Fairness in Recommendation Systems

Author names: Kavya Ravella, Nikita Khedekar

Abstract*—Recommendation Systems are one of the most important applications in machine learning which provides many services to give relevant suggestions to the users and match them to products and information. Fairness in recommendation system is an essential to make sure there is no unbalanced recommendations. When a user has rated 8 Indian restaurants and 2 Italian restaurants, it is more likely that that the user will be a personalized list of restaurant recommendations where 80% are Indian restaurants and 20% are Italian restaurants. The recommendations will comprise of all his preferences in specific percentages without ignoring the least rated ones and hence leading to balanced recommendations. This is called calibration. Calibration has recently received renewed attention in the context of fairness in machine learning. In Calibration based on the user's history the areas of interest are taken into consideration with their corresponding proportions. Calibration is used to improve accuracy of the recommendations. The outline metrics and re-ranking algorithms are simple yet effective and are used to post-process the output of the recommendation systems.

I. Introduction

A recommendation system is a class in machine learning which provides a subclass of information filtering system that predicts the preferences of a user and the rating they would give to an item. They are mostly used for commercial purposes. They are a reason for some industries to develop economically and also stand out significantly from their competitors.

Items are ranked according to their relevancy and the most relevant ones are shown to the user. Recommendation systems determine this relevancy with the help of historical data. They are divided into two main categories: Content-based and Collaborative filtering.

In this project, we will be using Collaborative filtering. The recommendation system will be trained to generate the recommended items to the user based on his main areas of interests. It is done by training the recommender system with ranking metrics. The most accurate ones will be given the first preference and vice versa. The items the user is least interested in will be overlooked or even absent. This will lead to unbalanced recommendations. This leads to a set of users not provided with fair recommendations while the other set get it according to their exact preferences. Hence there is a need to evaluate the fairness in recommendation systems.

Calibration is a concept of machine learning where the actual output is compared with the expected output. This concept will be used to evaluate the fairness in the recommendation systems. In this concept, the machine learning model will be made to improve using the training data such that the probabilities of the recommendations predicted will be similar to the probabilities observed.

<u>Goal</u>:

Evaluate the fairness of objects in a recommendation system using calibration.

There are three ways to achieve this:

1. Fairness for objects
2. Fairness for subjects
3. Multi-stakeholder recommendation

In our project, we are targeting to achieve fairness in recommendation system using techniques such as SVD++ and Calibrated Filtering. Evaluate these approaches and design a model which will help to improve the prediction accuracy based on proportions of user's interests, history of ratings and likings.

<u>Data</u>:

There are two datasets provided:

1. **Yelp Hotel Dataset:** It contains Yelp hotel ratings, reviewer information and hotel information.
2. **Yelp Restaurant Dataset:** It contains Yelp restaurant ratings, reviewer information and restaurant information.

We worked on the Yelp Restaurant dataset.

II. Data preprocessing

Useful attributes like userID, itemID, rating and item categories were extracted from the provided Yelp Restaurant dataset. As per requirement, the unique users, items and item categories were extracted. A data frame was created with the above attributes.

For creating a dataset, inner joins were applied on review, restaurant and reviewer tables. Inner joins were applied on review and restaurant table on restaurant ID and on review and reviewer table on reviewerID. This data frame had attributes as
userID: reviewerID from reviewer table
itemID: restaurantID from restaurant table
rating: rating from review table
itemCategory: categories from restaurant table.

The dataset contains:
16941 unique userIDs
235717 unique itemIDs
21516 unique item categories.

Further processing on the data frame was done to get more detailed data for our experiments. We applied different ranking approach to reduce size of dataset as it is too huge. After ranking it, according to the average rating, the dataset was still huge and kernel issues were being faced.

The dataset was further made more specific by using the concept of quantile on the total ratings. The number of reviewers per restaurant were calculated and stored in the total ratings column. We have mentioned the quantile as 80 according to which all restaurants with more than 39 reviewers were only considered in this project. This was done to tackle with memory and kernel issues.

Item mapping was done on the dataset to store each restaurants' category distribution. The item mapping plays a n important role in determining whether our recommendations are calibrated or not.

Tools/ Packages/ Libraries:

Python, Jupyter, numpy (performing calculations), Scikit, pandas (Processing and creating dataframes), SciPy, matplotlib (visualization), surprise (KNNBasic), matplotlib (ploting graphs), scipy.sparse (build sparse matrix), implicit.bpr (Bayesian Personalized Ranking), implicit.evaluation (train_test_split and precision_at_k), astropy.table (creating tables from lists)

III. Experiments and Results

Approaches:

Approach 1: Singular Value Decomposition

The first approach was user based collaborative filtering with SVD algorithm. In this approach the users were classified based on their attributes. The rating matrix was composed using the SVD algorithm, this matrix was combined with the new matrix to calculate the similarities between users. Using the nearest neighbors KNN approach in the collaborative filtering recommendation system, the ratings of the restaurants were predicted and hence recommendations were given.

Approach 2: Calibrated Filtering

When a user has rated 8 Indian restaurants and 2 Italian restaurants, it is more likely that that the user will be a personalized list of restaurant recommendations where 80% are Indian restaurants and 20% are Italian restaurants. The recommendations will comprise of all his preferences in specific percentages without ignoring the least rated ones and hence leading to balanced recommendations. This is called calibration.

A recommendation that actually reflect most if not all of the user's interest is considered a Calibrated Recommendation. For quantifying the degree of calibration, we outline metrics and use simple yet effective re-ranking approach for postprocessing the output of recommendation systems.

Implementation and Experiments:

Singular Value Decomposition-

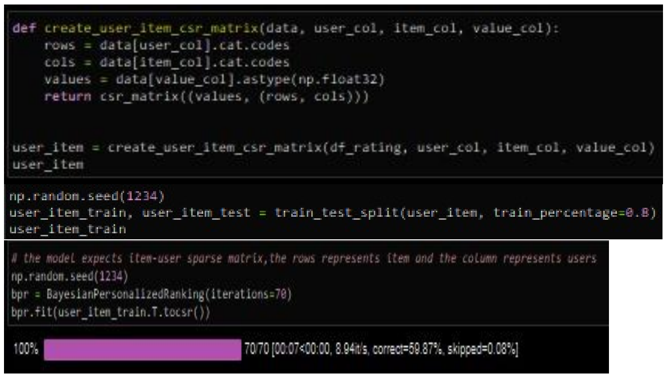
The rating matrix composed using Singular Value Decomposition algorithm was used to calculate the similarities between users. KNN nearest neighbors algorithm was used in the collaborative filtering recommendation system to predict the ratings of the restaurants and were ranked according to their scores.

The data is further split into training and test data where training is 70% and test data is 30%. With the help of this recommendations were given to the users. Recommendations remained same for all the users as it was based on scores hence making it popularity-based recommendation system.

A matrix of actual ratings given by the users were built where the rows were users and columns were items. A user based collaborative filtering system was built based on this matrix and by using the pandas pivot table function. With the help of SVD, we calculated the predicted ratings of the users who didn’t give any ratings to the restaurants.

Calibrated Filtering –

Once we have our final data frame, we constructed a sparse matrix to split the data randomly into train/test set. Fed the training set into a collaborative filtering model to train the model and got item recommendations for users.



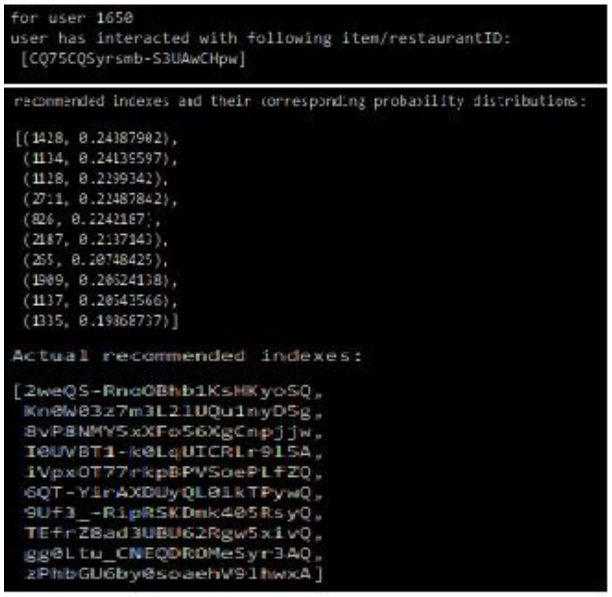
```
def create_user_item_csr_matrix(data, user_col, item_col, value_col):
    rows = data[user_col].cat.codes
    cols = data[item_col].cat.codes
    values = data[value_col].astype(np.float32)
    return csr_matrix((values, (rows, cols)))


user_item = create_user_item_csr_matrix(df_rating, user_col, item_col, value_col)
user_item
```

```
np.random.seed(1234)
user_item_train, user_item_test = train_test_split(user_item, train_percentage=0.8)
user_item_train
```

```
# the model expects item-user sparse matrix,the rows represents item and the column represents users
np.random.seed(1234)
bpr = BayesianPersonalizedRanking(iterations=70)
bpr.fit(user_item_train.T.tocsr())
```

```
100%                              70/70 [00:07<00:00, 8.94it/s, correct=59.87%, skipped=0.08%]
```

For test user (user-index: 1650), we used bpr.recommend method to recommend top n recommendations for test user. We mapped actual indexes to actual item IDs.



```
for user 1650
user has interacted with following item/restaurantID:
 [CQ75CQSyrsmb-S3UAwCHpw]
```

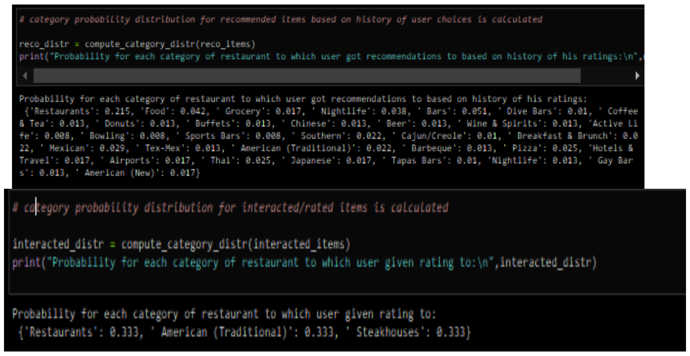
```
recommended indexes and their corresponding probability distributions:

[(1428, 0.24387902),
 (1134, 0.24135597),
 (1128, 0.2299342),
 (2711, 0.22487842),
 (826, 0.2242187),
 (2187, 0.2137143),
 (265, 0.20748425),
 (1909, 0.20524138),
 (1137, 0.20543566),
 (1335, 0.19868737)]

Actual recommended indexes:

[2weQS-RnoOBhb1KsHKyoSQ,
 Kn0W03z7m3L21UQu1nyD5g,
 8vP8NMY5xXFo56XgCnpjjw,
 I0UVBT1-k0LqUICRLr915A,
 iVpxOT77rkpBPVSoePLfZQ,
 6QT-YirAXDUyQL01kTPywQ,
 9Uf3_-RipRSKDmk405RsyQ,
 TEfrZ8ad3UBU62Rgw5xivQ,
 gg0Ltu_CNEQDROMeSyr3AQ,
 zPhbGU6by0soaehV91hwxA]
```

We have taken test user to see whether our recommendations are calibrated or not. We are having category distributions c for each restaurant(item) i, p(c|i), what we are interested is whether p(c|u) is like q(c|u) where,

- p(c|u) is the distribution over category c of the set of restaurant R played by user u in the past. $p(c|u) = \sum i \in R \; p(c|i)$

- q(c|u) is the distribution over category c of the set of restaurants I we recommended to user u. $q(c|u) = \sum i \in I \; p(c|i)$

After this we wrote a function which is calculating restaurant category distribution for list of items. We have called this function for recommended restaurant categories based on history and restaurant categories user has rated already. Following are results of same,



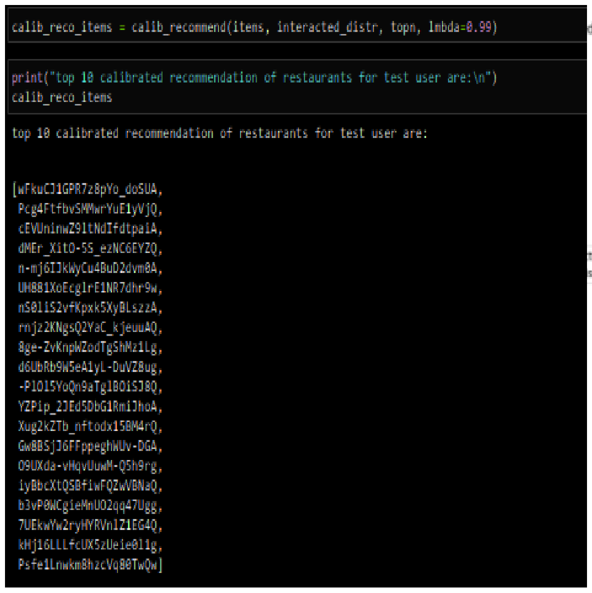
```
# category probability distribution for recommended items based on history of user choices is calculated

reco_distr = compute_category_distr(reco_items)
print("Probability for each category of restaurant to which user got recommendations to based on history of his ratings:\n",
```

```
Probability for each category of restaurant to which user got recommendations to based on history of his ratings:
 {'Restaurants': 0.215, 'Food': 0.042, ' Grocery': 0.017, ' Nightlife': 0.038, ' Bars': 0.051, ' Dive Bars': 0.01, ' Coffee
& Tea': 0.013, ' Donuts': 0.013, ' Buffets': 0.013, ' Chinese': 0.013, ' Beer': 0.013, ' Wine & Spirits': 0.013, 'Active Li
fe': 0.008, ' Bowling': 0.008, ' Sports Bars': 0.008, ' Southern': 0.022, ' Cajun/Creole': 0.01, ' Breakfast & Brunch': 0.0
22, ' Mexican': 0.029, ' Tex-Mex': 0.013, ' American (Traditional)': 0.022, ' Barbeque': 0.013, ' Pizza': 0.025, 'Hotels &
Travel': 0.017, ' Airports': 0.017, ' Thai': 0.025, ' Japanese': 0.017, ' Tapas Bars': 0.01, 'Nightlife': 0.013, ' Gay Bar
s': 0.013, ' American (New)': 0.017}
```

```
# category probability distribution for interacted/rated items is calculated

interacted_distr = compute_category_distr(interacted_items)
print("Probability for each category of restaurant to which user given rating to:\n",interacted_distr)
```

```
Probability for each category of restaurant to which user given rating to:
 {'Restaurants': 0.333, ' American (Traditional)': 0.333, ' Steakhouses': 0.333}
```

Following plot shows for each Category of restaurants Category Distribution between test User's Historical ratings v.s. Recommended Rating.



```
calib_reco_items = calib_recommend(items, interacted_distr, topn, lmbda=0.99)
```

```
print("top 10 calibrated recommendation of restaurants for test user are:\n")
calib_reco_items
```

```
top 10 calibrated recommendation of restaurants for test user are:


[wFkuCJ1GPR7z8pYo_doSUA,
 Pcg4FtfbvSMMwrYuE1yVjQ,
 cEVUninwZ91tNdIfdtpaiA,
 dMEr_XitO-5S_ezNC6EYZQ,
 n-mj6IJkWyCu4BuD2dvm0A,
 UHB81XoEcg1rE1NR7dhr9w,
 nS01iS2vfKpxk5XyBLszzA,
 rnjz2KNgsQ2YaC_kjeuuAQ,
 8ge-ZvKnpWZodTgShMz1Lg,
 d6UbRb9W5eA1yL-DuVZ8ug,
 -P1O15YoQn9aTg1BOiSJ8Q,
 YZPip_2JEd5DbG1RmiJhoA,
 Xug2kZTb_nftodx15BM4rQ,
 GwBBSjJ6FFppeghWUv-DGA,
 O9UXda-vHqvUuwM-Q5h9rg,
 iyBbcXtQSBfiwFQZwVBNaQ,
 b3vP0WCgieMnUO2qq47Ugg,
 7UEkwYw2ryHYRVnlZ1EG4Q,
 kHj16LLLfcUX5zUeie0l1g,
 Psfe1LnwkmBhzcVq80TwQw]
```

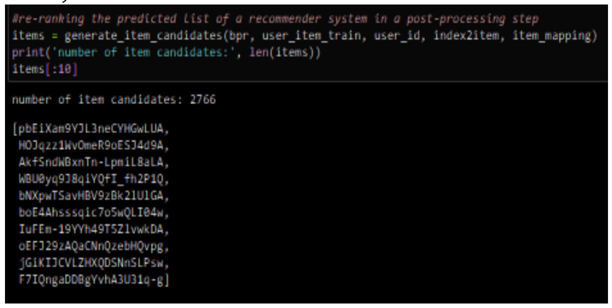
We can see from above graph and readings that users interacted restaurant categories were, “restaurant: 0.333”, “American(traditional): 0.333” and “steakehouse: 0.333” but in recommendation list probability distributions for these categories are comparatively low. So In short our recommendations are not calibrated. To scale this type of comparison, we'll now define our calibration metric CC. There are various methods to compare whether two distributions are similar to each other, and one popular choice is KL-divergence.

$$C(p,q) = D_{KL}(p||q) = \sum_{g} p(g|u) \cdot \log \frac{p(g|u)}{\tilde{q}(g|u)}$$

The denominator in the formula should be q(g|u), but given that the formula would be undefined if q(g|u)=0 and p(g|u)>0 for a category g. We instead use:

$$\tilde{q}(g|u) = (1-\alpha) \cdot q(g|u) + \alpha \cdot p(g|u)$$

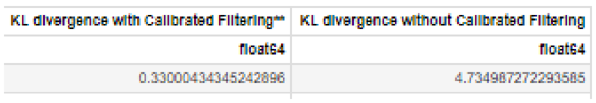
With a small αα such as $q(g|u) \approx \tilde{q}(g|u)$. 0.01,as



```
#re-ranking the predicted list of a recommender system in a post-processing step
items = generate_item_candidates(bpr, user_item_train, user_id, index2item, item_mapping)
print('number of item candidates:', len(items))
items[:10]
```

```
number of item candidates: 2766

[pbEiXam9YJL3neCYHGwLUA,
 HOJqzz1WvOmeR9oESJ4d9A,
 AkfSndWBxnTn-LpmiL8aLA,
 WBU8yq9J8qiYQfI_fh2P1Q,
 bNXpwTSavHBV9zBk21U1GA,
 boE4Ahsssqic7o5wQLI04w,
 IuFEm-19YYh49T5ZlvwkDA,
 oEFJ29zAQaCNnQzebHQvpg,
 jGiKIJCVLZHXQDSNnSLPsw,
 F7IQngaDDBgYvhA3U31q-g]
```

So when we computed KL divergence without and with above logic value was our results were as follows,



KL divergence with Calibrated Filtering**	KL divergence without Calibrated Filtering
float64	float64
0.33000434345242896	4.734987272293585

Being able to compute the calibration metric between p(g|u) and q(g|u) is all well and good, but how can we generate a recommendation list that is more calibrated becomes the next important and interesting question.

Different recommendation algorithm's objective function might be completely different, thus instead of going to hard-route of incorporating it into the objective function right off the bat and spend two weeks writing the customized algorithm in an efficient manner, we will start with an alternative approach of re-ranking the predicted list of a recommender system in a post-processing step.

To determine the optimal set I^* of N recommended items, we'll be using maximum marginal relevance.

$$I^* = \underset{I,|I|=N}{\text{argmax}} \; (1-\lambda) \cdot s(I) - \lambda \cdot C(p, q(I))$$

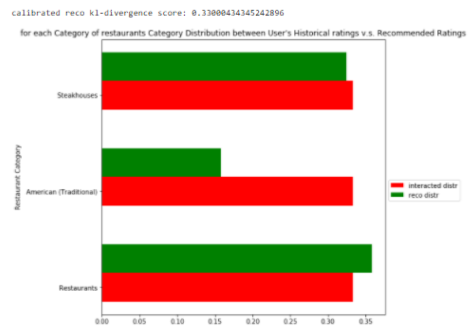
Finding the optimal set $I^*$is a combinatorial optimization problem and can be a topic by itself. We won't do a deep dive into it, but instead leverage a popular greedy submodular optimization to solve this problem. The process is as follows:

- We start out with the empty set.
- Iteratively append one item ii at a time, and at step n when we already have the set I_{n-1} comprised of n−1items, the item ii that maximizes the objective function defined above for the set $I_{n-1} \cup i$ is added to obtain I_n
- Repeat the process the generate the full $I*$ of size N.

The recommendations after performing calibrated filtering are as follows,

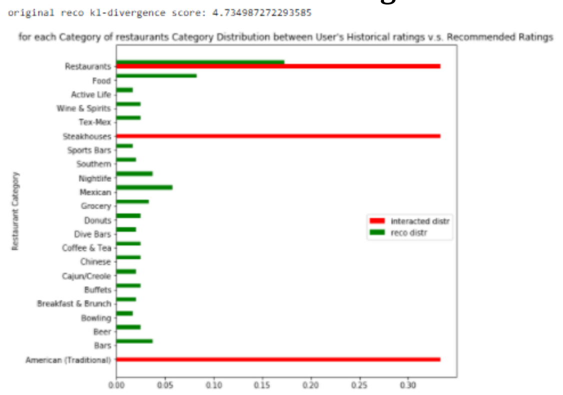
Following is comparison of results of KL divergence and graph of recommended categories of restaurants and interacted categories of restaurants with and without Calibrated filtering,

With Calibrated Filtering:



```
calibrated reco kl-divergence score: 0.33000434345242896
```

Without Calibrated Filtering:



```
original reco kl-divergence score: 4.734987272293585
```

We can see from above example we have achieved fairness in recommendation with calibrated filtering. Also KL Divergence value is low with Calibrated filtering.

Similarity measures used-

The similarity measure is the measure of how much alike two data objects are. Similarity measure in a data mining context is a distance with dimensions representing features of the objects. If this distance is small, it will be the high degree of similarity where large distance will be the low degree of similarity. The similarity is subjective and is highly dependent on the domain and application. For example, two fruits are similar because of color or size or taste.

1. **Cosine Similarity:** Cosine similarity metric finds the normalized dot product of the two word vectors. By determining the cosine similarity, we would effectively try to find the cosine of the angle between the two objects. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgement of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors. In our project, cosine similarity is being used in the Singular Value Decomposition approach for user based collaborative filtering.

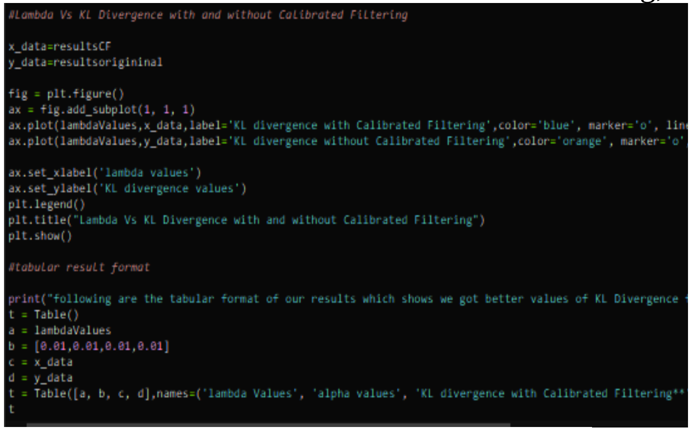
$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

2. **KL Divergence:** Kullback-Leibler divergence calculates a score that measures the divergence of one probability distribution from another. The KL divergence between two distributions P and Q can stated as KL(P||Q) where '||' operator indicates the divergence of P from Q. The KL divergence is large when the probability of event P is higher than probability of event Q. The KL divergence still remains large when the probability of event P is small and event Q is large. In the latter case, the KL divergence is not as large as the first and the integral of events is calculated rather than discrete event's sum of probabilities of. The KL divergence can be calculated between discrete and continuous probability distributions. The KL Divergence scores are not symmetrical.
KL(P||Q) != KL(Q||P).
The measure of dissimilarity of two probability distributions P and Q is called KL Divergence. In our project, we are implementing the KL Divergence concept in Calibrated Filtering approach and it has proved to give better results.

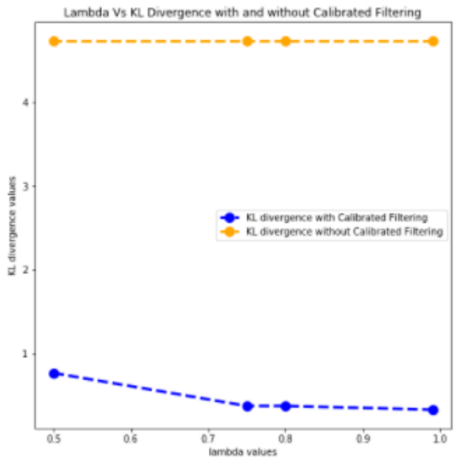
$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

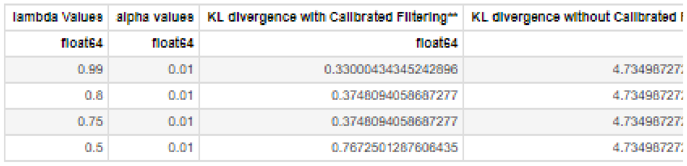
IV. PARAMETER TUNING

There are two parameter we can tune lambda and alpha. We tried different values of alpha and understood low value of alpha is better so kept it as 0.01 then performed experiment for different values of lambda to see changes in KL Divergence values with and without Calibrated filtering,



```
#Lambda Vs KL Divergence with and without Calibrated Filtering

x_data=resultsCF
y_data=resultsoriginal

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(lambdaValues,x_data,label='KL divergence with Calibrated Filtering',color='blue', marker='o', lin
ax.plot(lambdaValues,y_data,label='KL divergence without Calibrated Filtering',color='orange', marker='o'

ax.set_xlabel('lambda values')
ax.set_ylabel('KL divergence values')
plt.legend()
plt.title("Lambda Vs KL Divergence with and without Calibrated Filtering")
plt.show()

#tabular result format

print("following are the tabular format of our results which shows we got better values of KL Divergence
t = Table()
a = lambdaValues
b = [0.01,0.01,0.01,0.01]
c = x_data
d = y_data
t = Table([a, b, c, d],names=('lambda Values', 'alpha values', 'KL divergence with Calibrated Filtering**
t
```





lambda Values	alpha values	KL divergence with Calibrated Filtering**	KL divergence without Calibrated I
float64	float64	float64	
0.99	0.01	0.33000434345242896	4.73498727
0.8	0.01	0.3748094058687277	4.73498727
0.75	0.01	0.3748094058687277	4.73498727
0.5	0.01	0.7672501287606435	4.73498727

V. ANALYSIS AND CONCLUSION

Preprocessing the data is very crucial.
Item based collaborative filtering method gives more accurate values compared to user based collaborative filtering approach.In SVD approach, popularity based ranking takes place and hence the accuracy is low.Calibrated Filtering using KL Divergence gives better accuracy.We received comparatively high values for KL divergence without calibration.

According to our results, with calibrated filtering our results for KL divergence were better. Items are ranked according to their relevancy and the most relevant ones are shown to the user. Recommendation systems determine this relevancy with the help of historical data.

In our project, we achieved fairness in recommendation system using Calibration concept of machine learning where the actual output is compared with the expected output. Using this concept, we evaluated the fairness in the recommendation systems.

The machine learning model we designed helps to improve predictions using the training data such that the probabilities of the recommendations predicted will be similar to the probabilities observed.

VI. CONTRIBUTION

	Kavya	**Nikita**
Creating a dataset using join	✓	✓
Preprocessing data	✓	✓
Reducing the data size	✓	✓
Item Mapping	✓	✓
Singular Value Decomposition	✓	✓
Ranking	✓	✓
Calibrated Filtering	✓	✓
Reranking	✓	✓
Documentation	✓	✓

VII. REFERENCES

https://dl.acm.org/doi/pdf/10.1145/3240323.3240372

https://ieeexplore.ieee.org/abstract/document/6615466?casa_token=EDTO5xbtcpwAAAAA:LIRZ0TEWsIF_oNgabPOWxdyXuMZuZdNUJ_DkmUk0CepuGTKkY1ioeZzBaAv8xztIXmv7hbN_4skP

https://dl.acm.org/doi/abs/10.1145/3292500.3330745

https://dl.acm.org/doi/abs/10.1145/3240323.3240372

https://machinelearningmastery.com/divergence-between-probability-distributions/

https://ieeexplore.ieee.org/document/5369510