What is Next (WIN)

Exploring Next Best Action Strategies

Batoy, De Leon, Mariano

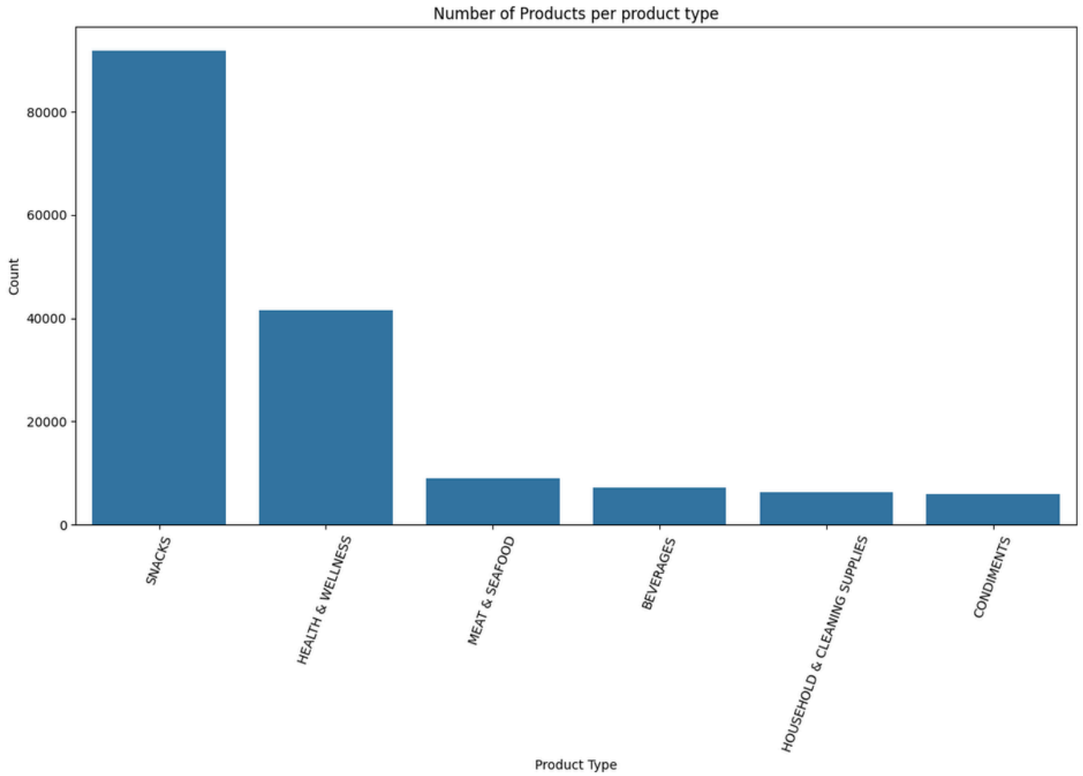
Task

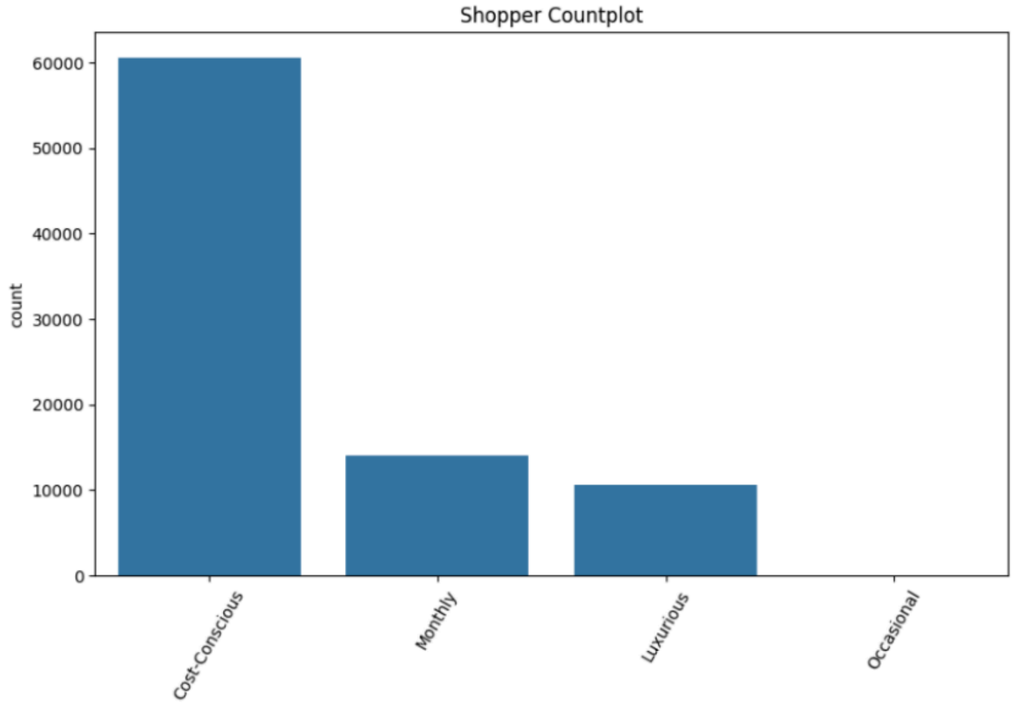
Given the dataset that contains information on customer grocery basket transaction history, provide a source of info that can be leveraged to predict and suggest the next best action (NBA) for each customer.

The Dataset

	CST_ID	TRANS_DATE	PRODUCT_TYPE	PRODUCT_BRAND	SHOPPER	CST_SINCE
0	157105.0	2023-09-28	SNACKS	Ruffles	Monthly	2023-08-09
1	374554.0	2024-03-17	HEALTH & WELLNESS	Nordic Naturals	Cost-Conscious	2022-01-08
2	374554.0	2024-03-12	HEALTH & WELLNESS	Amazing Grass	Cost-Conscious	2022-01-13
3	374554.0	2022-04-03	SNACKS	Ruffles	Cost-Conscious	2022-01-09
4	374554.0	2022-02-26	SNACKS	Ruffles	Cost-Conscious	2022-01-09

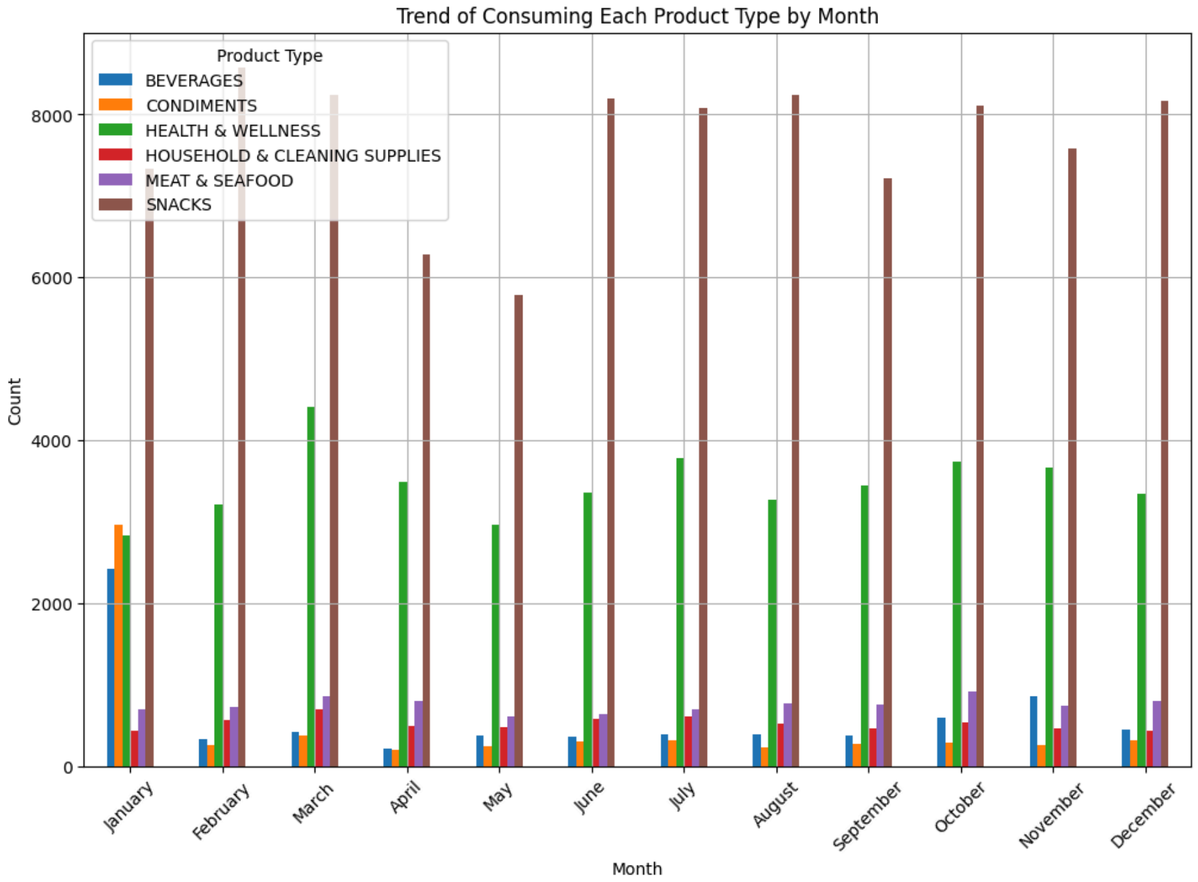
Exploratory Data Analysis





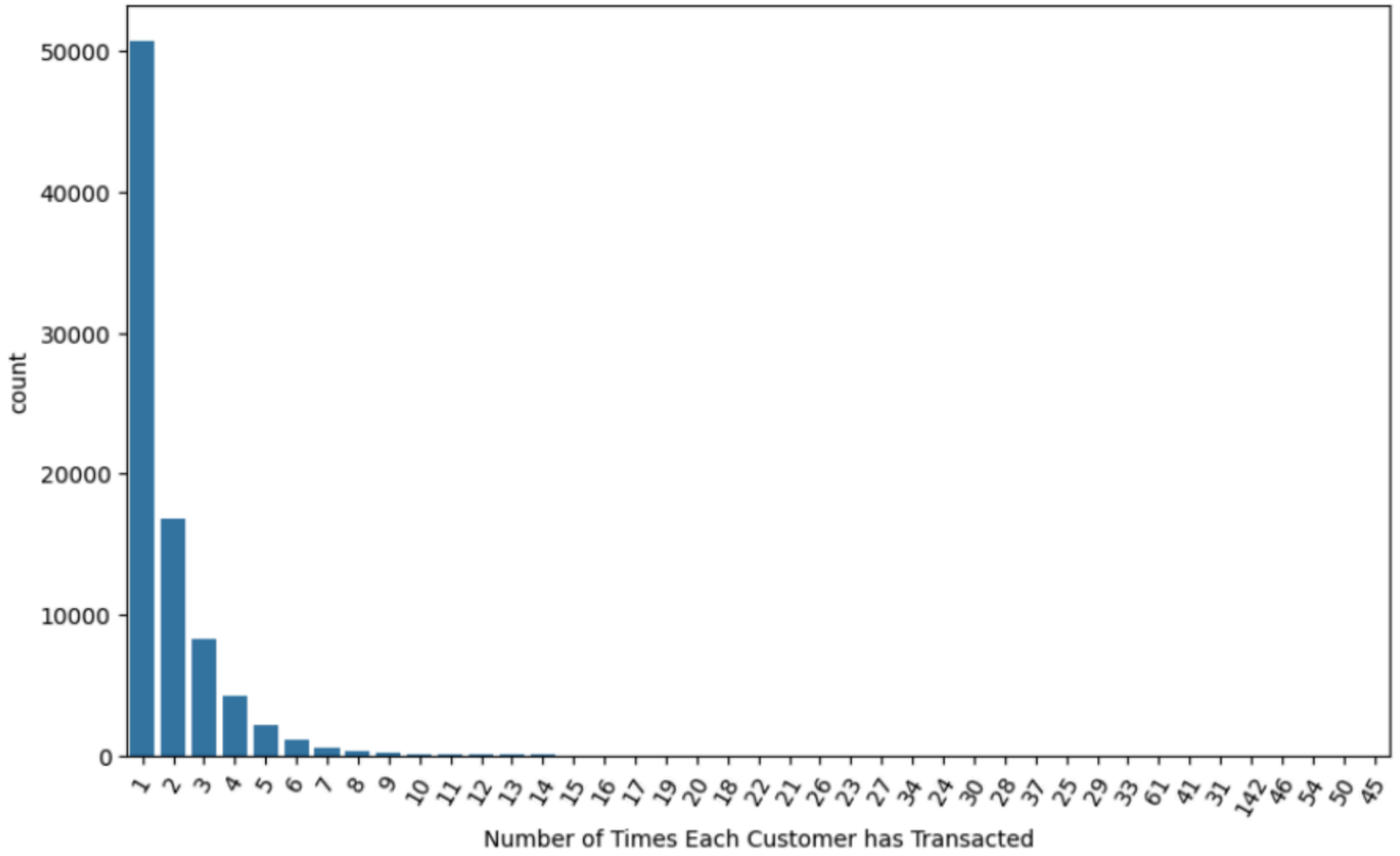
Majority of customers bought snacks and are cost conscious.
There are only 32 occasional shoppers.

Exploratory Data Analysis



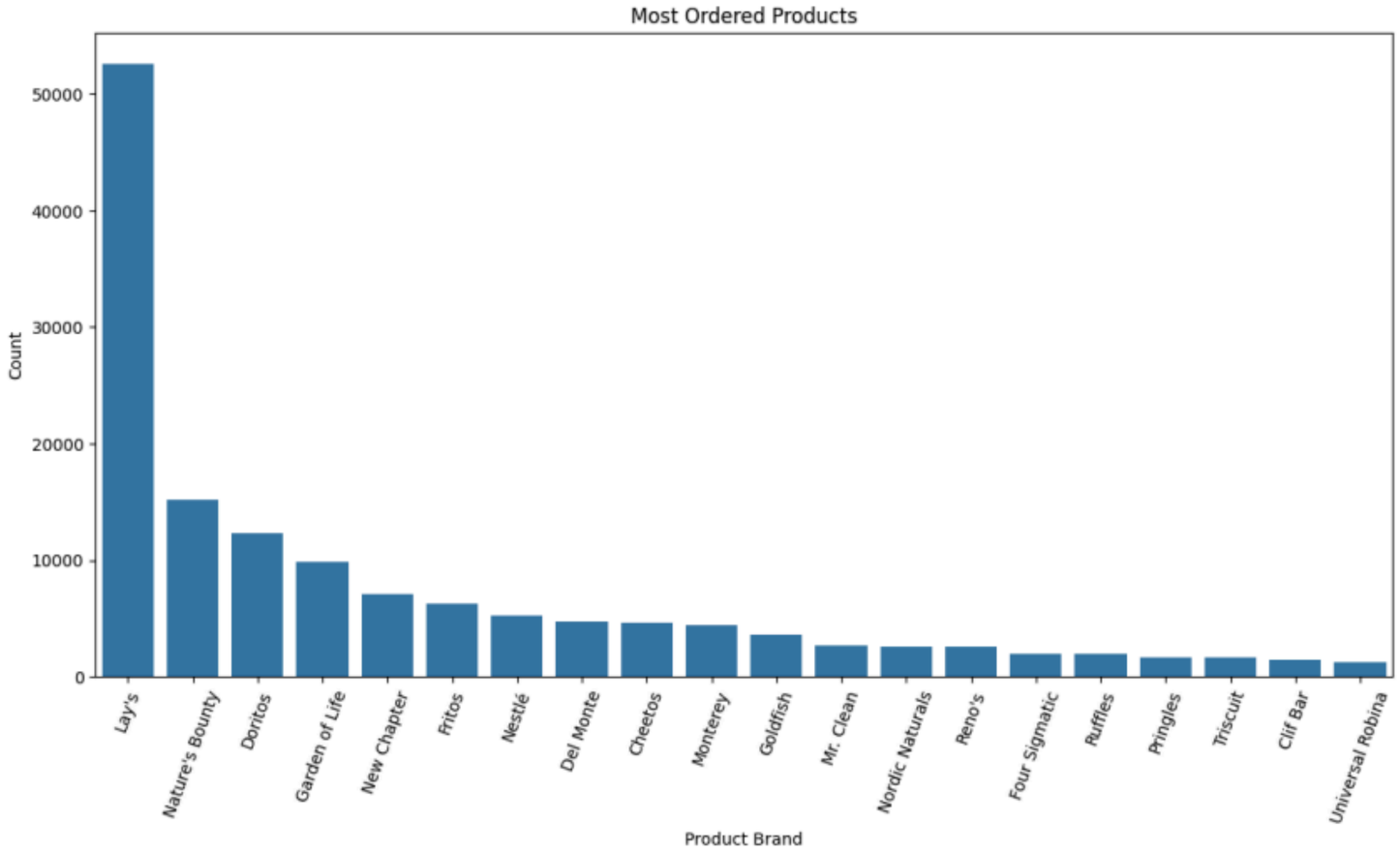
In January, there is a notable increase in purchase of beverages and condiments.

Exploratory Data Analysis



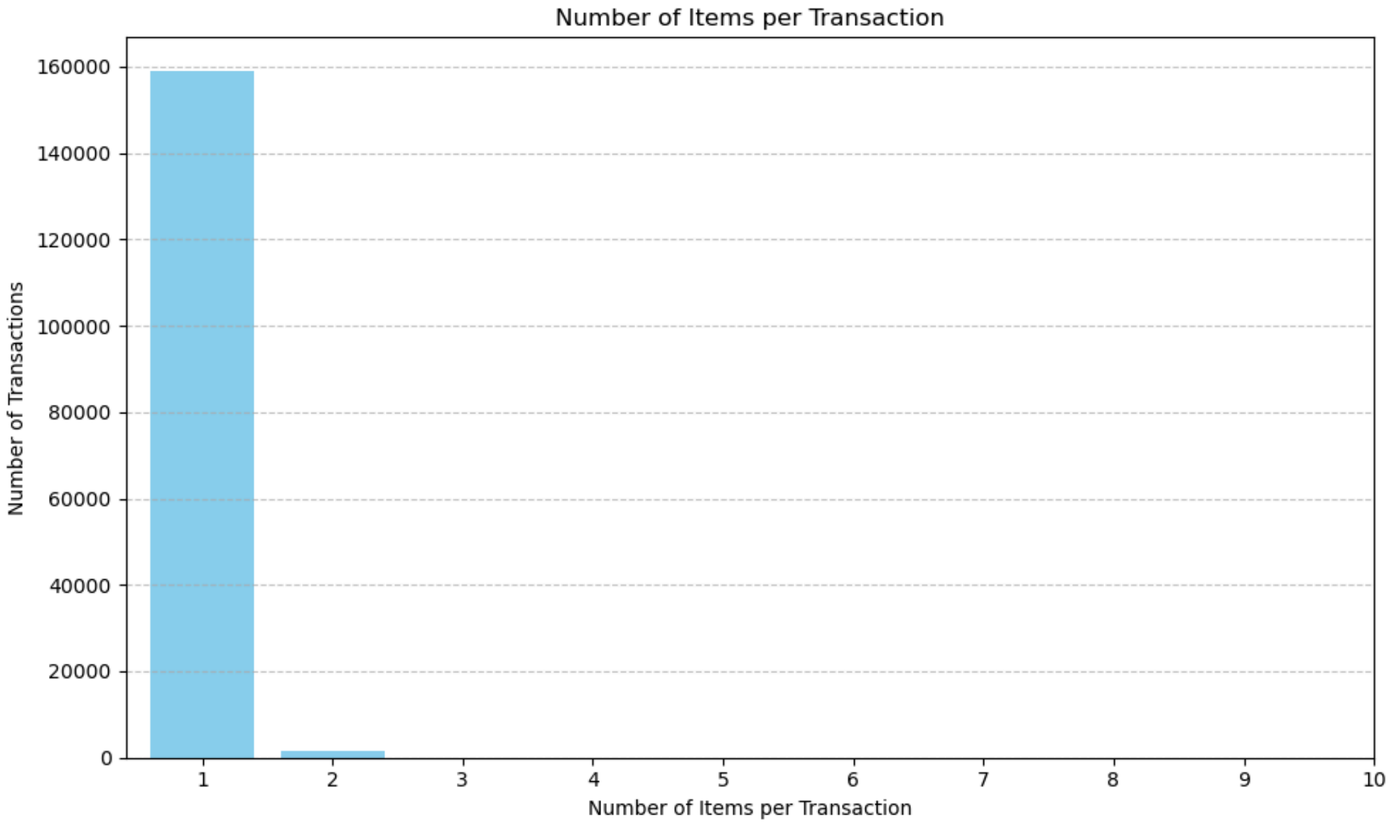
Most customers only transacts once.

Exploratory Data Analysis



Lay’s is the most frequently bought item.

Exploratory Data Analysis



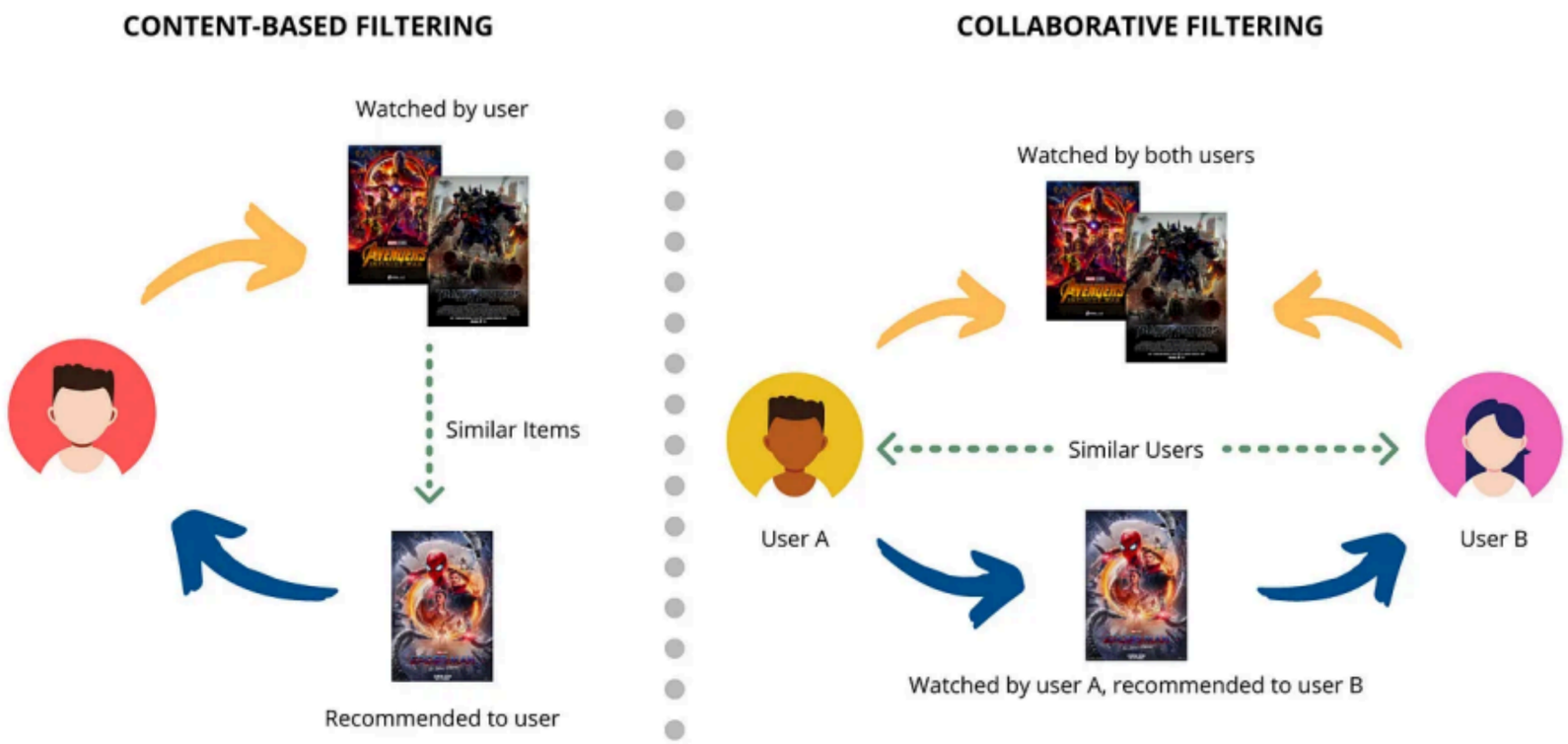
Almost all transactions* consists of one item only.

Recommender System

Collaborative Filtering

Collaborative Filtering

Collaborative filtering is a type of recommender system. It groups users based on similar behavior, recommending new items according to group characteristics. -IBM



Collaborative Filtering

DATA REQUIREMENT

1. Set of Items
2. Set of Users
3. Ratings - explicit (rating on a scale of 1 to 5, likes or dislikes) or implicit (viewing an item, adding it to a wish list, the time spent on an article).

	i_1	i_2	i_3	i_4	i_5
u_1	5		4	1	
u_2		3		3	
u_3		2	4	4	1
u_4	4	4	5		
u_5	2	4		5	2

Collaborative Filtering

ALGORITHMS

1. **Memory-Based** - in which statistical techniques are applied to the entire dataset to calculate the predictions.

Example: KNN

1. **Model-Based** - which involves a step to reduce or compress the large buT sparse user-item matrix

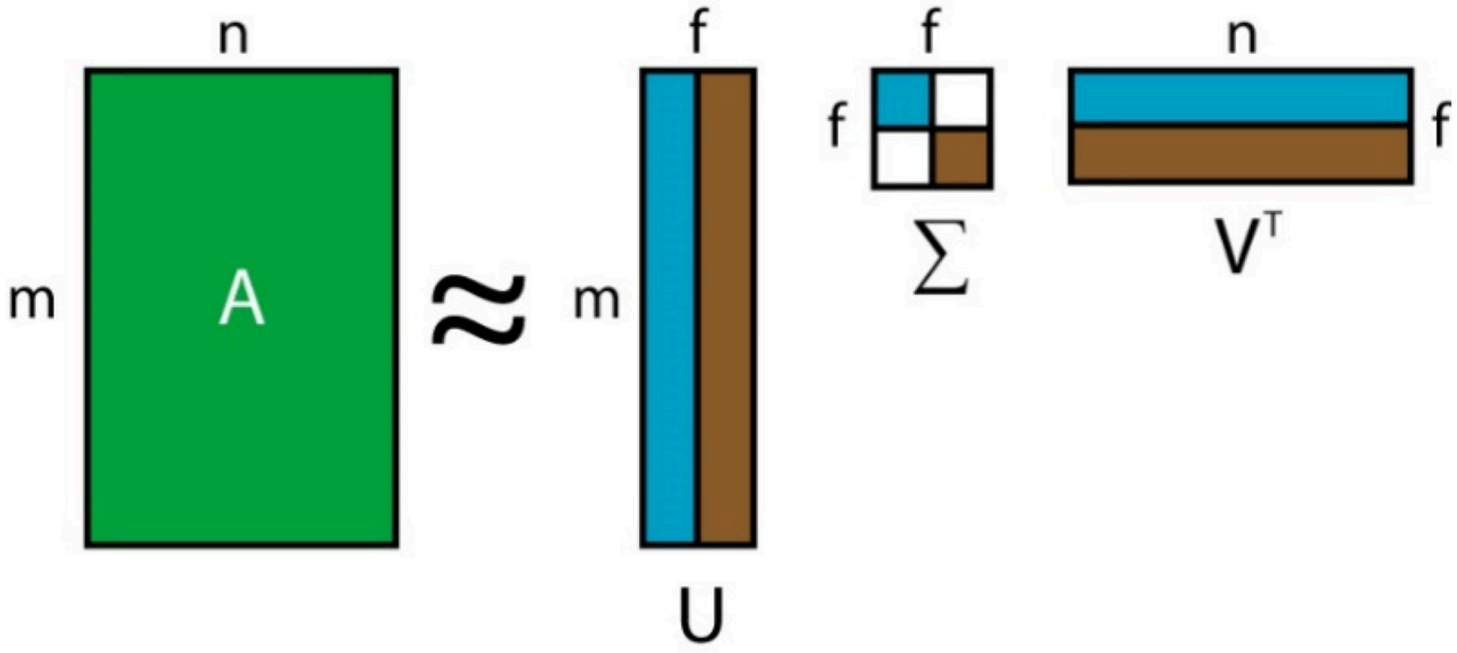
Example: SVD and ALS

Recommender System

Singular Value Decomposition (SVD)

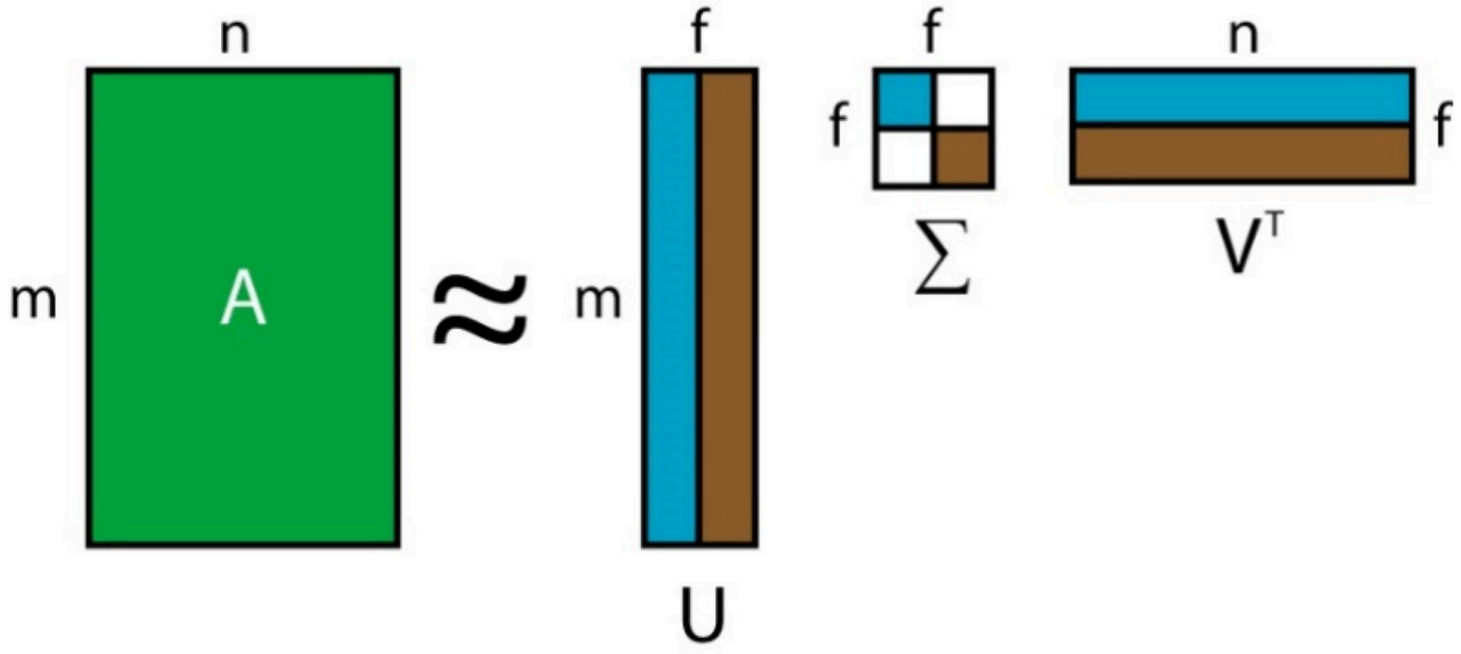
Matrix Factorization Technique

- We have a user-item interaction matrix A
  - A_{mn} = rating of user **m** to item **n**
- Decomposes matrix A into matrices U, Σ, and V^T
  - U = user features
  - Σ = singular values, representing the importance of each latent factor.
  - V^T = item features



Matrix Factorization Technique

- Reduce the dimensionality by keeping only the top k singular values. This reduces the noise and focuses on the most significant features.
- Predicts missing values by reconstructing an approximation of A



Implementation

work user	10000	10001	1000167	10001797	10005525	10007394	10007399	10009	10012725	10012975	...
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
-Eva-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
06nwingert	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1983mk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1dragones	5.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
zjakkelien	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
zmagic69	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
zquilts	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
zwaantje	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
zzshupinga	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

Needs to be in pivot table format (this is done by surprise library)

Implementation

	CST_ID	PRODUCT_BRAND_ID	rating
0	157105.0	51.0	1.0
1	157105.0	67.0	0.0
2	157105.0	102.0	0.0
3	157105.0	9.0	0.0
4	157105.0	82.0	0.0
...	...	...	...
10900987	871744.0	57.0	0.0
10900988	871744.0	11.0	0.0
10900989	871744.0	91.0	0.0
10900990	871744.0	128.0	0.0
10900991	871744.0	99.0	0.0

10900992 rows × 3 columns

```
# New Reader
tic = time.time()
reader = Reader(rating_scale=(0, 5)) #change to (1-5)
readtime = time.time()-tic
print(readtime)
# The columns must correspond to user id, item id and ratings (in that order).
toc = time.time()
new_rec_data = Dataset.load_from_df(new_rec_df, reader)
loadtime = time.time()-toc
print(loadtime)
# sample random trainset and testset
# test set is made of 25% of the ratings.
tac = time.time()
new_trainset, new_testset = train_test_split(new_rec_data, test_size=.25, random_state=12)
splittime = time.time()-tac
print(splittime)
```

Implementation

```
new_svd_algo = SVD()

new_svd_algo.fit(new_trainset)

new_predictions = new_svd_algo.test(new_testset)
```

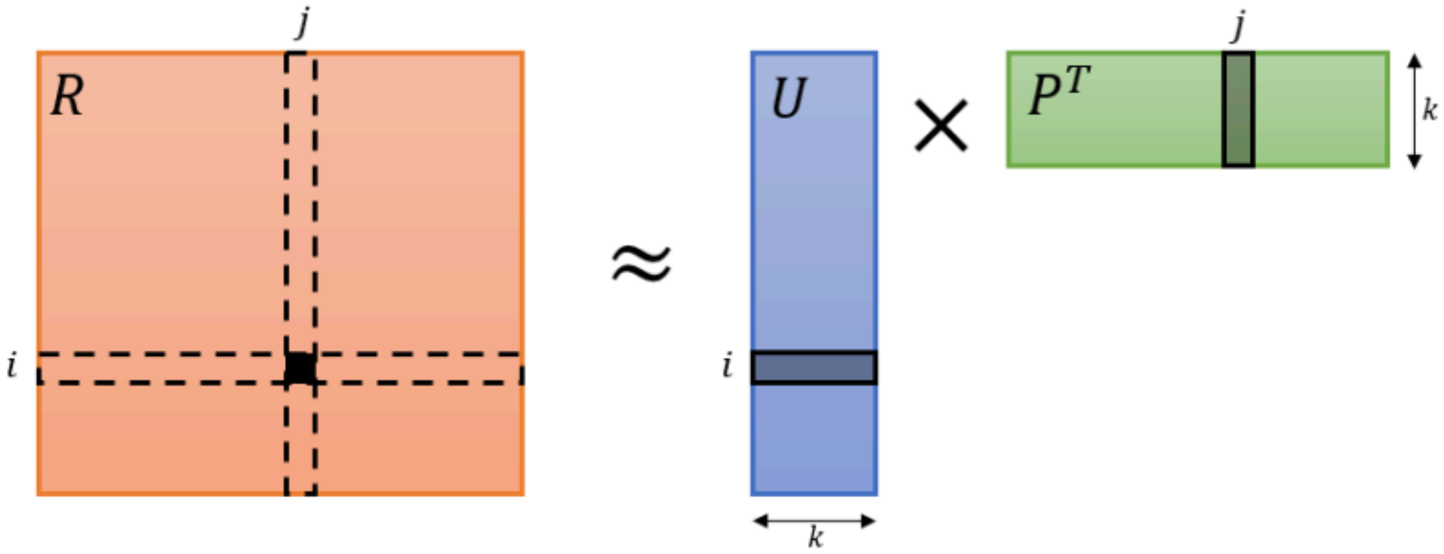
Calculated RMSE and tested predictions

Recommender System

Alternating Least Squares (ALS)

Matrix Factorization Technique

- We have a user-item interaction matrix R
  - R_{ij} = rating of user **i** to item **j**
- Decomposes matrix R into matrices U and P^T
  - U = user latent features
  - V = item latent features



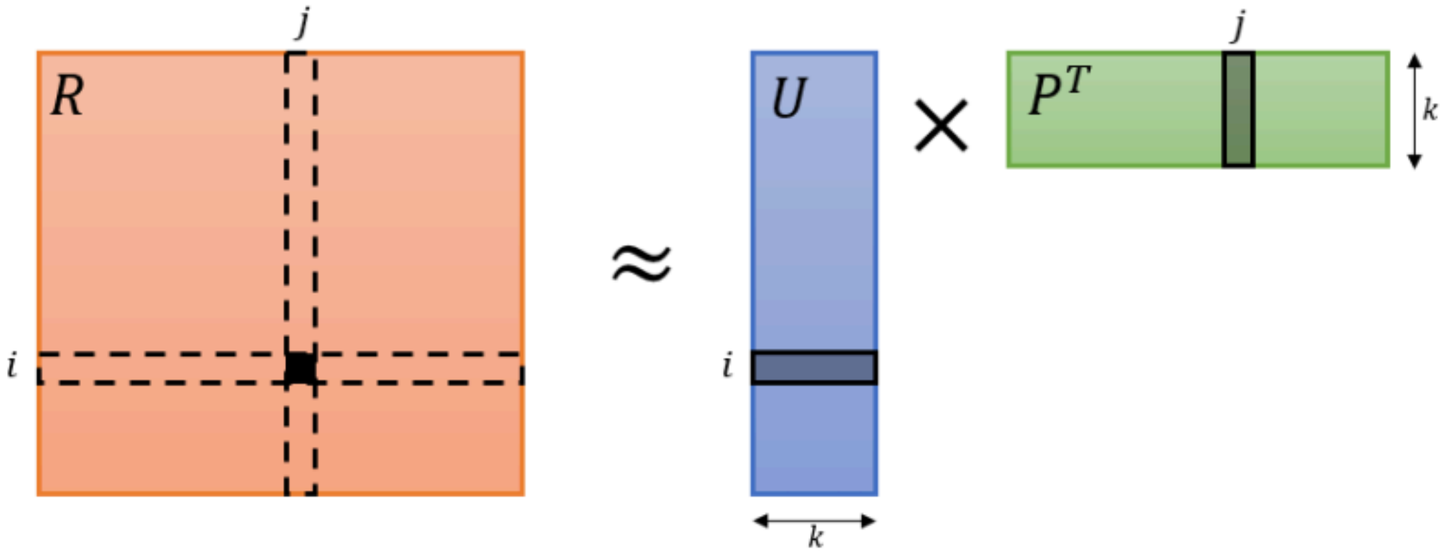
Matrix Factorization Technique

The objective function for ALS is given by:

$$\min_{U,P} \sum_{(i,j)\in\kappa} (r_{ij} - u_i^T p_j)^2 + \lambda \left(\|u_i\|^2 + \|p_j\|^2 \right)$$

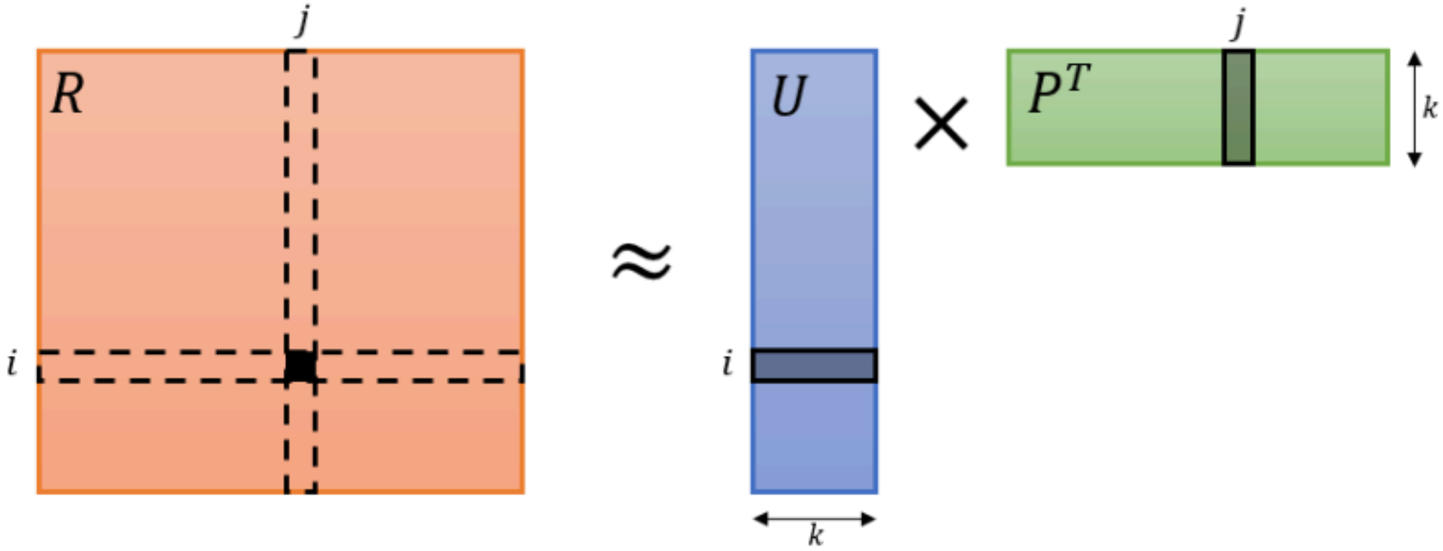
where:

- κ is the set of observed user-item pairs.
- r_{ij} is the actual rating given by user i to item j.
- u_i is the i-th row of U.
- p_j is the j-th row of P.
- λ is a regularization parameter to prevent overfitting.



Matrix Factorization Technique

- U and P first have small random values
- ALS alternates between fixing U and updating P, and fixing P and updating for U. This is why it's called "Alternating" Least Squares.
- This is done until you get an approximation of R with the least error
- Missing values are predicted in the process



Implementation

userId	itemId	rating
833.0	39.0	1.0
74775.0	68.0	1.0
74904.0	51.0	1.0
158803.0	15.0	1.0
158803.0	69.0	2.0
185494.0	69.0	1.0
244597.0	28.0	1.0
413791.0	51.0	1.0
424155.0	60.0	1.0
450950.0	69.0	1.0
480179.0	26.0	1.0
480179.0	37.0	1.0
480179.0	51.0	1.0
485791.0	69.0	1.0
494155.0	51.0	2.0
503650.0	28.0	1.0
576374.0	117.0	1.0
593226.0	89.0	2.0
603584.0	39.0	1.0
629243.0	3.0	1.0

Implementation

```
# Create Spark session
spark = SparkSession.builder.appName('ALSRecommendation').getOrCreate()

# Convert pandas DataFrame to Spark DataFrame
spark_df = spark.createDataFrame(df_reordered_desc)

# Select required columns and rename them for ALS
als_data = spark_df.selectExpr('CST_ID as userId', 'PRODUCT_BRAND_ID as itemId', 'REORDERED as rating')

# Show the data
als_data.show()
```

```
# Initialize the ALS model
als = ALS(
    maxIter=10,
    regParam=0.1,
    userCol="userId",
    itemCol="itemId",
    ratingCol="rating",
    coldStartStrategy="drop"
)

# Split the data into training and test sets
(training, test) = als_data.randomSplit([0.8, 0.2])

# Train the model
model = als.fit(training)
```

Implementation

```python
# Make predictions
predictions = model.transform(test)

# Evaluate the model
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating", predictionCol="prediction")
rmse = evaluator.evaluate(predictions)
print(f"Root-mean-square error = {rmse}")

# Show some predictions
predictions.show()
```

```
Root-mean-square error = 0.44615383048377333
+--------+------+------+----------+
|  userId|itemId|rating|prediction|
+--------+------+------+----------+
|   833.0|  39.0|   1.0| 0.8509407|
| 74775.0|  68.0|   1.0| 0.6192534|
| 74904.0|  51.0|   1.0| 0.9062147|
|158803.0|  15.0|   1.0|0.86456954|
|158803.0|  69.0|   2.0|0.93598175|
|185494.0|  69.0|   1.0| 0.8697497|
|244597.0|  28.0|   1.0|0.55100703|
|413791.0|  51.0|   1.0| 0.8748223|
|424155.0|  60.0|   1.0|0.96044034|
|450950.0|  69.0|   1.0|0.94032764|
|480179.0|  26.0|   1.0| 0.7297675|
|480179.0|  37.0|   1.0| 0.6630554|
|480179.0|  51.0|   1.0|0.75611144|
|485791.0|  69.0|   1.0| 0.7400789|
|494155.0|  51.0|   2.0| 0.9062147|
|503650.0|  28.0|   1.0|0.75150627|
|576374.0| 117.0|   1.0|0.82115996|
|593226.0|  89.0|   2.0| 0.9432781|
|603584.0|  39.0|   1.0|0.87056136|
|629243.0|   3.0|   1.0|0.86316764|
+--------+------+------+----------+
```

Ratings Feature

Feature Engineering Attempts

Ratings Feature

- Frequency as rating
  - 1 to 125
- Adjusted frequency as rating
  - 1 to 5, where 5 represents frequencies equal to or greater than 5 occurrences
- TF-IDF-inspired rating
  - item as term, customers as documents
  - considers frequently bought items as essential needs
  - scaled to 1-5

$$\text{Rating} = 1 + 4 \cdot \text{minmax}(tf \cdot \log(\tfrac{N}{df})).$$

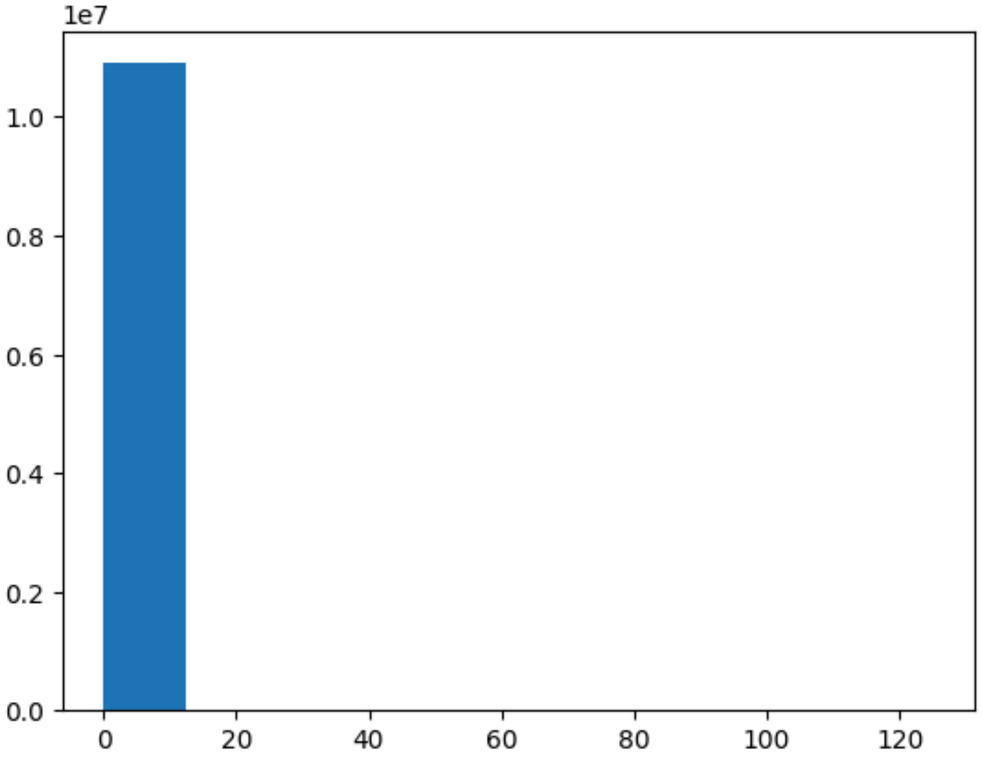
- Adjusted ratio rating
  - ratio of frequency to number of transactions
  - considers comparative ratings of a customer across other items
  - scaled to 1-5

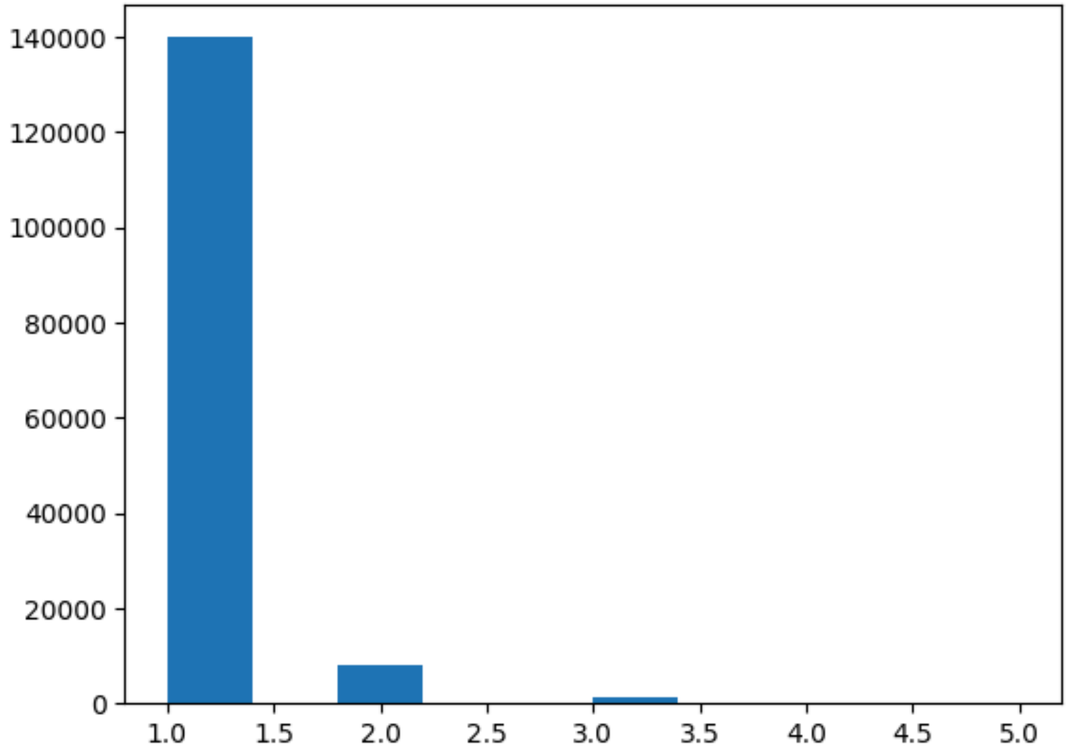
$$\text{Rating} = 1 + 4 \cdot \text{minmax}(\tfrac{X}{Y}).$$

Ratings Feature

Frequency

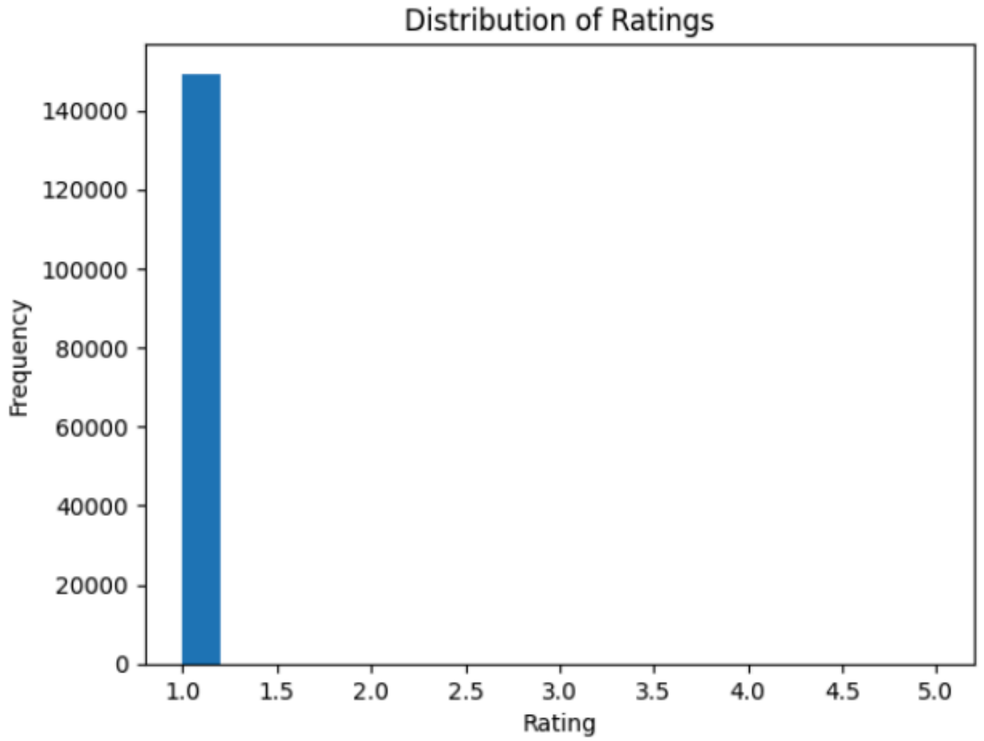
(1-5)





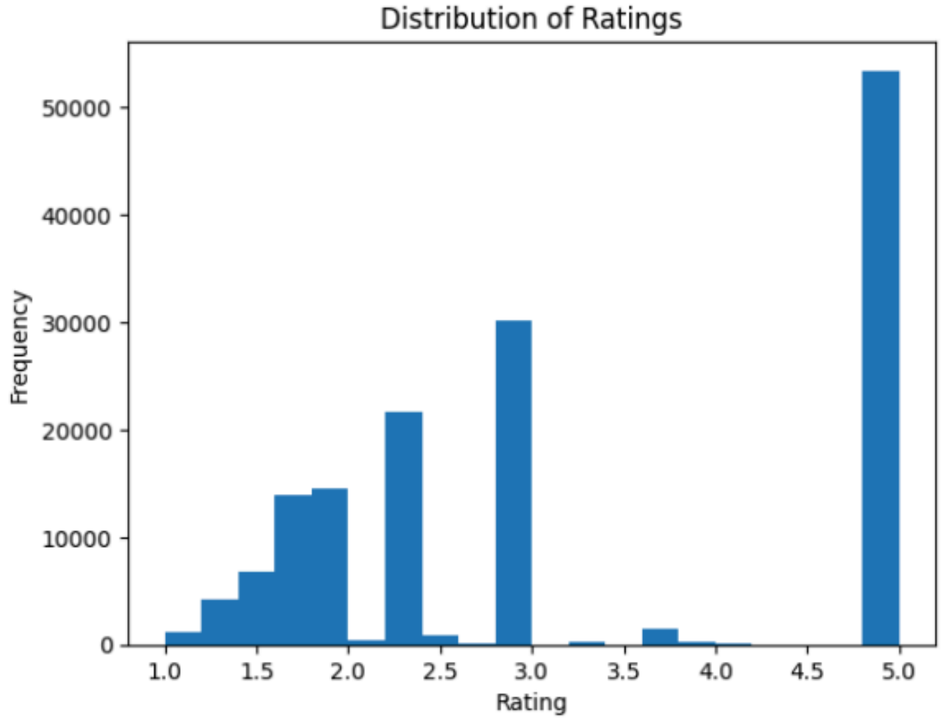
Ratings Feature

TF-IDF



$$\text{Rating} = 1 + 4 \cdot \text{minmax}(tf \cdot \log(\tfrac{N}{df})).$$

Uniform-Ratio



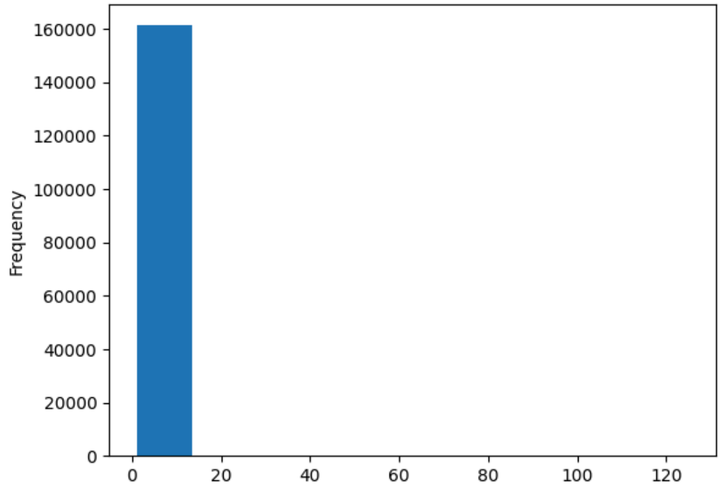
$$\text{Rating} = 1 + 4 \cdot \text{minmax}(\tfrac{X}{Y}).$$

Attempt 1

Frequency-Based

- We simply used the frequency that a customer has bought a product and used that as the rating
- Pro:
  - higher frequency means higher rating
- Cons:
  - low frequency does not always mean low rating
  - There are a lot of one-time buyers who only bought one item; this has rating of 1
  - Range of the rating is high (1-125)

	CST_ID	PRODUCT_BRAND_ID	REORDERED
0	157105.0	51	1
1	374554.0	67	1
2	374554.0	39	1
3	374554.0	51	2
4	374554.0	51	2
...	...	...	...
161686	480556.0	51	1
161687	554477.0	51	1
161688	522219.0	51	1
161689	633293.0	51	1
161690	871744.0	51	1



Attempt 1

Results

SVD

RMSE: 0.1176

ALS

RMSE: 0.5561

Interpretation

Since there are a lot of one-time buyers who only bought one product, a lot of entries are 1. This explains the small RMSE (< 1).

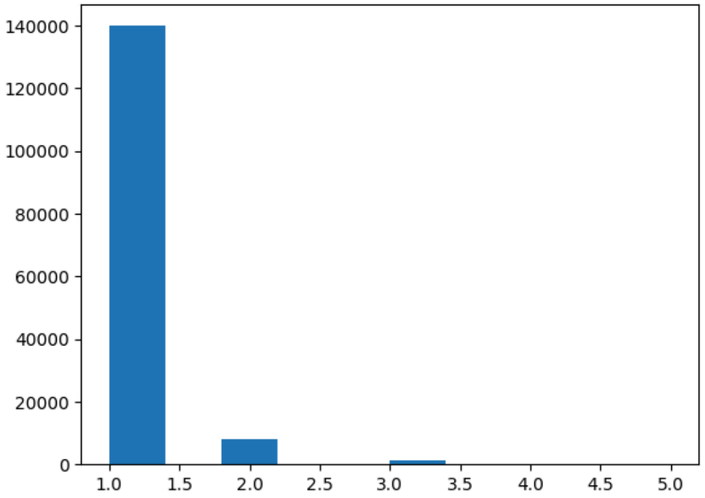
Attempt 2

1 - 5 Rating

- We still used frequency but converted everything above 5 to a rating of 5

- Pro:
  - Lower range
- Cons:
  - low frequency does not always mean low rating
  - There are a lot of one-time buyers who only bought one item; this has rating of 1

	CST_ID	PRODUCT_BRAND_ID	rating
10900354	480556.0	67	1
10900480	554477.0	51	1
10900608	522219.0	51	1
10900736	633293.0	51	1
10900864	871744.0	51	1



Attempt 2

Results

SVD

RMSE: 0.3275

ALS

RMSE: 0.4462

Interpretation

Since there are a lot of one-time buyers who only bought one product, a lot of entries are 1. This explains the small RMSE (< 1).

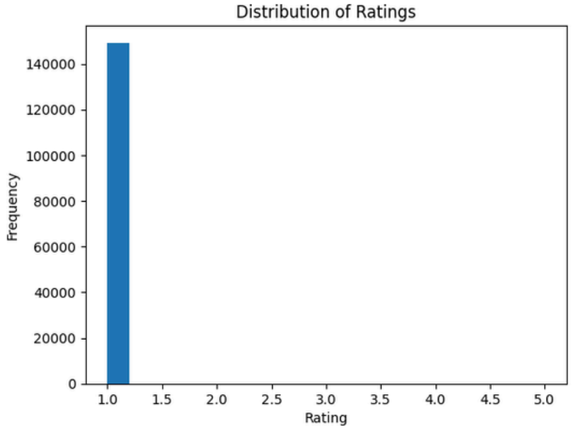
Attempt 3

TF-IDF-inspired

- We used the formula for TF-IDF and converted to 1 - 5 rating
  - DF = Total number of unique customers who have purchased the brand at least once (popularity)
  - TF = Number of times a product has been bought by a single customer
  - N = number of customers

$$\text{Rating} = 1 + 4 \cdot \text{minmax}(tf \cdot \log(\frac{N}{df})).$$

	CST_ID	PRODUCT_BRAND	DF	TF	N	RATING
0	517952.0	14	785	1	85164	1.025666
1	334094.0	14	785	1	85164	1.025666
2	973756.0	14	785	1	85164	1.025666
3	803423.0	14	785	1	85164	1.025666
4	728381.0	14	785	1	85164	1.025666



- Pro:
  - Reduces popularity bias
- Cons:
  - Lay’s has very low rating
  - Still a lot of 1 ratings

Attempt 3

Results

SVD

RMSE: 0.0185

ALS

RMSE: 0.1181

Interpretation

Since there are a lot of one-time buyers who only bought one product, a lot of entries are 1. This explains the small RMSE (< 1).

Attempt 4

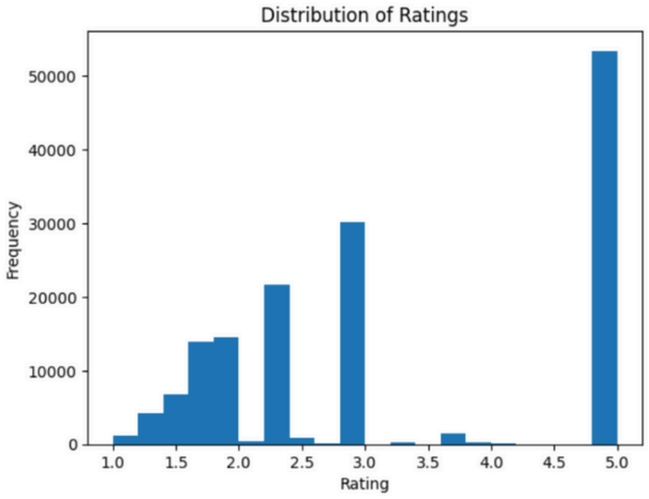
Adjusted Ratio

- Used ratio of two feature engineered variables
  - X = Number of times a product has been bought by a single customer
  - Y = Total Transactions per Customer

$$\text{Rating} = 1 + 4 \cdot \text{minmax}\left(\frac{X}{Y}\right).$$

- Pro:
  - Not very skewed to the right anymore
  - Considers customer behavior only
- Cons:
  - Lots of 5 ratings (one-time buyer with one item = 5)
  - if a customer buys 1 item that is typically bought one at a time for many days but the customer also buys lots of other items along with the 1 item, the rating will go down.

	CST_ID	PRODUCT_BRAND	TOTAL_TRANSACTIONS	PRODUCT_PURCHASE_COUNT	RATING
0	517952.0	14	6	1	1.643026
1	517952.0	68	6	2	2.314421
3	517952.0	116	6	2	2.314421
5	517952.0	50	6	1	1.643026
6	334094.0	14	4	1	1.978723



Attempt 3

Results

SVD

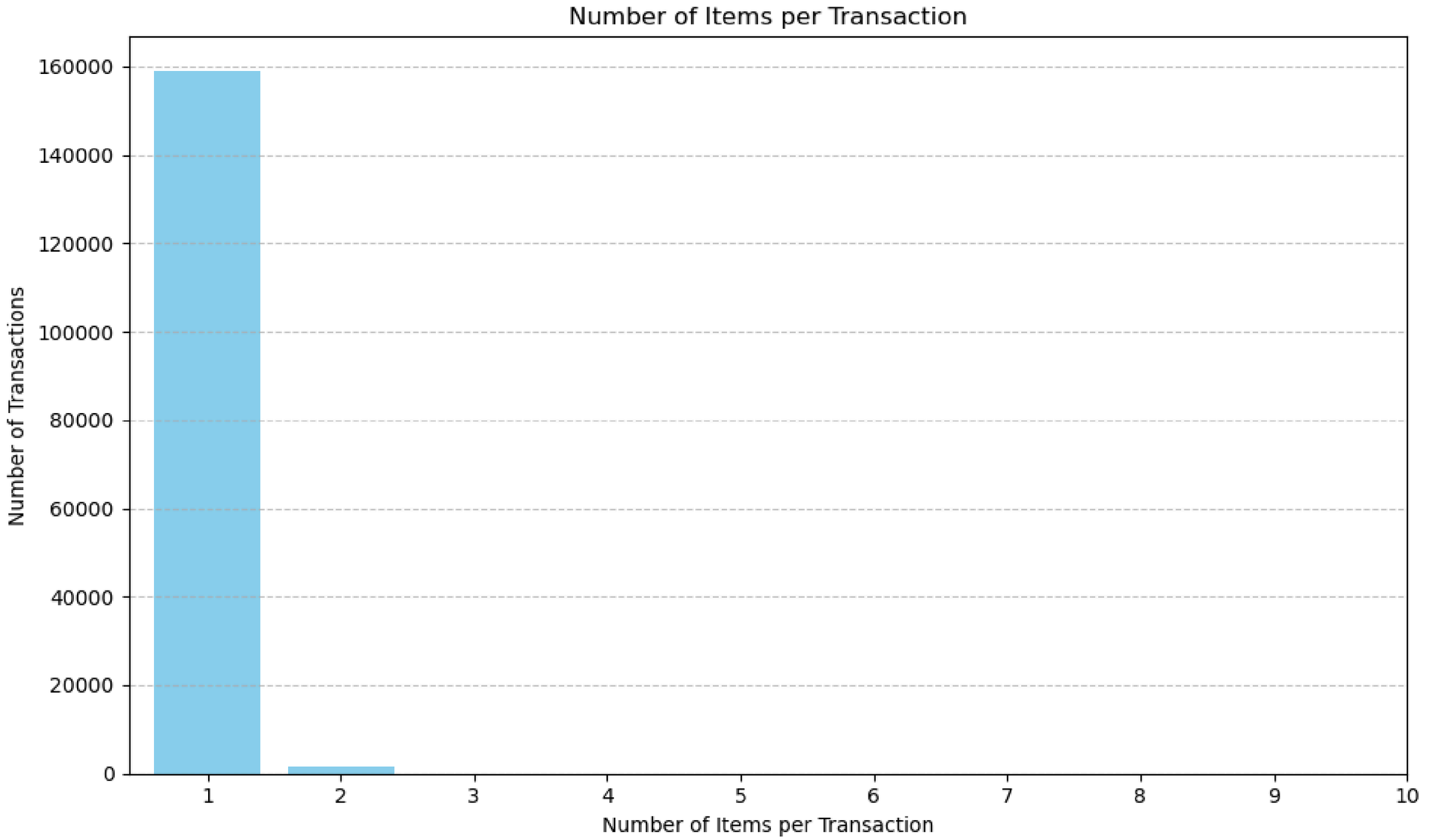
RMSE: 1.1513

ALS

RMSE: 0.4944

Interpretation

Slightly high RMSE (maybe due to the high number of 5's).


Number of Items per Transaction
Number of Transactions
160000
140000
120000
100000
80000
60000
40000
20000
0
1
2
3
4
5
6
7
8
9
10
Number of Items per Transaction

Recommender System

Sequential and Association Rule

Frequent Pattern Algorithm

- a frequent itemset mining algorithm used to discover frequently occurring patterns in large datasets
- gives the same result as apriori algorithm but runs faster for larger datasets

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

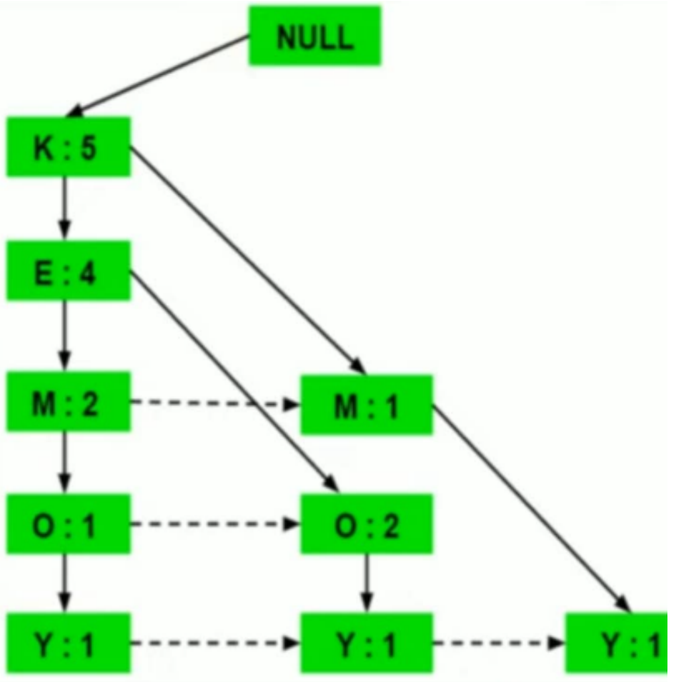
$L = \{K : 5, E : 4, M : 3, O : 3, Y : 3\}$

Frequent Pattern Set (L)

Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}

Ordered Item Set
(minimum support = 3)

Frequent Pattern Algorithm



Frequent Pattern Tree

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

Conditional Pattern Base

Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 3>}
K	

Frequent Patterns

Frequent Pattern Algorithm

Assumption

- all items bought by a customer on the same day are considered part of a single transaction (itemset), similar to items in a shopping cart
- minimum support was set to be significantly small

Data Preparation

```
[["Lay's", 'Ruffles'],
 ['Downy', 'Mr. Clean'],
 ["Lay's", 'Ruffles'],
 ['Doritos', "Lay's"],
 ["Lay's", 'Ruffles']]
```

Frequent Pattern Algorithm

Actual Data

```
[['Garden of Life'],
 ["Nature's Bounty"],
 ['Cheetos'],
 ["Lay's"],
 ['Doritos'],
 ['Monterey'],
 ['Goldfish'],
 ["Lay's"],
 ['Fritos'],
 ['New Chapter'],
 ['Monterey'],
 ['Garden of Life'],
 ['Monterey'],
 ['Monterey'],
 ['Doritos'],
 ['Nestlé'],
 ["Lay's"],
 ['Doritos'],
 ["Lay's"],
```

Result

	support	itemsets
3	0.411558	(Lay's)
4	0.099387	(Doritos)
1	0.075225	(Nature's Bounty)
0	0.054272	(Garden of Life)
7	0.041067	(Fritos)
2	0.038363	(Cheetos)
9	0.034373	(Nestlé)
6	0.029396	(Goldfish)
11	0.029040	(Del Monte)
5	0.027397	(Monterey)
8	0.026020	(New Chapter)

Frequent Pattern Algorithm

Data (Multiple Items)

```
[["Lay's", 'Ruffles'],
 ['Downy', 'Mr. Clean'],
 ["Lay's", 'Ruffles'],
 ['Doritos', "Lay's"],
 ["Lay's", 'Ruffles'],
 ["Lay's", 'Quaker Chewy Bars'],
 ['Garden of Life', 'Nordic Naturals'],
 ['Garden of Life', 'Nordic Naturals'],
 ['Cheetos', "Lay's"],
 ["Lay's", 'Pringles'],
 ['Pringles', 'Ruffles'],
 ['Fig Newtons', 'Teddy Grahams'],
 ["Lay's", 'Mr. Clean'],
 ['Downy', 'Zonrox'],
 ['Pringles', 'Ruffles'],
 ['Tide', 'Wheat Thins'],
 ["Lay's", 'Pringles'],
 ["Lay's", 'Ruffles'],
 ['Garden of Life', "Lay's"],
```

Result

	support	itemsets
65	0.132056	(Ruffles, Lay's)
118	0.125000	(Nature's Bounty, Garden of Life)
91	0.115927	(Pringles, Lay's)
116	0.054435	(New Chapter, Garden of Life)
85	0.046371	(Nordic Naturals, Garden of Life)
152	0.032258	(Fritos, Lay's)
106	0.025202	(Del Monte, Lay's)
176	0.019153	(Chips Ahoy!, Wheat Thins)
161	0.017137	(Nestlé, Lay's)
76	0.014113	(Quaker Chewy Bars, Lay's)
137	0.014113	(Garden of Life, Four Sigmatic)
73	0.013105	(Doritos, Lay's)
88	0.012097	(Cheetos, Lay's)
147	0.012097	(Nature's Bounty, Monterey)
69	0.012097	(Mr. Clean, Downy)

Frequent Pattern Algorithm

Result & Conclusions

- Highly frequent pairs:
  - Ruffles & Lay's (13.21%), Nature's Bounty & Garden of Life (12.5%), Pringles & Lay's (11.59%)
- Considering a low support threshold of 1%, there are 19 itemsets
- Notably, snack brands and health supplement brands are frequently brought together.
- Marketing Strategy: Focus on promoting the most frequent pairs together to boost sales.
- Sales: Create bundle discounts for frequently bought-together items.

Association Rule Mining

Using Apriori

Apriori Algorithm

Result & Conclusions

- The resulting itemsets were identical to those produced by FP-Growth.
- The runtime was nearly instantaneous, showing no significant difference.
- Literature suggests that FP-Growth typically has faster performance.

	Items	Support
24	Lay's	0.448589
20	Garden of Life	0.288306
33	Nature's Bounty	0.169355
46	Ruffles	0.154234
405	Lay's, Ruffles	0.132056

	support	itemsets
405	0.132056	(Ruffles, Lay's)
343	0.125000	(Nature's Bounty, Garden of Life)
397	0.115927	(Pringles, Lay's)
344	0.054435	(New Chapter, Garden of Life)
349	0.046371	(Nordic Naturals, Garden of Life)

Frequent Pattern Algorithm

Assumption

- all items bought by a customer on the same day are considered part of a single transaction (itemset), similar to items in a shopping cart

Data Preparation

```
[["Lay's", 'Ruffles'],
 ['Downy', 'Mr. Clean'],
 ["Lay's", 'Ruffles'],
 ['Doritos', "Lay's"],
 ["Lay's", 'Ruffles']]
```

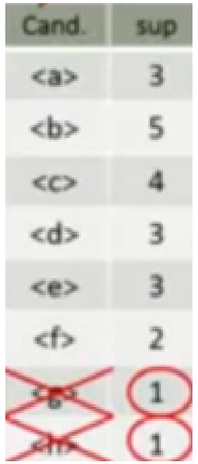
Sequential Rule Mining

Using Generalized Sequential Pattern (GSP) Mining

GSP

- a technique used to identify patterns in sequential data

SID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>



Cand.	sup
<a>	3
<b>	5
<c>	4
<d>	3
<e>	3
<f>	2
[illegible]	1
[illegible]	1

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

Sequences

Length-k Candidates

Length-k+1 Candidate Sequences

GSP

Assumption

- all items bought by a customer on the same day are considered part of a single transaction (itemset)
- a sequence is all transactions of customers arranged chronologically by date
- minimum support was set to be significantly small

Data Preparation

```
[['Garden of Life', "Nature's Bounty"],
 ['Cheetos'],
 ["Lay's"],
 ['Doritos'],
 ['Monterey'],
 ['Goldfish'],
 ["Lay's"],
```

GSP

Actual Data

```
[['Garden of Life', "Nature's Bounty"],
 ['Fritos',
  'New Chapter',
  'Monterey',
  'Garden of Life',
  'Monterey',
  'Monterey'],
 ['Nestlé', "Lay's"],
 ["Lay's", "Lay's", "Nature's Bounty", 'Garden of Life'],
 ['New Chapter', 'Garden of Life', 'Monterey'],
 ['Del Monte', "Lay's"],
 ["Nature's Bounty", 'Nestlé'],
 ["Lay's", "Nature's Bounty", 'Doritos'],
 ["Nature's Bounty", 'Garden of Life'],
 ["Lay's", 'Doritos'],
 ['Fresh Options', 'San Miguel'],
 ['Pringles', "Lay's", 'Ruffles'],
 ["Lay's", 'Triscuit'],
 ["Lay's", "Lay's"],
```

Result

Support > 0.1

- Lay’s -> Lay’s
- Nature’s Bounty -> Garden of Life

Support > 0.01

- Lay’s -> Nature’s Bounty
- Lay’s -> Doritos
- Lay’s -> Nestle
- Lay’s -> Nature’s Bounty -> Garden of Life
- Lay’s -> Ruffles

Frequent Pattern Algorithm

Result & Conclusions

- Highly frequent pairs:
  - Lay’s to Lay's (3859), Nature's Bounty to Garden of Life (2696)
- Considering a low support threshold of 1%, there are 39 sequences
- A notable sequence is from Lay’s to another snack
  - Lay's chips show strong brand loyalty with repeat purchases and frequent sequential buying with other snack brands like Doritos, Pringles, and Ruffles.
- Health brands are frequently bought in sequence
- Marketing: promote items frequently bought in sequence
- Note: we also tried SPADE (Sequential Pattern Discovery using Equivalence classes) but we were not able to run it

Thank you!

Thank you so mu-