

# **RECOMMENDED SYSTEM WITH SENTIMENT ANALYSIS**

*Submitted by*

**NAGAR MANTHAN RAMESHKUMAR**

**190130107070**

*In partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

*in*

**Computer Engineering**

**Government Engineering College, Gandhinagar**



**Gujarat Technological University, Ahmedabad**

**July, 2022**



## Government Engineering College, Gandhinagar

### CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Recommended system with sentiment analysis** has been carried out by **Nagar Manthan Rameshkumar 190130107070** under my guidance in partial fulfillment for the degree of Bachelor of Engineering in Computer Engineering, 7<sup>th</sup> Semester of Gujarat Technological University, Ahmedabad during the academic year 2022-23.

Prof. D P Khem  
Internal Guide

Mr. Dhaval Parikh  
Head of the Department



## Government Engineering College, Gandhinagar

### DECLARATION

I hereby declare that the Summer Internship report submitted along with the Summer Internship entitled **Recommended System with Sentiment Analysis** submitted in partial fulfillment for the degree of Bachelor of Engineering in Computer Engineering to Gujarat Technological University, Ahmedabad, is a bonafide record of original project work carried out by me at BrainyBeam Technologies Pvt. Ltd. under the supervision of Mr. Sagar Jasani and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due references.

Name of the Student

Sign of Student

## LIST OF FIGURES

Figure 1.1 Traditional programming vs Machine learning .....	10
Figure 1.2 Types Of Machine Learning System .....	12
Figure 2.1 Output (1) .....	16
Figure 2.2 Output (2) .....	17
Figure 2.3 Output (3) .....	17
Figure 2.4 Output (4) .....	17
Figure 2.5 Output (5) .....	18
Figure 2.6 Output (6) .....	18
Figure 2.7 Output (7) .....	19
Figure 2.8 Output (8) .....	20
Figure 2.9 Output (9) .....	20
Figure 2.10 Output (10) .....	21
Figure 2.11 Output (11) .....	22
Figure 2.12 Output (12) .....	22
Figure 2.13 Output (13) .....	23
Figure 2.14 Output (14) .....	23
Figure 2.15 Output (15) .....	24
Figure 2.16 Tokenization .....	25
Figure 2.17 Lemmatization .....	26
Figure 2.18 Stemming .....	27
Figure 2.19 Frequency Distribution .....	28
Figure 2.20 Standardisation vs Min-Max Normalization .....	30
Figure 2.21 Filter Method vs Wrapper Method vs Embedded Method .....	31
Figure 2.22 Label Encoding .....	32
Figure 2.23 TF-IDF Example .....	33
Figure 2.24 Term Frequency .....	33
Figure 2.25 Inverse Document Frequency .....	34
Figure 2.26 TF-IDF Vectorizer .....	35
Figure 2.27 Sparse Matrix .....	36
Figure 3.1 KNN Classifier .....	38

Figure 3.2 KNN Classifier Graph .....	38
Figure 3.3 Cosine Similarity Equation.....	39
Figure 3.4 Cosine Similarity .....	39
Figure 3.5 User-Based Filtering.....	40
Figure 3.6 Item-Based Filtering.....	40
Figure 3.7 Matrix Factorization .....	42
Figure 4.1 Output (1) .....	43
Figure 4.2 Output (2) .....	44
Figure 4.3 Output (3) .....	45
Figure 4.4 Output (4) .....	46
Figure 4.5 Output (5) .....	47
Figure 4.6 Output (6) .....	48
Figure 4.7 Output (7) .....	49
Figure 4.8 Output (8) .....	50
Figure 4.9 Output (9) .....	50
Figure 4.10 Output (10) .....	51
Figure 4.11 Output (11) .....	52

## TABLE OF CONTENT

DECLARATION .....	I
ACKNOWLEDGEMENT .....	<b>Error! Bookmark not defined.</b>
ABSTRACT.....	<b>Error! Bookmark not defined.</b>
LIST OF FIGURES .....	IV
TABLE OF CONTENT .....	VI
CHAPTER 1 .....	8
INTRODUCTION .....	8
1.1    WHAT IS DATA SCIENCE?.....	8
1.2    WHAT IS MACHINE LEARNING? .....	9
1.3    HOW MACHINE LEARNING AND DATA SCIENCE IMPACT ORGANIZATION?.....	10
1.4    TYPES OF DATA .....	11
1.5    TYPES OF MACHINE LEARNING SYSTEM.....	12
1.6    RECOMMENDATION SYSTEM.....	15
CHAPTER 2 .....	16
DATA PRE-PROCESSING .....	16
2.1    PYTHON EXTERNAL LIBRARIES.....	16
2.1.1    Numpy : .....	16
2.1.2    Pandas : .....	18
2.1.3    Matplotlib : .....	21
2.1.4    Nltk : .....	24
2.2    NOISE REMOVAL .....	29
2.3    NORMALIZATION .....	29

2.4	FEATURE SELECTION .....	30
2.5	FEATURE EXTRACTION .....	31
2.5.1	Label Encoding : .....	31
2.5.2	TF-IDF Vectorizer : .....	32
2.5.3	Sparse Matrix : .....	35
CHAPTER 3 .....		37
MODEL BUILDING .....		37
3.1	K-NEAREST NEIGHBOR .....	37
3.2	COSINE SIMILARITY .....	38
3.3	COLLABORATIVE FIELTERING .....	40
3.4	TESTING MODEL .....	41
3.5	MATRIX FACTORIZATION .....	41
CHAPTER 4 .....		43
PRODUCT RECOMMENDATION SYSTEM.....		43
4.1	LOAD DATASET.....	43
4.2	CLAEAN THE TEXT USING STOP WORDS AND REGULAR EXPRESSION FROM REVIEW TEXT .....	43
4.3	GENERATE A NEW SCORE BY MERGING THE RATING AND SENTIMENT SCORE .....	45
4.4	FIND THE COSINE SIMILARITY BETWEEN PRODUCTS .....	47
4.5	FIND NEAREST NEIGHBORS AND THEIR DISTANCE VALUES WITH THEIR INDICES .....	50
4.6	BUILD A RECOMMENDATION ENGINE USING KNN .....	51
CHAPTER 5 .....		53
CONCLUSION.....		53
REFERENCES .....		54

# CHAPTER 1

## INTRODUCTION

### 1.1 WHAT IS DATA SCIENCE?

- Data science combines math and statistics, specialized programming, advanced analytics, artificial intelligence (AI), and machine learning with specific subject matter expertise to uncover actionable insights hidden in an organization's data. These insights can be used to guide decision making and strategic planning.
- The accelerating volume of data sources, and subsequently data, has made data science is one of the fastest growing field across every industry. As a result, it is no surprise that the role of the data scientist was dubbed the “sexiest job of the 21st century” by Harvard Business Review (link resides outside of IBM). Organizations are increasingly reliant on them to interpret data and provide actionable recommendations to improve business outcomes.
- The data science lifecycle involves various roles, tools, and processes, which enables analysts to glean actionable insights. Typically, a data science project undergoes the following stages:
- Data ingestion: The lifecycle begins with the data collection--both raw structured and unstructured data from all relevant sources using a variety of methods. These methods can include manual entry, web scraping, and real-time streaming data from systems and devices. Data sources can include structured data, such as customer data, along with unstructured data like log files, video, audio, pictures, the Internet of Things (IoT), social media, and more.
- Data storage and data processing: Since data can have different formats and structures, companies need to consider different storage systems based on the type of data that needs to be captured. Data management teams help to set standards around data storage and structure, which facilitate workflows around analytics,



machine learning and deep learning models. This stage includes cleaning data, deduplicating, transforming and combining the data using ETL (extract, transform, load) jobs or other data integration technologies. This data preparation is essential for promoting data quality before loading into a data warehouse, data lake, or other repository.

- **Data analysis:** Here, data scientists conduct an exploratory data analysis to examine biases, patterns, ranges, and distributions of values within the data. This data analytics exploration drives hypothesis generation for a/b testing. It also allows analysts to determine the data's relevance for use within modeling efforts for predictive analytics, machine learning, and/or deep learning. Depending on a model's accuracy, organizations can become reliant on these insights for business decision making, allowing them to drive more scalability.
- **Communicate:** Finally, insights are presented as reports and other data visualizations that make the insights—and their impact on business—easier for business analysts and other decision-makers to understand. A data science programming language such as R or Python includes components for generating visualizations; alternately, data scientists can use dedicated visualization tools.

## **1.2 WHAT IS MACHINE LEARNING?**

- Machine learning is a “Field of study that gives computers the capability to learn without being explicitly programmed”. In a very layman's manner, Machine Learning(ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance.
- The process starts with feeding good quality data and then training our machines(computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate. Example: Training of students during exams. While preparing for the exams students don't actually cram the subject but try to learn it with complete understanding. Before the examination, they feed their machine(brain) with a good amount of high-quality data (questions

and answers from different books or teachers' notes, or online video lectures). Actually, they are training their brain with input as well as output i.e. what kind of approach or logic do they have to solve a different kinds of questions. Each time they solve practice test papers and find the performance (accuracy /score) by comparing answers with the answer key given, Gradually, the performance keeps on increasing, gaining more confidence with the adopted approach.

- That's how actually models are built, train machine with data (both inputs and outputs are given to the model), and when the time comes test on data (with input only) and achieve our model scores by comparing its answer with the actual output which has not been fed while training. Researchers are working with assiduous efforts to improve algorithms, and techniques so that these models perform even much better.

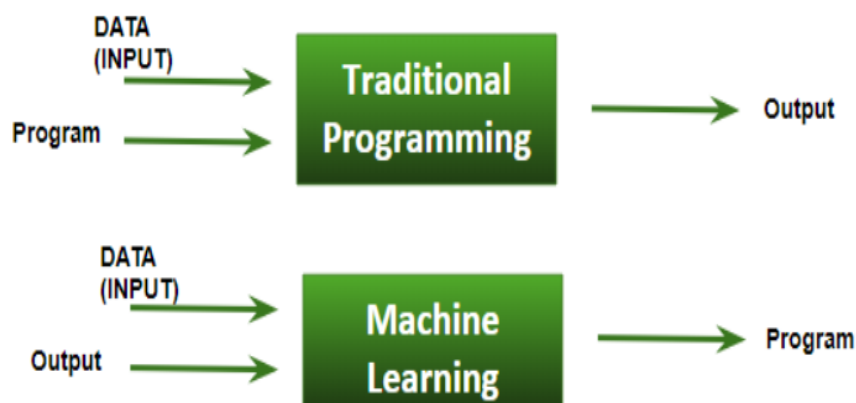


Figure 1.1 Traditional programming vs Machine learning

### 1.3 HOW MACHINE LEARNING AND DATA SCIENCE IMPACT ORGANIZATION?

The developed machine learning algorithms are used in various applications are:

- Vision processing
- Language processing

- Forecasting things like stock market trends, weather
- Pattern recognition
- Games
- Data mining
- Expert systems
- Robotic

## 1.4 TYPES OF DATA

### ➤ **Structured data –**

Structured data is data whose elements are addressable for effective analysis. It has been organized into a formatted repository that is typically a database. It concerns all data which can be stored in database SQL in a table with rows and columns. They have relational keys and can easily be mapped into pre-designed fields. Today, those data are most processed in the development and simplest way to manage information.

Example: Relational data.

### ➤ **Semi-Structured data –**

Semi-structured data is information that does not reside in a relational database but that has some organizational properties that make it easier to analyze. With some processes, you can store them in the relation database (it could be very hard for semi-structured data), but Semi-structured exist to ease space.

Example:XMLdata.

### ➤ **Unstructured data –**

Unstructured data is a data which is not organized in a predefined manner or does not have a predefined data model; thus, it is not a good fit for a mainstream

relational database. So, for Unstructured data, there are alternative platforms for storing and managing, it is increasingly prevalent in IT systems and is used by organizations in a variety of business intelligence and analytics applications.

Example: Word, PDF, Text, Media logs.

## 1.5 TYPES OF MACHINE LEARNING SYSTEM

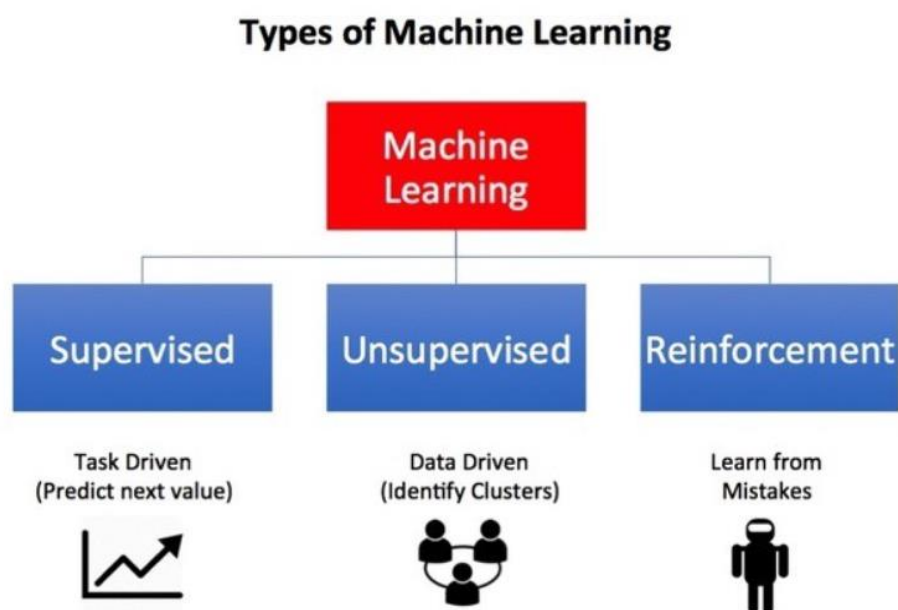


Figure 1.2 Types Of Machine Learning System

### ➤ Supervised Learning:-

Supervised learning is the most popular paradigm for machine learning. It is the easiest to understand and the simplest to implement. It is very similar to teaching a child with the use of flash cards.

Given data in the form of examples with labels, we can feed a learning algorithm these example-label pairs one by one, allowing the algorithm to predict the label for each example, and giving it feedback as to whether it predicted the right answer or not. Over time, the algorithm will learn to approximate the exact nature of the relationship between examples and their labels. When fully-trained, the supervised learning algorithm will be able to observe a new, never-before-seen example and

predict a good label for it.

Supervised learning is often described as task-oriented because of this. It is highly focused on a singular task, feeding more and more examples to the algorithm until it can accurately perform on that task. This is the learning type that we will most likely encounter, as it is exhibited in many of the following common applications:

**Spam Classification:** If you use a modern email system, chances are you've encountered a spam filter. That spam filter is a supervised learning system. Fed email examples and labels (spam/not spam), these systems learn how to preemptively filter out malicious emails so that their user is not harassed by them. Many of these also behave in such a way that a user can provide new labels to the system and it can learn user preference.

**Face Recognition:** Do you use Facebook? Most likely your face has been used in a supervised learning algorithm that is trained to recognize your face. Having a system that takes a photo, finds faces, and guesses who that is in the photo (suggesting a tag) is a supervised process. It has multiple layers to it, finding faces and then identifying them, but is still supervised nonetheless.

➤ **Unsupervised Learning:-**

Unsupervised learning is very much the opposite of supervised learning. It features no labels. Instead, our algorithm would be fed a lot of data and given the tools to understand the properties of the data. From there, it can learn to group, cluster, and/or organize the data in a way such that a human (or other intelligent algorithm) can come in and make sense of the newly organized data.

What makes unsupervised learning such an interesting area is that an overwhelming majority of data in this world is unlabeled. Having intelligent algorithms that can take our terabytes and terabytes of unlabeled data and make sense of it is a huge source of potential profit for many industries. That alone could help boost productivity in a number of fields.

**Recommender Systems:** If you've ever used YouTube or Netflix, you've most likely encountered a video recommendation system. These systems are often times

placed in the unsupervised domain. We know things about videos, maybe their length, their genre, etc. We also know the watch history of many users. Taking into account users that have watched similar videos as you and then enjoyed other videos that you have yet to see, a recommender system can see this relationship in the data and prompt you with such a suggestion.

**Buying Habits:** It is likely that your buying habits are contained in a database somewhere and that data is being bought and sold actively at this time. These buying habits can be used in unsupervised learning algorithms to group customers into similar purchasing segments. This helps companies market to these grouped segments and can even resemble recommender systems.

➤ **Reinforcement Learning :-**

Reinforcement learning is fairly different when compared to supervised and unsupervised learning. Where we can easily see the relationship between supervised and unsupervised (the presence or absence of labels), the relationship to reinforcement learning is a bit murkier. Some people try to tie reinforcement learning closer to the two by describing it as a type of learning that relies on a time-dependent sequence of labels, however, my opinion is that that simply makes things more confusing.

Reinforcement learning is learning that learn from mistakes. Place a reinforcement learning algorithm into any environment and it will make a lot of mistakes in the beginning. So long as we provide some sort of signal to the algorithm that associates good behaviors with a positive signal and bad behaviors with a negative one, we can reinforce our algorithm to prefer good behaviors over bad ones. Over time, our learning algorithm learns to make less mistakes than it used to.

Reinforcement learning is very behavior driven. It has influences from the fields of neuroscience and psychology. If you've heard of Pavlov's dog, then you may already be familiar with the idea of reinforcing an agent, albeit a biological one. The application of reinforcement learning are below.

**Video Games:** One of the most common places to look at reinforcement learning is in learning to play games. Look at Google's reinforcement learning application,

AlphaZero and AlphaGo which learned to play the game Go. Our Mario example is also a common example. Currently, I don't know any production-grade game that has a reinforcement learning agent deployed as its game AI, but I can imagine that this will soon be an interesting option for game devs to employ.

Resource Management: Reinforcement learning is good for navigating complex environments. It can handle the need to balance certain requirements. Take, for example, Google's data centers. They used reinforcement learning to balance the need to satisfy our power requirements, but do it as efficiently as possible, cutting major cost.

## **1.6 RECOMMENDATION SYSTEM**

- A recommendation system is a subclass of Information filtering Systems that seeks to predict the rating or the preference a user might give to an item.
- It is an algorithm that suggests relevant items to users. Eg: In the case of Netflix which movie to watch, In the case of e-commerce which product to buy, or In the case of kindle which book to read, etc.
- Recommender systems are a way of suggesting like or similar items and ideas to a users specific way of thinking.
- Collaborative system aggregation of consumers' preferences and recommendations to other users based on similarity in behavioral patterns
- Content-based system – supervised machine learning used to induce a classifier to discriminate between interesting and uninteresting items for the user.

## CHAPTER 2

### DATA PRE-PROCESSING

#### 2.1 PYTHON EXTERNAL LIBRARIES

##### 2.1.1 Numpy :

- NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

- **Methods of Numpy:-**

1. **Array():**

**Code:**

```
import numpy as np
a = np.array([1,2,3])
print(a)
```

**Output:**

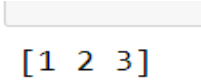


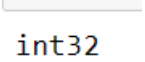
Figure 2.1 Output (1)

2. **dtype():**

**Code:**

```
import numpy as np
d = np.dtype(np.int32)
print(d)
```



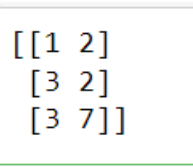
**Output:**

```
int32
```

Figure 2.2 Output (2)

**3. empty():****Code:**

```
import numpy as np
x = np.empty([3,2],dtype = int)
print(x)
```

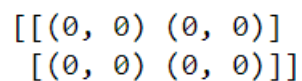
**Output:**

```
[[1 2]
 [3 2]
 [3 7]]
```

Figure 2.3 Output (3)

**4. zeros():****Code:**

```
import numpy as np
x = np.zeros((2,2),dtype=[('x','i4'),('y','i4')])
print(x)
```

**Output:**

```
[(0, 0) (0, 0)]
[(0, 0) (0, 0)]
```

Figure 2.4 Output (4)

**5. arange():****Code:**

```
import numpy as np
x = np.arange(5)
print(x)
```

**Output:**

---

[0 1 2 3 4]

Figure 2.5 Output (5)

### 2.1.2 Pandas :

- Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

- **Methods of Pandas:-**

1. **read\_csv():**

**Code:**

```
import pandas as pd
x = pd.read_csv('subject.csv')
x
```

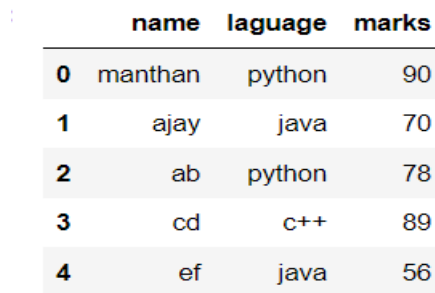
**Output:**

	name	language	marks
0	manthan	python	90
1	ajay	java	70
2	ab	python	78
3	cd	c++	89
4	ef	java	56
5	gh	java	87
6	ij	python	67
7	k	ds	48
8	l	python	86
9	manthan	ml	99

Figure 2.6 Output (6)

**2. head():****Code:**

```
import pandas as pd  
x.head()
```

**Output:**

	name	language	marks
0	manthan	python	90
1	ajay	java	70
2	ab	python	78
3	cd	c++	89
4	ef	java	56

Figure 2.7 Output (7)

**3. Describe():****Code:**

```
import pandas as pd  
x.describe()
```

**Output:**

marks	
count	3.000000
mean	28.333333
std	17.559423
min	10.000000
25%	20.000000
50%	30.000000
75%	37.500000
max	45.000000

Figure 2.8 Output (8)

#### 4. astype():

##### Code:

```
import pandas as pd
x['mearge'] = x['name'] + x['marks'].astype(str)
x
```

##### Output:

	name	language	marks	merge
0	manthan	python	90	manthan90
1	ajay	java	70	ajay70
2	ab	python	78	ab78
3	cd	c++	89	cd89
4	ef	java	56	ef56
5	gh	java	87	gh87
6	ij	python	67	ij67
7	k	ds	48	k48
8	l	python	86	l86
9	manthan	ml	99	manthan99

Figure 2.9 Output (9)

**5. loc()****Code:**

```
import pandas as pd  
x.loc[[2,4,6],['marks']]
```

**Output:**

marks	
2	78
6	67
4	56

Figure 210 Output (10)

**2.1.3 Matplotlib :**

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.
- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.
- **Methods of Matplotlib:-**

**1.show() and plot():****Code:**

```
import matplotlib.pyplot as plt  
x = [5, 2, 9, 4, 7]  
y = [10, 5, 8, 4, 2]  
plt.plot(x,y)  
plt.show()
```

**Output:**

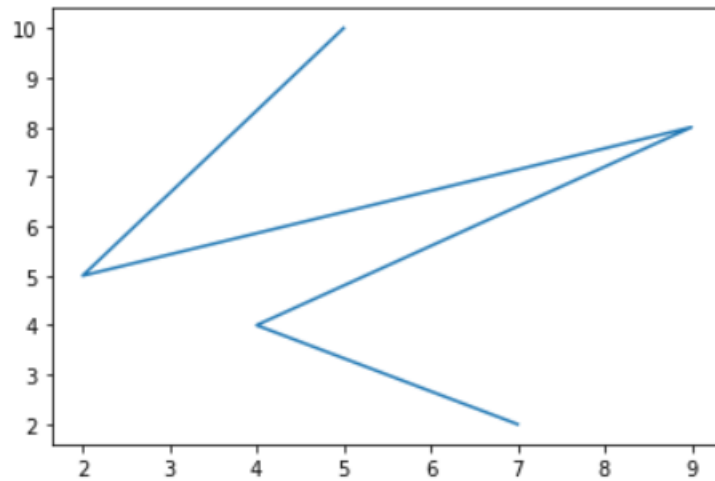


Figure 2.11 Output (11)

**2.subplot:****Code:**

```
x = np.array([2,4,3,6,4,8,7])
y = np.array([1,2,3,4,5,6,7])
plt.subplot(1, 2, 1)
plt.plot(x,y)
```

```
x = np.array([0, 1, 2, 3])
y = np.array([2,4,6,8])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.show()
```

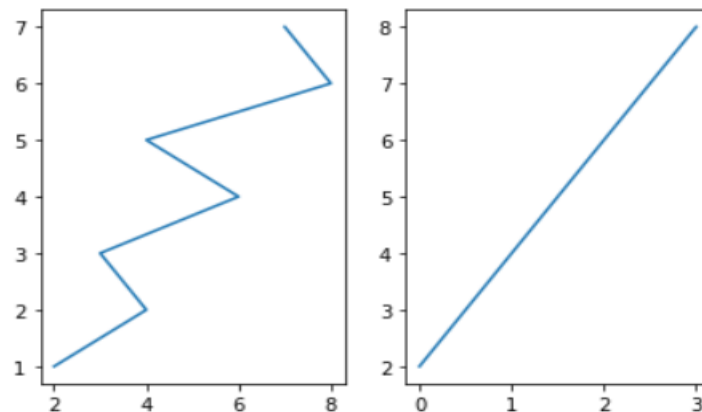
**Output:**

Figure 2.12 Output (12)

**3.pie:****Code:**

```
y = np.array([10,20,20,60])  
mylabels = ["USA", "India", "Russia", "China"]  
plt.pie(y, labels=mylabels,autopct='% 1.1f%%') # autopct for show percentage  
plt.show()
```

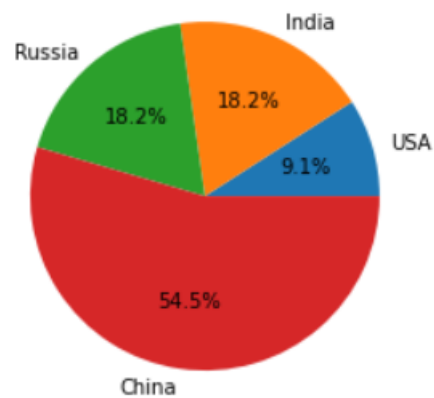
**Output:**

Figure 2.13 Output (13)

**4.bar:****Code:**

```
x = np.array(["A", "B", "C", "D"])  
y = np.array([2,4,6,8])  
plt.bar(x,y)  
plt.show()
```

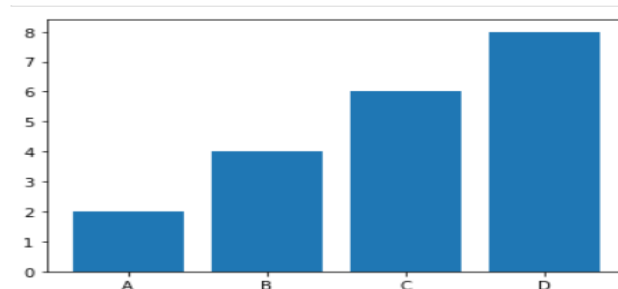
**Output:**

Figure 2.14 Output (14)

**5.scatter:****Code:**

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y)
plt.show()
```

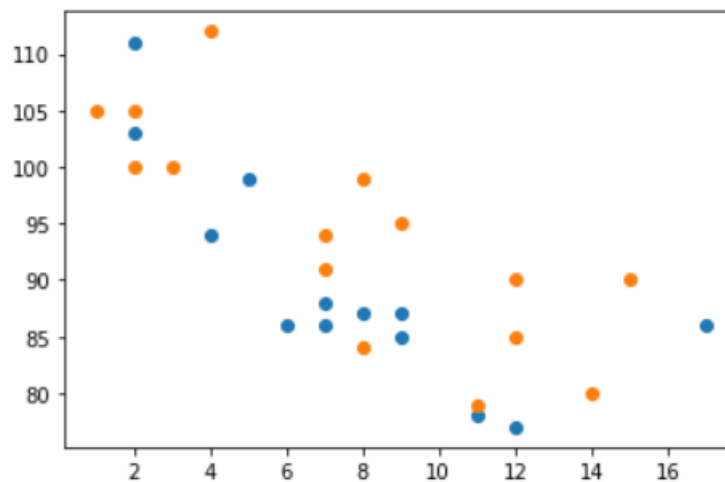
**Output:**

Figure 2.15 Output (15)

**2.1.4 Nltk :**

- **NLTK (Natural Language Toolkit)** Library is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response.

- **Tokenization :**

By tokenizing, you can conveniently split up text by word or by sentence. This will allow you to work with smaller pieces of text that are still relatively coherent and meaningful even outside of the context of the rest of the text. It's your first step in turning unstructured data into structured data, which is easier to analyze.



```
sentences = corpus.sents()
print("Total Sentences:",len(sentences))
print("First sentence:",sentences[0])

Total Sentences: 1
First sentence: ['C', 'programming', 'language', ',', 'generally', ',', 'the', 'special', 'symbols', 'have', 'some', 'special',
'meaning', 'and', 'they', 'cannot', 'be', 'used', 'for', 'other', 'purposes', '.', 'Some', 'of', 'the', 'special', 'symbols',
'that', 'are', 'used', 'in', 'C', 'programming', 'are', 'as', 'follows', '[-[]', '()', '{}', ';', '*', '=', '#.']

print("Words in the corpus:",corpus.words())

Words in the corpus: ['C', 'programming', 'language', ',', 'generally', ',', ...]
```

Figure 2.16 Tokenization

➤ **Lemmatization :**

Lemmatization in NLTK is the algorithmic process of finding the lemma of a word depending on its meaning and context. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word known as the lemma. The NLTK Lemmatization method is based on WordNet's built-in morph function. Text pre-processing includes both stemming as well as lemmatization. Many people find the two terms confusing. Some treat these as the same, but there is a difference between stemming vs lemmatization. Lemmatization is preferred over the former because of the below reason.

```

from nltk.stem import WordNetLemmatizer
wnl = WordNetLemmatizer()
word = corpus.words()
for words in word:
    print(words + " ---> " + wnl.lemmatize(words))

C ---> C
programming ---> programming
language ---> language
, ---> ,
generally ---> generally
, ---> ,
the ---> the
special ---> special
symbols ---> symbol
have ---> have
some ---> some
special ---> special
meaning ---> meaning
and ---> and
they ---> they
cannot ---> cannot
be ---> be
used ---> used
for ---> for
other ---> other
purposes ---> purpose
. ---> .
Some ---> Some
of ---> of
the ---> the
special ---> special
symbols ---> symbol
that ---> that
are ---> are
used ---> used
in ---> in
C ---> C
programming ---> programming
are ---> are
as ---> a
follows ---> follows
-[] ---> -[]
() ---> ()
{}, ---> {},
; ---> ;
* ---> *
= ---> =
#. ---> #.

```

Figure 2.17 Lemmatization

### ➤ Stemming:

Stemming is a method of normalization of words in Natural Language Processing. It is a technique in which a set of words in a sentence are converted into a sequence to shorten its lookup. In this method, the words having the same meaning but have some variations according to the context or sentence are normalized.

In another word, there is one root word, but there are many variations of the same words. For example, the root word is “eat” and it’s variations are “eats, eating, eaten and like so”. In the same way, with the help of Stemming in Python, we can find the root word of any variations.

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
ps = PorterStemmer()
words = corpus.words()
for x in words:
    print(x, " : ", ps.stem(x))
```

```
C : c
programming : program
language : languag
, : ,
generally : gener
, : ,
the : the
special : special
symbols : symbol
have : have
some : some
special : special
meaning : mean
and : and
they : they
cannot : cannot
be : be
used : use
for : for
other : other
purposes : purpos
. : .
Some : some
of : of
the : the
special : special
symbols : symbol
that : that
are : are
used : use
in : in
C : c
programming : program
are : are
as : as
follows : follow
-[] : -[]
() : ()
{}, : {},
; : ;
* : *
= : =
#. : #.
```

Figure 2.18 Stemming

### ➤ Frequency Distribution :

Counting the frequency of occurrence of a word in a body of text is called as Frequency distribution.

```
from nltk.probability import FreqDist
f=FreqDist()
word = corpus.words()
for i in word:
    f[i]+=1
f
FreqDist({'special': 3, 'C': 2, 'programming': 2, ',': 2, 'the': 2, 'symbols': 2, 'used': 2, 'are': 2, 'language': 1, 'generall
y': 1, ...})
```

Figure 2.19 Frequency Distribution

### ➤ Stop words :

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

```
{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during', 'out', 'very',
'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such', 'into', 'of', 'most', 'itself',
'other', 'off', 'is', 's', 'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves', 'until', 'below', 'are',
'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me', 'were', 'her', 'more', 'himself', 'this', 'down',
'should', 'our', 'their', 'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any',
'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that',
'because', 'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself', 'has', 'just',
'where', 'too', 'only', 'myself', 'which', 'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my',
'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}
```

Figure 2.20 Stop Words

## 2.2 NOISE REMOVAL

- Noise: Random error or variance in a measure variable
- For numeric values, box plots and scatter plots can be used to identify outliers. To deal with these anomalous values, data smoothing techniques are applied, which are described below.
- Binning: Using binning method smooths sorted value by using the values around it. The sorted values are then divided into 'bins'. There are various approaches to binning. Two of them are smoothing by bin means where each bin is replaced by the mean of bin's values, and smoothing by bin medians where each bin is replaced by the median of bin's values.
- Regression: Linear Regression and multiple linear regression can be used to smooth the data, where the values are conformed to a function.
- Outlier Analysis: Approaches such as clustering can be used to detect outliers and deal with them.

## 2.3 NORMALIZATION

- **Standardization or Z-Score Normalization** is the transformation of features by subtracting from mean and dividing by standard deviation. This is often called as Z-score.

$$X_{\text{new}} = (X - \text{mean})/\text{Std}$$

Standardization can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. It translates the data to the mean vector of original data to the origin and squishes or expands the points if std is 1 respectively. We can see that we are just changing mean and standard deviation to a standard normal distribution which is still normal thus the shape of the distribution is not affected.

Standardization does not get affected by outliers because there is no predefined range of transformed features.

- Normalization or Min-Max Scaling is used to transform features to be on a similar scale. The new point is calculated as:

$$X_{\text{new}} = (X - X_{\text{min}}) / (X_{\text{max}} - X_{\text{min}})$$

This scales the range to [0, 1] or sometimes [-1, 1]. Geometrically speaking, transformation squishes the n-dimensional data into an n-dimensional unit hypercube. Normalization is useful when there are no outliers as it cannot cope up with them. Usually, we would scale age and not incomes because only a few people have high incomes but the age is close to uniform.

Standardisation			Max-Min Normalization		
	Age	Salary		Age	Salary
0	0.758874	7.494733e-01	0	0.739130	0.685714
1	-1.711504	-1.438178e+00	1	0.000000	0.000000
2	-1.275555	-8.912655e-01	2	0.130435	0.171429
3	-0.113024	-2.532004e-01	3	0.478261	0.371429
4	0.177609	6.632192e-16	4	0.565217	0.450794
5	-0.548973	-5.266569e-01	5	0.347826	0.285714
6	0.000000	-1.073570e+00	6	0.512077	0.114286
7	1.340140	1.387538e+00	7	0.913043	0.885714
8	1.630773	1.752147e+00	8	1.000000	1.000000
9	-0.258340	2.937125e-01	9	0.434783	0.542857

Figure 2.20 Standardisation vs Min-Max Normalization

## 2.4 FEATURE SELECTION

- **Filter method:**

In this method, features are filtered based on general characteristics (some metric such as correlation) of the dataset such correlation with the dependent variable. Filter method is performed without any predictive model. It is faster and usually the better approach when the number of features are huge. Avoids overfitting but sometimes may fail to select best features.

### ➤ **Wrapper method**

In wrapper method, the feature selection algorithm exists as a wrapper around the predictive model algorithm and uses the same model to select best features (more on this from this excellent research paper). Though computationally expensive and prone to overfitting, gives better performance.

### ➤ **Embedded method**

In embedded method, feature selection process is embedded in the learning or the model building phase. It is less computationally expensive than wrapper method and less prone to overfitting.

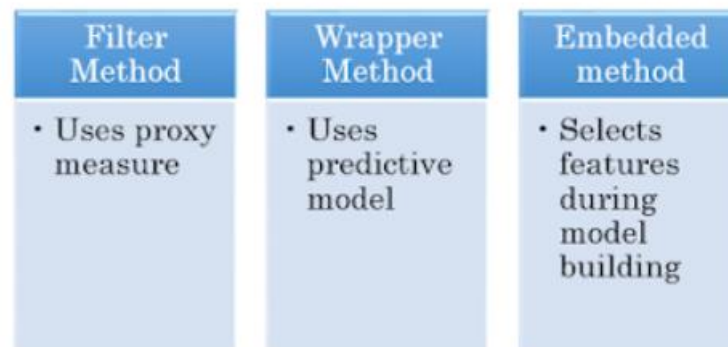


Figure 2.21 Filter Method vs Wrapper Method vs Embedded Method

## 2.5 **FEATURE EXTRACTION**

### 2.5.1 **Label Encoding :**

- **Label Encoding** refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

➤ Example:-

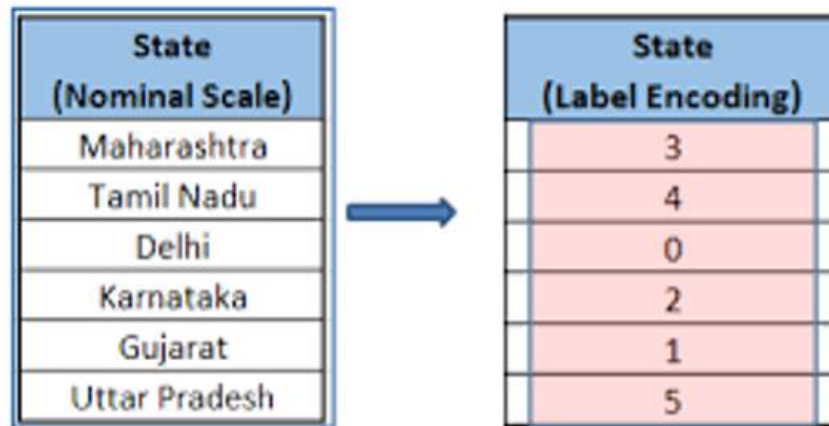


Figure 2.22 Label Encoding

### 2.5.2 TF-IDF Vectorizer :

- Bag of words (BoW) converts the text into a feature vector by counting the occurrence of words in a document. It is not considering the importance of words.
- **Term frequency — Inverse document frequency (TF-IDF)** is based on the Bag of Words (BoW) model, which contains insights about the less relevant and more relevant words in a document. The importance of a word in the text is of great significance in information retrieval.
- **Example** — If you search something on the search engine, with the help of TFIDF values, search engines can give us the most relevant documents related to our search.
- We will be discussing in detail how TFIDF can tell us which word is more important? We will first look into term frequency (TF) and inverse document frequency (IDF) separately and then combine it at the end.
- **Term Frequency (TF):**  
It is a measure of the frequency of a word (w) in a document (d). TF is defined as the ratio of a word's occurrence in a document to the total number of words in a document. The denominator term in the formula is to normalize since all the corpus documents are of different lengths.



➤ **EXAMPLE**

Documents	Text	Total number of words in a document
<b>A</b>	Jupiter is the largest planet	5
<b>B</b>	Mars is the fourth planet from the sun	8

Figure 2.23 TF-IDF Example

- The initial step is to make a vocabulary of unique words and calculate TF for each document. TF will be more for words that frequently appear in a document and less for rare words in a document.

Words	TF (for A)	TF (for B)
Jupiter	1/5	0
Is	1/5	1/8
The	1/5	2/8
largest	1/5	0
Planet	1/5	1/8
Mars	0	1/8
Fourth	0	1/8
From	0	1/8
Sun	0	1/8

Figure 2.24 Term Frequency

➤ **Inverse Document Frequency (IDF):-**

It is the measure of the importance of a word. Term frequency (TF) does not consider the importance of words. Some words such as 'of', 'and', etc. can be most frequently present but are of little significance. IDF provides weightage to each word based on its frequency in the corpus D.

- In our example, since we have two documents in the corpus,  $N=2$ .

Words	TF (for A)	TF (for B)	IDF
Jupiter	1/5	0	$\ln(2/1) = 0.69$
Is	1/5	1/8	$\ln(2/2) = 0$
The	1/5	2/8	$\ln(2/2) = 0$
largest	1/5	0	$\ln(2/1) = 0.69$
Planet	1/5	1/8	$\ln(2/2) = 0$
Mars	0	1/8	$\ln(2/1) = 0.69$
Fourth	0	1/8	$\ln(2/1) = 0.69$
From	0	1/8	$\ln(2/1) = 0.69$
Sun	0	1/8	$\ln(2/1) = 0.69$

Figure 2.25 Inverse Document Frequency

- After applying TF IDF, text in A and B documents can be represented as a TF IDF vector of dimension equal to the vocabulary words. The value corresponding to each word represents the importance of that word in a particular document.

Words	TF (for A)	TF (for B)	IDF	TFIDF (A)	TFIDF (B)
Jupiter	1/5	0	$\ln(2/1) = 0.69$	0.138	0
Is	1/5	1/8	$\ln(2/2) = 0$	0	0
The	1/5	2/8	$\ln(2/2) = 0$	0	0
largest	1/5	0	$\ln(2/1) = 0.69$	0.138	0
Planet	1/5	1/8	$\ln(2/2) = 0$	0.138	0
Mars	0	1/8	$\ln(2/1) = 0.69$	0	0.086
Fourth	0	1/8	$\ln(2/1) = 0.69$	0	0.086
From	0	1/8	$\ln(2/1) = 0.69$	0	0.086
Sun	0	1/8	$\ln(2/1) = 0.69$	0	0.086

Figure 2.26 TF-IDF Vectorizer

### 2.5.3 Sparse Matrix :

- If most of the elements of the matrix have **0 value**, then it is called a sparse matrix. The two major benefits of using sparse matrix instead of a simple matrix are:
- **Storage:** There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.
- **Computing time:** Computing time can be saved by logically designing a data structure traversing only non-zero elements.
- Sparse matrices are generally utilized in applied machine learning such as in data containing data-encodings that map categories to count and also in entire subfields of machine learning.

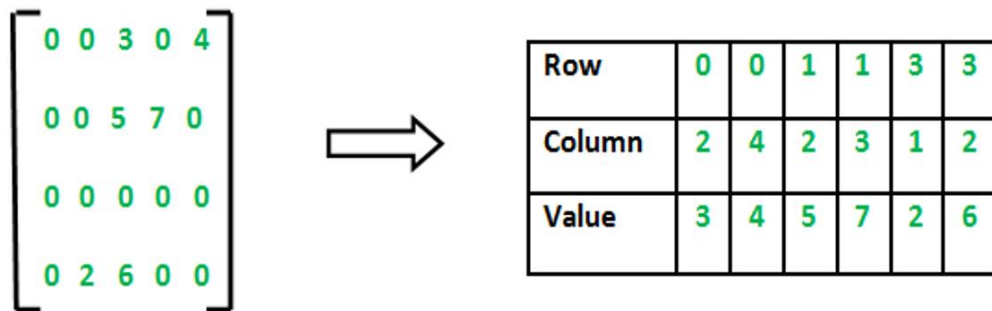


Figure 2.27 Sparse Matrix

## CHAPTER 3

### MODEL BUILDING

#### 3.1 K-NEAREST NEIGHBOR

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



Figure 3.1 KNN Classifier

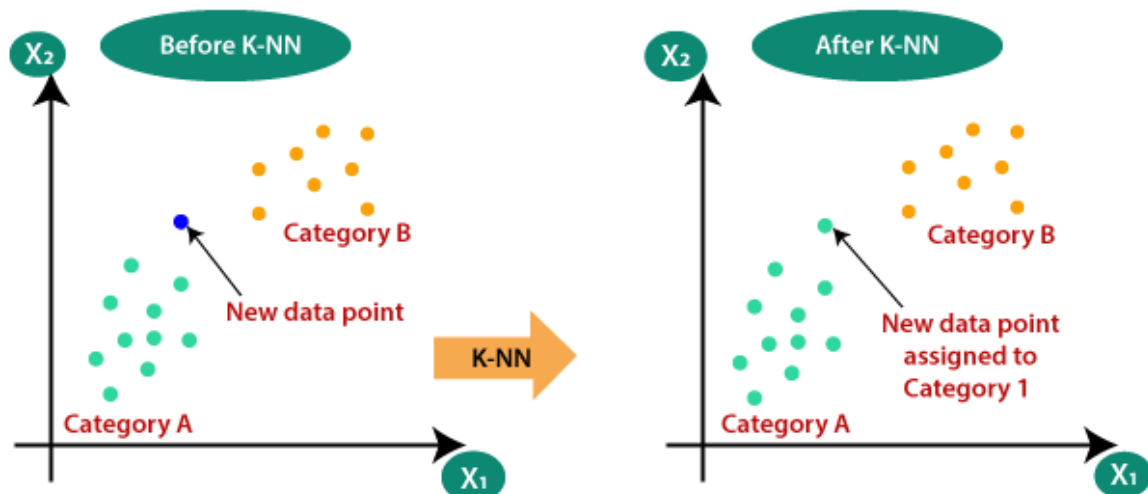


Figure 3.2 KNN Classifier Graph

### 3.2 COSINE SIMILARITY

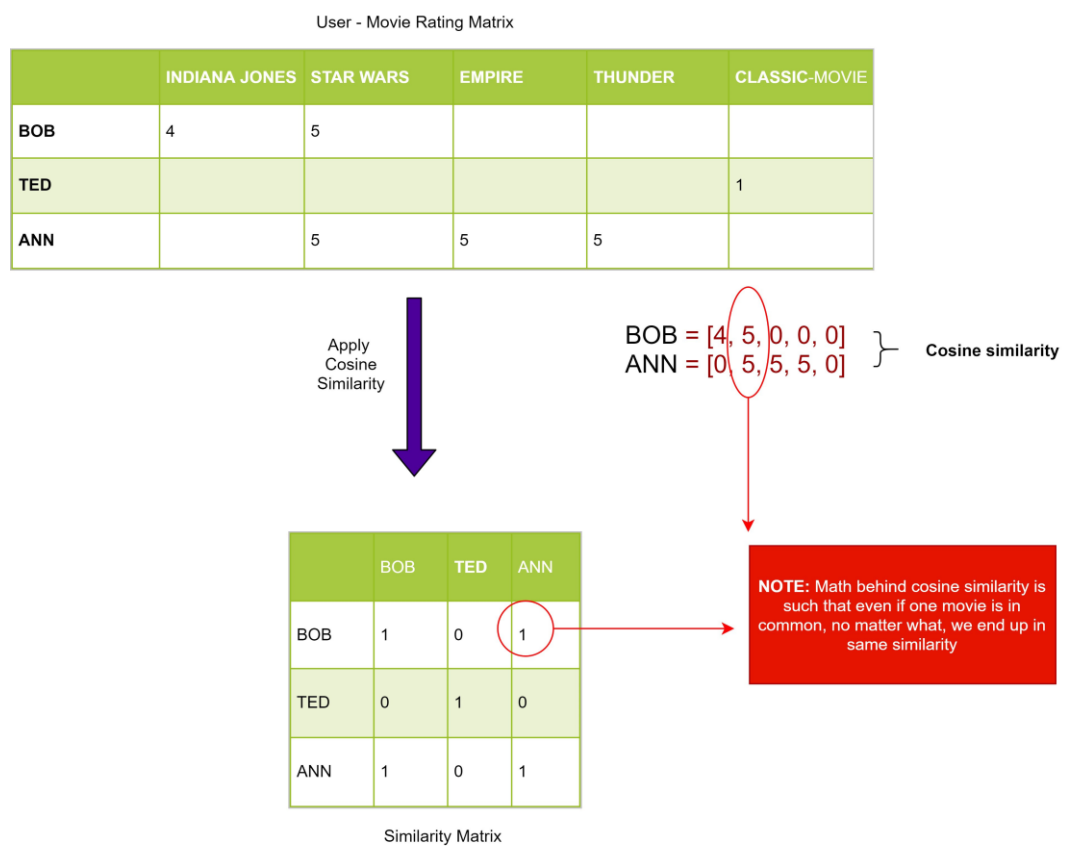
- **Item to Item Similarity:** The very first step is to build the model by finding similarity between all the item pairs. The similarity between item pairs can be found in different ways. One of the most common methods is to use cosine similarity.

$$I1 = 5U2 + 3U3 \text{ and,}$$

$$I2 = 2U2 + 3U3$$

$$\text{Similarity}(I1, I2) = \frac{(5*2)+(3*3)}{\sqrt{5^2+3^2}\sqrt{2^2+3^2}} = 0.90$$

Figure 3.3 Cosine Similarity Equation



You may reduce computation by removing doubles from similarity matrix!

Figure 3.4 Cosine Similarity

### 3.3 COLLABORATIVE FILTERING

- There are two classes of Collaborative Filtering:
- User-based Filtering:- which measures the similarity between target users and other users.

$$Sim(a, b) = \frac{\sum_p (r_{ap} - \bar{r}_a)(r_{bp} - \bar{r}_b)}{\sqrt{\sum_p (r_{ap} - \bar{r}_a)^2} \sqrt{\sum_p (r_{bp} - \bar{r}_b)^2}}$$

$r_{up}$  : rating of user  $u$  against item  $p$   
 $p$  : items

Figure 3.5 User-Based Filtering

- Item-based Filtering:- which measures the similarity between the item that target users rate or interact with and other items.

$$Similarity(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}|| * ||\vec{B}||}$$

Figure 3.6 Item-Based Filtering

- Collaborative filtering encompasses techniques for matching people with similar interests and making recommendations on this basis.



### 3.4 TESTING MODEL

- Train and test on the entire dataset. Train the model on the entire dataset. Test the model on the same dataset, and evaluate how well we did by comparing the predicted response values with the true response values.
- Instead, test model performance by doing monitoring, data slicing, or property-based testing targeted at real world problems.
- Combine this with test types that examine specifically the internal behavior of trained models (post-train tests):

- Invariance Test :

Invariance test defines input changes that are expected to leave model outputs unaffected.

The common method for testing invariance is related to data augmentation. We pair up modified and unmodified input examples and see how much this affects the model output.

- Directional Expectation Test :

We can run directional expectation tests to define input distribution changes' expected effects on the output.

A typical example is testing assumptions about the number of bathrooms or property size when predicting house prices. A higher number of bathrooms should mean a higher price prediction. Seeing a different result might reveal wrong assumptions about the relationship between our input and output or the distribution of our dataset.

- Minimum Functionality Test :

The minimum functionality test helps to decide whether individual model components behave as expect. The reasoning behind these tests is that overall, output-based performance can conceal critical upcoming issues in your model.

### 3.5 MATRIX FACTORIZATION

- Matrix decomposition methods, also called matrix factorization methods, are a foundation of linear algebra in computers, even for basic operations such as solving systems of linear equations, calculating the inverse, and calculating the determinant of a matrix.

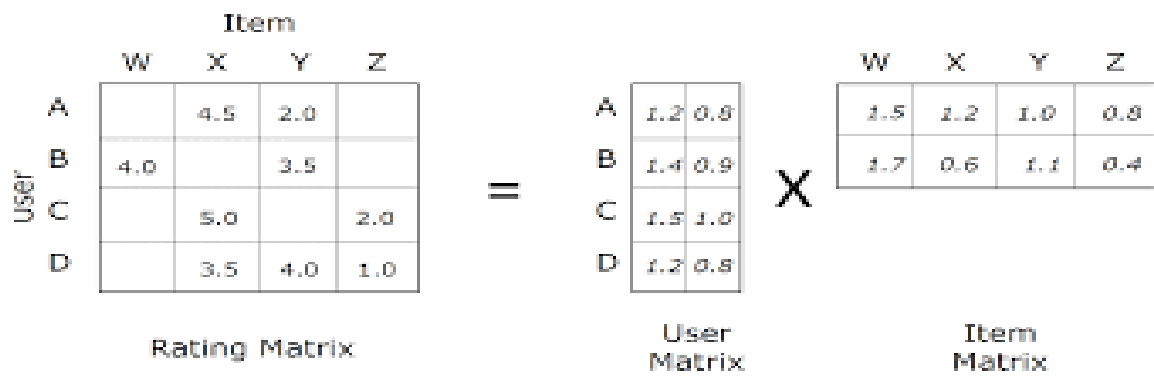


Figure 3.7 Matrix Factorization

## CHAPTER 4

### PRODUCT RECOMMENDATION SYSTEM

#### 4.1 LOAD DATASET

- Here we have 'final\_intern.csv' file as a dataset and we are performing different algorithms and analysis on this dataset.

- **Code:-**

```
df=pd.read_csv('final_intern.csv')
```

```
df.head()
```

**Output:-**

	id	name	rating	text	username
0	AV1l8zRZvKc47QAVhnAv	Olay Regenerist Deep Hydration Regenerating Cream	1	cream not much face throat area gone back gene...	An anonymous customer
1	AV1l8zRZvKc47QAVhnAv	Olay Regenerist Deep Hydration Regenerating Cream	4	wonderful product price reasonable product wor...	An anonymous customer
2	AV1l8zRZvKc47QAVhnAv	Olay Regenerist Deep Hydration Regenerating Cream	4	loved product took yrs blotches not noticeable...	An anonymous customer
3	AV1l8zRZvKc47QAVhnAv	Olay Regenerist Deep Hydration Regenerating Cream	4	great stuff face cuz old wrinkly makes look yo...	An anonymous customer
4	AV1l8zRZvKc47QAVhnAv	Olay Regenerist Deep Hydration Regenerating Cream	5	like rich dewy thick consistency great night c...	An anonymous customer

Figure 4.1 Output (1)

#### 4.2 CLAEAN THE TEXT USING STOP WORDS AND REGULAR EXPRESSION FROM REVIEW TEXT

- **Code:-**

```
with open('contractions.json', 'r') as f:
```

```
    contractions_dict = json.load(f)
```

```
contractions = contractions_dict
```

```
stop = stopwords.words('english')
```

```
new_stop=[]
```

```
not_removable_stop=['but','not','few','more','most','no','nor','very','too','same','only',  
                    '']
```

```

for e in stop:

    if e not in not_removable_stop:

        new_stop.append(e)

def cleaned_text(text):

    text = re.sub(r"\d+", " ", str(text))

    text = re.sub(r"\b[a-zA-Z]\b", "", str(text))

    text = re.sub(r"^\w\s", " ", str(text))

    text = re.sub(r"\s+", " ", str(text))

    text = text.lower()

    for word in text.split():

        if word.lower() in contractions:

            text = text.replace(word, contractions[word.lower()])

    return ' '.join([word for word in text.split() if word not in (new_stop)])

return text

df['cleaned_text'] = df['text'].apply(cleaned_text)

df.sample(5)

```

### Output:-

	id	name	rating	text	username	cleaned_text
570	AVpgNzjwLJeJML43Kpxn	AmazonBasics AAA Performance Alkaline Batterie...	5	work great	ByAmazon Customer	work great
1741	AVqklhkhv8e3D10-lebZ	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi...	4	like lot works well everything need would like...	Josh	like lot works well everything need would like...
1554	AVqVGWQDv8e3D10-lIdFr	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi...	5	son totally loves size right pretty fast compl...	Cindy	son totally loves size right pretty fast compl...
1091	AVpe7xlELJeJML43yplZ	AmazonBasics AA Performance Alkaline Batteries...	5	great batteries last long	ByMike	great batteries last long
1810	AVzNj1Y0GV-KLJ3aarCo	All-New Fire HD 8 Tablet with Alexa, 8 HD Disp...	5	purchased kindled past love ease use size	Jason	purchased kindled past love ease use size

Figure 4.2 Output (2)

### 4.3 GENERATE A NEW SCORE BY MERGING THE RATING AND SENTIMENT SCORE

- **Step 1: To find polarity of the sentiment using textblob.**

**Code:**

```
from textblob import TextBlob

def sentiment(text):

    try:

        return TextBlob(str(text)).sentiment.polarity

    except:

        return None

df['sentiment'] = df['cleaned_text'].apply(sentiment)

df
```

**Output:-**

	id	name	rating	text	username	cleaned_text	sentiment
872	AVpgNzjwLJeJML43Kpxn	AmazonBasics AAA Performance Alkaline Batterie...	5	last fairly long comparable batteries purchase...	ByAmazon Customer	last fairly long comparable batteries purchase...	-0.100000
2252	AVqVGWLKnn1JgDc3jF1	Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16...	5	daughter loves much faster last tablet	Dave	daughter loves much faster last tablet	0.100000
2256	AVqVGZSEQMlgsOJE6eUc	Kindle E-reader - White, 6 Glare-Free Touchscr...	4	item work well easy read day light	Bill	item work well easy read day light	0.416667
1976	AVpjEN4jLJeJML43rpUe	Fire Tablet with Alexa, 7 Display, 16 GB, Blue...	5	kids lot fun time games learning	Bobby	kids lot fun time games learning	0.300000
183	AVpf2tw1iIAPnD_xjflC	Red (special Edition) (dvdvideo)	5	movie very enjoyable recommended others	Anonymous	movie very enjoyable recommended others	0.650000

Figure 4.3 Output (3)

- **Step 2: Now we multiply rating and polarity to update score.**

**Code:**

```
df['updated_score'] = df['rating']*df['sentiment']
```

```
df.sample(5)
```

### Output:-

	id	name	rating	text	username	cleaned_text	sentiment	new_sentiment	updated_score
290	AVpgNzjwLJeJML43Kpxn	AmazonBasics AAA Performance Alkaline Batterie...	1	batteries sligtly bigger not fit electronics t...	ByAmazon Customer	batteries sligtly bigger not fit electronics t...	0.193333	positive	0.193333
2364	AVqklhxunnc1JgDc3kg_	Fire HD 8 Tablet with Alexa, 8 HD Display, 16 ...	4	easy use good size great ereader not expensive...	Chris	easy use good size great ereader not expensive...	0.545833	positive	2.183333
1479	AVqVGWQDv8e3D10-IdFr	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	4	item perfect online reading internet browsing ...	Nikki	item perfect online reading internet browsing ...	0.500000	positive	2.000000
1699	AVpgdkC8iIAPnD_xsvyi	Fire Tablet, 7 Display, Wi-Fi, 16 GB - Include...	5	good way get started kids play learn	Bobby	good way get started kids play learn	0.700000	positive	3.500000
1368	AVpe7xlELJeJML43ypLz	AmazonBasics AA Performance Alkaline Batteries...	1	worst batteries ever bought less one hour inst...	ByAmazon Customer	worst batteries ever bought less one hour inst...	-0.057792	negative	-0.057792

Figure 4.4 Output (4)

- **Step 3: Now we multiply rating and polarity to update score.**

### Code:

```
rating=[1,2,3,4,5]
```

```
sentiment = [-1,-0.75,-0.5,-0.25,0,0.25,0.5,0.75,1]
```

```
final = []
```

```
for i in rating:
```

```
    for j in sentiment:
```

```
        final.append(i*j)
```

```
len(np.unique(final))
```

```
a =np.unique(final)
```

```
a.sort()
```

```
dict1 = {'1':[-5,-4,-3.75,-3,-2.5,-2.25,-2,-1.5,-1.25,-1,-0.75,-0.5,-0.25],'2':[-0.24,0.25,0.5,0.75,1] , '3': [1.01,1.25,1.5,2],'4': [2.01,2.25,2.5,3],'5':[3.01,3.75,4,5]}
```

```
def new_score(val):
```

```
    for i in dict1:
```

```
        if val >= dict1[i][0] and val <= dict1[i][-1]:
```

```
            return int(i)
```

```
df['new_score'] = df['updated_score'].apply(new_score)
```

```
df.sample(5)
```

### Output:-

	id	name	rating	text	username	cleaned_text	sentiment	new_sentiment	updated_score	new_score
1268	AVpe7xIEJJeJML43ypLz	AmazonBasics AA Performance Alkaline Batteries...	1	like batteries but couple times putting opened...	ByAmazon Customer	like batteries but couple times putting opened...	0.000000	neutral	0.000000	2.0
1779	AVqVGZO3nnc1JgDc3jGK	Kindle Oasis E-reader with Leather Charging Co...	4	great reading sunlight well light easy wrist g...	Bobby	great reading sunlight well light easy wrist g...	0.449167	positive	1.796667	3.0
2116	AVph0EeEilAPnD_x9myq	Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16...	3	few problems games loading but good	Debbie	few problems games loading but good	0.250000	positive	0.750000	2.0
1760	AVpgdkC8ilAPnD_xsvyi	Fire Tablet, 7 Display, Wi-Fi, 16 GB - Include...	1	very cheap not impressed never	Tablet	very cheap not impressed never	0.010000	positive	0.010000	2.0
828	AVpgNzjwLJeJML43Kpxn	AmazonBasics AAA Performance Alkaline Batterie...	1	package duds lasted minutes get go trouble ret...	ByAmazon Customer	package duds lasted minutes get go trouble ret...	-0.200000	negative	-0.200000	2.0

Figure 4.5 Output (5)

## 4.4 FIND THE COSINE SIMILARITY BETWEEN PRODUCTS

### ➤ Step 1:- Build Pivot Table:-

**Code:**

```
df_pivot
=df.pivot_table(index='name',columns='username',values='new_score').fillna(0)

df_pivot
```

**Output:-**

username	0	1	2	3	4	5	6	7	8	9	...	78	79	80	81	82	83	84	85	86	87
name																					
0	0.000000	0.000000	0.00	0.000000	0.000000	2.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	3.000000	0.0	2.0	0.000000
1	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	3.0	0.0	0.000000
2	0.000000	0.000000	1.00	0.000000	0.000000	2.000000	0.000000	0.00	0.0	0.0	...	2.0	5.0	0.0	0.0	0.0	0.000000	0.000000	3.0	3.0	0.000000
3	5.000000	0.000000	0.00	3.500000	0.000000	3.000000	0.000000	2.50	0.0	1.0	...	4.0	0.0	2.0	2.5	0.0	0.000000	5.000000	5.0	0.0	0.000000
4	0.000000	0.000000	0.00	3.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	2.5	5.0	0.0	0.0	0.0	0.000000	0.000000	0.0	3.0	0.000000
5	3.500000	0.000000	2.00	3.000000	2.666667	2.000000	3.000000	3.00	3.5	4.0	...	4.0	4.0	3.5	2.0	0.0	3.000000	3.000000	0.0	3.0	0.000000
6	0.000000	0.000000	2.00	3.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
7	0.000000	0.000000	0.00	3.000000	0.000000	0.000000	0.000000	4.00	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	0.000000	0.000000	0.0	0.0	0.000000
8	0.000000	0.000000	0.00	3.000000	0.000000	0.000000	0.000000	4.00	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	0.000000	0.000000	0.0	0.0	0.000000
9	0.000000	0.000000	0.00	3.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	0.000000	0.000000	0.0	0.0	0.000000
10	0.000000	0.000000	0.00	5.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
11	0.000000	0.000000	0.00	0.000000	5.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	5.0	0.0	0.0	0.000000	0.000000	3.0	0.0	0.000000
12	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
13	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
14	0.000000	3.166667	0.00	0.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
15	0.000000	2.800000	0.00	0.000000	0.000000	0.000000	0.000000	0.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	3.500000
16	0.000000	0.000000	0.00	0.000000	0.000000	0.000000	0.000000	2.00	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
17	3.666667	0.000000	2.75	3.000000	2.500000	2.666667	3.000000	3.00	3.5	4.0	...	3.0	4.0	4.0	2.0	3.0	3.000000	2.666667	0.0	3.0	0.000000

Figure 4.6 Output (6)

➤ **Step 2:-Create Sparse Matrix****Code:**

```

from scipy.sparse import csr_matrix

df_pivot_matrix = csr_matrix(df_pivot.values)

print(df_pivot_matrix)

```



**Output:-**

(0, 5)	2.0
(0, 10)	1.0
(0, 39)	2.0
(0, 49)	3.0
(0, 84)	3.0
(0, 86)	2.0
(1, 10)	4.0
(1, 34)	2.0
(1, 36)	3.0
(1, 42)	3.0
(1, 46)	3.0
(1, 47)	4.0
(1, 53)	5.0
(1, 56)	2.0
(1, 57)	4.0
(1, 59)	2.0
(1, 64)	3.0
(1, 69)	2.0
(1, 70)	3.0
(1, 76)	4.0
(1, 85)	3.0
(2, 2)	1.0
(2, 5)	2.0
(2, 30)	4.0
(2, 34)	2.0
:	:
(36, 38)	5.0
(36, 40)	3.0
(36, 44)	3.0
(36, 56)	3.0
(36, 61)	3.0

Figure 4.7 Output (7)

**➤ Step 3:-Cosine Similarity Between Products:-****Code:**

```

from sklearn.metrics.pairwise import cosine_similarity

similarity_matrix = cosine_similarity(df_pivot)

similarity_matrix

```

**Output:**

```
array([[1.          , 0.05697451, 0.11940565, ..., 0.          , 0.          ,
        0.17065541],
       [0.05697451, 1.          , 0.46133372, ..., 0.19241861, 0.22508907,
        0.37990628],
       [0.11940565, 0.46133372, 1.          , ..., 0.50004955, 0.08324746,
        0.28821029],
       ...,
       [0.          , 0.19241861, 0.50004955, ..., 1.          , 0.12348086,
        0.15790651],
       [0.          , 0.22508907, 0.08324746, ..., 0.12348086, 1.          ,
        0.13880752],
       [0.17065541, 0.37990628, 0.28821029, ..., 0.15790651, 0.13880752,
        1.          ]])
```

Figure 4. 8 Output (8)

#### 4.5 FIND NEAREST NEIGHBORS AND THEIR DISTANCE VALUES WITH THEIR INDICES

➤ **Code to find nearest neighbours:**

```
from sklearn.neighbors import NearestNeighbors

model_knn = NearestNeighbors(metric='cosine',n_neighbors=20,radius=1)

model_knn.fit(df_pivot_matrix)
```

**Output:**

```
: NearestNeighbors(metric='cosine', n_neighbors=20, radius=1)
```

Figure 4.9 Output (9)

➤ **Code for distance values with their indices:**

```
product_id = (input("Enter Product Name:"))

n = int(input("How much recommendation you want:"))

data = df_pivot.index.to_list()

query_index = data.index(product_id)
```

```

distance,indices =
model_knn.kneighbors(df_pivot.iloc[query_index,:].values.reshape(1,-
1),n_neighbors = n+1)

print('distance values',d)

print('Indices',indices)

```

**Output:**


---

```

Enter Product Name:Fire Tablet with Alexa, 7 Display, 16 GB, Magenta - with Special Offers
How much recommendation you want:5
distance values ['0.00', '0.11', '0.12', '0.35', '0.46', '0.46']
Indices [[23 24 22 17  5 19]]

```

Figure 4.10 Output (10)

**4.6 BUILD A RECOMMENDATION ENGINE USING KNN****➤ Code :**

```

product_id = (input("Enter Product Name:"))

n = int(input("How much recommendation you want:"))

data = df_pivot.index.to_list()

query_index = data.index(product_id)

from sklearn.neighbors import NearestNeighbors

model_knn = NearestNeighbors(metric='cosine',n_neighbors=20,radius=1)

model_knn.fit(df_pivot_matrix)

distance,indices =
model_knn.kneighbors(df_pivot.iloc[query_index,:].values.reshape(1,-
1),n_neighbors = n+1)

distance = np.float32(distance)

i=df_pivot.index[indices.flatten()]

```

```

d=distance.flatten()

d= ['%.2f' % elem for elem in d]

new = list(zip(i,d))

final =pd.DataFrame(new)

final= final.iloc[1: ,:]

final.rename(columns={ 0: 'Recommended Products', 1: 'Distance'},inplace=True)

final

```

### Output :

Enter Product Name:Fire Tablet with Alexa, 7 Display, 16 GB, Magenta - with Special Offers  
How much recommendation you want:5

	Recommended Products	Distance
1	Fire Tablet, 7 Display, Wi-Fi, 16 GB - Include...	0.11
2	Fire Tablet with Alexa, 7 Display, 16 GB, Blue...	0.12
3	Fire HD 8 Tablet with Alexa, 8 HD Display, 16 ...	0.35
4	All-New Fire HD 8 Tablet, 8 HD Display, Wi-Fi,...	0.46
5	Fire Kids Edition Tablet, 7 Display, Wi-Fi, 16...	0.46

Figure 4.11 Output (11)

## **CHAPTER 5**

### **CONCLUSION**

Recommendation system helps customer to get right product to purchase in ecommerce websites like amazon and flipkart. This provides customers satisfaction of their purchase for that I use collaborative filtering based algorithm to get similar recommendation for similar user. So this is what the main intention to use recommendation system in ecommerce businesses to get up to date users with their data. Recommendation is not used in only product recommendation but also in widely in any platform to get users what they are find. like movie recommendation uses in such ott apps like Netflix and prime.

## REFERENCES

- [Machine Learning - GeeksforGeeks](#)
- [Python - Data Science Tutorial \(tutorialspoint.com\)](#)
- [Find Open Datasets and Machine Learning Projects | Kaggle](#)
- [Pandas - Python Data Analysis Library \(pydata.org\)](#)
- [Machine Learning Tutorial | Machine Learning with Python - Javatpoint](#)
- [Sklearn.neighbors.KNeighborsClassifier — scikit-learn 1.1.2 documentation](#)