**linkcode**

**Product Recommendation System for e-commerce businesses¶**

A well developed recommendation system will help businesses improve their shopper's experience on website and result in better customer acquisition and retention.

The recommendation system, I have designed below is based on the journey of a new customer from the time he/she lands on the business’s website for the first time to when he/she makes repeat purchases.

The recommendation system is designed in 3 parts based on the business context:

* **Recommendation system part I:** Product pupularity based system targetted at new customers
* **Recommendation system part II:** Model-based collaborative filtering system based on customer's purchase history and ratings provided by other users who bought items similar items
* **Recommendation system part III:** When a business is setting up its e-commerce website for the first time withou any product rating

When a new customer without any previous purchase history visits the e-commerce website for the first time, he/she is recommended the most popular products sold on the company's website. Once, he/she makes a purchase, the recommendation system updates and recommends other products based on the purchase history and ratings provided by other users on the website. The latter part is done using collaborative filtering techniques.

**linkcode**

**Recommendation System - Part I¶**

**Product popularity based recommendation system targeted at new customers¶**

* Popularity based are a great strategy to target the new customers with the most popular products sold on a business's website and is very useful to cold start a recommendation engine.
* **Dataset :** [Amazon product review dataset](https://www.kaggle.com/skillsmuggler/amazon-ratings)

**linkcode**

Importing libraries¶

**linkcode**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

*# %matplotlib inline*

plt.style.use("ggplot")

import sklearn

from sklearn.decomposition import TruncatedSVD

**linkcode**

Loading the dataset¶

**linkcode**

amazon\_ratings = pd.read\_csv('../input/amazon-ratings/ratings\_Beauty.csv')

amazon\_ratings = amazon\_ratings.dropna()

amazon\_ratings.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | UserId | ProductId | Rating | Timestamp |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | A39HTATAQ9V7YF | 0205616461 | 5.0 | 1369699200 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | A3JM6GV9MNOF9X | 0558925278 | 3.0 | 1355443200 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | A1Z513UWSAAO0F | 0558925278 | 5.0 | 1404691200 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | A1WMRR494NWEWV | 0733001998 | 4.0 | 1382572800 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | A3IAAVS479H7M7 | 0737104473 | 1.0 | 1274227200 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

linkcode

amazon\_ratings.shape

(2023070, 4)

**linkcode**

popular\_products = pd.DataFrame(amazon\_ratings.groupby('ProductId')['Rating'].count())

most\_popular = popular\_products.sort\_values('Rating', ascending=False)

most\_popular.head(10)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Rating |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ProductId |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B001MA0QY2 | 7533 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B0009V1YR8 | 2869 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B0043OYFKU | 2477 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B0000YUXI0 | 2143 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B003V265QW | 2088 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B000ZMBSPE | 2041 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B003BQ6QXK | 1918 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B004OHQR1Q | 1885 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B00121UVU0 | 1838 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B000FS05VG | 1589 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

linkcode

most\_popular.head(30).plot(kind = "bar")

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fd439e493c8>

**linkcode**

**Analysis:**

* The above graph gives us the most popular products (arranged in descending order) sold by the business.
* For eaxmple, product, ID # B001MA0QY2 has sales of over 7000, the next most popular product, ID # B0009V1YR8 has sales of 3000, etc.

**linkcode**

**Recommendation System - Part II¶**

**Model-based collaborative filtering system¶**

* Recommend items to users based on purchase history and similarity of ratings provided by other users who bought items to that of a particular customer.
* A model based collaborative filtering technique is closen here as it helps in making predictinfg products for a particular user by identifying patterns based on preferences from multiple user data.

**linkcode**

Utility Matrix based on products sold and user reviews¶

**Utility Matrix :** An utlity matrix is consists of all possible user-item preferences (ratings) details represented as a matrix. The utility matrix is sparce as none of the users would buy all teh items in the list, hence, most of the values are unknown.

**linkcode**

*# Subset of Amazon Ratings*

amazon\_ratings1 = amazon\_ratings.head(10000)

**linkcode**

ratings\_utility\_matrix = amazon\_ratings1.pivot\_table(values='Rating', index='UserId', columns='ProductId', fill\_value=0)

ratings\_utility\_matrix.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ProductId | 0205616461 | 0558925278 | 0733001998 | 0737104473 | 0762451459 | 1304139212 | 1304139220 | 130414089X | 130414643X | 1304146537 | ... | B000052YPE | B000052YPF | B000052YPG | B000052YPH | B000052YPM | B000052YPU | B000052YPV | B000052YPY | B000052YQ0 | B000052YQ2 |
| UserId |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A00205921JHJK5X9LNP42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A024581134CV80ZBLIZTZ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A03056581JJIOL5FSKJY7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A03099101ZRK4K607JVHH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A0505229A7NSH3FRXRR4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 886 columns

**linkcode**

As expected, the utility matrix obtaned above is sparce, I have filled up the unknown values wth 0.

**linkcode**

ratings\_utility\_matrix.shape

(9697, 886)

**linkcode**

Transposing the matrix

**linkcode**

X = ratings\_utility\_matrix.T

X.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| UserId | A00205921JHJK5X9LNP42 | A024581134CV80ZBLIZTZ | A03056581JJIOL5FSKJY7 | A03099101ZRK4K607JVHH | A0505229A7NSH3FRXRR4 | A05492663T95KW63BR75K | A059547920Q3LZVFHLPI3 | A07410232KYRFR25CIUGJ | A082796624UNM47DSAI6K | A0864963DOAY7LXGS5I6 | ... | AZW1HXXYAC15B | AZWRTJPN7NXT | AZWTXHXZXFAYP | AZYQEFB9Y5N22 | AZZHB6U54UDYW | AZZHJZP4GQPPZ | AZZNK89PXD006 | AZZOFVMQC0BJG | AZZQXL8VDCFTV | AZZTJQ7CQZUD8 |
| ProductId |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0205616461 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0558925278 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0733001998 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0737104473 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0762451459 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 9697 columns

**linkcode**

X.shape

(886, 9697)

**linkcode**

Unique products in subset of data

**linkcode**

X1 = X

**linkcode**

**Decomposing the Matrix¶**

**linkcode**

SVD = TruncatedSVD(n\_components=10)

decomposed\_matrix = SVD.fit\_transform(X)

decomposed\_matrix.shape

(886, 10)

**linkcode**

**Correlation Matrix¶**

**linkcode**

correlation\_matrix = np.corrcoef(decomposed\_matrix)

correlation\_matrix.shape

(886, 886)

**linkcode**

correlation\_matrix

**linkcode**

**Isolating Product ID # 6117036094 from the Correlation Matrix¶**

Assuming the customer buys Product ID # 6117036094 (randomly chosen)

**linkcode**

X.index[99]

'6117036094'

**linkcode**

Index # of product ID purchased by customer

**linkcode**

i = "6117036094"

product\_names = list(X.index)

product\_ID = product\_names.index(i)

product\_ID

99

**linkcode**

Correlation for all items with the item purchased by this customer based on items rated by other customers people who bought the same product

**linkcode**

correlation\_product\_ID = correlation\_matrix[product\_ID]

correlation\_product\_ID.shape

(886,)

**linkcode**

**Recommending top 10 highly correlated products in sequence¶**

**linkcode**

Recommend = list(X.index[correlation\_product\_ID > 0.90])

*# Removes the item already bought by the customer*

Recommend.remove(i)

Recommend[0:9]

['0733001998',

'1304139212',

'1304139220',

'130414089X',

'130414643X',

'130414674X',

'1304174778',

'1304174867',

'1304174905']

**linkcode**

**Product Id #** Here are the top 10 products to be displayed by the recommendation system to the above customer based on the purchase history of other customers in the website.

**linkcode**

**Recommendation System - Part III¶**

* For a business without any user-item purchase history, a search engine based recommendation system can be designed for users. The product recommendations can be based on textual clustering analysis given in product description.
* **Dataset :** [Home Depot's dataset with product dataset.](https://www.kaggle.com/c/home-depot-product-search-relevance/data)

**linkcode**

*# Importing libraries*

from sklearn.feature\_extraction.text import TfidfVectorizer, CountVectorizer

from sklearn.neighbors import NearestNeighbors

from sklearn.cluster import KMeans

from sklearn.metrics import adjusted\_rand\_score

**linkcode**

**Item to item based recommendation system based on product description¶**

Applicable when business is setting up its E-commerce website for the first time

**linkcode**

product\_descriptions = pd.read\_csv('../input/home-depot-product-search-relevance/product\_descriptions.csv')

product\_descriptions.shape

(124428, 2)

**linkcode**

Checking for missing values¶

**linkcode**

*# Missing values*

product\_descriptions = product\_descriptions.dropna()

product\_descriptions.shape

product\_descriptions.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | product\_uid | product\_description |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 100001 | Not only do angles make joints stronger, they ... |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 100002 | BEHR Premium Textured DECKOVER is an innovativ... |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | 100003 | Classic architecture meets contemporary design... |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | 100004 | The Grape Solar 265-Watt Polycrystalline PV So... |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | 100005 | Update your bathroom with the Delta Vero Singl... |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

linkcode

product\_descriptions1 = product\_descriptions.head(500)

*# product\_descriptions1.iloc[:,1]*

product\_descriptions1["product\_description"].head(10)

0 Not only do angles make joints stronger, they ...

1 BEHR Premium Textured DECKOVER is an innovativ...

2 Classic architecture meets contemporary design...

3 The Grape Solar 265-Watt Polycrystalline PV So...

4 Update your bathroom with the Delta Vero Singl...

5 Achieving delicious results is almost effortle...

6 The Quantum Adjustable 2-Light LED Black Emerg...

7 The Teks #10 x 1-1/2 in. Zinc-Plated Steel Was...

8 Get the House of Fara 3/4 in. x 3 in. x 8 ft. ...

9 Valley View Industries Metal Stakes (4-Pack) a...

Name: product\_description, dtype: object

**linkcode**

Feature extraction from product descriptions¶

Converting the text in product description into numerical data for analysis

**linkcode**

vectorizer = TfidfVectorizer(stop\_words='english')

X1 = vectorizer.fit\_transform(product\_descriptions1["product\_description"])

X1

<500x8932 sparse matrix of type '<class 'numpy.float64'>'

with 34817 stored elements in Compressed Sparse Row format>

**linkcode**

Visualizing product clusters in subset of data¶

**linkcode**

*# Fitting K-Means to the dataset*

X=X1

kmeans = KMeans(n\_clusters = 10, init = 'k-means++')

y\_kmeans = kmeans.fit\_predict(X)

plt.plot(y\_kmeans, ".")

plt.show()

**linkcode**

F

**linkcode**

def print\_cluster(i):

print("Cluster %d:" % i),

for ind in order\_centroids[i, :10]:

print(' %s' % terms[ind]),

print

**linkcode**

**Output¶**

* Recommendation of product based on the current product selected by user.
* To recommend related product based on, Frequently bought together.

**linkcode**

Top words in each cluster based on product description¶

**linkcode**

*# # Optimal clusters is*

true\_k = 10

model = KMeans(n\_clusters=true\_k, init='k-means++', max\_iter=100, n\_init=1)

model.fit(X1)

print("Top terms per cluster:")

order\_centroids = model.cluster\_centers\_.argsort()[:, ::-1]

terms = vectorizer.get\_feature\_names()

for i in range(true\_k):

print\_cluster(i)

Top terms per cluster:

Cluster 0:

concrete

stake

ft

coating

apply

epoxy

drying

sq

garage

formula

Cluster 1:

wood

patio

bamboo

natural

frame

outdoor

rug

size

steel

dining

Cluster 2:

used

trim

painted

65

proposition

nbsp

residents

california

project

32

Cluster 3:

door

lbs

easy

dog

nickel

solid

roof

plastic

house

adjustable

Cluster 4:

cutting

saw

tool

blade

design

cut

pliers

grip

metal

non

Cluster 5:

wall

piece

finish

tile

design

use

color

easy

installation

water

Cluster 6:

light

watt

bulb

led

fixture

volt

bulbs

lighting

use

power

Cluster 7:

helps

water

easy

snow

handle

nozzle

year

features

tool

control

Cluster 8:

air

ft

water

unit

room

installation

fan

cooling

use

easy

Cluster 9:

post

fence

gate

ft

screen

vinyl

posts

aluminum

brackets

spline

**linkcode**

Predicting clusters based on key search words¶

**linkcode**

def show\_recommendations(product):

*#print("Cluster ID:")*

Y = vectorizer.transform([product])

prediction = model.predict(Y)

*#print(prediction)*

print\_cluster(prediction[0])

**linkcode**

* **Keyword :** cutting tool

**linkcode**

show\_recommendations("cutting tool")

Cluster 4:

cutting

saw

tool

blade

design

cut

pliers

grip

metal

non

**linkcode**

* **Keyword :** spray paint

**linkcode**

show\_recommendations("spray paint")

Cluster 2:

used

trim

painted

65

proposition

nbsp

residents

california

project

32

**linkcode**

* **Keyword :** steel drill

**linkcode**

show\_recommendations("steel drill")

Cluster 8:

air

ft

water

unit

room

installation

fan

cooling

use

easy

**linkcode**

In case a word appears in multiple clusters, the algorithm chooses the cluster with the highest frequency of occurance of the word.

**linkcode**

* **Keyword :** water

**linkcode**

show\_recommendations("water")

Cluster 8:

air

ft

water

unit

room

installation

fan

cooling

use

easy

**linkcode**

Once a cluster is identified based on the user's search words, the recommendation system can display items from the corresponding product clusters based on the product descriptions.

**linkcode**

Summary:¶

This works best if a business is setting up its e-commerce website for the first time and does not have user-item purchase/rating history to start with initally. This recommendation system will help the users get a good recommendation to start with and once the buyers have a purchased history, the recommendation engine can use the model based collaborative filtering technique.