TATA CONSULTANCY SERVICES
Experience certainty.

Online Product Recommendation System

Submitted By:

Name	Employee ID
Sarthak .S. Bharadwaj	2039020

Mentored By:

Name	Employee ID
Sudipta Bandyopadhyay	151777
Diptayan Dutta	1121206

Table of Contents

Problem Statement:

The use of e commerce platform has gained widespread popularity since its inception. Nowadays, we don't need to visit any shop or market to buy any product of our choice. Each and everything is available within our reach within a single click. When we use to visit market for a particular product, the shopkeeper will show us a range of similar products to choose from. The reason for showing similar products is to increase sales as well as giving the user an opportunity to compare between multiple products and help them to take an insightful decision. Similarly, in the e commerce sites the users might need to see the products which are similar to that of the original searched product so that they can compare different features from the recommended products. Recommendation systems are widely used to help the users take decisions by showing various products of the same category/brands.

Overview:

- **What is a Recommendation System:**

Recommendation Engines seek to predict the 'rating' or 'preference' that user would give to an item.This analyses the pattern or similarity between different users.

- **Types:**

Collaborative Recommender System:

It's the most sought after, most widely implemented and most mature technologies that is available in the market. Collaborative recommender systems aggregate ratings or recommendations of objects, recognize commonalities between the users on the basis of their ratings, and generate new recommendations based on inter-user comparisons. The greatest strength of collaborative techniques is that they are completely independent of any machine-readable representation of the objects being recommended and work well for complex objects where variations in taste are responsible for much of the variation in preferences.

Content based Recommender System:

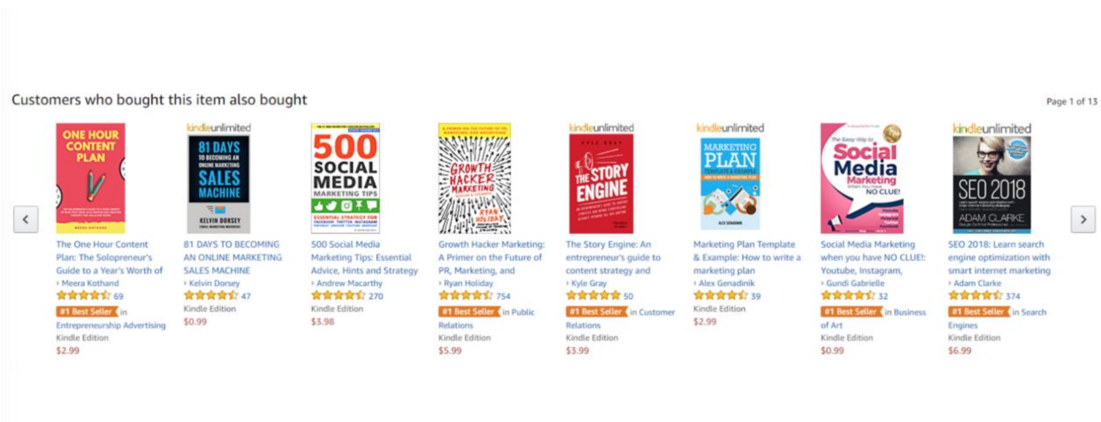
It's mainly classified as an outgrowth and continuation of information filtering research. In this system, the objects are mainly defined by their associated features. A content-based recommender learns a profile of the new user's interests based on the features present; in objects the user has rated. It's basically a keyword specific recommender system here keywords are used to describe the items.

Hybrid Recommender System:

Combining any of the two systems in a manner that suits a particular industry is known as Hybrid Recommender system. This is the most sought after Recommender system that many companies look after, as it combines the strengths of more than two Recommender systems and also eliminates any weakness which exist when only one recommender system is used.

Demographic based Recommender System:

This system aims to categorize the users based on attributes and make recommendations based on demographic classes. Many industries have taken this kind of approach as it's not that complex and easy to implement. In Demographic-based recommender system the algorithms first need a proper market research in the specified region accompanied with a short survey to gather data for categorization. Demographic techniques form "people-to-people" correlations like collaborative ones, but use different data. The benefit of a demographic approach is that it does not require a history of user ratings like that in collaborative and content based recommender systems.



Companies like Amazon use recommendation system to increase their sales. Similarly a lot of OTT platforms use recommendation engines to recommend new movies and series based on user past activities.

Limitations:-

In this project we do not have search by picture option. Only search by text is done.

Project Description:

This project automates the recommendation which in turn improves the chances of user buying the recommended product.
This ensures that the user has to spend minimal time on the site to buy a specific product.
New feature of seasonal sale is also added where in when the user searches for a specific season and the best products of that season will be recommended.

Here we used cosine similarity as the distance similarity varies largely with dataset size.

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. ***The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together****. The smaller the angle, higher the cosine similarity.*

When plotted on a multi-dimensional space, where each dimension corresponds to a word in the document, the cosine similarity captures the orientation (the angle) of the documents and not the magnitude. If you want the magnitude, compute the Euclidean distance instead.

WHY TO AVOID DISTANCE MEASUREMENT AND GO WITH COSINE ANGLE

The Three Documents and Similarity Metrics

Considering only the 3 words from the above documents: 'sachin', 'dhoni', 'cricket'

Doc Sachin: Wiki page on Sachin Tendulkar
Dhoni - 10
Cricket - 50
Sachin - 200

Doc Dhoni: Wiki page on Dhoni
Dhoni - 400
Cricket - 100
Sachin - 20

Doc Dhoni_Small: Subsection of wiki on Dhoni
Dhoni - 10
Cricket - 5
Sachin - 1

Document - Term Matrix (Word Counts)

Word Counts	"Dhoni"	"Cricket"	"Sachin"
Doc Sachin	10	50	200
Doc Dhoni	400	100	20
Doc Dhoni_Small	10	5	1

Similarity Metrics

Similarity or Distance Metrics	Total Common Words	Euclidean distance	Cosine Similarity
Doc Sachin & Doc Dhoni	10 + 50 + 10 = 70	432.4	0.15
Doc Dhoni & Doc Dhoni_Small	20 + 10 + 7 = 37	204.0	0.23
Doc Sachin & Doc Dhoni_Small	10 + 10 + 7 = 27	401.85	0.77

All three documents are connected by a common theme – the game of Cricket.

Our objective is to quantitatively estimate the similarity between the documents.

For understanding consider only the top 3 common words between the documents: 'Dhoni', 'Sachin' and 'Cricket'.

You would expect Doc B and Doc C, that is the two documents on Dhoni would have a higher similarity over Doc A and Doc B, because, Doc C is essentially a snippet from Doc B itself.

However, if we go by the number of common words, the two larger documents will have the most common words and therefore will be judged as most similar, which is exactly what we want to avoid. The results would be more congruent when we use the cosine similarity score to assess the similarity.

It turns out, the closer the documents are by angle, the higher is the Cosine Similarity (Cos theta).

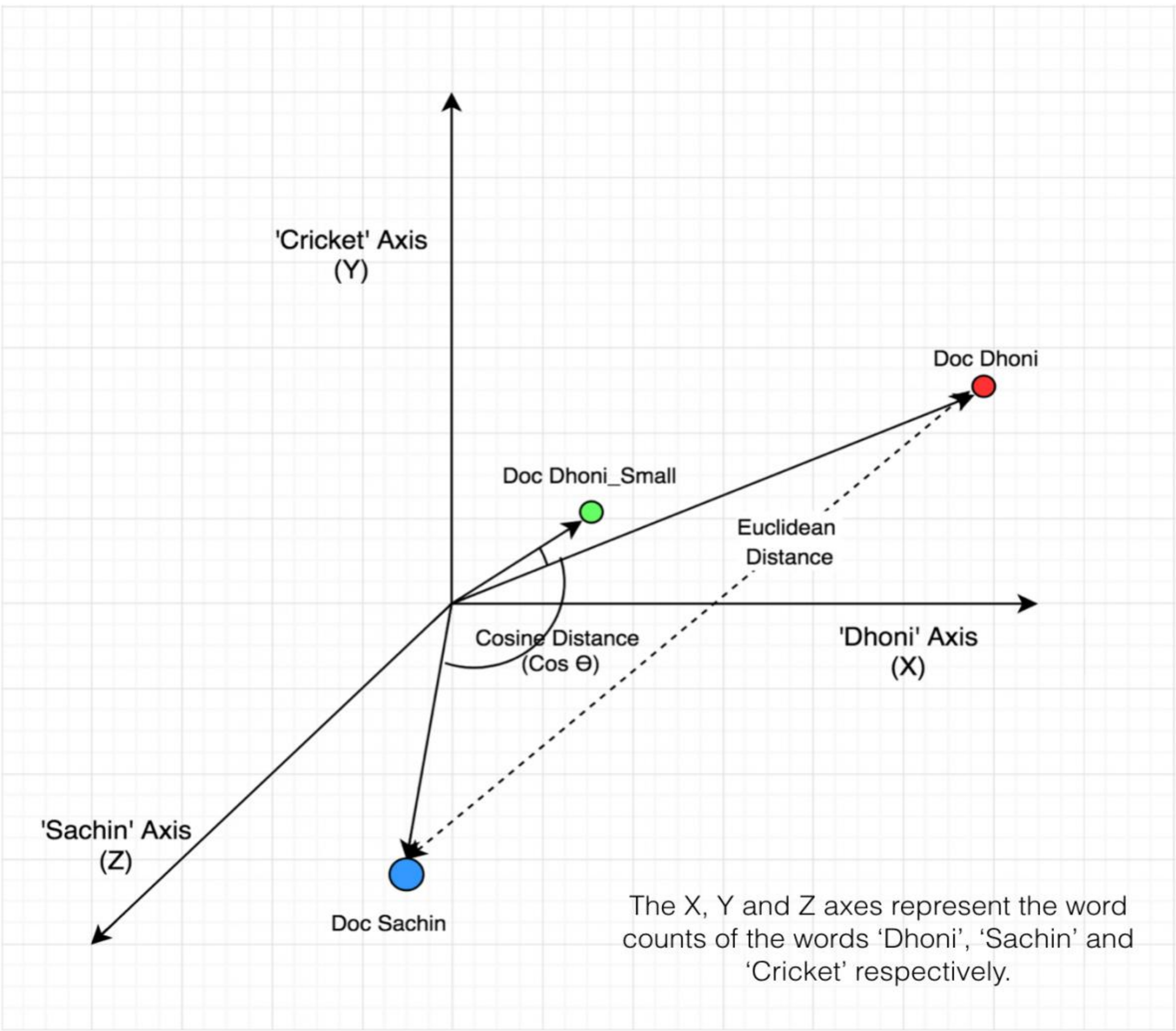
$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

As you include more words from the document, it’s harder to visualize a higher dimensional space. But you can directly compute the cosine similarity using this math formula.

In our Recommendation engine case we used columns **“product-name”, “brand”, “product-category -tree” and “description”** to find the similarity between the products**.**

Projection of Documents in 3D Space



Libraries Used:

- **Flask** -a web application framework used for API.
- **numpy** - for multidimensional data structures , predefined trigonometric functions.
- **pandas** - for loading ,manipulating and analyzing data
- **nltk**- natural language toolkit . used for tokkenising lemmatising and correcting the wrongly spelt words.

    tokenize = breaking sentences or phrases into words
    stemming = finding the root words
    lemmatize = brings words into a common form . eats -> eat
    stopwords = includes commonly occurring eng words , doesn't add meaning to sentences (the/and)

- **seaborn** and **matplotlib** - libraries for plotting graphs.
- **string**- for splitting , joining (present by default)
- **Sklearn** - used for data modelling

Code Snippets And Its Use:-

```
from flask import Flask, request , json

from flipkart_rec import Recommendation
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import string
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
```

Importing the required libraries to use the predefined functions throughout the program.
It is considered good programming practice to keep all the libraries at the beginning of the file.

```python
@app.route('/recommend/', methods=['GET', 'POST'])
def getData():
    user_input = request.args.get('product')
    pre_df=pd.read_csv("E:\\TCS internship related\\recommendation_system\\Dataset\\flipkart_data.csv"
        na_values=["No rating available"])
    default_input = "FabHomeDecor Fabric Double Sofa Bed"

    if user_input not in pre_df.values:
        user_input = default_input[:]

    print( user_input )
    #print(pre_df.head())
    #pre_df.info()
```

Here we are creating the route where it will get deployed.

We take the input from the user and if the user inputs any value that is not present in the data-set or user simply doesn't provide any input then we consider our default input.

Since the data-set is present in the local system we read it from there and converts all the “No rating available ” in the data-set to “NaN ”.

We can download the data-set from /kaggle/input/flipkart-products/flipkart_com-ecommerce_sample.csv
In the next statement we print the user input.

```python
pre_df['product_category_tree']=pre_df['product_category_tree'].map(lambda x:x.strip('[]'))
pre_df['product_category_tree']=pre_df['product_category_tree'].map(lambda x:x.strip('"'))
pre_df['product_category_tree']=pre_df['product_category_tree'].map(lambda x:x.split('>>'))
```

Since out data-set column named “**product_category_tree**” has “**>>,”,[ ]**” we remove those to get the actual meaningful data out of it.

Map function takes in a function and a data-structure operates on it and returns the values.
It generally has two parameters first the function and second the data structure.

A lambda function is a small anonymous function. The function can take any number of arguments, but can only have one expression at max.

```
del_list=['crawl_timestamp', 'product_url','image',"retail_price","discounted_price",
"is_FK_Advantage_product","product_rating","overall_rating","product_specifications"]
pre_df=pre_df.drop(del_list,axis=1)


lem = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
exclude = set(string.punctuation)

#print(pre_df.head())
#print(pre_df.shape)
```

Here we are dropping the columns from the data-set that we wont be using. This will reduce the time needed to complete the execution.
We will be using the columns: **Product_name + Brand +Product category tree + Description**

```
smd=pre_df.copy()
smd.drop_duplicates(subset ="product_name",
                     keep = "first", inplace = True)
#print(smd.shape)

print("\nRUMMAGING THE STOREROOM PLEASE WAIT!\n")

def filter_keywords(doc):
    doc=doc.lower()
    stop_free = " ".join([i for i in doc.split() if i not in stop_words])
    punc_free = "".join(ch for ch in stop_free if ch not in exclude)
    word_tokens = word_tokenize(punc_free)
    filtered_sentence = [(lem.lemmatize(w)) for w in word_tokens]
    return filtered_sentence

#applying the filter
smd['product'] = smd['product_name'].apply(filter_keywords)
smd['description'] = smd['description'].astype("str").apply(filter_keywords)
smd['brand'] = smd['brand'].astype("str").apply(filter_keywords)
```

Here we drop the similar rows after copying the dataset.
Inside filter keywords function we remove the common English words, punctuation marks and keep only the relevant words. Then finally we return the filtered keywords.

We apply these filters to all the columns we used to recommend the product.

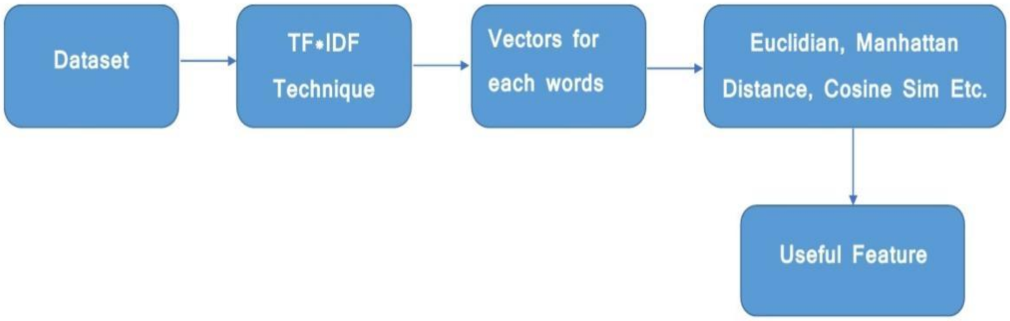
```
smd["all_meta"]=smd['product']+smd['brand']+ pre_df['product_category_tree']+smd['description']
smd["all_meta"] = smd["all_meta"].apply(lambda x: ' '.join(x))

#print(smd["all_meta"].head())

tf = TfidfVectorizer(ngram_range=(1, 2),min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(smd['all_meta'])

cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

Copied the columns data to one single column so that it will be easier to compare. Then we create a matrix so that we can compare the similarity of products through cosine angle between the products.



```python
def get_recommendations(title):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:51]#1:31
    product_indices = [i[0] for i in sim_scores]
    return titles.iloc[product_indices]

smd = smd.reset_index()
titles = smd['product_name']
indices = pd.Series(smd.index, index=smd['product_name'])

#print(get_recommendations(user_input).head(50))
result = get_recommendations(user_input).head(50)
```

In the above code snippet we calculate the similarity and return the indices of top 50 products with best matches to the product that the consumer searched for.

```python
    #print(get_recommendations(user_input).head(50))
    result = get_recommendations(user_input).head(50)
    #print(result["product_name"].tolist())
    #print(json.jsonify(result.to_json()))
    #print(type(result))
    response = app.response_class(
        response=json.dumps(result.to_json()),
        status=200,
        mimetype='application/json'
    )
    return response
if __name__ == '__main__':
    app.run(host='127.0.0.1', port=105)
```

This piece of code gives the output as we have deployed it using flask.

Output:

```
E:\TCS internship related\recommendation_system\src>flipkart_rec.py

 ENTER THE PRODUCT FROM THE DATASET
        :-Packman 8 x 10 inches Security Bags Without POD Jacket Courier Bag Security Bag
Packman 8 x 10 inches Security Bags Without POD Jacket Courier Bag Security Bag

RUMMAGING THE STOREROOM PLEASE WAIT!

5606     Emagica Home Security 1 Channel Home Security ...
7382              Puma Fundamentals Sports Bag S Sport Bag
1040           Vency creation Waterproof Multipurpose Bag
1762                             JDK NOVELTY Hand-held Bag
3723                 Oriflame Waterproof Multipurpose Bag
12234                                  Prajo Hand-held Bag
1757                              JDK NOVELTY Shoulder Bag
1939                          Synergy SFJB0105 Grocery Bag
9055                               Brandvilla Shoulder Bag
11845                                  DLOOP Hand-held Bag
11843                                Yumlookup Shoulder Bag
8652                         Bendly Outrider Rucksack  - 60 L
872                                    X-WELL Shoulder Bag
1761                                     Edel Shoulder Bag
9054                                   ALIFS Hand-held Bag
11842                             Butterflies Hand-held Bag
941                     Dealcrox Kangaroo Keeper Bags Organizer
12246                                Histeria Hand-held Bag
11837                                Priority Hand-held Bag
9056                                     SSM Hand-held Bag
9053                                  BagsHub Shoulder Bag
11378      United Bags Girls Super Mario Bag 13 L Backpack
965                           Dolphin Product Shoulder Bag
12437                              ALL DAY 365 Shoulder Bag
12235                                   Maayas Shoulder Bag
12201                                   ESBEDA Shoulder Bag
11835                              Adore London Hand-held Bag
11348                 MOOI-ZAK Girls, Women Blue PU Sling Bag
995                          YBC Women Pink PU Sling Bag
11345                 MOOI-ZAK Girls, Women Pink PU Sling Bag
7183               Total TS_NCK184 Gymnastic Stick - 10 inch
7187               Total TS_NCK185 Gymnastic Stick - 10 inch
1347                                  Dinero Hand-held Bag
```

If we take the input "Packman 8 x 10 inches… bag " we get the output as shown above. Along with index we get the product name.

As we can see we recommend different kinds of bags here to the user.

```
 ENTER THE PRODUCT FROM THE DATASET
        :-Aries Gold G 729 S-BK Analog Watch  - For Men, Boys
Aries Gold G 729 S-BK Analog Watch  - For Men, Boys

RUMMAGING THE STOREROOM PLEASE WAIT!

215       D'Signer 681GM_WHT Analog Watch  - For Men, Boys
181            A Avon PK_504 Analog Watch  - For Men, Boys
217            A Avon PK_741 Analog Watch  - For Men, Boys
5924     Flippd FD040102 Casual Analog Watch  - For Men...
6301     Flippd FD040107 Casual Analog Watch  - For Men...
166          Positif Pfbk612 Analog Watch  - For Men, Boys
6113     Flippd FD040103 Formal Analog Watch  - For Men...
71       Camerii WM64 Elegance Analog Watch  - For Men,...
9589                     R.S D&G16 Analog Watch  - For Men
113      Franck Bella FB0128B Analog Watch  - For Men, ...
182      Franck Bella FB0122DD Analog Watch  - For Men,...
203      Franck Bella FB123D Analog Watch  - For Men, Boys
6187     Times 314TMS314 Party-Wedding Analog Watch  - ...
180       fastrack 38015PL01 Analog Watch  - For Men, Boys
5883         Now B140-SRR12 Analog Watch  - For Men, Women
209       Lenco Bdblue Tango Analog Watch  - For Men, Boys
167      D'SIGNER 688RGM _BRN Analog Watch  - For Men, ...
65       Cobra Paris CO6394A1 Analog Watch  - For Men, ...
158                 Now SP-ETHNIC Analog Watch  - For Boys
122      Colat COLAT_M08 Roman Numerals Analog Watch  -...
5892      Sonata Gold Plated GOLDP Analog Watch  - For Men
5994         Maxima 24742LMGY Gold Analog Watch  - For Men
6207         Maxima 06896LMGY Gold Analog Watch  - For Men
6197     HMT Sonata Gold Plated Watch For Men Sonata An...
6101         Maxima 01427CMGY Gold Analog Watch  - For Men
5960         Maxima 04615CMGY Gold Analog Watch  - For Men
6153         Maxima 19431CMGY Gold Analog Watch  - For Men
5922         Maxima 01433CMGY Gold Analog Watch  - For Men
5904                Sonata Everyday Analog Watch - For Men
6082         Maxima 04608CMGY Gold Analog Watch  - For Men
6015         Maxima 06362CMGY Gold Analog Watch  - For Men
6006         Maxima 09321CMGY Gold Analog Watch  - For Men
6273         Maxima 29126LMGY Gold Analog Watch  - For Men
```

If we take the input “Aries gold …Mens, Boys ” we get the output as shown above. Along with index we get the product name.

As we can see we recommend different kinds of watches for Men and Boys here to the user.

```
E:\TCS internship related\recommendation_system\src>flipkart_rec.py

 ENTER THE PRODUCT FROM THE DATASET
        :-Salt N Pepper 13-455 Pisa Black Boots
Salt N Pepper 13-455 Pisa Black Boots

RUMMAGING THE STOREROOM PLEASE WAIT!

103                Salt N Pepper 13-167 Marsha Red Boots
214      Salt N Pepper 14-475 Dorthea Almond Boots Boots
194            Salt N Pepper 13-552 Rebacca Almond Boots
98                  Salt N Pepper 13-516 Greta Red Boots
86          Salt N Pepper 13-019 Femme Black Boots Boots
89          Salt N Pepper 14-664 Denny Black Boots Boots
1060                 salt n pepper 16-510 BLACK Lace Up
183                   Salt N Pepper 12-298 Taupe Boots
5152       Kraftnation Gourmet 2 Piece Salt & Pepper Set
168         Salt N Pepper 13-019 Femme Taupe Boots Boots
111                               La Briza Black Boots
10043                          Bacca Bucci Black Boots
66         TEN TEN Women's Black Knee Length Boots Boots
78                                      Catwalk Boots
109                                       Kielz Boots
176           TEN TEN Women's Tan Mid Length Boots Boots
207                                     CatBird Boots
81                                       Rialto Boots
137                                      Credos Boots
68                                      Carlton Boots
91                                        Crocs Boots
130                                        Roxy Boots
200                          Willy Winkies Black Boots
170                            Get Glamr Stylish Boots
146                           Selfie Black Denim Boots
161                                Remson India Boots
136                                  Foot Candy Boots
9985                      TEN Stylish and Elegant Boots
119                                Sneha Unique Boots
75                             Steppings Trendy Boots
94                             Myra Comfortable Boots
82                                 Kielz Ladies Boots
141                               Anand Archies Boots
```

Similarly when we take input “Salt n Pepper … Boots” we get the output as shown above. Along with index we get the product name.

As we can see here we recommend different kinds of Boots to the user

References:

- https://www.kaggle.com/PromptCloudHQ/flipkart-products/ data-set for the engine
- https://app.diagrams.net/ : draw.io
- https://en.wikipedia.org/wiki/Recommender_system
- https://www.machinelearningplus.com/nlp/cosine-similarity/#3cosinesimilarityexample
- https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- https://www.geeksforgeeks.org/sklearn-feature-extraction-with-tf-idf/
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html
- https://goodboychan.github.io/python/datacamp/natural_language_processing/2020/07/17/04-TF-IDF-and-similarity-scores.html