**“EXTRACTION OF FASHION TRENDS FROM SOCIAL MEDIA USING SELENIUM”**

# 

# A project report

**Submitted in partial fulfillment of requirements to**

# 

# RGUKT - SRIKAKULAM

# For the award of the degree

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

## 4th year B. Tech (A.Y- 2019-2023)

## Submitted by

**P.Vinay Kumar [ S170528 ]**  **B.Devaki [ S170283 ]**

## M.Bhargavi [ S171051 ]

Under the Esteemed Guidance of

**Ms. Musidi Roopa, M. Tech,**

Assistant Professor

CSE Department



**Department of Computer Science and Engineering, RGUKT-SRIKAKULAM, ETCHERLA.**

## A.Y 2019-2023



**CERTIFICATE**

This is to certify that the thesis work titled “Extraction of Trends from Social Media using Selenium” was successfully completed by **P.Vinay Kumar(S170528), B.Devaki (S170283), M.Bhargavi (S171051),** in partial fulfillment of the requirements for the major project in Computer Science and Engineering of Rajiv Gandhi University of Knowledge Technologies under the guidance and output of the work carried out is satisfactory.

**Project Guide Project Co-ordinator Head of The Department**

---------------------------- -------------------------------- -------------------------------

**Ms. Musidi Roopa M.Tech Mr. Chilla Satish Kumar M.Tech Mr. N. Sesha Kumar M.Tech**

**Assistant Professor Assistant Professor Assistant Professor Department of CSE Department of CSE Department of CSE**

## DECLARATION

I declared that this thesis work titled **“**Extraction of Fashion Trends from Social Media using Selenium**”** is carried out by me during the year 2022-23 in partial fulfillment of the requirements for the Major Project in **Computer Science and Engineering.**

I further declare that this dissertation has not been submitted elsewhere for any Degree. The matter embodied in this dissertation report has not been submitted elsewhere for any other degree. Furthermore, the technical details furnished in various chapters of this thesis are purely relevant to the above project and there is no deviation from the theoretical point of view for design, development and implementation.

**P.Vinay Kumar - S170528  
 B.Devaki - S170283**

## M.Bhargavi - S171051

**ACKNOWLEDGEMENT**

I would like to articulate my profound gratitude and indebtedness to my project guide **Ms. M. Roopa M.Tech, ,** Assistant Professor who has always been a constant motivation and guiding factor throughout the project time. It has been a great pleasure for me to get an opportunity to work under his guidance and complete the thesis work successfully.

I wish to extend my sincere thanks to **N.SESHA KUMAR ,** Head of the Computer Science and Engineering Department, for her constant encouragement throughout the project.

I wish to extend my sincere thanks to **CH. Satish Kumar,** Project Coordinator Computer Science and Engineering Department, for her constant encouragement throughout the project.

I thank one and all who have rendered help to me directly or indirectly in the completion of my thesis work. I am also grateful to other members of the department without their support my work would have been carried out so successfully.

**Project Associate**

**P.Vinay Kumar - S170528  
 B.Devaki - S170283**

## M.Bhargavi - S171051

# 

# ABSTRACT

## “Extraction of Fashion Trends from Social Media using Automation Selenium”

Nowadays fashion trends are changing day by day, so identifying fashion trends is very important because trend analysis helps understand how a business has evolved in various areas, especially research areas such as marketing where current business activities and practices take place. So, all we need is a place where trends are available. Social Networks have become an important environment for collective trends extraction. Currently, the most relevant and popular Social Networks are Instagram, Twitter etc. These were created to share posts, comments and opinions and that can now use to extract trending products, which provide information on trending products that people are talking about, so the retailers can source them to meet their needs. In other words, if they offer a more popular product, a larger audience will be willing to buy it.

The extraction of fashion trends from social media typically involves using data mining and machine learning techniques to analyze user-generated content, such as photos, hashtags, and captions. Natural language processing (NLP) algorithms can be used to identify relevant keywords and phrases related to fashion and style, while computer vision techniques can be used to analyze images and identify patterns in clothing and accessories.

Once the data has been analyzed, fashion trend experts can use the insights gained to identify emerging fashion trends, popular styles, and changes in consumer preferences. This information can then be used by fashion retailers and designers to guide their product development, marketing strategies, and overall business decisions.

|  |  |  |
| --- | --- | --- |
|  | **CONTENTS** | **PG.NO** |
| **1** | **INTRODUCTION** | **1** |
|  | 1.1 Introduction | 1 |
|  | 1.2 Problem of the Statement | 1 |
|  | 1.3 Objectives | 1 |
|  | 1.4 Goals | 1 |
|  | 1.5 Scope | 2 |
|  | 1.6 Limitations | 2 |
| **2** | **LITERATURE SURVEY** | **3** |
|  | 2.1 Collecting Information | 3 |
|  | 2.2 Study | 4 |
|  | 2.3 Benefits | 4 |
|  | 2.4 Summary | 4 |
| **3** | **SYSTEM ANALYSIS** | **5** |
|  | 3.1 Existing System | 5 |
|  | 3.2 Disadvantages | 5 |
|  | 3.3 Proposed System | 5 |
|  | 3.4 Advantages | 5 |
|  | 3.5 System Requirements | 6 |
| **4** | **SYSTEM DESIGN** | **7** |
|  | 4.1 Class Diagram | 7 |
|  | 4.2 Use Case Diagram | 8 |
|  | 4.3 Sequence Diagram | 9 |
|  | 4.4 Data Flow Diagram | 10 |
| **5** | **SAMPLE SOURCECODE** | **12** |
| **6** | **SYSTEM TESTING** | **21** |
|  | 7.1 Testing Introduction | 21 |
|  | 7.2 Types Of Testing | 21 |
|  | 7.3 Levels Of Testing | 23 |
| **7** | **CONCLUSION** | **25** |
| **8** | **FUTURE ENHANCEMENT** | **26** |
| **9** | **REFERENCES** | **27** |

# Chapter - 1

# INTRODUCTION

## Introduction

## 

Nowadays trends are changing day by day, so identifying fashion trends is very important because trend analysis helps understand how a business has evolved in various areas, especially research areas such as marketing where current business activities and practices take place. So, all we need is a place where trends are available. Social Networks have become an important environment for collective trends extraction. The interactions amongst users provide information of their preferences and relationships. This information can be used to measure the influence of ideas, or opinions, and how they are spread within the Network.

## Statement of the problem

Social media platforms generate vast amounts of data every day, providing valuable insights into various trends and topics. However, extracting meaningful insights from this data can be challenging due to its unstructured nature and the sheer volume of information. Therefore, the problem statement is to develop an efficient and accurate method for extracting fashion trends from social media data in clothing category that can provide valuable insights for businesses, researchers, and individuals. This method should be able to process large amounts of data from different social media platforms, identify relevant topics, and provide insights on how these topics are evolving over time. Additionally, the method should be able to filter out noise and irrelevant information, allowing for more accurate trend analysis. The ultimate goal is to provide actionable insights to help users make informed decisions based on the trends identified in social media data.

## Objectives

* Process large amounts of unstructured social media data from various sources.
* Identify and filter out irrelevant information and noise to ensure accurate trend analysis.
* Utilize machine learning algorithms and statistical models to identify patterns and trends in the data.
* Visualize the trends and insights in an intuitive and easy-to-understand way.

## Goals

**1**. Develop an efficient and accurate method for extracting trends from social media data.

**2**. Improve the accuracy of trend analysis by filtering out noise and irrelevant information, allowing for more precise trend identification.

**3**. Continuously refine and improve the method to keep up with the rapidly changing social media landscape, ensuring that the insights provided remain relevant and useful to stakeholders.

## Scope

A fashion retailer wants to source ongoing and upcoming fashion trends from major online fashion portals and online magazines in a consumable and actionable format, so that they are able to effectively and efficiently design an upcoming fashion product portfolio.

Extracted trends are used to map with e-commerce websites like Flipkart, Amazon, etc. which provide information on trending products that people are talking about, so the retailers can source them to meet their needs. In other words, if they offer a more popular product, a larger audience will be willing to buy it.

It is also useful for the common people to know what is the trend going on to follow.

* 1. **Limitations**

1. Incomplete data: The project relies on social media data, which may not capture all relevant information about a particular topic or trend. Some users may not use social media, or their content may be inaccessible, resulting in incomplete data.

2. Inaccurate or irrelevant information: Social media platforms are vulnerable to spam, fake news, and disinformation campaigns, which can skew the analysis results. Additionally, irrelevant information, such as advertisements or personal posts, can also affect the accuracy of trend analysis.

3. Dynamic nature of social media: Social media trends and topics are constantly changing, and new trends can emerge quickly. It can be challenging to keep up with these changes, and the extracted trends and insights may become obsolete quickly.

# 

# Chapter 2

# LITERATURE SURVEY

## Collect Information

1. "Real-time trend analysis of social media data" by Khan et al. (2019): This paper proposes a real-time trend analysis system that uses machine learning algorithms to detect and track trends in social media data.

2. "Mining social media data: A survey of techniques and applications" by Aggarwal and Zhai (2012): This paper provides a general survey of various techniques used to mine social media data, including trend analysis.

## Benefits

1. Improved understanding of consumer behavior

2. Real-time tracking of trends

3. Enhanced customer engagement

## Summary

The extraction of trends from social media involves analyzing data from social media platforms to identify emerging patterns, preferences, and opinions. This can provide valuable insights that can help businesses to improve their marketing strategies, respond quickly to changing market conditions, identify emerging opportunities, and build stronger relationships with customers.

**CHAPTER - 3**

**ANALYSIS**

**Existing system**

* Google Trends is a website by Google that analyzes the popularity of top search queries in Google Search across various regions and languages. The website uses graphs to compare the search volume of different queries over time.
* Hootsuite Insights: Hootsuite Insights is a social media monitoring and analytics tool that provides real-time insights into social media trends. It allows users to track mentions, hashtags, and keywords across various social media platforms, including Twitter, Instagram, and Facebook.

## Disadvantages

* They provide trends of particular things over periods, based on google search results but not analyzing the social media data.
* HootSuite is not a free social analyzing tool.

## Proposed System

* In the Proposed system we can find images of trending products based on search keywords.
* We can also find the preferred number of most trending products of the searched results.

## Advantages

* Insights into customer behavior
* Competitive intelligence
* Cost-effective

## 3.5 System Requirements:

**Software Requirements:**

* Python
* Google Colab
* Jupyter Notebook
* Windows 10

**Hardware Requirements:**

* RAM: 4GB above
* Hard disk: 500 GB above

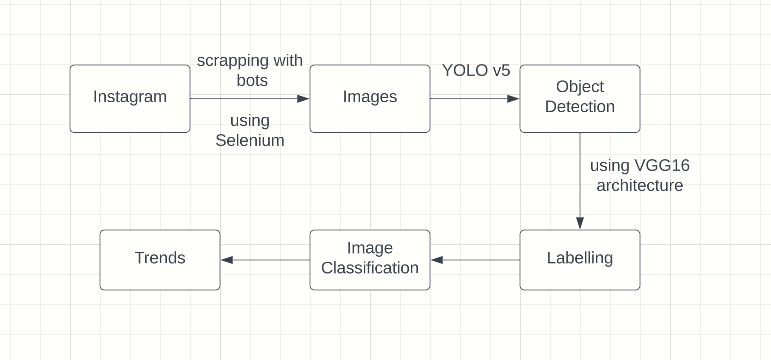
# 

# CHAPTER - 4

# SYSTEM DESIGN

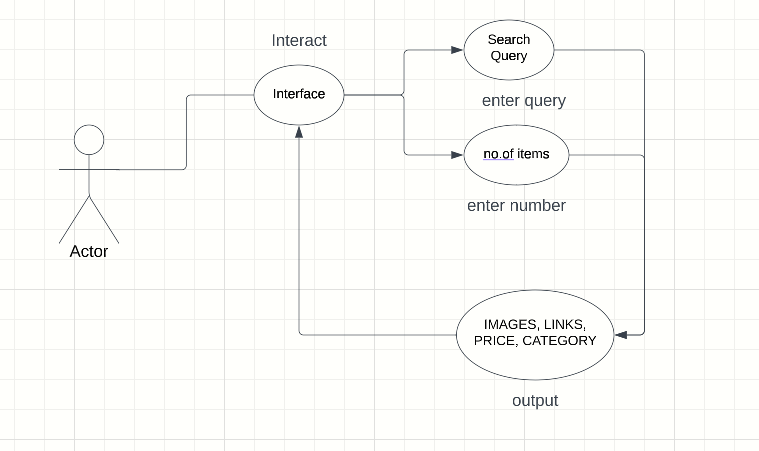
## 4.1 Class diagram:

Class diagram in the Unified Modelling Language (UML), is a kind of static structure diagram that describes the constitution of a process through showing the system's classes, their attributes, and the relationships between the class. The motive of a class diagram is to depict the classes within a model. In an object-oriented software, classes have attributes (member variables), operations (member capabilities) and relation.



**4.2 Use Case Diagram**

The use case diagram is a graphic depiction of the interactions among the elements int the system. It represents the methodology used in the system analysis, to identify, clarify and organize system requirement of the system. The main actors of the system are admin, warden and student. Who perform the different type of use cases such has manage hostels and stocks.



**CHAPTER -5**

**SAMPLE SOURCE CODE**

#### 5.1 Instagram Scrapping

#imports here

from selenium import webdriver

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.common.by import By

from selenium.webdriver.support.wait import WebDriverWait

import time

import pandas as pd

import os

import wget

driver = webdriver.Chrome(executable\_path="chromedriver")

#open the webpage

driver.get("http://www.instagram.com")

#target username

username = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "input[name='username']")))

password = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "input[name='password']")))

#enter username and password

username.clear()

username.send\_keys("9100769197")

password.clear()

password.send\_keys("devaki283")

#target the login button and click it

button = WebDriverWait(driver, 2).until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "button[type='submit']"))).click()

not\_now = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="x1i10hfl xjqpnuy xa49m3k xqeqjp1 x2hbi6w xdl72j9 x2lah0s xe8uvvx xdj266r x11i5rnm xat24cr x1mh8g0r x2lwn1j xeuugli x1hl2dhg xggy1nq x1ja2u2z x1t137rt x1q0g3np x1lku1pv x1a2a7pz x6s0dn4 xjyslct x1ejq31n xd10rxx x1sy0etr x17r0tee x9f619 x1ypdohk x1i0vuye xwhw2v2 xl56j7k x17ydfre x1f6kntn x2b8uid xlyipyv x87ps6o x14atkfc x1d5wrs8 x972fbf xcfux6l x1qhh985 xm0m39n xm3z3ea x1x8b98j x131883w x16mih1h xt0psk2 xt7dq6l xexx8yu x4uap5 x18d9i69 xkhd6sd x1n2onr6 xjbqb8w x1n5bzlp x173jzuc x1yc6y37"][contains(text(), "Not Now")]'))).click()

not\_now2 = WebDriverWait(driver, 10).until(EC.element\_to\_be\_clickable((By.XPATH, '//button[contains(text(), "Not Now")]'))).click()

import time

keyword = "fashion"

driver.get("https://www.instagram.com/explore/tags/" + keyword + "/")

# Wait for 5 seconds

time.sleep(5)

#scroll down to scrape more images

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

#target all images on the page

images = driver.find\_elements(By.TAG\_NAME, 'img')

images = [image.get\_attribute('src') for image in images]

images = images[:-2]

print('Number of scraped images: ', len(images))

import os

import wget

path = os.getcwd()

path = os.path.join(path, keyword + "s")

#create the directory

os.mkdir(path)

path

#download images

counter = 0

for image in images[3:]:

save\_as = os.path.join(path, keyword + str(counter) + '.jpg')

wget.download(image, save\_as)

counter += 1

**5.2 Blog Scrapping**

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.common.by import By

from selenium.webdriver.support.wait import WebDriverWait

import time

links = [

['pinterest-men','https://www.pinterest.com/Marcellthekid/men-fashion-catalog/'],

['pinterest-women',"https://www.pinterest.com/Kuricho/womens-t-shirts/"],

['vogue-fashion','https://www.vogue.in/fashion/fashion-trends']

]

option = 2

driver = webdriver.Chrome(executable\_path="chromedriver")

#open the webpage

driver.get(links[option][1])

driver.execute\_script("window.scrollTo(0, 4000);")

images = driver.find\_elements(by=By.TAG\_NAME, value='img')

images = [image.get\_attribute('src') for image in images]

images[:-2]

import os

import wget

path = os.getcwd()

counter = 0

for image in images:

save\_as = os.path.join(path+'/current-fashion', links[option][0] + str(counter) + '.jpg')

wget.download(image, save\_as)

counter += 1

**5.3 Object Detection using Yolov3**

! ls

!mkdir output

! git clone https://github.com/kritanjalijain/Clothing\_Detection\_YOLO.git

# Commented out IPython magic to ensure Python compatibility.

# %cd Clothing\_Detection\_YOLO

! git pull

# Commented out IPython magic to ensure Python compatibility.

# %cd yolo/weights

!gdown --id 1BaWJ6j5HGC136h6f4kl\_eo2LNPfjgyjq

# Commented out IPython magic to ensure Python compatibility.

# %cd ..

# Commented out IPython magic to ensure Python compatibility.

# %cd ..

! ls

from google.colab import drive

drive.mount('/content/MyDrive')

import torch

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

#from yolo.utils.utils import \*

from predictors.YOLOv3 import YOLOv3Predictor

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

torch.cuda.empty\_cache()

print(device)

#YOLO PARAMS

yolo\_modanet\_params = { "model\_def" : "yolo/modanetcfg/yolov3-modanet.cfg",

"weights\_path" : "yolo/weights/yolov3-modanet\_last.weights",

"class\_path":"yolo/modanetcfg/modanet.names",

"conf\_thres" : 0.5,

"nms\_thres" :0.4,

"img\_size" : 416,

"device" : device}

#DATASET

dataset = 'modanet'

yolo\_params = yolo\_modanet\_params

#Classes

def load\_classes(path):

fp = open(path, "r")

names = fp.read().split("\n")

return names

classes = load\_classes(yolo\_params["class\_path"])

#Colors

cmap = plt.get\_cmap("rainbow")

colors = np.array([cmap(i) for i in np.linspace(0, 1, 13)])

#np.random.shuffle(colors)

model = 'yolo'

detectron = YOLOv3Predictor(params=yolo\_params)

folder = "/content/MyDrive/MyDrive/multicategory/models" #input('input image directory')

parent\_dir = '/content/output'

images=[]

detections = []

#path = input('img path: ')

#if not os.path.exists(path):

# print('Img does not exists..')

# break#continue

for filename in os.listdir(folder):

path = os.path.join(folder,filename)

#print(path)

img = cv2.imread(path)

if img is not None:

images.append(img)

#print('image appended')

detections = detectron.get\_detections(img)

#print(detections)

#print(type(detections))

#print(type(images))

count = 1

if len(detections) != 0 :

detections.sort(reverse=False ,key = lambda x:x[4])

img\_id = path.split('/')[-1].split('.')[0]

print(img\_id)

for x1, y1, x2, y2, cls\_conf, cls\_pred in detections:

print("\t+ Label: %s, Conf: %.5f" % (classes[int(cls\_pred)], cls\_conf))

color = colors[int(cls\_pred)]

color = tuple(c\*255 for c in color)

color = (.7\*color[2],.7\*color[1],.7\*color[0])

font = cv2.FONT\_HERSHEY\_SIMPLEX

x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)

text = "%s conf: %.3f" % (classes[int(cls\_pred)] ,cls\_conf)

#print(img)

#print(y1, y2, x1, x2)

img\_crop = img[y1:y2, x1:x2]

#print(img\_crop)

if classes[int(cls\_pred)] in ['boots' , 'footwear']:

directory = 'footwear'#str(classes[int(cls\_pred)])

shoe\_dir\_path = os.path.join(parent\_dir, directory)

try:

os.mkdir(shoe\_dir\_path)

except OSError as error:

print("directory exists")

crop\_path = shoe\_dir\_path + "/" + str(img\_id) +

str(classes[int(cls\_pred)])+str(count)+ '.png'

count = count+1

elif classes[int(cls\_pred)] in ['pants', 'shorts','skirt' ]:

directory = 'bottomwear'#str(classes[int(cls\_pred)])

bottom\_dir\_path = os.path.join(parent\_dir, directory)

try:

os.mkdir(bottom\_dir\_path)

except OSError as error:

print("directory exists")

crop\_path = bottom\_dir\_path + "/" + str(img\_id) + str(classes[int(cls\_pred)])+ '.png'

#crop\_path = "output/cropped/bottomwear/" + str(img\_id) + str(classes[int(cls\_pred)])+ '.jpg'

elif classes[int(cls\_pred)] in ['top', 'outer']:

directory = 'topwear'#str(classes[int(cls\_pred)])

top\_dir\_path = os.path.join(parent\_dir, directory)

try:

os.mkdir(top\_dir\_path)

except OSError as error:

print("directory exists")

crop\_path = top\_dir\_path + "/" + str(img\_id) + str(classes[int(cls\_pred)])+ '.png'

#crop\_path = "output/cropped/upper/" + str(img\_id) + str(classes[int(cls\_pred)])+ '.jpg'

else:

directory = str(classes[int(cls\_pred)])

new\_dir\_path = os.path.join(parent\_dir, directory)

try:

os.mkdir(new\_dir\_path)

except OSError as error:

print("directory exists")

crop\_path = new\_dir\_path + "/" + str(img\_id) + str(classes[int(cls\_pred)])+ '.png'

if((x1 > 0) & (x2 > 0) & (y1 > 0) & (y2 > 0)):

cv2.imwrite(crop\_path,img\_crop)

cv2.rectangle(img.copy(),(x1,y1) , (x2,y2) , color,3)

y1 = 0 if y1<0 else y1

y1\_rect = y1-25

y1\_text = y1-5

if y1\_rect<0:

y1\_rect = y1+27

y1\_text = y1+20

#break

print('Output saved')

print('End inner loop')

#break

#print("end of if loop")

print("End of while loop")

#zipping output folder to save

!zip -r /content/output.zip /content/output

import cv2 as cv

from google.colab.patches import cv2\_imshow # for image display

import torch

import os

import cv2

from yolo.utils.utils import \*

from predictors.YOLOv3 import YOLOv3Predictor

import glob

from tqdm import tqdm

import sys

import uuid

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

torch.cuda.empty\_cache()

print(device)

#YOLO PARAMS

yolo\_modanet\_params = { "model\_def" : "yolo/modanetcfg/yolov3-modanet.cfg",

"weights\_path" : "yolo/weights/yolov3-modanet\_last.weights",

"class\_path":"yolo/modanetcfg/modanet.names",

"conf\_thres" : 0.5,

"nms\_thres" :0.4,

"img\_size" : 416,

"device" : device}

#DATASET

dataset = 'modanet' #'df2'

yolo\_params = yolo\_modanet\_params

#Classes

classes = load\_classes(yolo\_params["class\_path"])

#Colors

cmap = plt.get\_cmap("rainbow")

colors = np.array([cmap(i) for i in np.linspace(0, 1, 13)])

#np.random.shuffle(colors)

model = 'yolo'

detectron = YOLOv3Predictor(params=yolo\_params)

while(True):

path = input('img path: ')

if not os.path.exists(path):

print('Img does not exists..')

continue

img = cv2.imread(path)

detections = detectron.get\_detections(img)

#detections = yolo.get\_detections(img)

#print(detections)

if len(detections) != 0 :

detections.sort(reverse=False ,key = lambda x:x[4])

for x1, y1, x2, y2, cls\_conf, cls\_pred in detections:

print("\t+ Label: %s, Conf: %.5f" % (classes[int(cls\_pred)], cls\_conf))

#color = bbox\_colors[np.where(unique\_labels == cls\_pred)[0]][0]

color = colors[int(cls\_pred)]

color = tuple(c\*255 for c in color)

color = (.7\*color[2],.7\*color[1],.7\*color[0])

font = cv2.FONT\_HERSHEY\_SIMPLEX

x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)

text = "%s conf: %.3f" % (classes[int(cls\_pred)] ,cls\_conf)

cv2.rectangle(img,(x1,y1) , (x2,y2) , color,3)

y1 = 0 if y1<0 else y1

y1\_rect = y1-25

y1\_text = y1-5

if y1\_rect<0:

y1\_rect = y1+27

y1\_text = y1+20

cv2.rectangle(img,(x1-2,y1\_rect) , (x1 + int(8.5\*len(text)),y1) , color,-1)

cv2.putText(img,text,(x1,y1\_text), font, 0.5,(255,255,255),1,cv2.LINE\_AA)

img\_id = path.split('/')[-1].split('.')[0]

cv2\_imshow(img)

print('End inner loop')

break

print("End of while loop")

**5.4 Google Image Scraping**

#Import libraries

import os

import concurrent.futures

from GoogleImageScraper import GoogleImageScraper

from patch import webdriver\_executable

def worker\_thread(search\_key):

image\_scraper = GoogleImageScraper(

webdriver\_path, image\_path, search\_key, number\_of\_images, headless, min\_resolution, max\_resolution)

image\_urls = image\_scraper.find\_image\_urls()

image\_scraper.save\_images(image\_urls, keep\_filenames)

#Release resources

del image\_scraper

if \_\_name\_\_ == "\_\_main\_\_":

#Define file path

webdriver\_path = os.path.normpath(os.path.join(os.getcwd(), 'webdriver', webdriver\_executable()))

image\_path = os.path.normpath(os.path.join(os.getcwd(), 'photos'))

#Add new search key into array ["cat","t-shirt","apple","orange","pear","fish"]

search\_keys = list(set(["Blazer", "Collar Shirt", "Designer saree", "Hoodies", "Jacket", "Jean pant", "t-shirt", "shorts", "oversized Tshirt", "striped shirt", "wide leg pants"]))

#Parameters

number\_of\_images = 50 # Desired number of images

headless = True # True = No Chrome GUI

min\_resolution = (0, 0) # Minimum desired image resolution

max\_resolution = (9999, 9999) # Maximum desired image resolution

max\_missed = 1000 # Max number of failed images before exit

number\_of\_workers = 1 # Number of "workers" used

keep\_filenames = False # Keep original URL image filenames

#Run each search\_key in a separate thread

#Automatically waits for all threads to finish

#Removes duplicate strings from search\_keys

with concurrent.futures.ThreadPoolExecutor(max\_workers=number\_of\_workers) as executor:

executor.map(worker\_thread, search\_keys)

**5.5 Labelling using VGG-16**

import os

import matplotlib.pyplot as plt

import matplotlib.image as img

import tensorflow.keras as keras

import numpy as np

import tensorflow as tf

config = tf.compat.v1.ConfigProto()

config.gpu\_options.allow\_growth = True

tf.compat.v1.keras.backend.set\_session(tf.compat.v1.Session(config=config))

import cv2

from google.colab import drive

drive.mount('/content/drive')

datasetdir = r'/content/drive/MyDrive/Google-Image-Scraper-master/photos'

os.chdir(datasetdir)

from tensorflow.keras.preprocessing.image import ImageDataGenerator

batch\_size = 3

def DataLoad(shape, preprocessing):

'''Create the training and validation datasets for

a given image shape.

'''

imgdatagen = ImageDataGenerator(

preprocessing\_function = preprocessing,

horizontal\_flip = True,

validation\_split = 0.1,

)

height, width = shape

train\_dataset = imgdatagen.flow\_from\_directory(

os.getcwd(),

target\_size = (height, width),

classes = ['Blazer','Collar shirt','Designer

saree','Hoodies',

'Jacket','jean pant','Oversized TShirt','shorts', 'Striped Shirt','Tshirt','Wide leg pants'],

batch\_size = batch\_size,

subset = 'training',

)

val\_dataset = imgdatagen.flow\_from\_directory(

os.getcwd(),

target\_size = (height, width),

classes = ['Blazer','Collar shirt','Designer saree','Hoodies',

'Jacket','jean pant','Oversized TShirt','shorts', 'Striped Shirt','Tshirt','Wide leg pants'],

batch\_size = batch\_size,

subset = 'validation'

)

return train\_dataset, val\_dataset

vgg16 = keras.applications.vgg16

conv\_model = vgg16.VGG16(weights='imagenet', include\_top=False)

train\_dataset, val\_dataset = DataLoad((224,224), preprocessing=vgg16.preprocess\_input)

X\_train, y\_train = next(train\_dataset)

conv\_model = vgg16.VGG16(weights='imagenet', include\_top=False, input\_shape=(224,224,3))

# flatten the output of the convolutional part:

x = keras.layers.Flatten()(conv\_model.output)

# three hidden layers

x = keras.layers.Dense(100, activation='relu')(x)

x = keras.layers.Dense(100, activation='relu')(x)

x = keras.layers.Dense(100, activation='relu')(x)

# final softmax layer with 15 categories

predictions = keras.layers.Dense(11, activation='softmax')(x)

# creating the full model:

full\_model = keras.models.Model(inputs=conv\_model.input, outputs=predictions)

full\_model.summary()

for layer in conv\_model.layers:

layer.trainable = False

full\_model.compile(loss='categorical\_crossentropy',

optimizer=keras.optimizers.Adamax(lr=0.001),

metrics=['acc'])

from tensorflow.keras.preprocessing.image import ImageDataGenerator

test\_dir=r'/content/drive/MyDrive/Google-Image-Scraper-master/photos'

test\_datagen = ImageDataGenerator()

test\_generator = test\_datagen.flow\_from\_directory(test\_dir, target\_size=(224, 224), batch\_size=3, class\_mode='categorical')

# X\_test, y\_test = next(test\_generator)

history = full\_model.fit\_generator(

train\_dataset,

validation\_data = val\_dataset,

workers=0,

epochs=3,)

from keras.preprocessing import image

from keras.applications.vgg16 import preprocess\_input

from tensorflow.keras.preprocessing import image

from tensorflow.keras.applications.vgg16 import preprocess\_input

img\_path = r'/content/drive/MyDrive/Google-Image-Scraper-master/Test Images/4.jpg'

img = image.load\_img(img\_path, target\_size=(224,224))

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

x = preprocess\_input(x)

plt.imshow(img)

img = image.load\_img(img\_path1, target\_size=(224,224))

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

x = preprocess\_input(x)

def get\_class\_string\_from\_index(index):

for class\_string, class\_index in test\_generator.class\_indices.items():

if class\_index == index:

return class\_string

preds = full\_model.predict(x)

pred\_labels = np.argmax(preds, axis=1)

print('Predicted\_Class is:', pred\_labels) #Get the rounded value of the predicted class

true\_index = 5

# print('true\_label is:', true\_labels) #Get the rounded value of the predicted class

print("Predicted label: " + get\_class\_string\_from\_index(pred\_labels))

**5.6 Trending Score**

# pip install pytrends

from pytrends.request import TrendReq

pytrends = TrendReq(hl="en-US", tz=330)

kw\_list = ['blazers', 'Striped shirt', 'OverSized', 'Designer Saree'] # list of keywords to get data

pytrends.build\_payload(kw\_list, cat=0, timeframe='now 7-d')

#1 Interest over Time

data = pytrends.interest\_over\_time()

data = data.reset\_index()

import plotly.express as px

fig = px.line(data, x="date", y=['blazers', 'Striped shirt', 'OverSized', 'Designer Saree'],

title='Keyword Web Search Interest Over Time')

fig.show()

d1 = data.mean()

import pandas as pd

data1 = pd.DataFrame.from\_dict(d1)

data1.columns = ["Score"]

data1

type(d1)

data1["Score"] = (data1["Score"] - data1["Score"].min(axis=0)) / (

data1["Score"].max(axis=0) - data1["Score"].min(axis=0))

data1["Score"] = data1["Score"] \* (5 - 3) + 3

data1

pytrends.build\_payload(kw\_list=['fashion india'])

related\_queries = pytrends.related\_queries()

list(related\_queries.values())[0]

**5.6 Trending Score**

# <script async src="https://cse.google.com/cse.js?cx=917788237b53c4e42">

# </script>

# <div class="gcse-search"></div>

# AIzaSyA9mYLE9-r-fgEN1aJgr3ztXyPUdlaCsJw ->API KEY

import requests

import json

# Define the API endpoint and parameters

endpoint = "https://www.googleapis.com/customsearch/v1"

params = {

"q": "blazers OR Striped shirt OR OverSized OR Designer Saree",

"cx": "917788237b53c4e42",

"key": "AIzaSyA9mYLE9-r-fgEN1aJgr3ztXyPUdlaCsJw",

"num": 10,

"start": 1,

}

# Make the API request

response = requests.get(endpoint, params=params)

# Parse the response data

json\_data = json.loads(response.text)

#print(json\_data)

# Extract the search interest data from the response

search\_interest\_data = {}

for item in json\_data["items"]:

name = item["title"]

link = item["link"]

snippet = item["snippet"]

search\_interest\_data[name] = {"link": link, "snippet": snippet}

# Print the search interest data

print(search\_interest\_data, end='\n')

import requests

import json

import pandas as pd

import plotly.express as px

# Define your API key and search engine ID

API\_KEY = 'AIzaSyA9mYLE9-r-fgEN1aJgr3ztXyPUdlaCsJw'

SEARCH\_ENGINE\_ID = '917788237b53c4e42'

# Define your list of keywords

kw\_list = ['blazers', 'Striped shirt', 'OverSized', 'Designer Saree']

# Define the time period for the search

timeframe = 'today 7-d'

# Define the base URL for the API request

base\_url = f"https://www.googleapis.com/customsearch/v1?key={API\_KEY}&cx={SEARCH\_ENGINE\_ID}&q="

# Make the requests and retrieve the search data for each keyword

data = []

for kw in kw\_list:

url = base\_url + kw.replace(' ', '+') + f'&dateRestrict={timeframe}'

response = requests.get(url)

if response.status\_code == 200:

results = json.loads(response.text)['queries']['request'][0]['totalResults']

data.append({'Keyword': kw, 'Searches': int(results)})

# Convert the search data into a pandas dataframe

data\_df = pd.DataFrame(data)

# Normalize the search data

data\_df['Normalized Searches'] = (data\_df['Searches'] - data\_df['Searches'].min()) / (data\_df['Searches'].max() - data\_df['Searches'].min())

data\_df['Normalized Searches'] = data\_df['Normalized Searches'] \* (5 - 3) + 3

# Plot the search data using plotly

fig = px.bar(data\_df, x='Keyword', y='Normalized Searches', title='Keyword Web Search Interest Over Time')

fig.show()

**5.7 Website**

import pandas as pd

import regex as re

from operator import itemgetter

import string

import urllib

import matplotlib.pyplot as plt

from matplotlib.pyplot import figure

from fuzzywuzzy import fuzz

from nltk.corpus import stopwords

import nltk

from nltk.corpus import wordnet

from nltk.tokenize import word\_tokenize

from nltk.stem import WordNetLemmatizer

data = pd.read\_json('flipkart\_fashion\_products\_dataset.json')

subdata = pd.read\_csv('flipkart\_com-ecommerce\_sample.csv')

def for\_fun(some\_text):

some\_text = str(some\_text)

some\_text = some\_text.lower()

if ('"women') in some\_text:

return 'women'

elif ('"men') in some\_text:

return 'men'

subdata['gender'] = subdata['product\_specifications'].apply(for\_fun)

testsub = subdata[subdata['gender'].isin(['men', 'women'])]

testsub = testsub[

['brand', 'description', 'product\_specifications', 'product\_name', 'product\_url', 'product\_category\_tree',

'gender']]

def for\_data\_fun(some\_text):

some\_text = str(some\_text)

some\_text = some\_text.lower()

if ('-women-') in some\_text and '-men-' in some\_text:

return 'both'

elif ('-women-') in some\_text:

return 'women'

elif ('-men-') in some\_text:

return 'men'

data['gender'] = data['url'].apply(for\_data\_fun)

main\_df = data.drop\_duplicates(subset=['title'], keep='first')

testing = main\_df.loc[~main\_df['gender'].isin(['men', 'both', 'women'])]

def something\_doing(some\_text):

some\_text = str(some\_text)

some\_text = some\_text.lower()

if ('women') in some\_text or ('woman') in some\_text:

return 'women'

elif ('men') in some\_text or ('man') in some\_text:

return 'men'

return None

testing['gender'] = testing['product\_details'].apply(something\_doing)

final\_testing = testing.loc[~testing['gender'].isin(['men', 'both', 'women'])]

final\_testing['gender'] = final\_testing['description'].apply(something\_doing)

final\_testing = final\_testing.dropna(axis=0, subset=['gender'])

testing = testing.dropna(axis=0, subset=['gender'])

main\_df = main\_df.dropna(axis=0, subset=['gender'])

dfs = [main\_df, testing, final\_testing]

merged1 = pd.concat(dfs)

merged1.reset\_index(inplace=True)

main\_df = subdata.drop\_duplicates(subset=['product\_name'], keep='first')

testing = main\_df.loc[~main\_df['gender'].isin(['men', 'both', 'women'])]

def something\_doing(some\_text):

some\_text = str(some\_text)

some\_text = some\_text.lower()

if (('-women-') in some\_text or ('-woman-') in some\_text) and (('-men-') in some\_text or ('-man-') in some\_text):

return 'both'

elif ('-men-') in some\_text or ('-man-') in some\_text:

return 'men'

elif ('-women-') in some\_text or ('-woman-') in some\_text:

return 'women'

return None

testing['gender'] = testing['product\_url'].apply(something\_doing)

final\_testing = testing.loc[~testing['gender'].isin(['men', 'both', 'women'])]

def something\_doing(some\_text):

some\_text = str(some\_text)

some\_text = some\_text.lower()

some\_text = some\_text.split(' ')

if ('woman') in some\_text or ('women') in some\_text:

return 'women'

elif ('man') in some\_text or ('men') in some\_text:

return 'men'

return None

final\_testing['gender'] = final\_testing['description'].apply(something\_doing)

final\_testing = final\_testing.dropna(axis=0, subset=['gender'])

testing = testing.dropna(axis=0, subset=['gender'])

main\_df = main\_df.dropna(axis=0, subset=['gender'])

dfs = [main\_df, testing, final\_testing]

merged2 = pd.concat(dfs)

merged2.reset\_index(inplace=True)

merged2.rename(columns={'product\_name': 'title', 'product\_category\_tree': 'category'}, inplace=True)

drop\_columns = ['\_id', 'actual\_price', 'average\_rating', 'crawled\_at', 'discount', 'images', 'out\_of\_stock', 'pid',

'selling\_price']

data.drop(drop\_columns, axis=1, inplace=True)

def remove\_irrelevant(some\_text):

return re.sub('[^a-zA-Z]', ' ', some\_text)

def make\_lowercase(some\_text):

some\_text = str(some\_text).lower()

if 't shirt' in some\_text:

some\_text = some\_text.replace('t shirt', 'tshirt')

return some\_text

nltk.download('punkt')

def do\_tokenization(some\_text):

return word\_tokenize(some\_text)

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

def remove\_stopwords(token):

return [item for item in token if item not in stop\_words]

nltk.download('wordnet')

nltk.download('omw-1.4')

lemma = WordNetLemmatizer()

def do\_lemmatization(token):

return [lemma.lemmatize(word=w, pos='v') for w in token]

def remove\_small(token):

return [i for i in token if len(i) > 2]

def convert\_to\_string(token):

return ' '.join(token)

def do\_cleaning():

merged1['ntitle'] = merged1['title'].apply(remove\_irrelevant)

merged2['ntitle'] = merged2['title'].apply(remove\_irrelevant)

merged1['ntitle'] = merged1['ntitle'].apply(make\_lowercase)

merged2['ntitle'] = merged2['ntitle'].apply(make\_lowercase)

merged1['ntitle'] = merged1['ntitle'].apply(do\_tokenization)

merged2['ntitle'] = merged2['ntitle'].apply(do\_tokenization)

merged1['ntitle'] = merged1['ntitle'].apply(remove\_stopwords)

merged2['ntitle'] = merged2['ntitle'].apply(remove\_stopwords)

merged1['ntitle'] = merged1['ntitle'].apply(do\_lemmatization)

merged2['ntitle'] = merged2['ntitle'].apply(do\_lemmatization)

merged1['ntitle'] = merged1['ntitle'].apply(remove\_small)

merged2['ntitle'] = merged2['ntitle'].apply(remove\_small)

merged1['ntitle'] = merged1['ntitle'].apply(convert\_to\_string)

merged2['ntitle'] = merged2['ntitle'].apply(convert\_to\_string)

def document\_cleaning():

document\_list['title'] = document\_list['title'].apply(remove\_irrelevant)

document\_list['title'] = document\_list['title'].apply(make\_lowercase)

document\_list['title'] = document\_list['title'].apply(do\_tokenization)

document\_list['title'] = document\_list['title'].apply(remove\_stopwords)

document\_list['title'] = document\_list['title'].apply(do\_lemmatization)

document\_list['title'] = document\_list['title'].apply(remove\_small)

document\_list['title'] = document\_list['title'].apply(convert\_to\_string)

do\_cleaning()

merged1.drop('index', axis=1, inplace=True)

merged2.drop('index', axis=1, inplace=True)

my\_data = [merged1['title'], merged2['title']]

documents = pd.concat(my\_data)

documents = documents.drop\_duplicates()

document\_list = pd.DataFrame(documents)

document\_cleaning()

document\_list.reset\_index(inplace=True)

document\_list.drop('index', axis=1, inplace=True)

final\_result = []

for i in document\_list.title:

lst = i.split(' ')

result = []

for word in lst:

my\_word = wordnet.synsets(word)

if len(my\_word) == 0:

result.append(word)

continue

for inner\_word in my\_word:

for x in inner\_word.lemma\_names():

result.append(x)

final\_result.append(' '.join(result))

clothing\_types = ['Outer wear', 'Active wear', 'Swimwear', 'Legwear', 'Neckwear', 'Shawls', 'Bridal', 'Pant', 'Undergarments', 'Uniform', 'Knit', 'Lounge', 'Shirts', 'Blouses', 'Bottoms', 'Sleep', 'Resort', 'Mourning', 'Afternoon', 'Afghan', 'line coat', 'line skirt', 'Apron', 'Ankle length Pants', 'Anklet', 'Apache Tie', 'Armband', 'Babushka', 'Baby Bonnet', 'Baby bunting', 'Baby doll dress', 'Baggy pants', 'Ball Gown', 'Balmacaan', 'Bandana', 'Ballerina skirt', 'Barfly apparel', 'Barn jacket', 'Basque waistline', 'Barrow coat', 'Bathing cap', 'Bathing suit', 'Bathrobe', 'Beach Pajamas', 'Bed Jacket', 'Bell bottom pants', 'Bell Boy Jacket', 'Belt', 'Bermuda Shorts', 'Bertha collar', 'Bias Slip', 'Bias skirt', 'Bike Tards', 'Bikini', 'Bishop sleeve', 'Bisht', 'Blazer', 'Bloomers', 'Blouse', 'Blouson', 'Bodycon dress',

'Bomber jacket', 'Bootcut', 'Boots', 'Bow Tie', 'Boxers', 'Boyshorts', 'Bow tie', 'Bra', 'Brassiere', 'KnickerSet', 'Bralette', 'Briefs', 'Burkha', 'Business suit', 'Bustier', 'Bustle', 'Button down shirt', 'Cap', 'Cape', 'Capri Pants', 'Cardigan', 'Cargos', 'Car coat', 'Circle skirt', 'Cloche', 'Cloak', 'Cocktail dress', 'Corset', 'Crew neck', 'Crop top', 'Cufflinks', 'Divided skirt', 'Doublet', 'Drape front', 'Dress', 'Dressing Gown', 'Dress Shirt', 'Dupatta', 'Flannel shirt', 'French pantyhose', 'Full Slips', 'Fur coat', 'Gaghra', 'Gartini', 'Gilet', 'Girdle', 'Gloves', 'G string', 'Hat', 'Hawaiian shirt', 'Hijab', 'Hipster', 'Hoody', 'House coat', 'Jacket', 'Jeans', 'Jeggings', 'Jumper', 'Kimono', 'Knee high socks', 'Knickers', 'Kurta', 'Leggings', 'Leg warmer', 'Lehanga', 'Lingerie', 'Lounge wear', 'Lounge pants', 'Maternity wear', 'Maxi', 'Mittens', 'Nightgown', 'Nightwear', 'Niqab', 'Oilskin clothing', 'Overalls', 'Overskirt', 'Padded Bra', 'Pajama', 'Panty', 'Pantyhose', 'Pencil skirt', 'Petticoat', 'Photo tshirts', 'Pleated skirts', 'Polo', 'Poet sleeve', 'Polo Shirt', 'Poncho', 'Pull on clothes', 'Pullover', 'Push up Bras', 'Pyjamas', 'Rain Coat', 'Robe', 'Running shorts', 'Sari', 'Saree', 'Sarong', 'Sash', 'Scarf', 'Shawl', 'Shervani', 'Shirt', 'Shorts', 'Slip', 'Ski Pants', 'Skirt', 'Socks', 'Stockings', 'Stole', 'Straight pants', 'Straight skirt', 'Suit', 'Summer Cardigan', 'Surplice', 'Suspenders', 'Swacket', 'Sweater', 'Sweatshirt', 'Sweats', 'Sweep train', 'Swimming Costume', 'Swimming Shorts', 'Swimming Trunks', 'Tap pants', 'T-Shirt', 'Tank top', 'Thong', 'Tie', 'Tiered skirt', 'Tights', 'Top', 'Toreador pants', 'Tracksuit', 'Trainers', 'Treggings', 'Trench Coat', 'Trousers', 'Trumpet skirt', 'Trousseau', 'Tulip hemmed skirt', 'Tummy shaper', 'Tunic', 'Turban', 'Turtleneck', 'Tuxedo Jacket', 'Tuxedo', 'Twin set', 'Ulster', 'Underwear', 'Uniform', 'Union suit', 'Unitard', 'Veil', 'Vest', 'Waistcoat', 'Watteau', 'Windsor Tie', 'Wrap around skirt', 'Whorts', 'Wind Pants', 'Wrap coat', 'Wylie coat', 'Yoga Pants', 'Zip front top']

def basic\_cleaning(query):

query = remove\_irrelevant(query)

query = make\_lowercase(query)

query = do\_tokenization(query)

query = remove\_stopwords(query)

query = do\_lemmatization(query)

query = remove\_small(query)

query = convert\_to\_string(query)

return query

final\_types = []

for clothing in clothing\_types:

clothing = basic\_cleaning(clothing)

if clothing not in final\_types:

final\_types.append(clothing)

def calculate\_score(query, term):

query = query.split(' ')

term = term.split(' ')

temp\_matches = set(final\_types).intersection(set(query))

matches = set(temp\_matches).intersection(set(term))

if len(matches) > 0:

return 80

else:

return 0

def find\_sim\_fuzz(query, n):

results = []

for x in range(len(final\_result)):

score = fuzz.token\_set\_ratio(query, final\_result[x]) + calculate\_score(query, final\_result[x])

results.append([score, x])

result = sorted(results, key=itemgetter(0), reverse=True)

final\_results = []

for x in result[:n]:

final\_results.append(document\_list['title'][x[1]])

return final\_results

def fetch\_results(result\_term):

n1 = merged1.loc[merged1['ntitle'] == result\_term].shape[0]

n2 = merged2.loc[merged2['ntitle'] == result\_term].shape[0]

if n1 > 0:

category = list(merged1.loc[merged1['ntitle'] == result\_term, 'category'])[0]

description = list(merged1.loc[merged1['ntitle'] == result\_term, 'description'])[0]

gender = list(merged1.loc[merged1['ntitle'] == result\_term, 'gender'])[0]

sub\_category = list(merged1.loc[merged1['ntitle'] == result\_term, 'sub\_category'])[0]

title = list(merged1.loc[merged1['ntitle'] == result\_term, 'title'])[0]

normal = list(merged1.loc[merged1['ntitle'] == result\_term, 'actual\_price'])[0]

selling = list(merged1.loc[merged1['ntitle'] == result\_term, 'selling\_price'])[0]

url = list(merged1.loc[merged1['ntitle'] == result\_term, 'url'])[0]

image = (list(merged1.loc[merged1['ntitle'] == result\_term, 'images'])[0])

image = image[0]

category = string.capwords(category)

description = string.capwords(description)

gender = string.capwords(gender)

sub\_category = string.capwords(sub\_category)

title = string.capwords(title)

print('Title: {}\nDescription: {}\nGender: {}\nCategory: {}\nSub-Category: {}\n\nMRP: Rs. {}\nDiscounted Price: Rs. {}\nLink: {}\n'.format(title, description, gender, category, sub\_category, normal, selling, url))

print()

try:

f = urllib.request.urlopen(image)

figure(figsize=(2.5, 2.5), dpi=80)

a = plt.imread(f, 0)

plt.imshow(a)

plt.axis('off')

plt.show()

print('\n')

except:

print("No image currently available")

elif n2 > 0:

category = list(merged2.loc[merged2['ntitle'] == result\_term, 'category'])[0][2:-2]

description = list(merged2.loc[merged2['ntitle'] == result\_term, 'description'])[0]

gender = list(merged2.loc[merged2['ntitle'] == result\_term, 'gender'])[0]

title = list(merged2.loc[merged2['ntitle'] == result\_term, 'title'])[0]

normal = list(merged2.loc[merged2['ntitle'] == result\_term, 'retail\_price'])[0]

selling = list(merged2.loc[merged2['ntitle'] == result\_term, 'discounted\_price'])[0]

url = list(merged2.loc[merged2['ntitle'] == result\_term, 'product\_url'])[0]

image = (list(merged2.loc[merged2['ntitle'] == result\_term, 'image'])[0].split(','))

image = str(image[0][2:-1])

category = string.capwords(category)

description = string.capwords(description)

gender = string.capwords(gender)

title = string.capwords(title)

print(

'Title: {}\nDescription: {}\nGender: {}\nCategory: {}\n\nMRP: Rs. {}\nDiscounted Price: Rs. {}\nLink: {}\n'.format(

title, description, gender, category, normal, selling, url))

print()

try:

f = urllib.request.urlopen(image)

figure(figsize=(2.5, 2.5), dpi=80)

a = plt.imread(f, 0)

plt.imshow(a)

plt.axis('off')

plt.show()

print('\n')

except:

print("No image currently available")

else:

return 'Not Found!'

def get\_top\_results(query, n):

print()

print('Query: {}\n'.format(query))

query = basic\_cleaning(query)

results = find\_sim\_fuzz(query, n)

print("Displaying Top {} Results,\n".format(n))

for result in range(0, len(results)):

print("Result #{}\n".format(result + 1))

fetch\_results(results[result])

query = input("Search Query: ")

number\_wanted = int(input("Number of Results: "))

get\_top\_results(query, number\_wanted)

# CHAPTER-6

# SYSTEM TESTING

## INTRODUCTION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation of change over methods. Apart from planning major task of preparing the implementation are education and training of users.

The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system. In network backup system no additional resources are needed. Implementation is the final and the most important phase.

## TYPES OF TESTS Unit Testing:

The software units in the system are modules and routines that are assembled and integrated to perform a specific function. As a part of unit testing, we executed the program for individual modules independently. This enables, to detect errors in coding and logic that are contained within each of the three modules.

This testing includes entering data that is filling forms and ascertaining if the value matches to the type and entered into the database. The various controls are tested to ensure that each performs its action as required.

**Integration Testing:**

Data can be lost across any interface, one module can have an adverse effect on another, sub functions when combined, may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are

combined and tested as a whole.

Here the includes module, index module and script module options are integrated and tested. This testing provides the assurance that the application is well integrated functional unit with smooth transition of data.

## Functional testing:

Functional Testing checks provide systematic demonstrations that capabilities established are to be had as particular by means of the business and technical specifications, method documentation, and consumer manuals. Functional testing is working on below mentioned data.

LEGITIMATE INPUT: Identified lessons of legitimate input ought to be accredited. INVALID ENTER: Recognized lessons of unacceptable effort must be rejected.

CAPABILITIES: recognized features ought to be exercised.

OUTPUT: recognized courses of software outputs have got to be exercised. SYSTEMS/PROCEDURES: performance of the system here was invoked

## System testing:

As the part of system testing, we execute the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance.

Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies, a test plan is carried out on each module. The various tests performed are unit testing, integration testing and user acceptance testing.

## 

## White Box Testing:

This testing is a trying out wherein where the application tester has competencies of the interior workings, constitution and software language, or at least its cause. It's rationale.

It's used to test areas that can't be reached from a black box stage.

## Black Box Testing:

This is testing the software with none advantage of the inside workings, establishment or words of the unit life form veteran. Black field checks, as most other sorts of methods.

## LEVELS OF TESTING

Unit checking out is most commonly performed as a part of a mixed code and unit experiment part of the software life cycle, though it be not exceptional for coding and unit checking out to be performed as two targeted phases.

## Test strategy and approach:

Field testing out can be carried out manually and sensible assessments shall be written in element.

## Test objectives:

* + - Each field must be work correctly.
    - Each page must be activated through the specified link.
    - Features to be tested Verify that the entries are of the correct format No duplicateentries should be allowed.

## Integration testing strategy

Software integration testing is the incremental integration checking out of two otherwise further included software gears on top of a solo stage to fabricate failure induced with the aid of interface defects. The project of the mixing scan is to check that components or program applications.

e.g., Components in a program approach or owe one step up owe software purposes at the company degree owe interact without error.

## Test Results:

All of the scan circumstances recounted above passed efficiently. No defects encountered.

## Acceptance Testing

User Acceptance testing trying out is a crucial section of any mission and requires enormous participation by the tip user. It additionally ensures that the procedure meets the functional specifications.

## Test Results:

The entire test cases recounted above passed effectually. No defects Encountered.

# 

# CONCLUSION

.

Social media has become a major fashion influencer for today's generation. All of us surf the internet and it happens a lot that we come across some products that we wish to buy but can't find. This model solves this problem.

It extracts top trending posts from social media and then forms a searchable query on E-Commerce and shows all the similar products which are then compared to the initial product.

# FUTURE ENHANCEMENT

**1. Increase the Generality in Scraping:** Currently specific scrapers need to be written, for specific websites. To extend the volume of images scraped in the future, a generalized scraper needs to be created that is able to extract relevant images and data from any website.

**2. Combination of Search Queries:** Currently, the model supports minimal combinations of queries. In the future, we would like to further work on our model to be able to classify images with multiple pieces of fashion such as, “Red Beanie, Black Bomber Jacket, Blue Jeans, Brown Leather Shoes, White Watch”

**3. Extending Approach to other Categories:** After fashion, electronics is the most sought after category and on arriving upon a set method, the idea used and approaches can be extended to other categories

# APPENDIX

# 

# REFERENCES

Kumar, A., & Morstatter, F. (2018). Social media analytics: A survey of techniques, tools and platforms. AI Communications, 31(3), 249-269.

<https://www.researchgate.net/publication/272591009_Social_media_analytics_A_survey_of_techniques_tools_and_platforms>