender systems, 83-90.

[3] M. Jahrer and A. T¨oscher. Collaborative filtering ensemble for ranking. In Proc. of KDD Cup Workshop at 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD ’11, San Diego, CA, USA, 2011.

[4] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In Proc. of 8th IEEE Int. Conf. on Data Mining, ICDM ’08, pages 263–272, Pisa, Italy, 2008.

[5] I. Pil´aszy, D. Zibriczky, and D. Tikk. Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In Proc. of the 4th ACM conference on Recommender Systems, RecSys ’10, pages 71–78, Barcelona, Spain, 2010.

[6] Zeping Yu, Jianxun Lian, Ahmad Mahmoody, Gongshen Liu, Xing Xie. Adaptive User Modeling with Long and Short-Term Preferences for Personailzed Recommendation. In Proceedings of the 28th International Joint Conferences on Artificial Intelligence, IJCAI’19, Pages 4213-4219. AAAI Press, 2019.

[7] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 426–434. ACM, 2008.

[8] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li,and Kun Gai. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery And Data Mining, KDD’18, pages 1059–1068, New York, NY, USA, 2018. ACM.

[9] Jing Li, Pengjie Ren, Zhumin Chen, ZhaochunRen, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pages 1419–1428. ACM, 2017.

[10] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In Proceedings of the tenth ACM international conference on web search and data mining, pages 495–503. ACM, 2017.

[11] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. Context-aware sequential recommendation. In 2016 IEEE 16th International Conference on Data Mining (ICDM), pages 1053–1058. IEEE, 2016.

[12] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang,Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: Modeling user behaviors by time-lstm. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI'17,pages 3602–3608. AAAI Press, 2017.

[13] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In ThirtyThird AAAI Conference on Artificial Intelligence, 2019.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, Domonkos Tikk. Session-based Recommendations with Recurrent Neural Networks. ICLR (Poster) 2016.

[15] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In UAI'09: 25th Conf. on Uncertainty in Artificial Intelligence, pp. 452–461, 2009. ISBN 978-0-9749039-5-8.

[16]. Tang, Jiaxi, and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. ACM, 2018.

[17] Steven Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In International Conference on World Wide Web. ACM, 811–820.

[18] R. He and J. McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In International Conference on Data Mining. IEEE.

[19] Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J. M., & He, X. A Simple Convolutional Generative Network for Next Item Recommendation. WSDM, 2019.

[20] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015).

[21] Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009, June). BPR: Bayesian personalized ranking from implicit feedback.

[22] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In IEEE International Conference on Data Mining (ICDM 2008), pages 263–272, 2008.

[23] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In IEEE International Conference on Data Mining (ICDM 2008), pages 502–511, 2008.

[24] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang & Meng Wang, LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation, 2020.

[25] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In KDD Workshop on Deep Learning Day.

[26] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In KDD (Data Science track). 974–983.

[27] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In WWW. 173–182.

[28] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In SIGIR. 515–524.

[29] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In RecSys. 140–144.

[30] Liang, Dawen, et al. "Variational autoencoders for collaborative filtering." Proceedings of the 2018 World Wide Web Conference. 2018.

[31] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. 263–272.

[32] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In Data Mining (ICDM), 2011 IEEE 11th International Conference on. 497–506.

[33] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. 153–162.

[34] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web. 173–182.

[35] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In WWW, pages 285–295, 2001.

[36] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In SIGIR, pages 549–558, 2016.

[37] R Salakhutdinov, A Mnih, G Hinton,Restricted Boltzmann machines for collaborative filtering,Proceedings of the 24th international conference on Machine learning, 791-798.

[38] Wang, Hongwei, et al. "DKN: Deep Knowledge-Aware Network for News

Recommendation." Proceedings of the 2018 World Wide Web Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2018.

[39] Steffen Rendle. 2012. Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST) 3, 3 (2012), 57.

[40] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In Proceedings of the International Joint Conference on Artificial Intelligence.

[41] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In CIKM. ACM, 2333–2338.

[42] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 7–10.

[43] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In Proceedings of the 26th International Joint Conference on Artificial Intelligence.

[44] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 191–198.

[45] Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In Proceedings of the 26th International Joint Conference on Artificial Intelligence.

[46] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems. 3146–3154.

[47] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu and Xing Xie: Neural News Recommendation with Long- and Short-term User Representations, ACL 2019.

[48] Heng-Tze Cheng, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, and Wei Chai. 2016. Wide & deep learning for recommender systems. In DLRS, pages 7–10.

[49] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In EMNLP, pages 1746–1751.

[50] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based news recommendation for millions of users. In KDD, pages 1933–1942.

[51] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang and Xing Xie: Neural News Recommendation with Attentive Multi-View Learning, IJCAI 2019.

[52] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang and Xing Xie: NPA: Neural News Recommendation with Personalized Attention, KDD 2019, ADS track.

[53] Pratik Jawanpuria, Arjun Balgovind, Anoop Kunchukuttan, Bamdev Mishra. _Learning Multilingual Word Embeddings in Latent Metric Space: A Geometric Approach_. Transaction of the Association for Computational Linguistics (TACL), Volume 7, p.107-120, 2019.

[54] Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, Fangxi Zhang. A Hybrid Collaborative Filtering Model withDeep Structure for Recommender Systems. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), p.1309-1315, 2017.

[55] Salakhutdinov, R., and Mnih, A. 2011. Probabilistic matrix factorization. In NIPS, volume 20, 1–8.

[56] Singh, A. P., and Gordon, G. J. 2008. Relational learning via collective matrix factorization. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 650–658. ACM.

[57] Wang, H.; Wang, N.; and Yeung, D.-Y. 2015. Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1235–1244. ACM.

[58] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah, Wide & Deep Learning for Recommender Systems, Proceedings of the 1st workshop on deep learning for recommender systems, 7-10.

[59] Asela Gunawardana and Guy Shani: A Survey of Accuracy Evaluation Metrics of Recommendation Tasks

[60] Dimitris Paraschakis et al, "Comparative Evaluation of Top-N Recommenders in e-Commerce: An Industrial Perspective", IEEE ICMLA, 2015, Miami, FL, USA.

[61] Lisa Li, et al, Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization, The Journal of Machine Learning Research, Volume 18 Issue 1, pp 6765-6816, January 2017.

[62] James Bergstrat et al, Algorithms for Hyper-Parameter Optimization, Procs 25th NIPS 2011.

[63] D Leony, A Pardo, HA Parada, C Delgado Kloos, A Cloud-based Architecture for an Affective Recommender System of Learning Resources, Proceedings of the 1st International Workshop on Cloud Education, 2012.

[64] Harini Suresh and John V Guttag. A framework for understanding unintended consequences of machine learning. 2019.

[65] N Mehrabi, F Morstatter, N Saxena, K Lerman, A Galstyan,A survey on bias and fairness in machine learning,arXiv preprint arXiv:1908.09635.

[66] Nazanin Alipourfard, Peter G Fennell, and Kristina Lerman. 2018. Can you Trust the Trend?: Discovering Simpson's Paradoxes in Social Data. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. ACM, 19–27.

[67] Nazanin Alipourfard, Peter G Fennell, and Kristina Lerman. 2018. Using Simpson A -Zs Paradox to Discover Interesting Patterns in Behavioral Data. In Twelfth International AAAI Conference on Web and Social Media.

[68] H Abdollahpouri, R Burke, B Mobasher, Controlling popularity bias in learning-to-rank recommendation, RecSys '17: Proceedings of the Eleventh ACM Conference on Recommender Systems, August 2017, Pages 42–46.

[69] Zhao X., Niu Z., Chen W. (2013) Opinion-Based Collaborative Filtering to Solve Popularity Bias in Recommender Systems. In: Decker H., Lhotská L., Link S., Basl J., Tjoa A.M. (eds) Database and Expert Systems Applications. DEXA 2013. Lecture Notes in Computer Science, vol 8056. Springer, Berlin, Heidelberg.

[70] H Abdollahpouri, R Burke, B Mobasher, Managing Popularity Bias in Recommender Systems with Personalized Re-ranking, The 32nd International FLAIRS Conference in Cooperation with AAAI.

[71] S Khenissi, O Nasraoui, Modeling and counteracting exposure bias in recommender systems, arXiv preprint arXiv:2001.04832.

[72] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. CoRR abs/1602.05352 (2016). arXiv:1602.05352.

[73] A Collins, D Tkaczyk, A Aizawa, J Beel, Position bias in recommender systems for digital libraries, International Conference on Information, 335-344.

[74] S Krishnan, J Patel, MJ Franklin, K Goldberg, A methodology for learning, analyzing, and mitigating social influence bias in recommender systems,Proceedings of the 8th ACM Conference on Recommender systems, 137-144.

[75] Roselli, Drew and Matthews, Jeanna and Talagala, Nisha, Managing bias in AI, Companion Proceedings of The 2019 World Wide Web Conference 539-544 2019.

[76] Benedikt Schifferer,Chris Deotte and Even Oldridge, RecSys '20: Fourteenth ACM Conference on Recommender SystemsSeptember 2020 Pages 754–755.

[77] https://developers.google.com/machine-learning/recommendation/.

[78] https://github.com/microsoft/recommenders/

[79] https://www.fast.ai/2018/08/07/hbr-bias-algorithms/

[80] Multivariate time-series anomaly detection via graph attention network Multivariate time-series anomaly detection via graph attention network Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, Qi Zhang 2020 IEEE International Conference on Data Mining (ICDM), 841-85 2020 IEEE International Conference on Data Mining (ICDM), 841-85.

[81] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, “Robust anomaly detection for multivariate time series through stochastic recurrent neural network,” in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2828– 2837.

[82] Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, “Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding,” in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 387–395.

[83] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: an ensemble of autoencoders for online network intrusion detection,” arXiv preprint arXiv:1802.09089, 2018.

[84] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, “Deep autoencoding gaussian mixture model for unsupervised anomaly detection,” in International Conference on Learning Representations, 2018.

[85] D. Li, D. Chen, J. Goh, and S.-K. Ng, “Anomaly detection with generative adversarial networks for multivariate time series,” arXiv preprint arXiv:1809.04758, 2018.

[86] D. Li, D. Chen, L. Shi, B. Jin, J. Goh, and S.-K. Ng, “Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks,” arXiv preprint arXiv:1901.04997, 2019.

[87] D. Park, Y. Hoshi, and C. C. Kemp, “A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder,” IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1544–1551, 2018.

[88] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou and Tony Xing, Mao Yang, Jie Tong, Qi Zhang. 2019. Time Series Anomaly Detection Service at Microsoft. In The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’19), August 4–8, 2019, Anchorage, AK, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330680.

[89] Rasheed, Peter Peng, Reda Alhajj, and Jon Rokne. 2009. Fourier transform based spatial outlier mining. In International Conference on Intelligent Data Engineering and Automated Learning. Springer, 317–324.

[90] Owen Vallis, Jordan Hochenbaum, and Arun Kejariwal. 2014. A Novel Technique for Long-Term Anomaly Detection in the Cloud. In 6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 14). USENIX Association, Philadelphia, PA.

[91] https://github.com/linkedin/luminol

[92] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. 2018. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In Proceedings of the 2018 World Wide Web Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 187–196.

[93] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. 2017. Anomaly detection in streams with extreme value theory. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 1067–1075.

[94] https://riverml.xyz/0.12.1/

[95] Q Guo, F Zhuang, C Qin, H Zhu, X Xie, H Xiong, Q He, A survey on knowledge graph-based recommender systems, IEEE Transactions on Knowledge and Data Engineering.

[96] Challenges and research opportunities in ecommerce search and recommendations, M Tsagkias, TH King, S Kallumadi, V Murdock, M de Rijke, ACM SIGIR Forum 54 (1), 1-23.

[97] Manos Tsagkias, Wouter Weerkamp, and Maarten de Rijke. News Comments: Exploring, Modseling, and Online Prediction. In ECIR, pages 191–203. Springer, 2010.