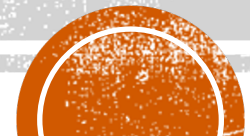
# DEVELOPING A COLLABORATIVE FILTERING BASED RECOMMENDER SYSTEM

For Amazon Kindle Store Data

By Krish Weragalaarachchi

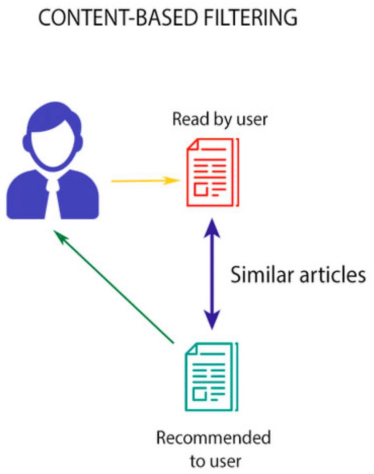


amazon

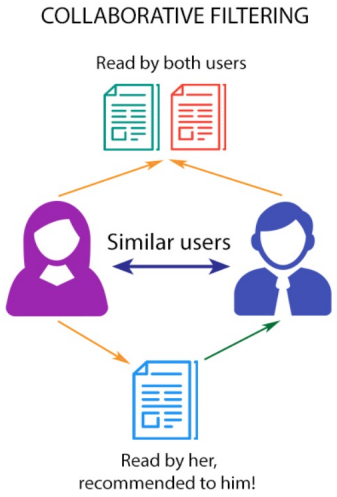
# INTRODUCTION

There are two common recommendation filtering techniques:

- Content Based filtering - needs the profile of both the users and the items so that the system can determine the recommendation according to properties of users and items




- Collaboration Filtering - requires the model to learn the connections between users so that it can generate the best recommendation options based on users' past choices



# INTRODUCTION

- Given the ratings of items, list of products will be recommended to a customer based on the purchase history of other customers on the website.
- Choose the best ML model.
- Recommendations from the ML model is ordered based on average sentiment score of corresponding reviews for the product and, 5 products with the highest score is filtered as the final recommendation to the customer.

Customers

→

Recommender System Based on Ratings

→

Average Sentiment Score of Reviews

→

# RELATED WORK

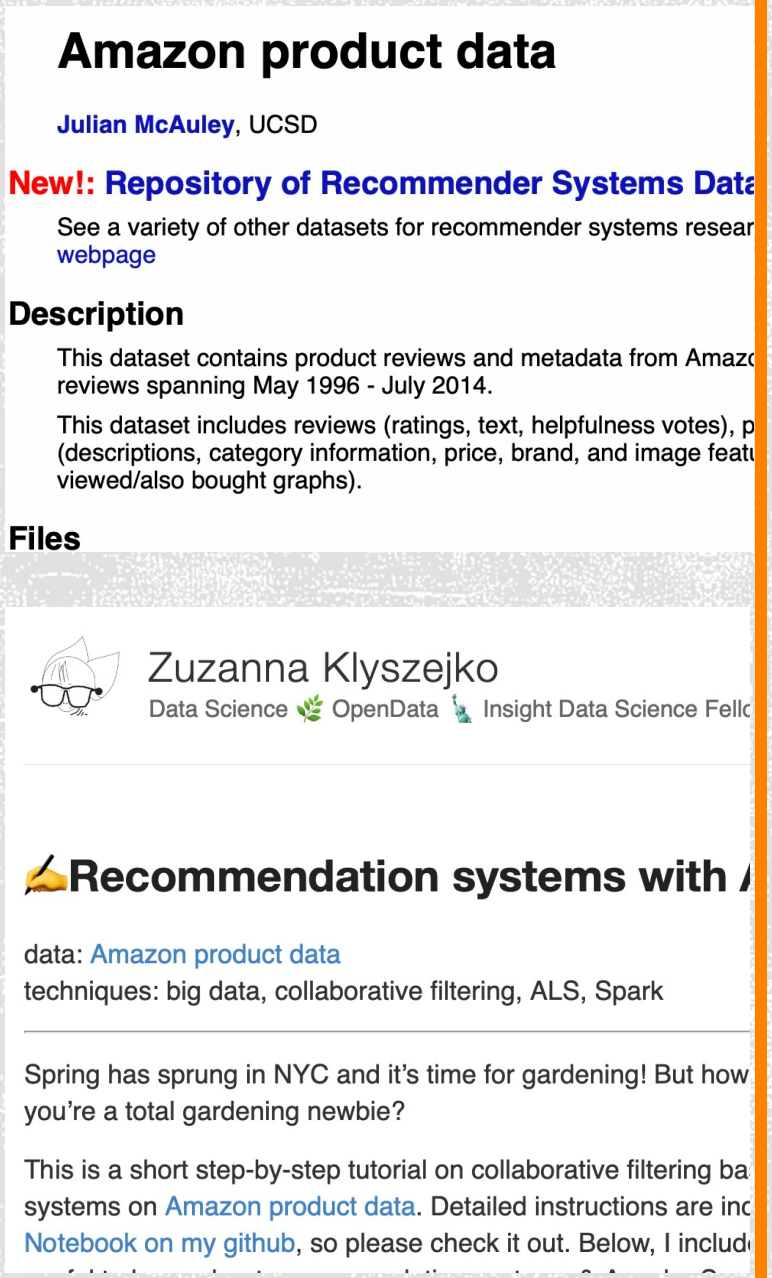
- There are so many recommendation systems built by many individuals using Amazon Data.

- Could not find anyone using the 2014 Amazon Kindle Store Data set.

2014 Kindle Store Data - Julian McAuley, UCSD
http://jmcauley.ucsd.edu/data/amazon/links.html

- Got insights from Zuzanna Klyzejko blog post featuring "**Recommendation systems with Apache Spark**" which she uses "amazon Patio, lawn and garden data set"

(Klyzejko, Zuzanna) 2017
https://zuzannna.github.io/Recommendation-System-with-Spark/



# DATA

- 2014 Amazon Kindle store product dataset -Julian McAuley lab at UCSD

  2014 Kindle Store Data - Julian McAuley, UCSD
  http://jmcauley.ucsd.edu/data/amazon/links.html

- 982,619 entries (827 MB)
- There are information on reviewer ID, Product ID, reviewer's name, whether the review helpful or not, review, overall rating, summary and Review time.

```
df_raw.show(5)
```

```
+----------+-------+-------+--------------------+----------+--------------+--------------------+-----------------+--------------+
|      asin|helpful|overall|          reviewText|reviewTime|    reviewerID|        reviewerName|          summary|unixReviewTime|
+----------+-------+-------+--------------------+----------+--------------+--------------------+-----------------+--------------+
|B000F83SZQ| [0, 0]|    5.0|I enjoy vintage b...| 05 5, 2014|A1F6404F1VG29J|           Avidreader|Nice vintage story|    1399248000|
|B000F83SZQ| [2, 2]|    4.0|This book is a re...| 01 6, 2014| AN0N05A9LIJEQ|             critters|      Different...|    1388966400|
|B000F83SZQ| [2, 2]|    4.0|This was a fairly...| 04 4, 2014| A795DMNCJILA6|                 dot|           Oldie|    1396569600|
|B000F83SZQ| [1, 1]|    5.0|I'd never read an...|02 19, 2014|A1FV0SX13TWVXQ|Elaine H. Turley ...|I really liked it.|    1392768000|
|B000F83SZQ| [0, 1]|    4.0|If you like perio...|03 19, 2014|A3SPTOKDG7WBLN|  Father Dowling Fan|    Period Mystery|    1395187200|
+----------+-------+-------+--------------------+----------+--------------+--------------------+-----------------+--------------+
only showing top 5 rows
```

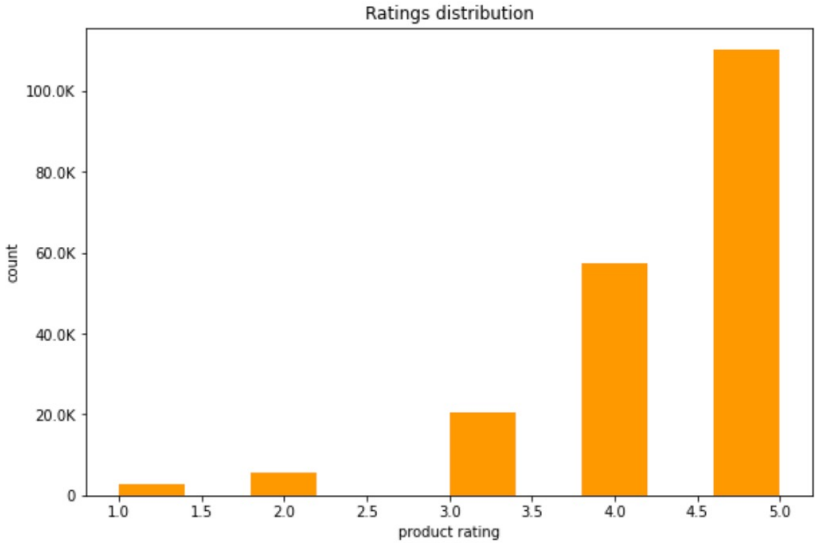
# DATA

- Data Cleaning
  - Check for non, missing or null values
  - Make less sparse customer - product matrix

    Sparsity decreased from 99.98% to 99.67% by splicing the data set so that it includes customers who gave more than 20 ratings.
  - Data summary

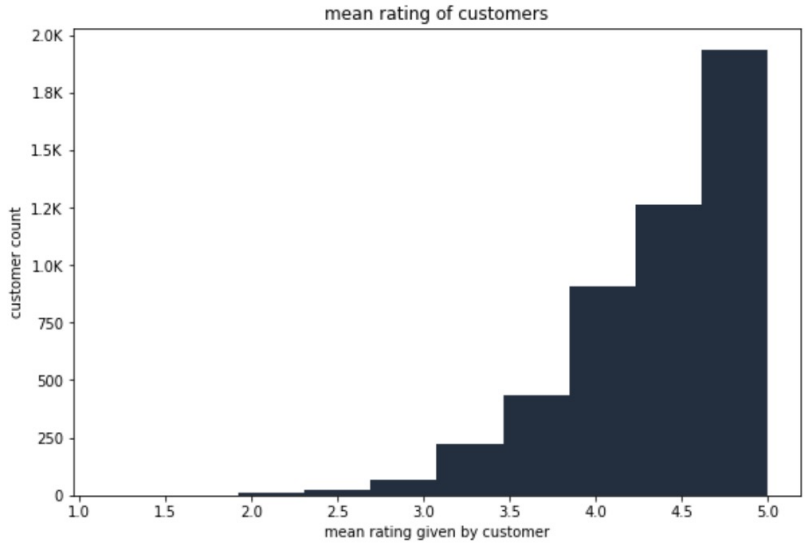
| | |
|---|---|
| Number of unique products | 12291 |
| Number of unique users | 4867 |
| Number of different rating | 5 |
| Number of ratings | 196190 |
| Number of reviews | 196190 |

## Rating distribution of Customers




| Rating | Count |
|---|---|
| 1 | 2631 |
| 2 | 5591 |
| 3 | 20477 |
| 4 | 57308 |
| 5 | 110183 |

Histogram showing customer count for mean rating of customer




** Rating distribution is highly skew to the right close to rating 5.0

Mean rating = 4.36

Median rating = 5.0

# METHODOLOGIES

- Selected the most basic recommendation (Baseline model ) as recommendations based on median or mean.
- Then tested following recommendation systems
  - Popularity based recommendation System
  - Collaborative recommendation System
    - Alternative Least Square Algorithm
    - K-Nearest Neighbor (KNNWithMeans, KNNBasics )
    - Singular Value Decomposition
- Incorporated sensitivity score of reviews form VADER to filtered recommendation to get the best (final) recommendation.

# BASELINE

The most basic recommendation is based on median value.

**The standard deviation of the residuals (prediction errors)**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values

$y_1, y_2, \ldots, y_n$ are observed values

$n$ is the number of observations

RMSE of baseline based on median = 1.087

RMSE of baseline based on mean = 0.878 **

Median rating = 5.0

Mean rating = 4.36

# POPULARITY BASED RECOMMENDATIONS

Works on the principle of popularity and or anything which is in trend. These systems check about the product or movie which are in trend or are most popular among the users and directly recommend those.

Steps :

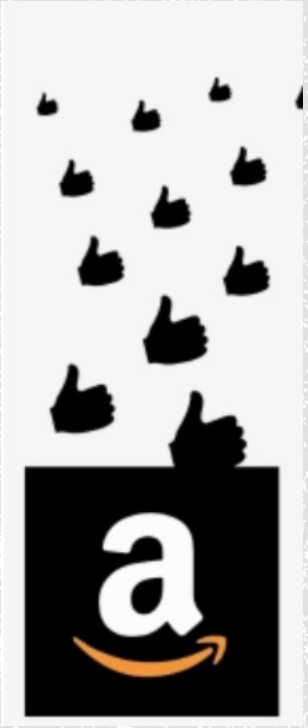
- Training testing split (80, 20)
- Calculate average rating for each product in the training set
- Merge average rating calculated to testing data (ignored nulls)

| product_ID | customer_ID | rating | review | product_Index | customer_Index | avg(rating) |
|---|---|---|---|---|---|---|
| B004OEIRNA | AUF0OPKICU6D4 | 4.0 | I rather enjoyed ... | 603 | 254 | 3.7435897435897436 |
| B004OEIRNA | A26F0QHBB1NW3H | 5.0 | So it was a quick... | 603 | 3967 | 3.7435897435897436 |
| B004OEIRNA | AV0DTTLGNW7C1 | 4.0 | I was very pleasa... | 603 | 545 | 3.7435897435897436 |
| B004OEIRNA | A3GFQDC7240EZR | 5.0 | I really enjoyed ... | 603 | 132 | 3.7435897435897436 |
| B004OEIRNA | A5NZK7PR5RCBX | 2.0 | Can't say I under... | 603 | 4436 | 3.7435897435897436 |
| B005C5TCR0 | A35POUEGQ4PGUK | 5.0 | This little gem t... | 8287 | 860 | 3.8333333333333335 |
| B005C5TCR0 | A3Q86MZSVUUAD8 | 4.0 | this was a well w... | 8287 | 194 | 3.8333333333333335 |

- Calculate RMSE (0.852)

RMSE of Popularity based model = 0.852

RMSE of baseline based on mean = 0.878



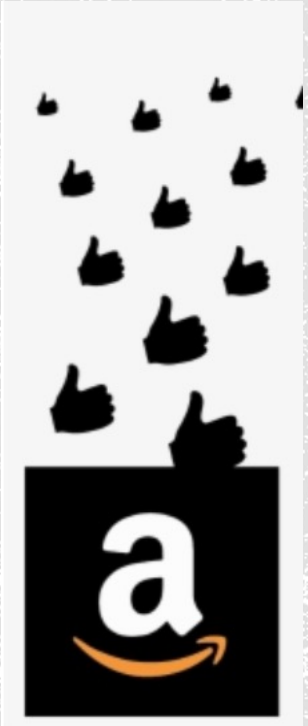
# POPULARITY BASED RECOMMENDATIONS

10 best recommendations to any customer based on popularity based recommendation system

| product_ID | avg(rating) |
|---|---|
| B00B7PBN4E | 5.0 |
| B0094P40JY | 5.0 |
| B007O3IKTY | 5.0 |
| B008MYSMN8 | 5.0 |
| B00EBVD7EU | 5.0 |
| B00IHBQVSY | 5.0 |
| B00AHVNEPK | 5.0 |
| B001BUPF62 | 5.0 |
| B00CDU1H98 | 5.0 |
| B008IEJTSE | 5.0 |

RMSE of Popularity based model = 0.852

RMSE of baseline based on mean = 0.878



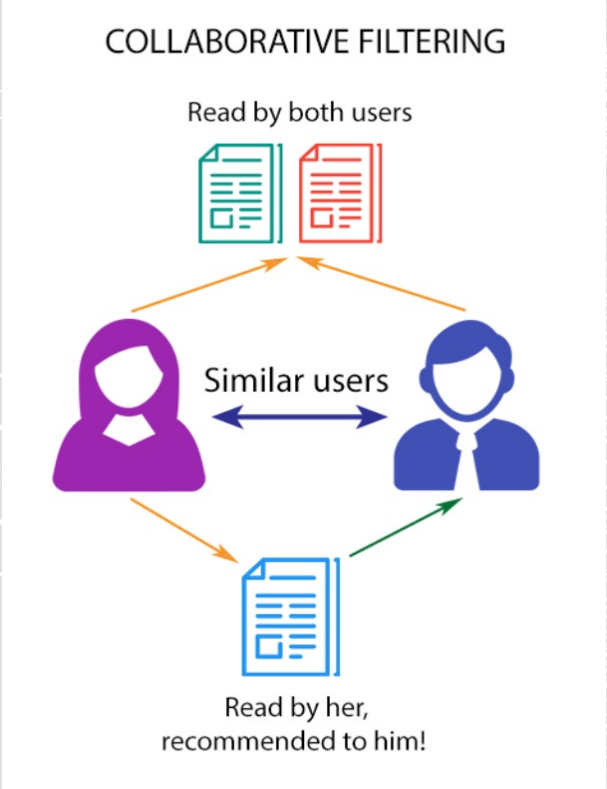
# COLLABORATIVE RECOMMENDATION SYSTEMS

Two types of collaborative filtering techniques

- User-User collaborative filtering
- Item-Item collaborative filtering

Let's say you know a friend who has the same taste as you because you both love Science, then you might like reading other books that your friend has read but you haven't.

This is the sole concept behind the collaborative filtering.



# ALTERNATIVE LEAST SQUARE (ALS) MODEL

```
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.sql import Row


als = ALS(userCol="customer_Index", itemCol="product_Index", ratingCol="rating",
          coldStartStrategy="drop",nonnegative = True, implicitPrefs = False)

from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
als_param_grid = ParamGridBuilder() \
 .addGrid(als.rank, [5, 10]) \
 .addGrid(als.maxIter, [5, 10]) \
 .addGrid(als.regParam, [.01, 0.1]) \
 .build()
# Define evaluator as RMSE
als_evaluator = RegressionEvaluator(metricName = 'rmse', labelCol = 'rating', predictionCol = 'prediction')
# Print length of evaluator
print ('Num models to be tested using param_grid: ', len(als_param_grid))
```

```
Num models to be tested using param_grid:  8
```

```
# Build cross validation using CrossValidator
als_cv = CrossValidator(estimator = als, estimatorParamMaps = als_param_grid, evaluator = als_evaluator, numFolds = 5)
als_model = als_cv.fit(training)
als_predictions = als_model.transform(testing)
als_predictions.show(n = 10)
```

RMSE of Alternative Least Square model = 0.787

RMSE of Popularity based model = 0.852

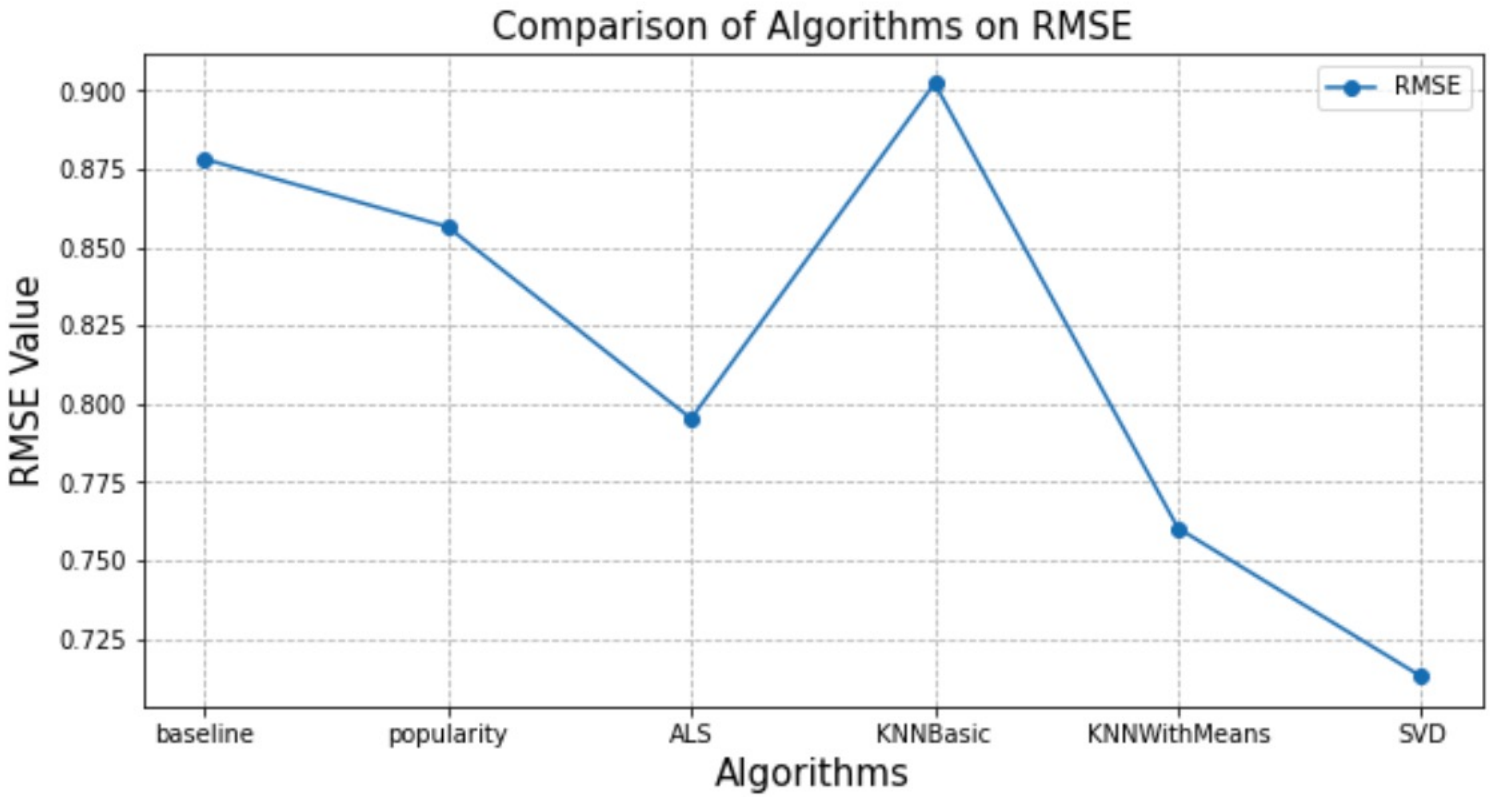
RMSE of baseline based on mean = 0.878

# K-NEAREST NEIGHBOR MODEL AND SINGULAR VALUE DECOMPOSITION MODELS (using python SURPRISE library)

| Algorithm | Parameters tested | Best model |
|---|---|---|
| K-Nearest Neighbor - considering user mean value (KNNwithMeans) | name : msd,cosine,pearson_baseline | name : msd |
| | min_support : 3, 4, 5 | min_support : 3 |
| | user_based : True | user_based : True |
| | k : 5 ,10 | k : 5 |
| | Verbose : True ,Fasle | Verbose : True |
| K-Nearest Neighbor (KNNBasics) | name : msd,cosine,pearson_baseline | name : pearson_baseline |
| | min_support : 3, 4, 5 | min_support : 5 |
| | user_based : True | user_based : True |
| | k : 5 ,10 | k : 5 |
| | Verbose : True ,Fasle | Verbose : True |
| Singular Value Decomposition (SVD) | n_epochs= 20, 25 | n_epochs= 25 |
| | lr_all=0.007, 0.009, 0.01 | lr_all=0.01 |
| | reg_all= 0.4 , 0.6 | reg_all= 0.4 |

# MODEL EVALUATION

Based on RMSE




| Model | RMSE |
|---|---|
| SVD | 0.713 |
| KNNWithMeans | 0.760 |
| ALS | 0.787 |
| Popularity based | 0.852 |
| Mean based | 0.878 |
| KNNBasic | 0.902 |

SVD model with parameters : n_epochs= 25, lr_all=0.01, reg_all= 0.4

Customers

Recommender System Based on Ratings
SVD algorithm

```
'ANS3TWQTZLXGN': [('B00FJ3CMNG', 4.694206657711124),
 ('B00E7IWEFU', 4.676140739691464),
 ('B008NBYHPC', 4.584243926195197),
 ('B00AOT3UDG', 4.582389131380704),
 ('B00JD03U1O', 4.581642240414649),
 ('B00BG1VMK4', 4.502174100271493),
 ('B007JCTY40', 4.501705814899906),
 ('B008CUKK7S', 4.458945427748505),
 ('B00B9KUZK0', 4.443662437842071),
 ('B00AKSL3TO', 4.09691646652649)],
```

Given a customer id, top 10 items are recommended

# TEXT PROCESSING AND SENTIMENT ANALYSIS OF REVIEWS

# TEXT PROCESSING

```
+----------------------------------------------------------------------------------------------------
|review
+----------------------------------------------------------------------------------------------------
|I enjoyed the two main characters: kick-ass Lilli and hunky smart motorcycle gang leader Isaac.There&#8217;s
|SUICIDE RUN by Michael Connelly.I was tempted to give this 5 stars for the last story.  Harry is up against a
|It's about Reacher when he was 13 years old in 1974.  His brother Joe was 15.  They had just moved with their
|INSUFFERABLE PROXIMITY by Z. Stefani.The best part is probably the numerous sex scenes where Julian gropes he
|BAY'S MERCENARY by C.L. Scholey.Imagine yourself in an alien world.  They think you are a pet like a puppy, g
+----------------------------------------------------------------------------------------------------
```

Reviews were cleaned by

- removing non - ASCII characters
- fixing abbreviations
- removing hyperlinks
- removing mentions
- removing numerals
- removing non alphanumeric characters and words less than 1 in length

```
+--------------------------------------------------------------------
|review
+--------------------------------------------------------------------
|enjoyed the two main lilli and hunky sma motorcycle gang leader lot of s
|suicide run by michael connelly.i was tempted to give this stars for the
|about reacher when he was years old in his brother joe was they had just
|insufferable proximity by z. stefani.the best pa is probably the numerou
|mercenary by c.l. scholey.imagine yourself in an alien world. they think
+--------------------------------------------------------------------
```

# SENTIMENT OF REVIEWS

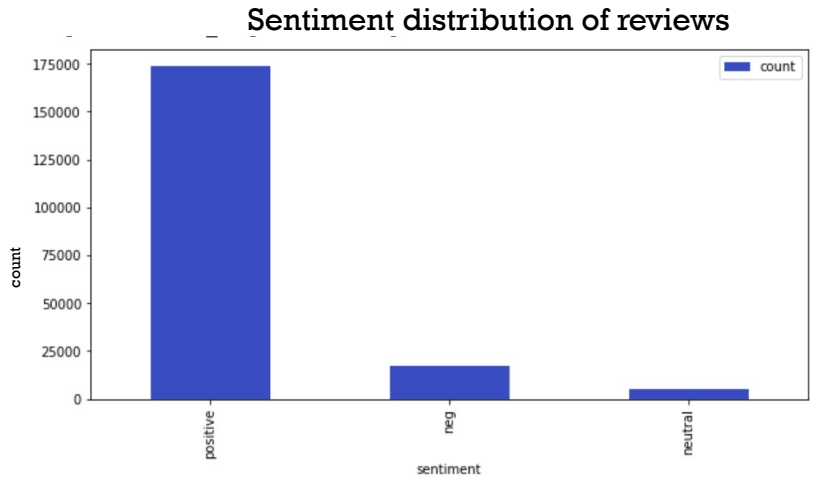
- Calculated the compound score for each review. (VADER : Valence Aware Dictionary and Sentiment Reasoner)
  - Stemming , lemmatization , stop word removal is not needed.

```
dtext = dtext.withColumn("sentiment_score", sentiment_analysis_udf(dtext['review'] ))
dtext.show(5,True)
```

```
+----------+--------------+------+--------------------+-------------+--------------+---------------+
|product_ID|   customer_ID|rating|              review|product_Index|customer_Index|sentiment_score|
+----------+--------------+------+--------------------+-------------+--------------+---------------+
|B00G8U2O0U|A1CNQTCRQ35IMM|   3.0|enjoyed the two m...|          522|           771|         0.6342|
|B0055PMJMW|A1CNQTCRQ35IMM|   4.0|suicide run by mi...|        10675|           771|         0.5122|
|B005D75Z8C|A1CNQTCRQ35IMM|   4.0|about reacher whe...|         3364|           771|        -0.5859|
|B007FIIF34|A1CNQTCRQ35IMM|   2.0|insufferable prox...|         5346|           771|        -0.7478|
|B00D4KG0DE|A1CNQTCRQ35IMM|   3.0|mercenary by c.l....|         1640|           771|          0.952|
+----------+--------------+------+--------------------+-------------+--------------+---------------+
only showing top 5 rows
```

- Sentiment Distribution

```
# sentiment distribution
dtext1=dtext.withColumn("sentiment",F.when(F.col('sentiment_score')>=0.1,'positive')\
                        .otherwise(F.when(F.col("sentiment_score")<=-0.1,'neg')\
                        .otherwise('neutral')))
```



## Average sentiment score of reviews for each product

```
dtext_ordred = dtext.groupBy("product_ID").mean('sentiment_score').sort(col('avg(sentiment_score)').desc())
dtext_ordred.show()
```

```
+----------+--------------------+
|product_ID|avg(sentiment_score)|
+----------+--------------------+
|B00DQZRQPI|  0.9980999827384949|
|B009JUZQVK|  0.9976000189781189|
|B00F63NG98|   0.996999979019165|
|B00BMKIWTS|  0.9968727285211737|
|B00CLG0TES|  0.9964312575757504|
|B00KFHBPKM|  0.9962200105190278|
|B008LQ9CIG|  0.9961000084877014|
|B009YLRBO4|  0.9958999752998352|
|B00FXCXGGA|   0.995681806044145|
|B00FDVBRKI|     0.9953666528066|
|B00JF9HC6W|   0.995199978351593|
|B004TQN0NU|  0.9950000047683716|
|B00ANV9EM6|  0.9947999715805054|
|B00BNZ6XV6|  0.9943647069089553|
|B00G4FCHH4|  0.9943000078201294|
|B00HTJH9J0|  0.9941000125624917|
|B004CJ812Y|  0.9936200022697449|
|B00D9IDSJU|  0.9934999942779541|
|B00I6M9EXS|  0.9930999875068665|
|B007HQ272E|   0.992900013923645|
+----------+--------------------+
only showing top 20 rows
```

# FINAL RESULT

Customers

Recommender System Based on Ratings SVD algorithm

Average Sentiment Score of Reviews


"Reading Renee"
A1ZCEJEA67P6DE

```
[('B005G7SUL8', 4.673435540415803),
 ('B00FNV2D9W', 4.520599007697548),
 ('B00IQYYYJ0', 4.482646796230664),
 ('B00K3IQCQK', 4.477839986997124),
 ('B00C6SI9UW', 4.465020262168067),
 ('B00HG15RHM', 4.457092005804411),
 ('B0060PEOR4', 4.425474214944406),
 ('B004LLIGNM', 4.117951761105915),
 ('B00E55HXLA', 4.110212391400843),
 ('B00BP6Q8M2', 3.879062743437701)]
```

```
+----------+--------------------+
|product_ID|avg(sentiment_score)|
+----------+--------------------+
|B00DQZRQPI|   0.9980999827384949|
|B009JUZQVK|   0.9976000189781189|
|B00F63NG98|    0.996999979019165|
|B00BMKIWTS|   0.9968727285211737|
|B00CLG0TES|   0.9964312575757504|
|B00KFHBPKM|   0.9962200105190278|
|B008LQ9CIG|   0.9961000084877014|
|B009YLRBO4|   0.9958999752998352|
|B00FXCXGGA|    0.995681806044145|
|B00FDVBRKI|      0.9953666528066|
|B00JF9HC6W|    0.995199978351593|
|B004TQN0NU|   0.9950000047683716|
|B00ANV9EM6|   0.9947999715805054|
|B00BNZ6XV6|   0.9943647069089553|
|B00G4FCHH4|   0.9943000078201294|
|B00HTJH9J0|   0.9941000125624917|
|B004CJ812Y|   0.9936200022697449|
|B00D9IDSJU|   0.9934999942779541|
|B00I6M9EXS|   0.9930999875068665|
|B007HQ272E|    0.992900013923645|
+----------+--------------------+
only showing top 20 rows
```

| product_ID | rating | avg(sentiment_score) |
|---|---|---|
| **B00IQYYYJ0** | 4.482238 | 0.914100 |
| **B00C6SI9UW** | 4.464350 | 0.871775 |
| **B004LLIGNM** | 4.118432 | 0.869914 |
| **B00E55HXLA** | 4.123048 | 0.833474 |
| **B00K3IQCQK** | 4.473289 | 0.814583 |

# CHALLENGES...

- Little time for brainstorming and reading
- Could not test Parameters in a wide range as it takes longer to finish in colab.
- Limited resources capabilities.
- Limited knowledge on using AWS and MongoDB.
- Fail to convert surprise output to pyspark dataframe due to numpy "type error " and had to use some python for coding.

# FUTURE WORK...

- Some more algorithms should be tested as RMSE is not in acceptable range (0.2 - 0.5) for a good prediction model

- Parameter search should be expanded .

# THANK YOU