THANKS FOR THE RECOMMENDATION

Recommendation Systems
August 26, 2024

Prepared by: Jeffandy St.Hubert

TABLE OF CONTENTS

BUSINESS PROBLEM

The rate, variety, and sheer amount of information are growing exponentially in e-commerce. This poses unique challenges but not without unique opportunities.

This increase in data allows for the most personalized customer experience in modern history. If Amazon can be at the edge of recommendation system innovation through ranking, collaborative filtering, and matrix factorization-based algorithms, the unanimous response from each customer will be “Thanks for the recommendation”. Amazon will continue to lead in the e-commerce space.

DATA OVERVIEW

- **userId**
  Every user identified with a unique id
- **productId**
  Every product identified with a unique id
- **Rating**
  The rating of the corresponding product by the corresponding user
- **timestamp**
  Time of the rating. We will not use this column to solve the current problem

EXPLORATORY DATA ANALYSIS

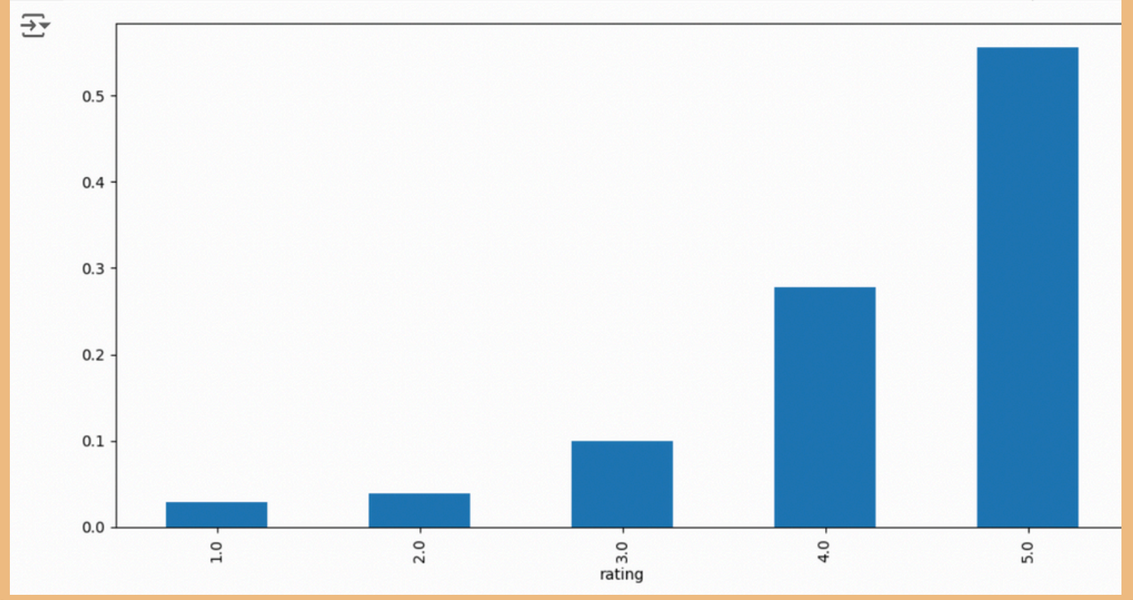
The average rating is 4.3 which is pretty high.

Bar plot shows it is left-skewed with most items rated 5/5

```
# Create the bar plot and provide observations
plt.figure(figsize = (12, 6))

df_final['rating'].value_counts(1).sort_values(ascending=True).plot(kind='bar')

plt.show()
```




```
# Summary statistics of 'rati
df_final['rating'].describe()
```

	rating
count	65290.000000
mean	4.294808
std	0.988915
min	1.000000
25%	4.000000
50%	5.000000
75%	5.000000
max	5.000000

dtype: float64

EXPLORATORY DATA ANALYSIS CONT.

```
# Top 10 users based on the number of ratings
most_rated = df_final.groupby('user_id').size().sort_values(ascending = False)[:10]
most_rated
```

user_id	
ADLVFFE4VBT8	295
A3OXHLG6DIBRW8	230
A1ODOGXEYECQQ8	217
A36K2N527TXXJN	212
A25C2M3QF9G7OQ	203
A680RUE1FDO8B	196
A22CW0ZHY3NJH8	193
A1UQBFCERIP7VJ	193
AWPODHOB4GFWL	184
A3LGT6UZL99IW1	179

dtype: int64

Users with the most number of ratings

RANK BASED MODEL

```
# Calculate the average rating for each product
average_rating = df_final.groupby('prod_id')['rating'].mean()

# Calculate the count of ratings for each product
count_rating = df_final.groupby('prod_id')['rating'].count()

# Create a dataframe with calculated average and count of ratings
final_rating = pd.DataFrame({'average_rating': average_rating, 'count_rating': count_rating})

# Sort the dataframe by average of ratings in descending order
final_rating = final_rating.sort_values('average_rating', ascending=False)

# See the first five records of the "final_rating" dataset
final_rating.head()
```

First, we created a rank based model by creating a data frame for final rating.

prod_id	average_rating	count_rating
B00LGQ6HL8	5.0	5
B003DZJQQI	5.0	14
B005FDXF2C	5.0	7
B00I6CVPVC	5.0	7
B00B9KOCYA	5.0	8

RANK BASED MODEL CONT.

Recommending top 5 products with 50 minimum interactions based on popularity.

```
[28] # Recommending top 5 products with 100 minimum interactions based on popularity
     top_5_products_100_interactions = top_n_products(final_rating, n=5, min_interaction=100)

     # Display the top 5 products
     print("Top 5 products with at least 100 interactions based on popularity:")
     print(top_5_products_100_interactions)

     Top 5 products with at least 100 interactions based on popularity:
     Index(['B003ES5ZUU', 'B000N99BBC', 'B002WE6D44', 'B007WTAJTO', 'B002V88HFE'], dtype='object', name='prod_id')
```

ITEM-ITEM SIMILARITY MODEL

This model is a collaborative filtering model.

Now let's build the **final model** by using **tuned values of the hyperparameters** which we received by using grid search cross-validation.

```
[62] # Using the optimal similarity measure for item-item based collaborative filtering
     sim_options = {'name': 'msd',
                    'user_based': False}

     # Creating an instance of KNNBasic with optimal hyperparameter values
     sim_item_item_optimized = KNNBasic(sim_options = sim_options, k = 30, min_k = 6, random_state = 1, verbose = False)

     # Training the algorithm on the train set
     sim_item_item_optimized.fit(trainset)

     # Let us compute precision@k and recall@k, f1_score and RMSE
     precision_recall_at_k(sim_item_item_optimized, testset=testset)
```

```
RMSE: 0.9576
Precision:  0.839
Recall:  0.88
F_1 score:  0.859
```

ITEM-ITEM SIMILARITY MODEL CONT.

Predicting top 5 products for userId = "A1A5KUIIIHFF4U" with similarity based recommendation system.

```
[66] # Making top 5 recommendations for user_id A1A5KUIIIHFF4U with similarity-based recommendation engine.
     recommendations = get_recommendations(df_final, "A1A5KUIIIHFF4U", 5, sim_item_item_optimized)
```

ChatGPT

```
# Building the dataframe for above recommendations with columns "prod_id" and "predicted_ratings"
pd.DataFrame(recommendations, columns = ['prod_id', 'predicted_ratings'])
```

	prod_id	predicted_ratings
0	1400532655	4.292024
1	1400599997	4.292024
2	9983891212	4.292024
3	B00000DM9W	4.292024
4	B00000J1V5	4.292024

ITEM-ITEM SIMILARITY MODEL CONT.

Comparison:

- Estimated Rating: Both models predict an identical estimated rating of 4.29 for the non-interacted product. This indicates that the optimization didn’t significantly affect the prediction for this particular case where the user had no prior interaction with the product.
- "Was Impossible" Reason: Both models flagged the prediction as "impossible" due to not having enough neighbors ('reason': 'Not enough neighbors.'). This suggests that for users who haven't interacted with the product, the models struggle to make confident predictions, and this remains a challenge even after optimization.

Summary:

- Interacted Product (User A3LDPF5FMB782Z): The optimized model provided a more accurate prediction closer to the actual rating compared to the baseline model, reflecting an improvement in model performance after tuning.
- Non-Interacted Product (User A34BZM6S9L7QI4): There was no noticeable difference between the baseline and optimized models, as both struggled due to insufficient neighbors. The challenge of predicting ratings for users with no prior interaction with a product remains even after optimization.

The optimized model performs better for users with a history of interaction, but its effectiveness is still limited for users with no prior interaction data.

MATRIX FACTORIZATION BASED MODEL

```
[77] # Build the optimized SVD model using optimal hyperparameter search. Use random_state = 1
     svd_optimized = SVD(n_epochs = 20, lr_all = 0.01, reg_all = 0.2, random_state = 1)

     # Train the algorithm on the train set
     svd_optimized = svd_optimized.fit(trainset)

     # Use the function precision_recall_at_k to compute precision@k, recall@k, F1-Score, and RMSE
     precision_recall_at_k(svd_optimized, testset=testset)
```

```
RMSE: 0.9037
Precision:  0.846
Recall:  0.841
F_1 score:  0.843
```

WE BUILT THE FINAL OPTIMIZED MODEL USING TUNED VALUES OF THE HYPERPARAMETERS, WHICH WE RECEIVED USING GRID SEARCH CROSS-VALIDATION ABOVE.

CONCLUSIONS

IN CONCLUSION, WE CREATED THREE TYPES OF MODELS: A RANK-BASED MODEL, A COLLABORATIVE FILTERING ITEM-ITEM BASED MODEL AND A MATRIX FACTORIZATION MODEL

- ACCURACY (RMSE): THE OPTIMIZED SVD MODEL PERFORMS BETTER WITH A LOWER RMSE, INDICATING MORE ACCURATE PREDICTIONS.
- PRECISION AND RECALL: THE OPTIMIZED SVD MODEL HAS SLIGHTLY HIGHER PRECISION, WHILE THE ITEM-TO-ITEM MODEL HAS HIGHER RECALL.
- F1 SCORE: THE ITEM-TO-ITEM MODEL HAS A BETTER F1 SCORE, SUGGESTING A MORE EFFECTIVE BALANCE BETWEEN PRECISION AND RECALL.

RECOMMENDATIONS

IF ACCURACY IS THE PRIMARY GOAL AND MAKING PRECISE PREDICTIONS IS CRUCIAL, THE OPTIMIZED SVD MODEL IS BETTER.

IF RECALL AND ENSURING THAT A HIGHER PROPORTION OF RELEVANT ITEMS ARE RECOMMENDED IS MORE IMPORTANT, THE ITEM-TO-ITEM MODEL MIGHT BE PREFERRED.

Given the balance of precision and recall, the item-to-item model has a better F1 score, making it a more balanced recommendation system overall. However, since the SVD model has the best precision and prediction accuracy this is ideal for Amazon. Giving a precice recommendation coupled with the truth "it is easier to sell to one who is already buying" makes this a surefire way to please the customer and increase Amazon's bottom line while making the company a go-to place to find what you didn't know you needed.

THANK YOU

Massachusetts Institute of Technology

IDSS
MIT INSTITUTE FOR DATA, SYSTEMS, AND SOCIETY

Great Learning
POWER AHEAD