Collaborative Filtering - based

RECOMMENDER SYSTEM

using Amazon reviews

JUST EAT Takeaway.com

Pham Dang Khoa

7 Dec 2020

OUTLINE

1. Introduction
2. Exploratory Data Analysis
3. Modelling
4. Conclusions
5. Q&A

OUTLINE

1. Introduction
    - 1.1 Datasets
    - 1.2 Business significance
    - 1.3 Research question
    - 1.4 Methods
2. Exploratory Data Analysis
3. Modelling
4. Conclusions
5. Q&A

1. INTRODUCTION

1.1 Datasets

```
# Load the datasets
meta = pd.read_csv('metadata_category_clothing_shoes_and_jewelry_only.csv')
reviews = pd.read_csv('reviews_Clothing_Shoes_and_Jewelry_5.csv')
```

```
meta[:2]
```

	metadataid	asin	salesrank	imurl	categories	title	description	price	related	brand
0	2005401	B00004SR8Z	{'Clothing': 1631}	http://ecx.images-amazon.com/images/I/41RfWLMD...	[['Clothing, Shoes & Jewelry', 'Luggage & Trav...	Lewis N. Clark Deluxe Neck Stash	NaN	12.8	{'also_bought': ['B004RJWFDU', 'B00920ZC3O', '...	Lewis N. Clark
1	2217897	B0000ZE74A	{'Clothing': 4742}	http://ecx.images-amazon.com/images/I/41ryA-RO...	[['Clothing, Shoes & Jewelry', 'Women', 'Cloth...	Vanity Fair Women's Lollipop Plus Size Cuff Le...	NaN	NaN	{'also_bought': ['B004PEHJ6U', 'B0000TF5VK', '...	NaN

```
meta.shape
```

```
(23033, 10)
```

1. INTRODUCTION

1.1 Datasets

```
reviews[:2]
```

	Unnamed: 0	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	0	A1KLRMWW2FWPL4	0000031887	Amazon Customer "cameramom"	[0, 0]	This is a great tutu and at a really great pri...	5.0	Great tutu- not cheaply made	1297468800	02 12, 2011
1	1	A2G5TCU2WDFZ65	0000031887	Amazon Customer	[0, 0]	I bought this for my 4 yr old daughter for dan...	5.0	Very Cute!!	1358553600	01 19, 2013

```
reviews.shape
```

```
(278677, 10)
```

```
print("Smallest number of reviews per user is ", min(reviews.groupby('reviewerID').asin.count()))
print("Smallest number of reviews per product is ", min(reviews.groupby('asin').reviewerID.count()))
```

```
Smallest number of reviews per user is  5
Smallest number of reviews per product is  5
```

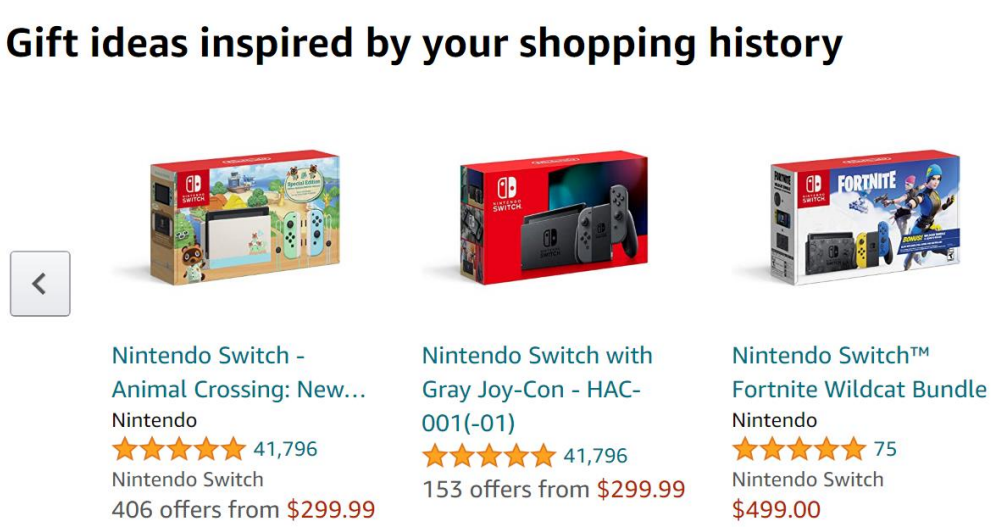
1. INTRODUCTION

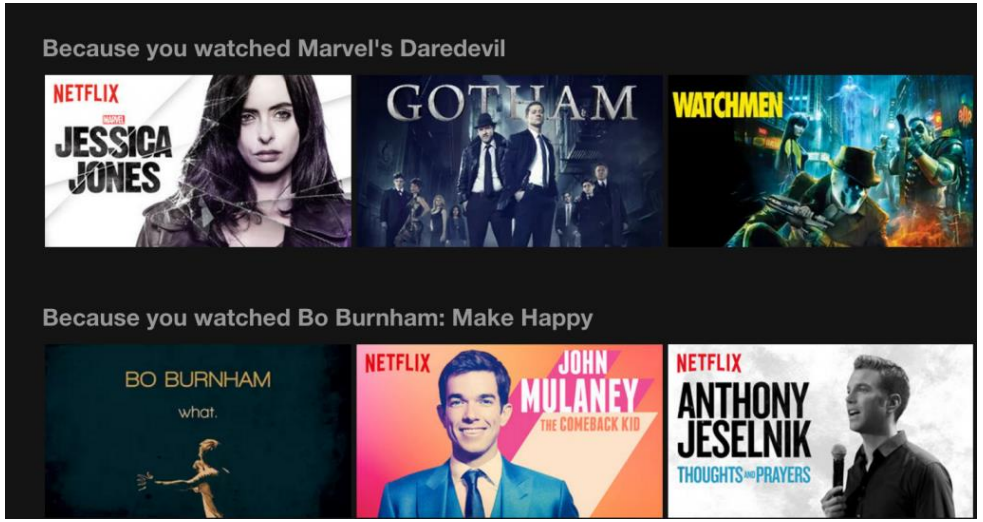
1.2 Business significance

```
reviews[:2]
```

	Unnamed: 0	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	0	A1KLRMWW2FWPL4	0000031887	Amazon Customer "cameramom"	[0, 0]	This is a great tutu and at a really great pri...	5.0	Great tutu- not cheaply made	1297468800	02 12, 2011
1	1	A2G5TCU2WDFZ65	0000031887	Amazon Customer	[0, 0]	I bought this for my 4 yr old daughter for dan...	5.0	Very Cute!!	1358553600	01 19, 2013

We can build a ***RECOMMENDER SYSTEM*** !!!





1. INTRODUCTION

1.2 Business significance

NETFLIX 80%

amazon 35%

Spotify®

YouTube

JUST EAT Takeaway.com

Users

- Personalised content
- Avoid information overload
- Explore new options

Companies

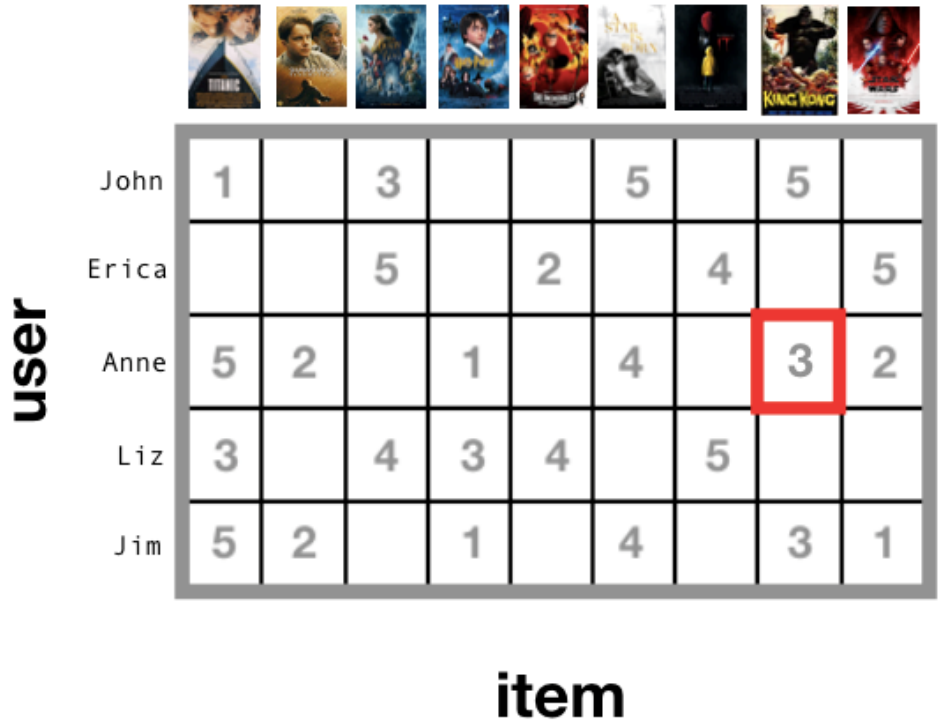
- Improve KPI (sales, click...)
- Promote less-popular items
- Assist marketing

1. INTRODUCTION

1.3 Research questions

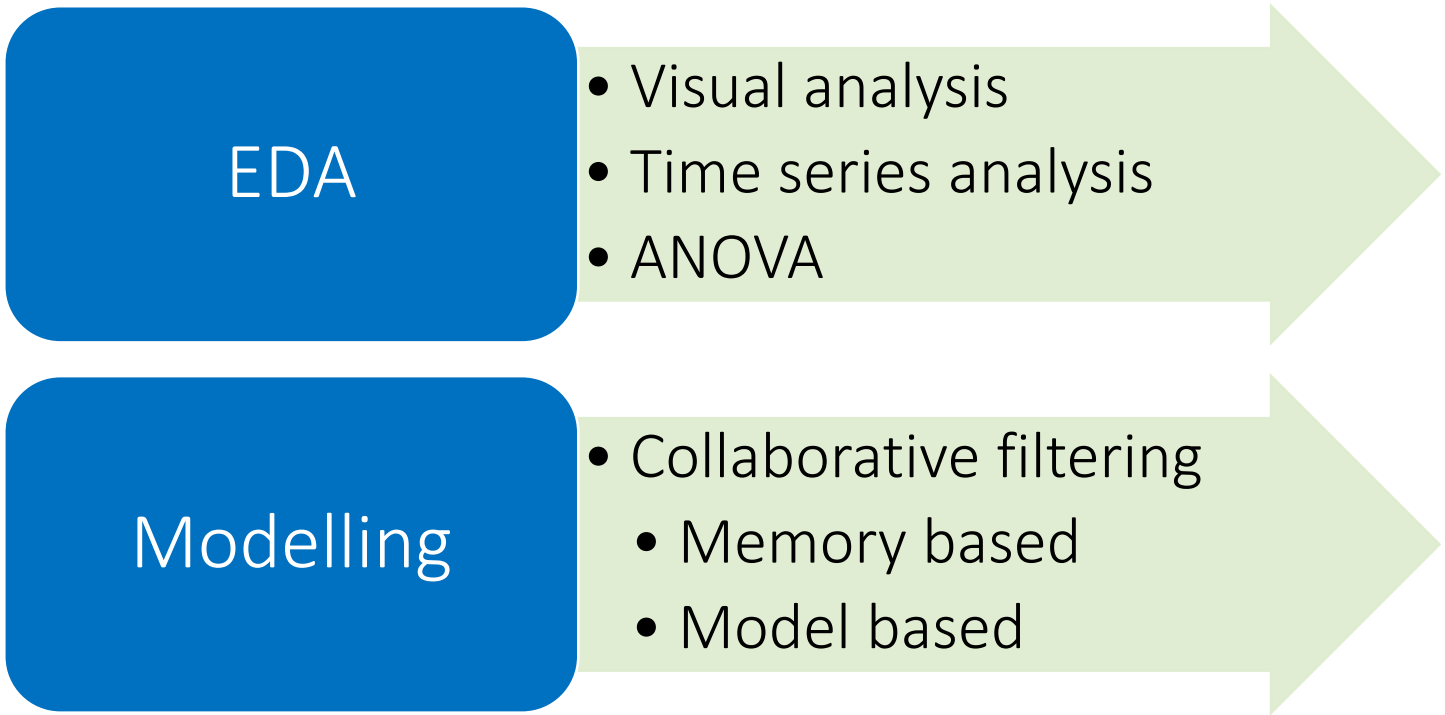
"How can we predict rating for new *(reviewer, product)* pairs from the Amazon reviews?"

- How many reviews, users, products are there?
- On average, how many reviews are there per user and per product?
- How does the rating distribution looks like?
- What is the most suitable algorithm?
- What set of parameters give the best accuracy?
- What is the min, max, average prediction error?
- How can we implement the model into a recommender system (giving a list of top rated products for each user)?



1. INTRODUCTION

1.4 Methods



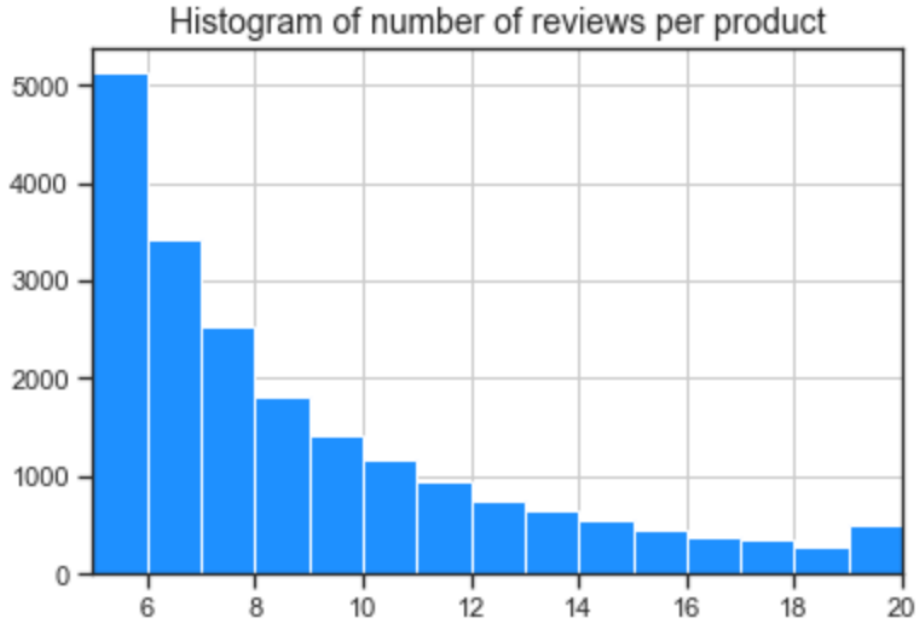
OUTLINE

1. Introduction
2. Exploratory Data Analysis
   - 2.1 Overview
   - 2.2 Breakdown of rating
   - 2.3 Word cloud of rating
   - 2.4 Time series
   - 2.5 Rating vs Helpfulness
3. Modelling
4. Conclusions
5. Q&A

2. EXPLORATORY DATA ANALYSIS

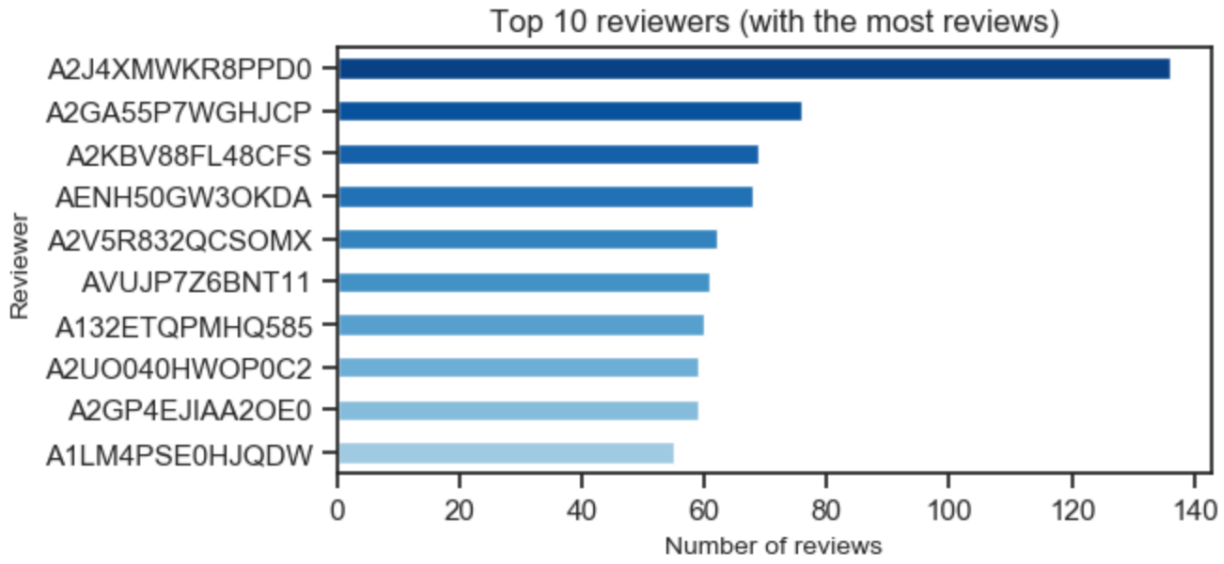
2.1 Overview

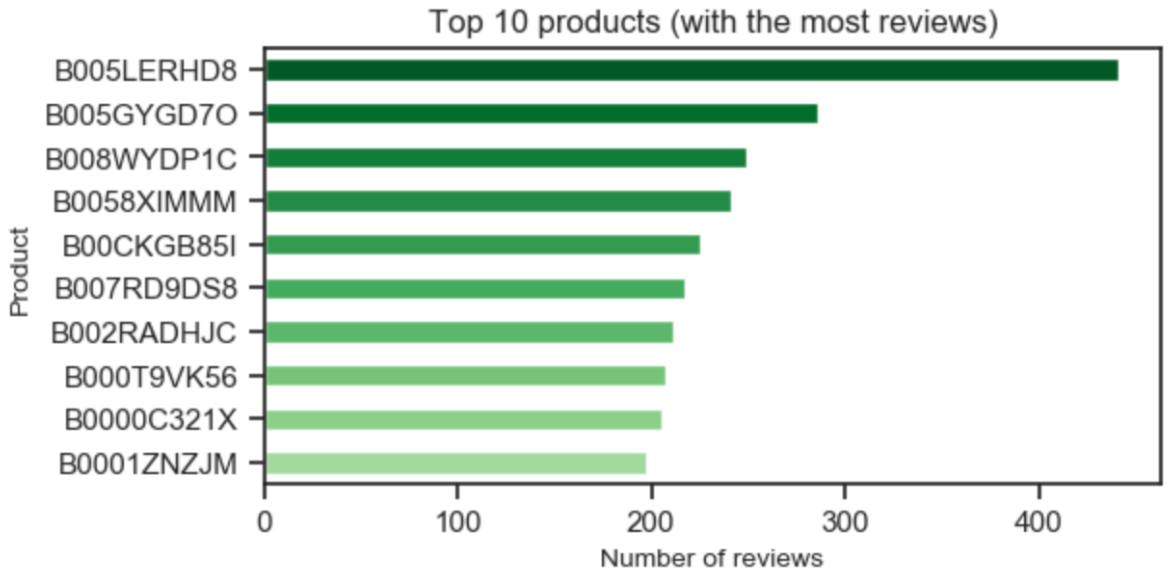
```
Total number of reviews:  278677
Total number of reviewers:  39387
Total number of products:  23033
On average, each reviewer gives : 7.08 reviews.
On average, each product has : 12.1 reviews.
```



2. EXPLORATORY DATA ANALYSIS

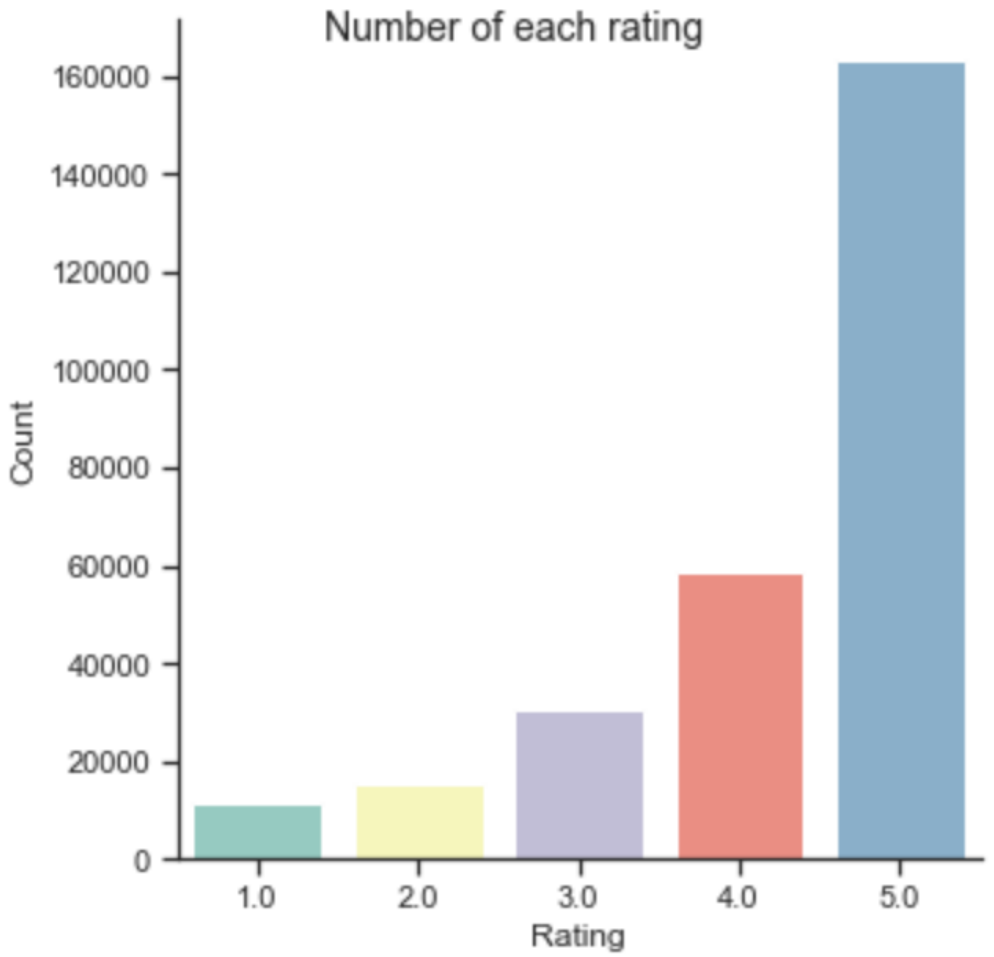
2.1 Overview

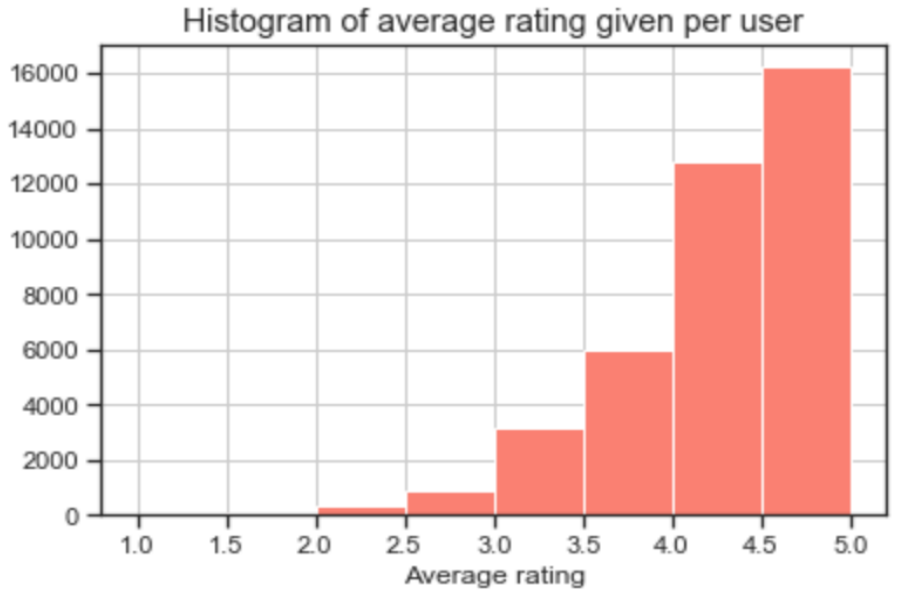


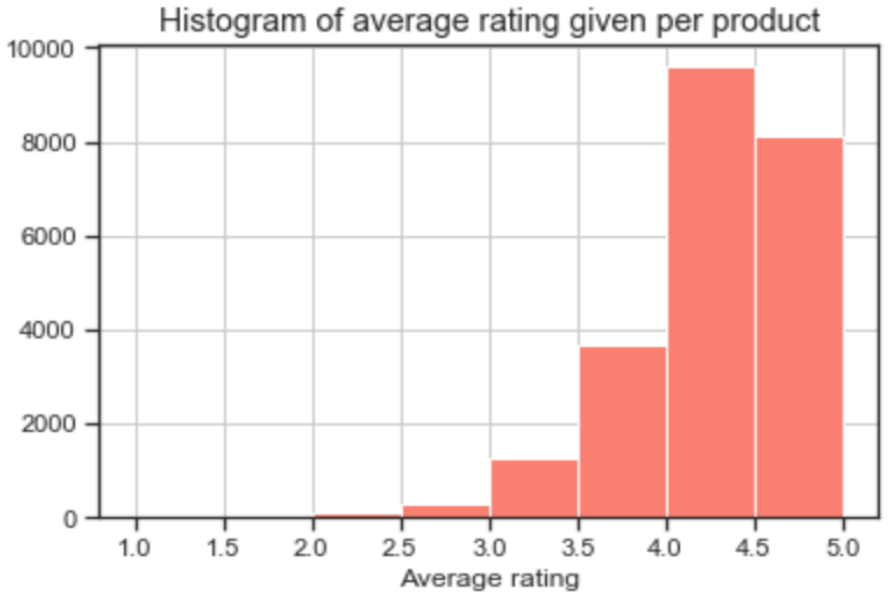


2. EXPLORATORY DATA ANALYSIS

2.2 Breakdown of rating

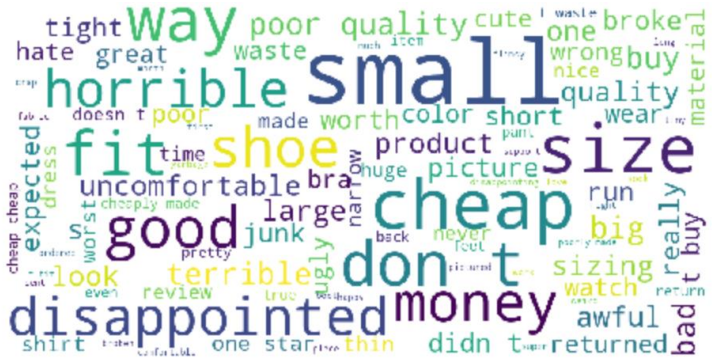






2. EXPLORATORY DATA ANALYSIS

2.3 Word cloud of each rating



Review Score One

Review Score Two

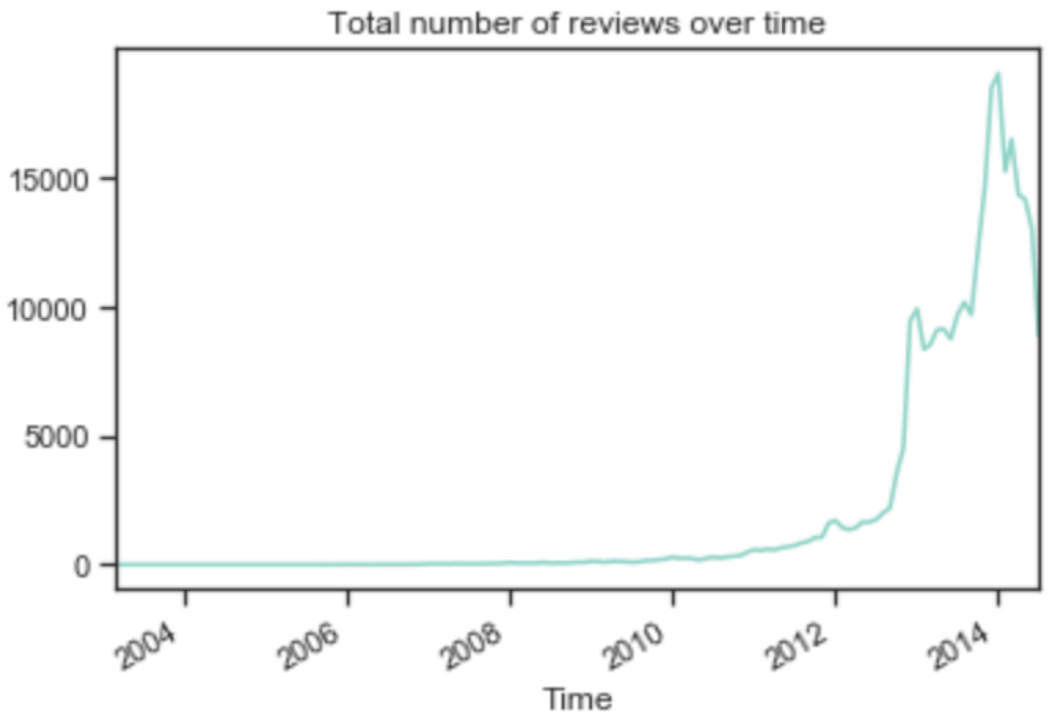
Review Score Three

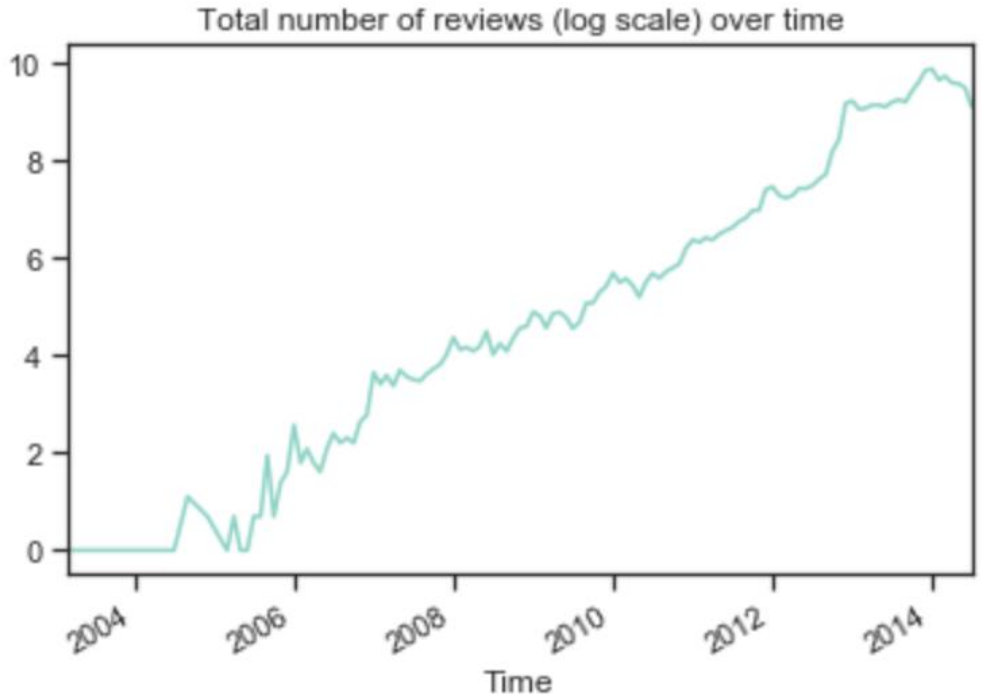
Review Score Four

Review Score Five

2. EXPLORATORY DATA ANALYSIS

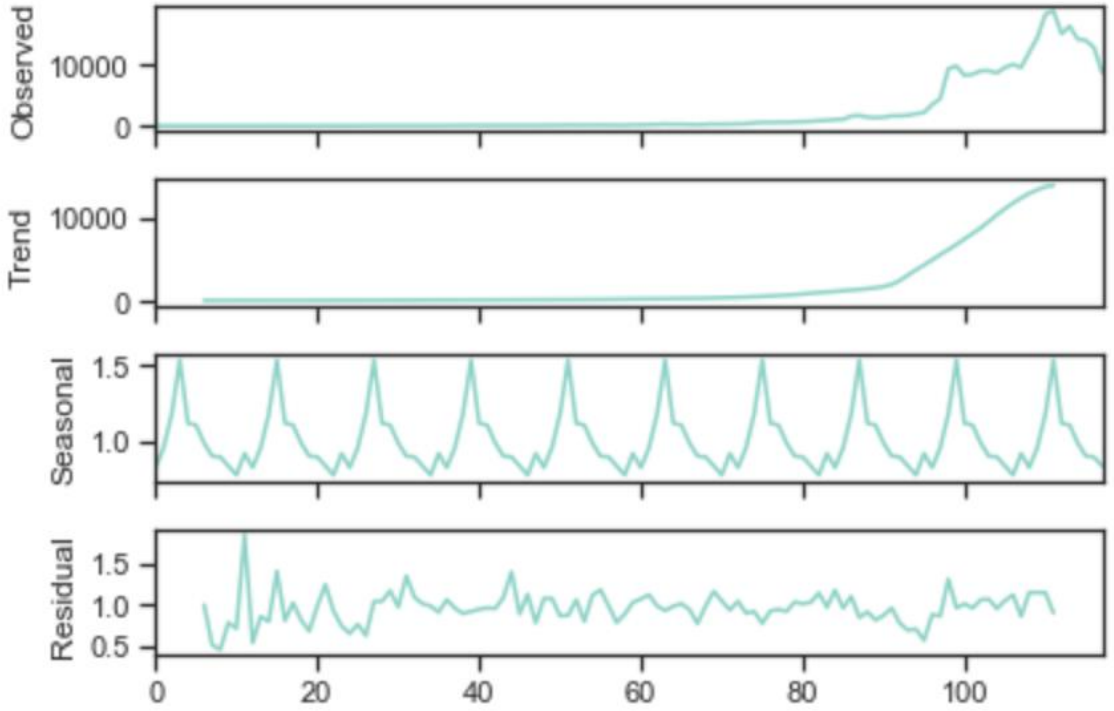
2.4 Time series

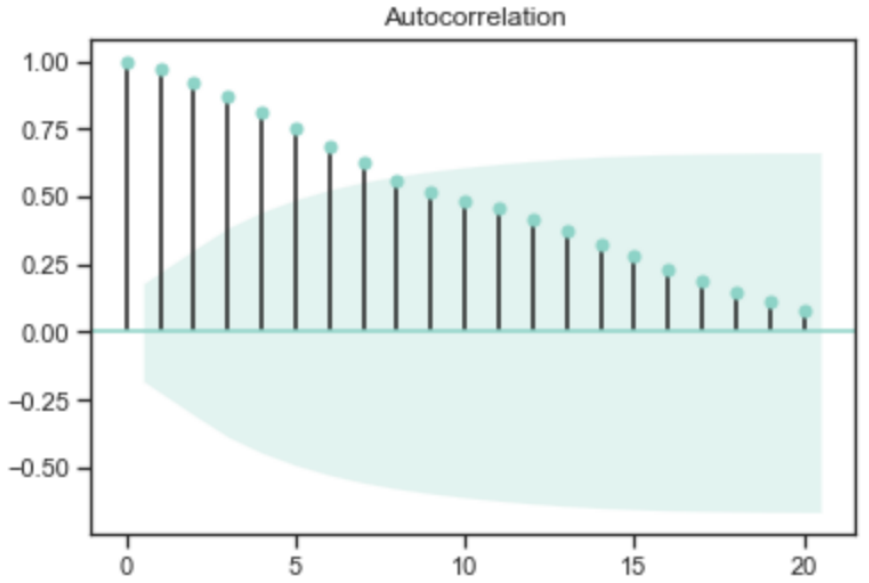


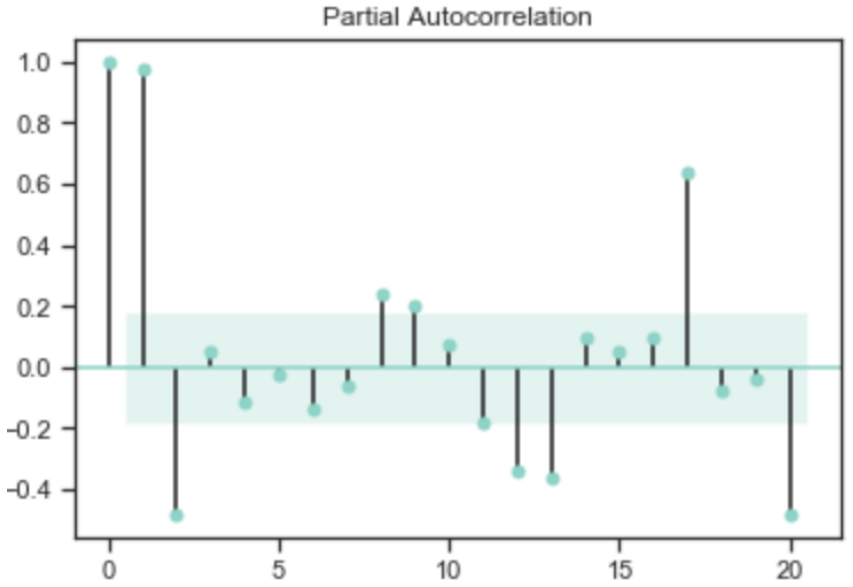


2. EXPLORATORY DATA ANALYSIS

2.4 Time series

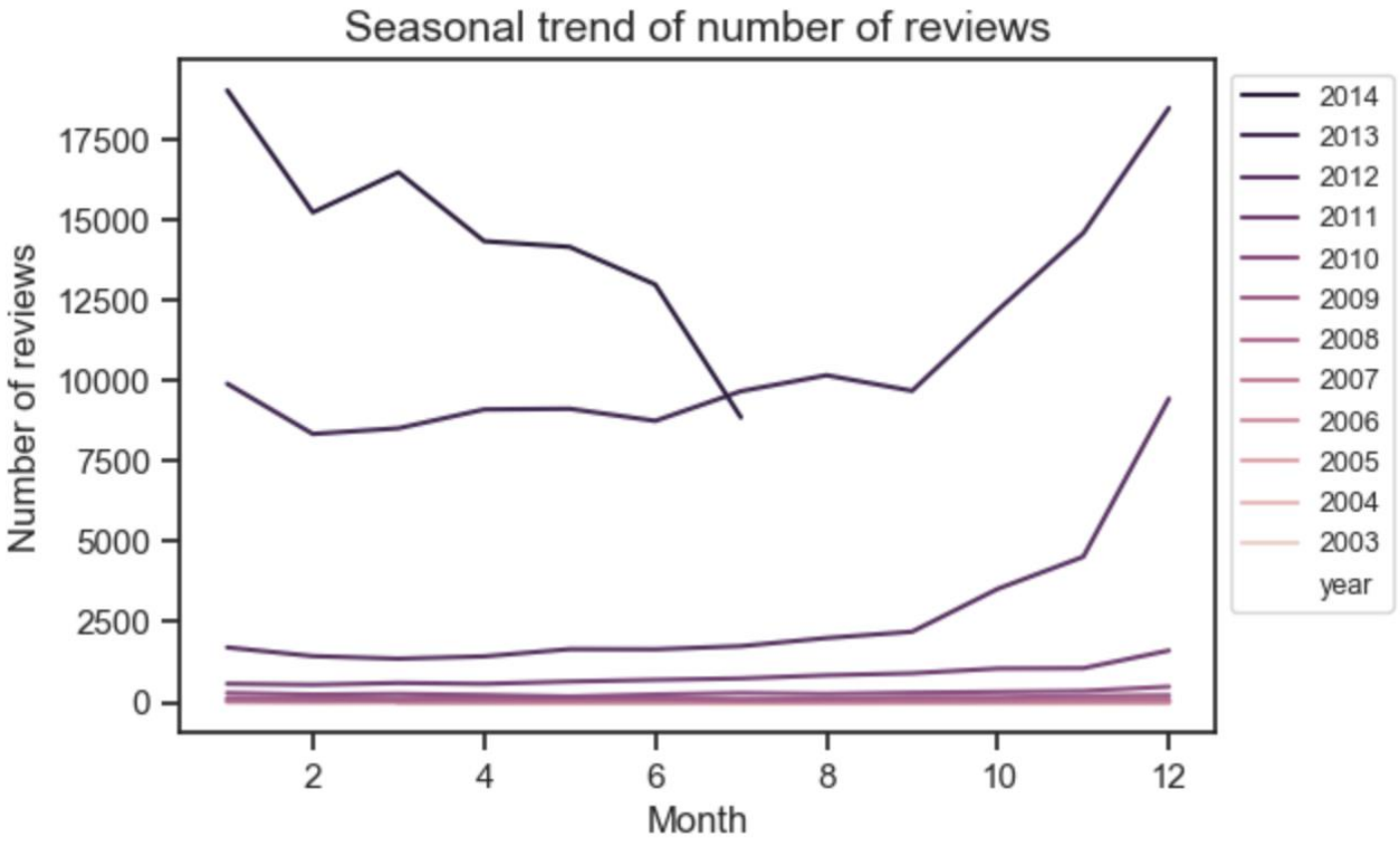






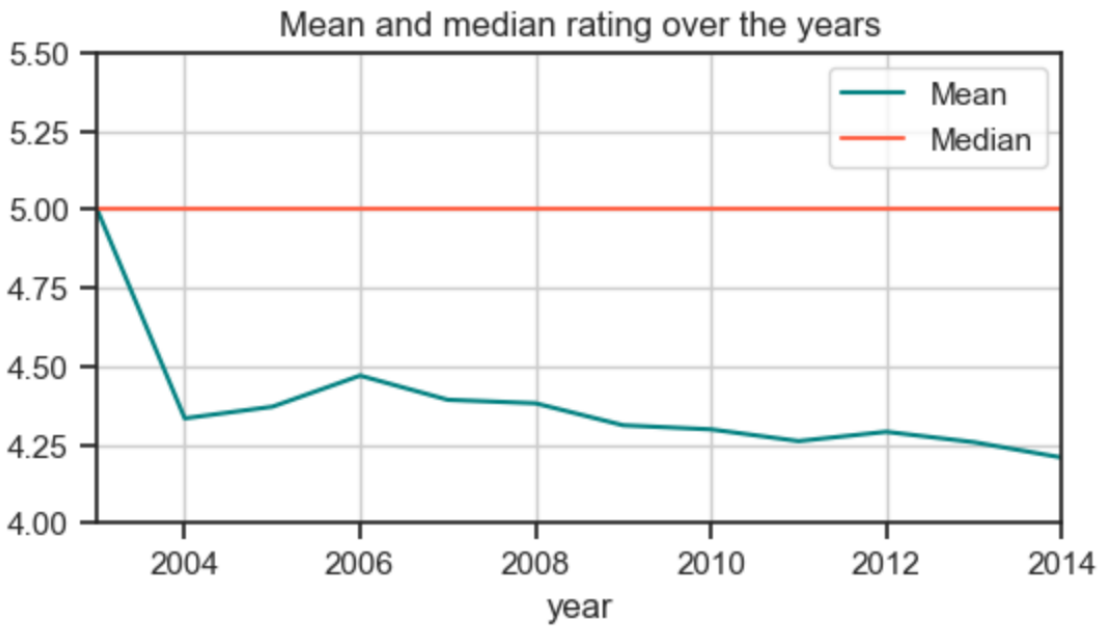
2. EXPLORATORY DATA ANALYSIS

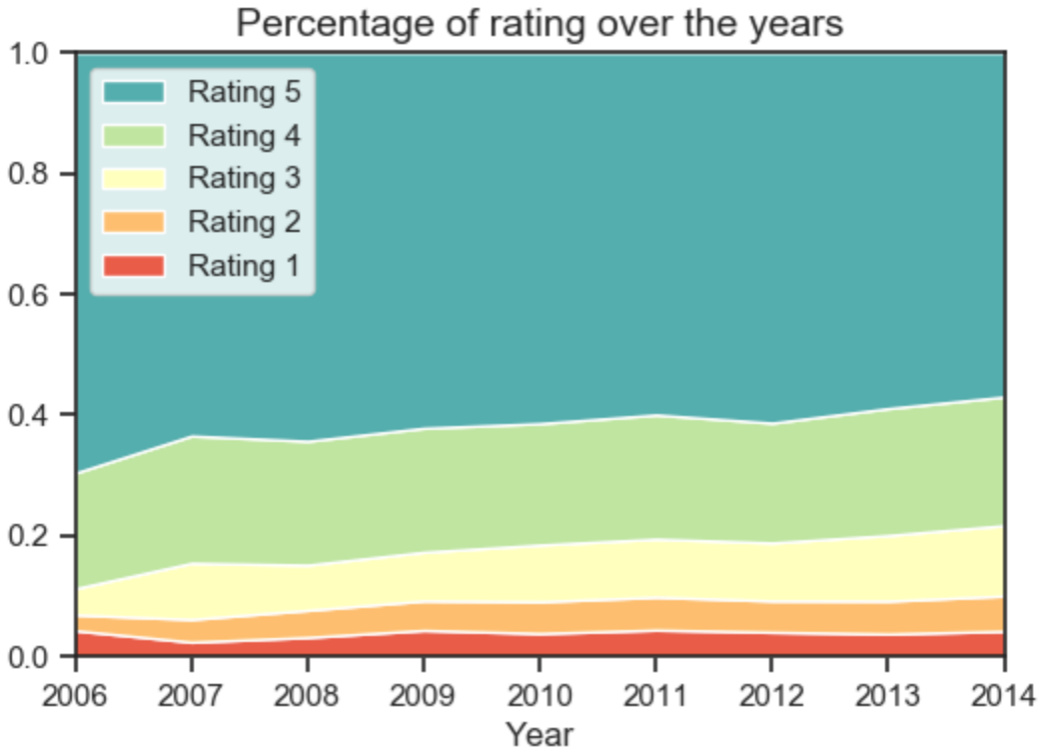
2.4 Time series



2. EXPLORATORY DATA ANALYSIS

2.4 Time series



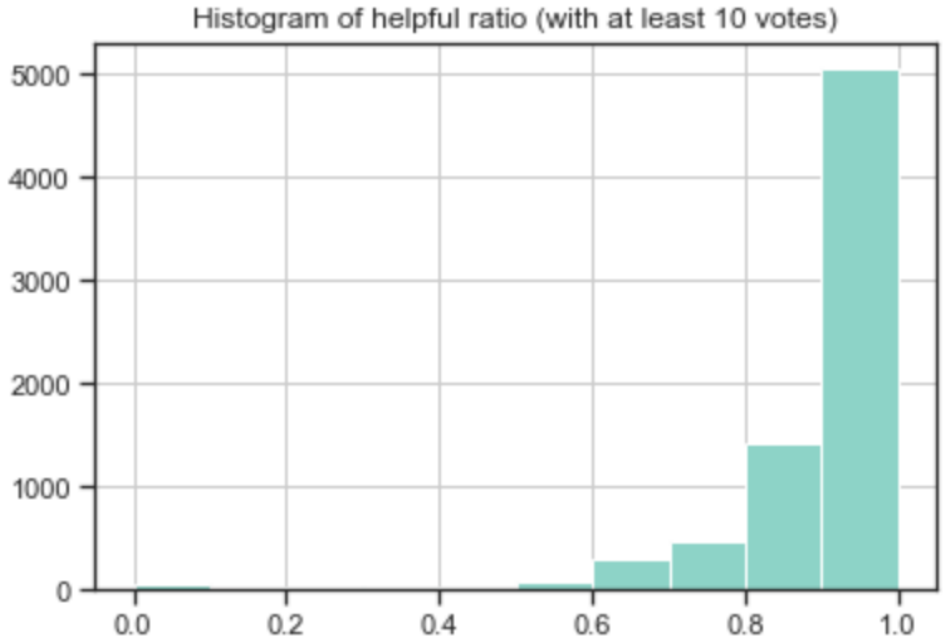


2. EXPLORATORY DATA ANALYSIS

2.5 Rating vs Helpfulness

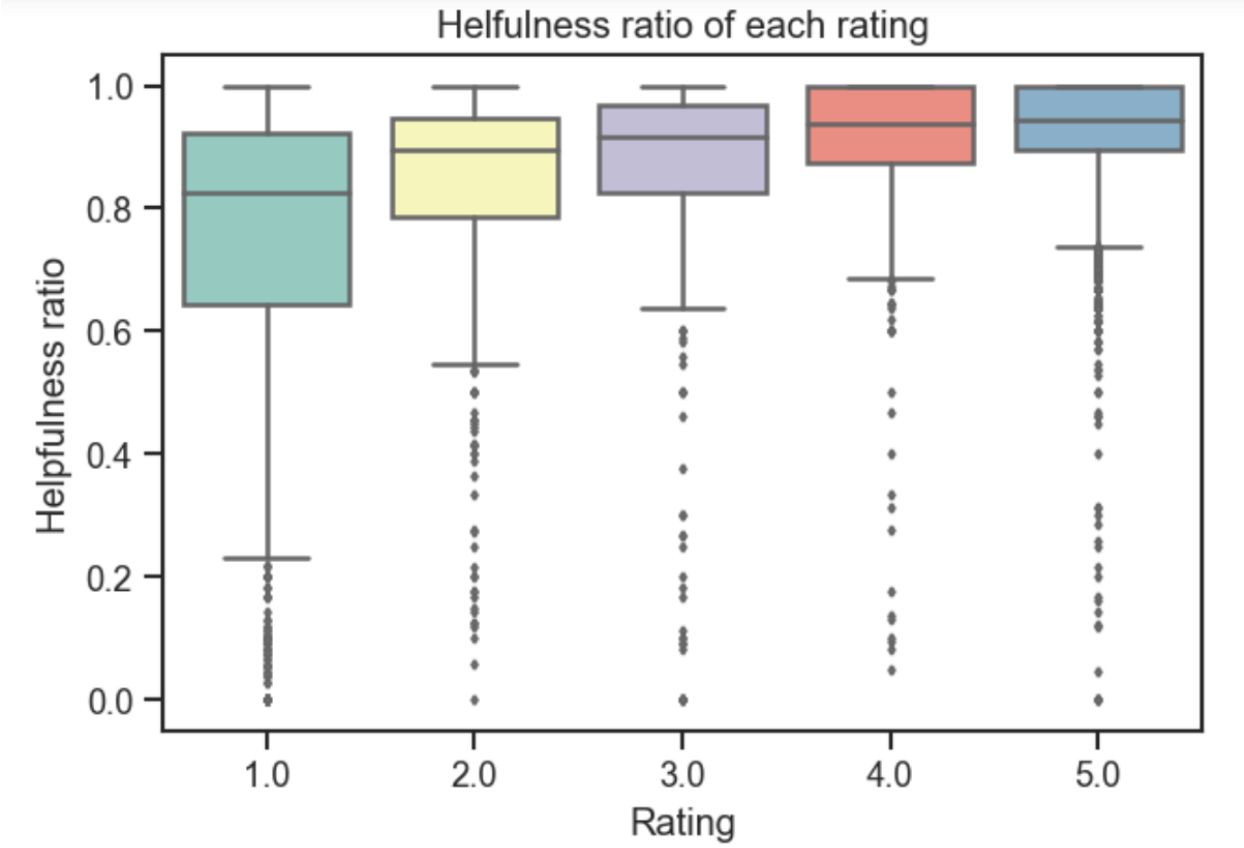
```
reviews[:2]
```

	Unnamed: 0	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	0	A1KLRMWW2FWPL4	0000031887	Amazon Customer "cameramom"	[0, 0]	This is a great tutu and at a really great pri...	5.0	Great tutu- not cheaply made	1297468800	02 12, 2011
1	1	A2G5TCU2WDFZ65	0000031887	Amazon Customer	[0, 0]	I bought this for my 4 yr old daughter for dan...	5.0	Very Cute!!	1358553600	01 19, 2013



2. EXPLORATORY DATA ANALYSIS

2.5 Rating vs Helpfulness

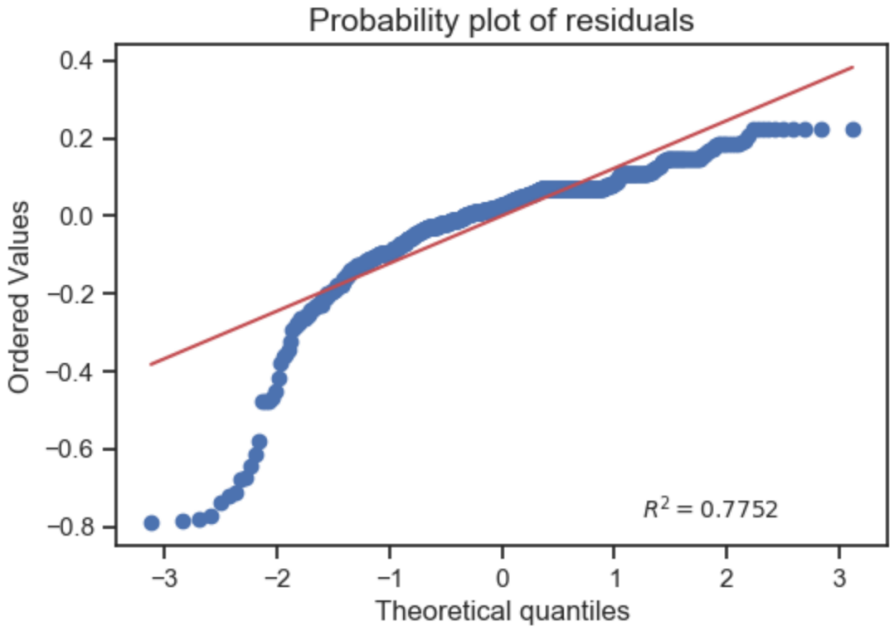


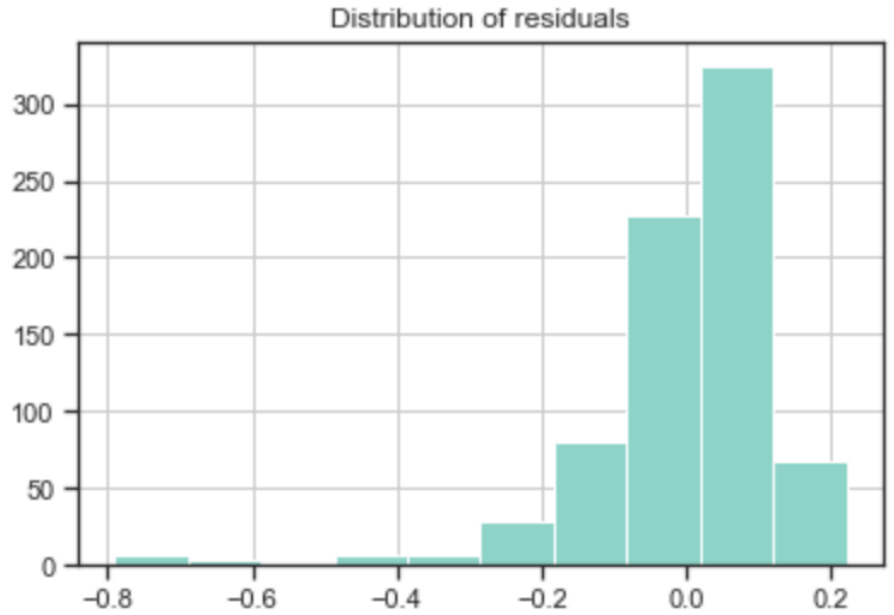
2. EXPLORATORY DATA ANALYSIS

2.5 Rating vs Helpfulness

```
# ANOVA model:
model = ols('helpful_ratio ~ overall', data=df_new.sample(frac=0.1)).fit()
t n>5000
aov_table = sm.stats.anova_lm(model, typ=2)
aov_table
```

	sum_sq	df	F	PR(>F)
overall	1.842036	1.0	95.26344	2.892893e-21
Residual	14.482837	749.0	NaN	NaN





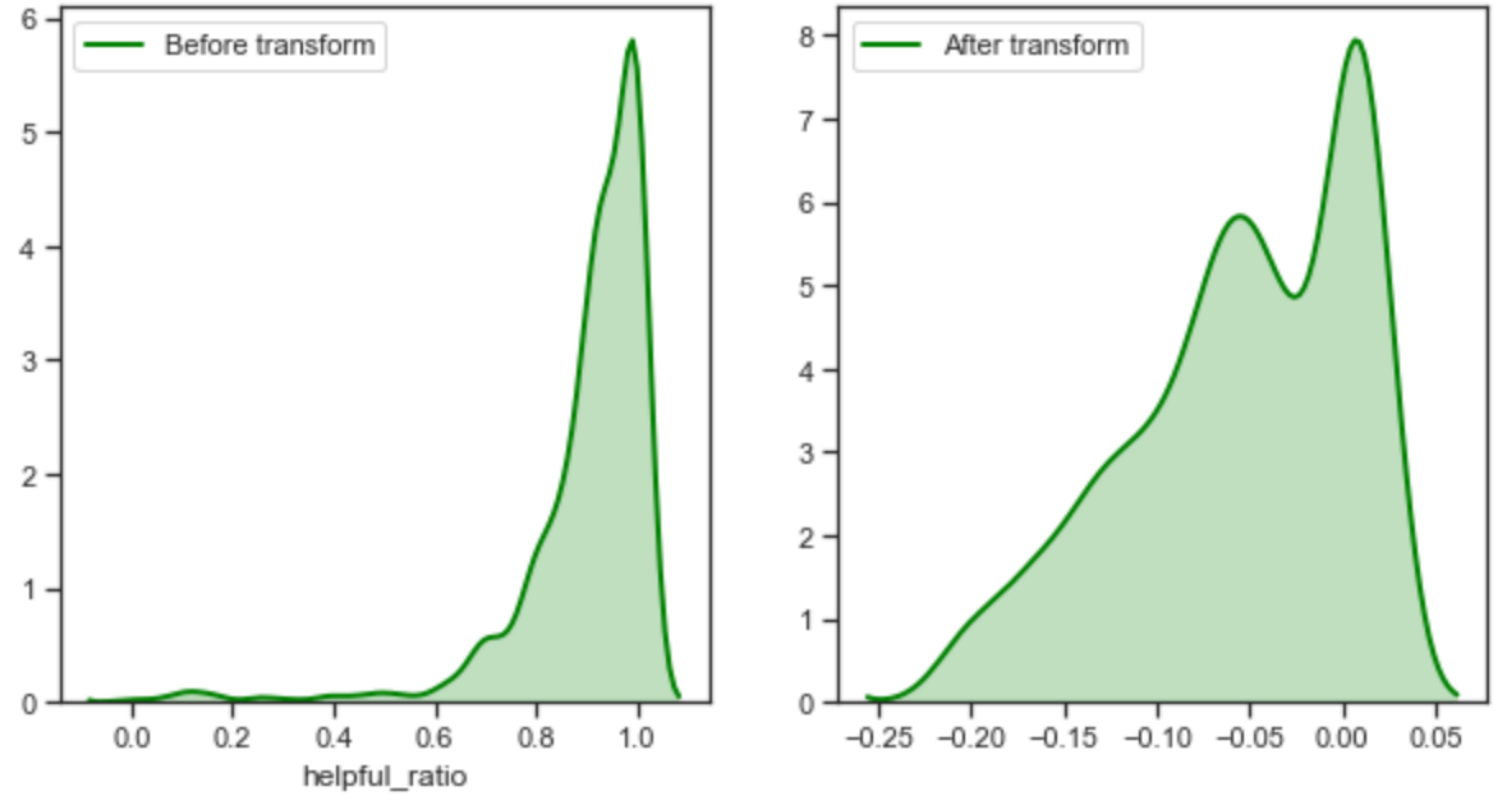
The residuals are not normally distributed, violating the assumption of ANOVA.

2. EXPLORATORY DATA ANALYSIS

2.5 Rating vs Helpfulness

Transform the helpful ratio (Box-cox method) and redo ANOVA and recheck residuals

```
Lambda value used for Transformation: 4.8949738757369
```

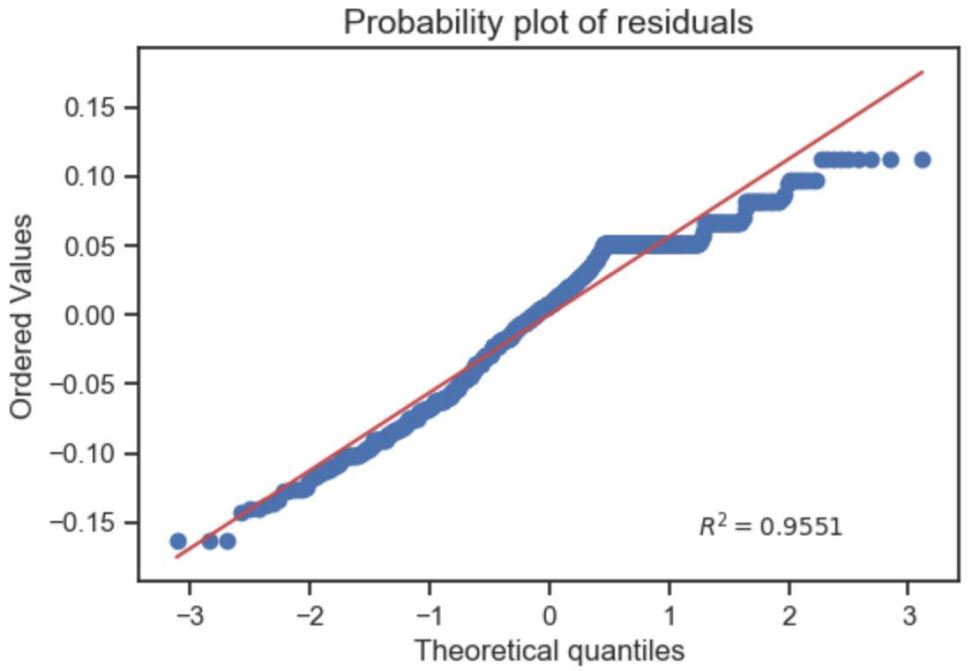


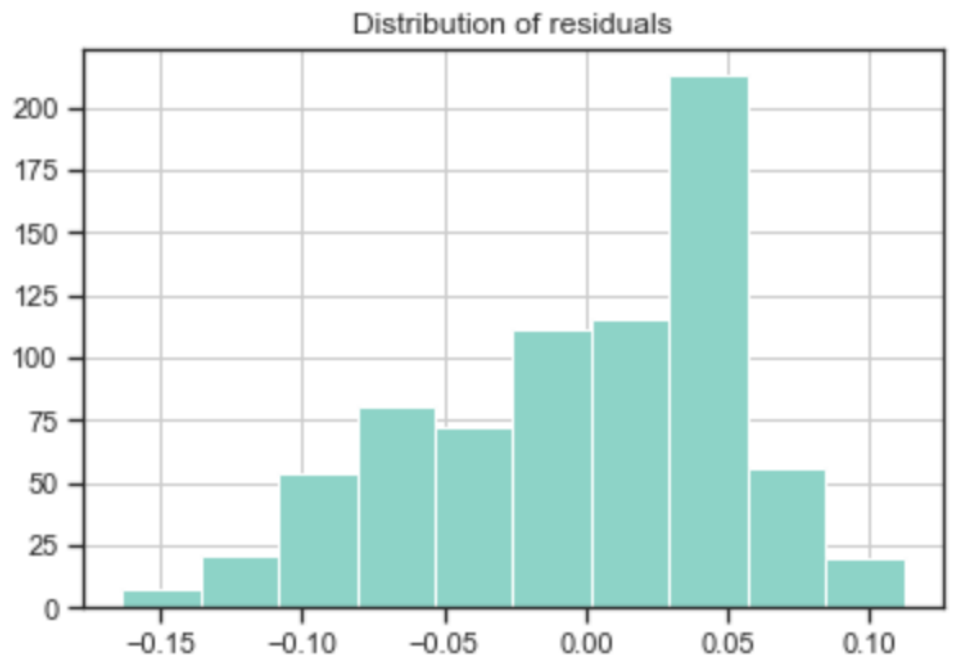
2. EXPLORATORY DATA ANALYSIS

2.5 Rating vs Helpfulness

```
# ANOVA model:
model = ols('transformed_helpful_ratio ~ overall', data=df_sample).fit()
aov_table = sm.stats.anova_lm(model, typ=2)
aov_table
```

	sum_sq	df	F	PR(>F)
overall	0.400514	1.0	100.99735	2.236043e-22
Residual	2.970225	749.0	NaN	NaN





The residuals are still not normally distributed.

2. EXPLORATORY DATA ANALYSIS

2.5 Rating vs Helpfulness

→ Proceed to use Kruskal-Wallis test (non-parametric equivalent of ANOVA)

```
# Kruskal-Wallis test
stats.kruskal(df_sample.overall, df_sample.helpful_ratio) # p < 0.05 means there is correlation
```

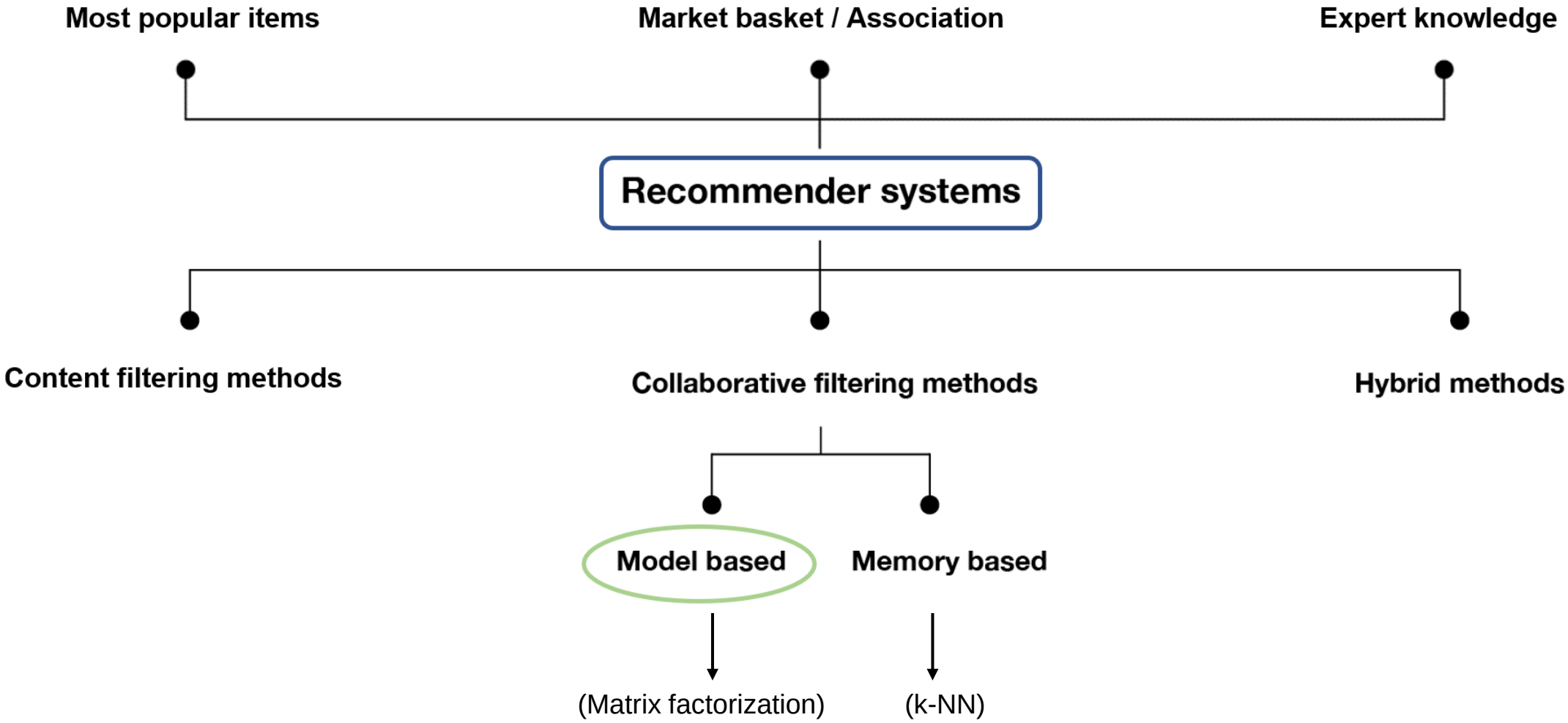
```
KruskalResult(statistic=1094.951857769904, pvalue=4.1311591638275195e-240)
```

OUTLINE

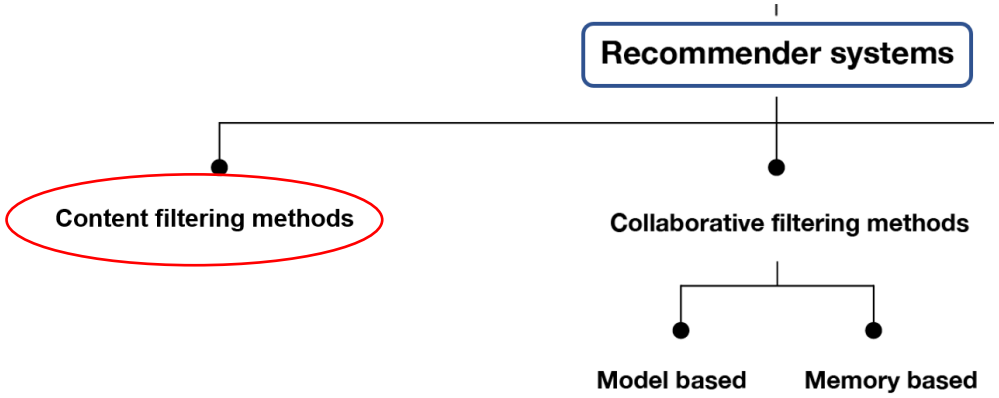
1. Introduction
2. Exploratory Data Analysis
3. Modelling
   - 2.1 Model Selection
   - 2.2 Parameter Tuning
   - 2.3 Evaluating prediction accuracy
   - 2.4 Applications
4. Conclusions
5. Q&A

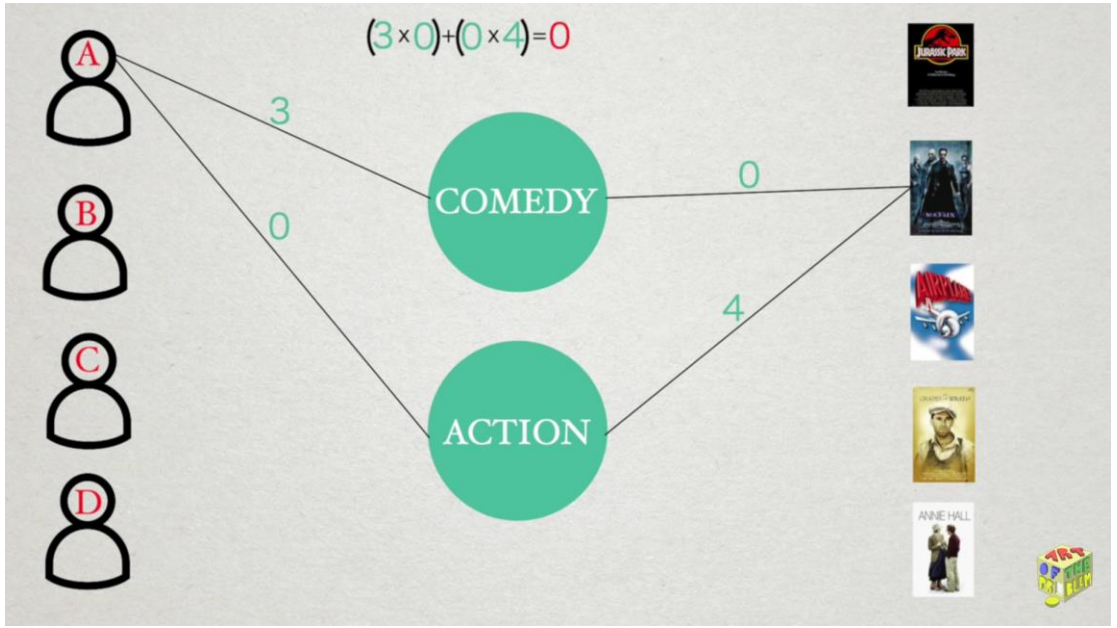
3. MODELLING

Recommender system is built upon filtering methods. Below are the most common filtering methods



3. MODELLING

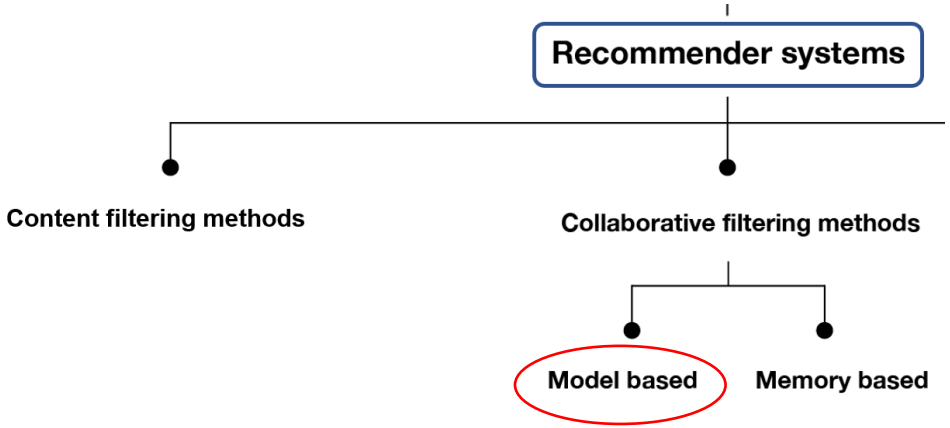




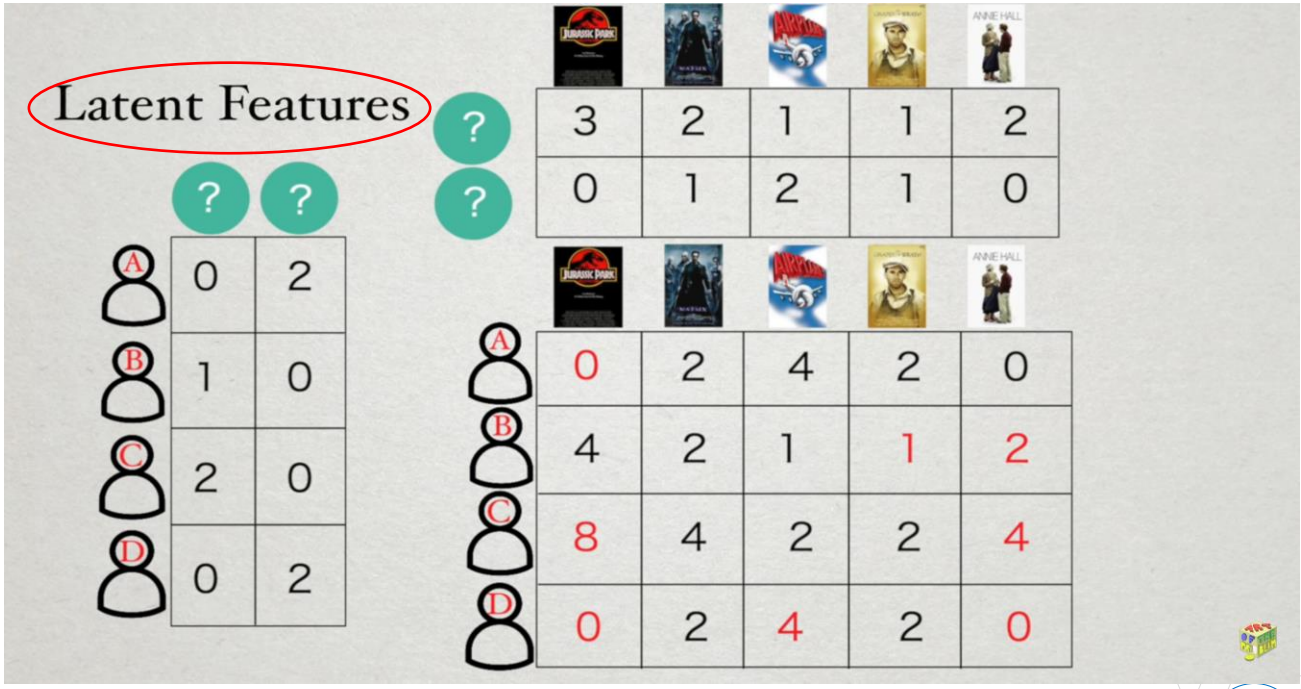
In general, how much do you like watching movies from the following genres?

	Really dislike	Dislike	Neither like nor dislike	Like	Really like
Action	○	◉	○	○	○
Adventure	○	○	◉	○	○
Animation	○	○	○	○	◉
Comedy	○	○	○	◉	○
Crime/Gangster	○	○	○	○	○
Documentary	○	○	○	○	○
Drama	○	○	○	○	○

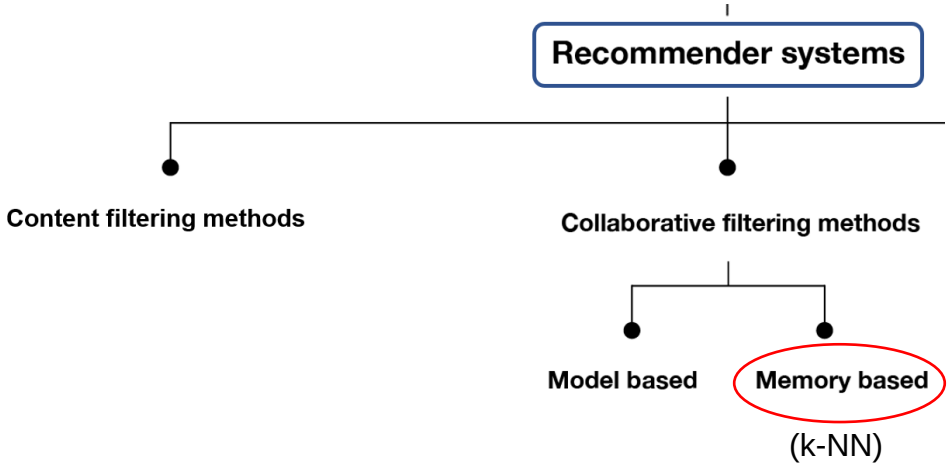
3. MODELLING

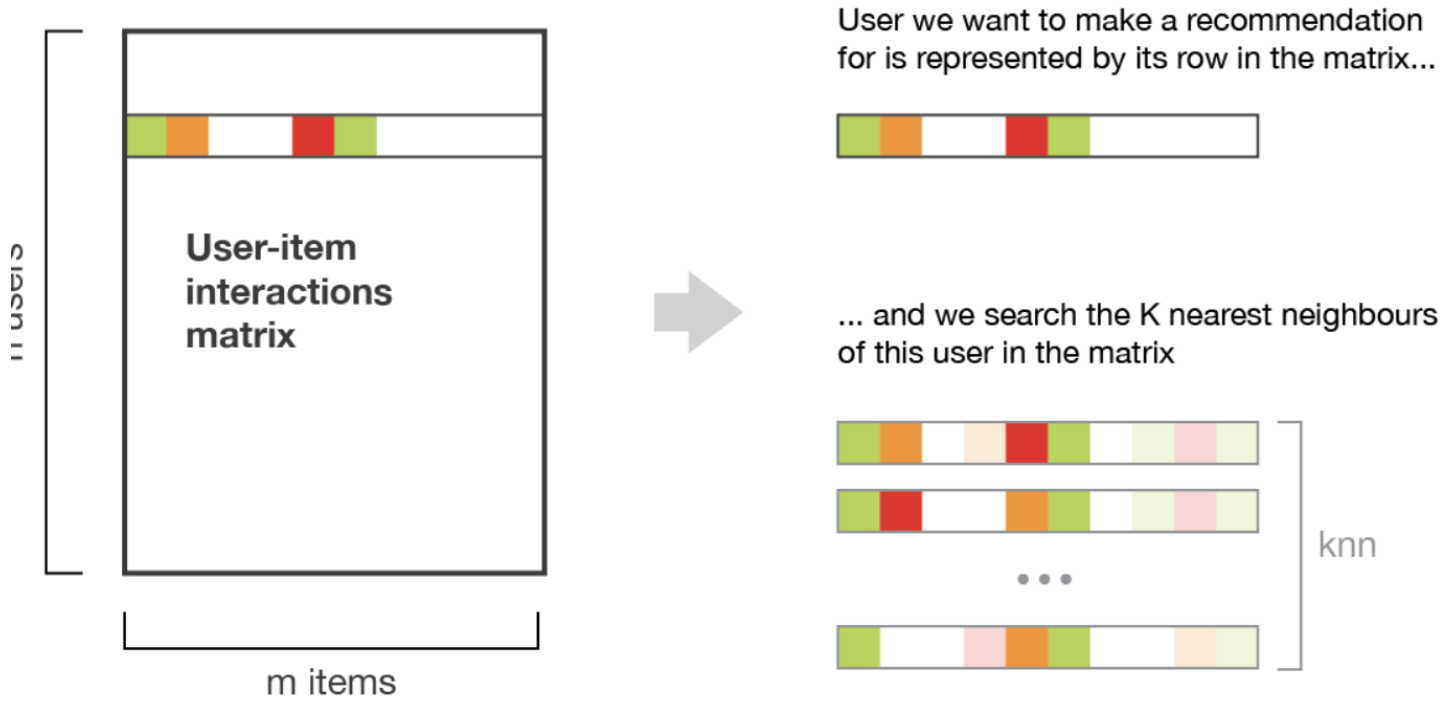


Matrix factorization



3. MODELLING


Recommender systems
Content filtering methods
Collaborative filtering methods
Model based
Memory based
(k-NN)


n users
User-item interactions matrix
m items
User we want to make a recommendation for is represented by its row in the matrix...
... and we search the K nearest neighbours of this user in the matrix
knn

3. MODELLING

	Content Filtering	Collaborative Filtering	
		Model based (Matrix Factorization)	**Memory based (k-NN)**
Advantages	• OK with cold start (new user/product)	• No metadata needed • No expert understanding of features needed	
Disadvantages	• Need to maintain large metadata of users and products • Need user profiling	• Cannot predict for new user/product • Overfitting issue	• Cannot predict for new user/product • Scalability issue (k-NN takes long time) • Lack of diversity (recommend popular choices too often)

3. MODELLING

How to build Collaborative Filtering model with “Surprise” library

```
# Step 1: Set the rating scale
reader = Reader(rating_scale=(1, 5))

# Step 2: Load the dataframe into a Dataset object
data = Dataset.load_from_df(df[['reviewerID', 'asin', 'overall']], reader)

# Step 3: Split into train and test set
trainset, testset = train_test_split(data, test_size=.25)

# Step 4: Choose the algorithm and train
algo = SVD()
algo.fit(trainset)

# Step 5: Estimate errors for testset
predictions = algo.test(testset)
accuracy.rmse(predictions)
```

```
RMSE: 1.0492
```

Out[20]: 1.0492043800084374

3. MODELLING

3.1 Model selection

"Surprise" library provides several algorithms for collaborative filtering:

- NormalPredictor
- BaselineOnly
- KNNBasic
- KNNWithMeans
- KNNWithZScore
- KNNBaseline
- SVD
- SVDpp
- NMF
- SlopeOne
- CoClustering

k-NN inspired algorithms
(**Memory based** Collaborative Filtering)

Matrix factorization-based algorithms
(**Model based** Collaborative Filtering)

Algorithm	test_rmse	fit_time	test_time
SVD	1.092876	2.173522	0.104054
BaselineOnly	1.094409	0.146988	0.122657
KNNBaseline	1.095214	11.023981	0.174190
KNNBasic	1.099781	12.776442	0.254311
KNNWithMeans	1.150865	9.712231	0.266952
KNNWithZScore	1.155883	13.179107	0.154920
SlopeOne	1.156990	7.324387	0.218415
CoClustering	1.185757	4.104849	0.127982
NMF	1.217904	5.337158	0.098736
NormalPredictor	1.417224	0.073801	0.141622

SVD, Baseline, KNNBaseline are the best. However, KNN has scalability issue → choose SVD for parameter tuning.

3. MODELLING

3.2 Parameter tuning

Find the best combination of parameters for SVD

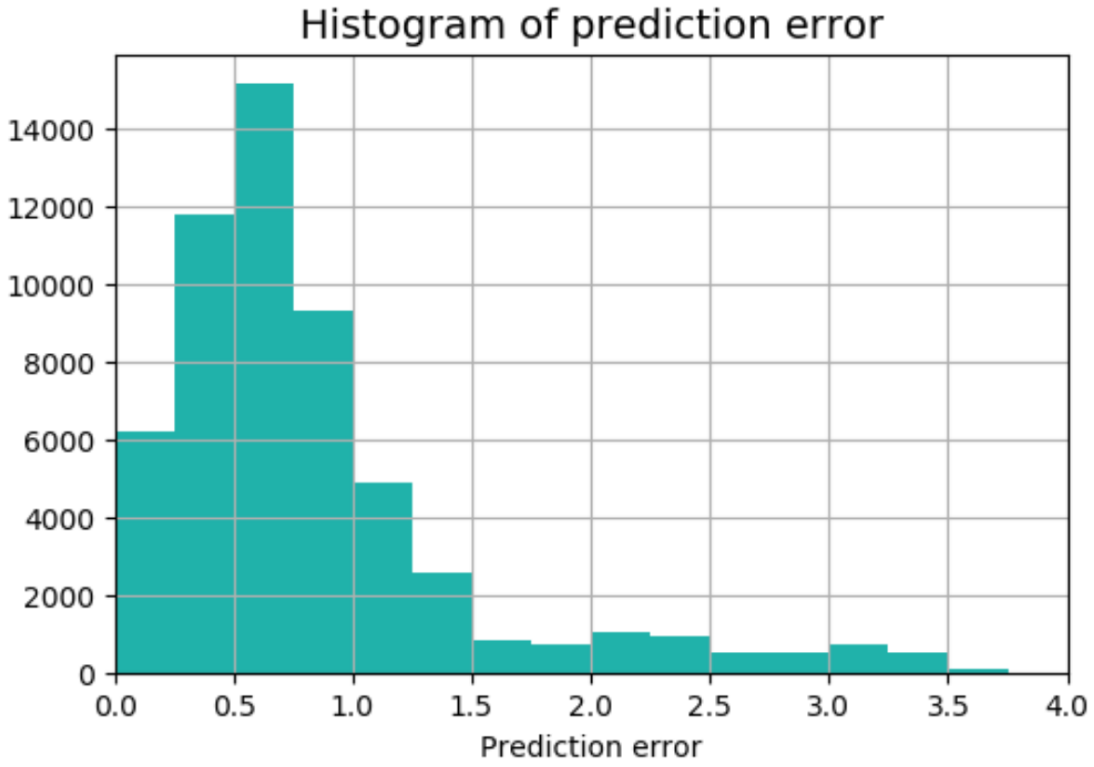
```
# I use the spare dataset (to save computational time) to find the best combinations of parameter

# Select your best algo with grid search.
param_grid = {'n_factors': [50,100,150],'n_epochs': [20,30], 'lr_all': [0.005,0.01],'reg_all':[0.

gs = GridSearchCV(SVD, param_grid, measures=['rmse'], cv=3)

gs.fit(data)

# best RMSE score
print('Best RMSE ', gs.best_score['rmse'])

# combination of parameters that gave the best RMSE score
print('Parameters that give best RMSE', gs.best_params['rmse'])
```

```
Best RMSE  1.0931169104096543
Parameters that give best RMSE {'n_factors': 50, 'n_epochs': 20, 'lr_all': 0.005, 'reg_all': 0.1}
```

3. MODELLING

3.3 Evaluating prediction accuracy

	uid	iid	rui	est	details	Iu	Ui	err
0	A2G5TCU2WDFZ65	0000031887	5.0	4.473038	{'was_impossible': False}	7	17	0.526962
1	A2G5TCU2WDFZ65	B000UU49GG	5.0	4.086050	{'was_impossible': False}	7	8	0.913950
2	A2G5TCU2WDFZ65	B005JJ2762	5.0	3.955912	{'was_impossible': False}	7	3	1.044088




```
Mean prediction error is  0.8226079650151646
Median prediction error is  0.6556078718578511
```

3. MODELLING

3.3 Evaluating prediction accuracy

Best 10 predictions (lowest errors):

```
best_predictions = df_predictions.sort_values(by='err')[:10]
best_predictions
```

3]:

	uid	iid	rui	est	details	Iu	Ui	err
288	A2J4XMWKR8PPD0	B0002FHFZQ	5.0	5.0	{'was_impossible': False}	108	3	0.0
50251	A2JA4IKKDP153M	B008KK0ZJ8	5.0	5.0	{'was_impossible': False}	22	128	0.0
8321	ACW3EKB58CX4D	B000B8P7Y0	5.0	5.0	{'was_impossible': False}	23	61	0.0
313	A2J4XMWKR8PPD0	B00A8FCVEK	5.0	5.0	{'was_impossible': False}	108	2	0.0
308	A2J4XMWKR8PPD0	B007D0MF26	5.0	5.0	{'was_impossible': False}	108	4	0.0
307	A2J4XMWKR8PPD0	B0064Y3PVW	5.0	5.0	{'was_impossible': False}	108	31	0.0
35565	A2SZWP16FQIMTO	B0041VMVJM	5.0	5.0	{'was_impossible': False}	21	19	0.0
289	A2J4XMWKR8PPD0	B000FOJZVQ	5.0	5.0	{'was_impossible': False}	108	29	0.0
297	A2J4XMWKR8PPD0	B001UEN2ZU	5.0	5.0	{'was_impossible': False}	108	14	0.0
291	A2J4XMWKR8PPD0	B000IDET4W	5.0	5.0	{'was_impossible': False}	108	11	0.0

3. MODELLING

3.3 Evaluating prediction accuracy

Worst 10 predictions (highest errors):

In [59]:
```
worst_predictions = df_predictions.sort_values(by='err')[-10:]
worst_predictions
```

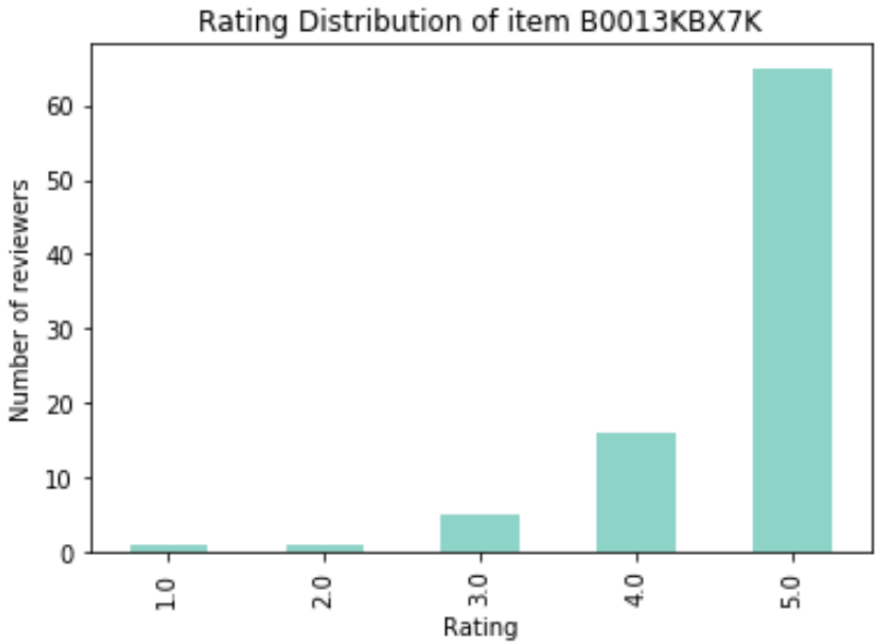
Out[59]:

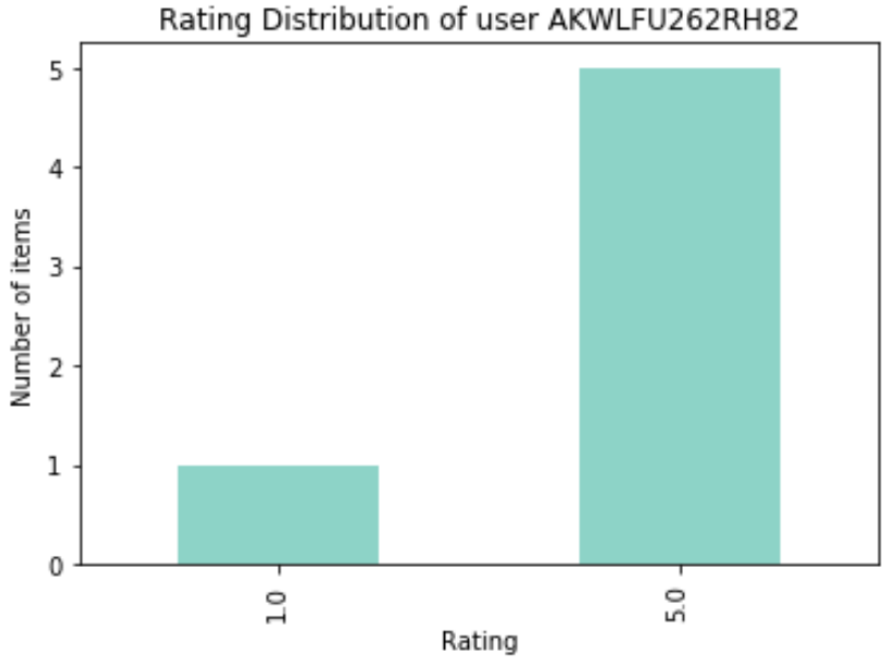
	uid	iid	rui	est	details	Iu	Ui	err
29560	A3S42Z04G0ECDJ	B004KP73D8	1.0	4.685305	{'was_impossible': False}	6	18	3.685305
55495	ARWA6MYXE6R9	B00FKDDJ46	1.0	4.695923	{'was_impossible': False}	5	7	3.695923
40696	AXA3C2UHJQPPS	B00540OWOQ	1.0	4.697257	{'was_impossible': False}	13	4	3.697257
20611	AKWLFU262RH82	B0013KBX7K	1.0	4.715403	{'was_impossible': False}	5	67	3.715403
14622	AXQW6IS0Z62BR	B000MXIMZK	1.0	4.726600	{'was_impossible': False}	4	31	3.726600
20654	A27JU93NC06175	B0013KDTTA	1.0	4.732063	{'was_impossible': False}	5	37	3.732063
17610	A3H40XHTYVCCCV	B000UJZAA6	1.0	4.750451	{'was_impossible': False}	15	19	3.750451
38160	AIZBSI36ZSV9E	B004KP73D8	1.0	4.772659	{'was_impossible': False}	5	18	3.772659
3996	A3M27ACKH1M1MZ	B0006AAS5Q	1.0	4.784892	{'was_impossible': False}	5	27	3.784892
28157	A211W8JLJFDIC0	B002HIJC8Y	1.0	4.822330	{'was_impossible': False}	13	32	3.822330

3. MODELLING

3.3 Evaluating prediction accuracy

Analysing the (user, item) circled in red in previous slide to find out reason for bad prediction.





The bad prediction seems to be an outlier. The majority of users gave that item a high rating, which is similar to our prediction, but that particular user gave an usually low rating.

3. MODELLING

3.4 Applications

Predict the ratings for new combinations of (user, item)

```
Predicted rating of item  B009G6MRLE by user  A1GKR5QUCDUPWP
```

Out[64]: `[4.183416085901702]`

Get top 10 recommendations for a user

```
Top 10 recommendations for user  AJP6FDRCQQQJI
```

Out[67]:

	asin	price	title
14298	B0081IZ3UA	0.99	925 Sterling Silver Cubic Zirconia Stud Earrings
2545	B000MARQFA	NaN	Propper Men's Canvas Tactical Pant
9758	B005JD3VKO	9.75	Gold Toe Boys 8-20 6 Pack Sport Quarter Sock
3581	B001PPE9G0	NaN	RARE Hour Glass Cufflinks Gift Boxed
16956	B007DLVLAW	7.99	Lock Laces Elastic Shoelace and Fastening System
16458	B005MKGOOY	131.25	Citizen Men's BM8475-26E "Eco-Drive"...
15622	B008R6TIFW	NaN	Hanes Men's 6 Pack Mid-rise Brief, Blue Assort...
4972	B000UTTKG6	NaN	Aerosoles Women's Blue Gene Boot
2910	B0007DG61C	NaN	Naturalizer Women's Malvina Loafer
5731	B0031U0QBO	17.99	Carhartt Men's Utility Suspender

4. CONCLUSIONS

EDA

- 280k reviews from 39k reviewers for 23k products.
- On average, each reviewer gives 7 reviews and each product has 12 reviews.
- More than half (160k out of 280k) of reviews has a rating of 5. Less than 20k reviews have a rating of 1 or 2.
- The number of reviews grows exponentially over the years. There is little seasonal variation in the number of reviews.
- Mean rating decreases slightly over the period.
- Higher rating has higher helpfulness ratio.

Modelling

- There are several filtering methods to make recommendation list. One of them is collaborative filtering, which includes matrix factorization algorithms.
- In matrix factorization, the matrix of ratings can be decomposed into a user matrix and a product matrix. Taking the dot product of those 2 matrices will give us prediction of new (user, product) pairs.
- Several collaborative filtering algorithms were tested on our dataset. SVD, a famous matrix factorization-based algorithm, gives a good prediction accuracy. The majority of prediction errors are less than 1.
- We can use the results to give the prediction to new (user, product) pairs and create a list of top 10 recommendations (highest predicted ratings).

Questions?