# Sending HTTP Requests using the requests library

import requests

# Define the URL

url = 'http://example.com'

# Send a GET request to the URL

response = requests.get(url)

# Check if the request was successful

if response.status\_code == 200:

print("Request was successful!")

else:

print(f"Failed to retrieve data. Status code: {response.status\_code}")

# Parsing HTML Content using BeautifulSoup (bs4)

from bs4 import BeautifulSoup

# Parse the HTML content of the response

soup = BeautifulSoup(response.content, 'html.parser')

# You can now use the soup object to extract elements from the HTML# Import necessary libraries

import requests

from bs4 import BeautifulSoup

import pandas as pd

# Define the URL to scrape

url = 'https://unacademy.com/'

# Send a GET request to the URL

response = requests.get(url)

# Check if the request was successful

if response.status\_code == 200:

print("Successfully accessed the website")

else:

print("Failed to access the website")

response.raise\_for\_status()

# This will raise an HTTPError if the HTTP request returned an unsuccessful status code

# Parse the HTML content of the page

soup = BeautifulSoup(response.content, 'html.parser')

# Print the title of the page

print("Page Title:", soup.title.string if soup.title else "No title found")

# Find and extract specific data from the page

# For this example, let's extract all headings (h1, h2, h3, etc.)

headings = {}

for level in range(1, 7):

tag = f'h{level}'

headings[tag] = [heading.get\_text(strip=True) for heading in soup.find\_all(tag)]

# Create a DataFrame from the extracted headings

# Since the lists may have different lengths, we need to handle this situation

max\_len = max(len(v) for v in headings.values())

for key in headings:

while len(headings[key]) < max\_len:

headings[key].append(None)

headings\_df = pd.DataFrame(headings)

# Find and extract content from the page

# For this example, let's extract the main content of the page

# Replace 'content\_tag' with the appropriate HTML tag for your content

content = soup.find('div') # Update 'div' with the appropriate tag

content\_text = content.get\_text(strip=True) if content else "No content found"

# Find and extract footer details from the page

# For this example, let's assume the footer is contained within a footer tag

footer = soup.find('footer') # Update 'footer' with the appropriate tag for your footer

footer\_text = footer.get\_text(strip=True) if footer else "No footer found"

# Display the content and footer details

print("Content:", content\_text)

print("Footer:", footer\_text)

# Display the DataFrame

print(headings\_df)

# Save the DataFrame to a CSV file

headings\_df.to\_csv('headings.csv', index=False)

# Import necessary libraries

import numpy as np

import pandas as pd

# Adjust display options to show all rows and columns

pd.set\_option("display.max\_rows", None)

pd.set\_option("display.max\_columns", None)

# Define the subsegments

subsegments = [

"Banking & Finance",

"Digital Subscriptions",

"E-commerce",

"Education & Career Development",

"Interest-Based Targeting",

"Legal & Regulatory",

"Pharmacy/Drugstore",

"Professional Development & Training",

"Remarketing",

"Test Preparation & Exam Readiness"

]

# Create a dictionary to store the data

data = {"Users": range(1, 1001)} # 1000 users, labeled from 1 to 1000

# Generate random values for each subsegment for each user (0 or 1, indicating browsing activity)

for subsegment in subsegments:

data[subsegment] = np.random.randint(0, 2, size=1000)

# Convert the dictionary to a DataFrame

df = pd.DataFrame(data)

# Display the DataFrame (contains the 1000 users and their browsing activity across the subsegments)

print(df)

# Import pandas library

import pandas as pd

# Create a dictionary with the data

data = {

"Users": range(1, 11), # 10 users

"Banking & Finance": [1, 0, 0, 0, 1, 1, 0, 1, 0, 1],

"Digital Subscriptions": [1, 1, 1, 0, 0, 1, 1, 1, 1, 0],

"E-commerce": [1, 0, 1, 1, 1, 0, 1, 0, 1, 1],

"Education & Career Development": [0, 0, 0, 1, 1, 1, 1, 0, 0, 0],

"Interest-Based Targeting": [0, 1, 0, 1, 0, 0, 0, 1, 1, 1],

"Legal & Regulatory": [1, 1, 1, 0, 1, 1, 1, 0, 1, 1],

"Pharmacy/Drugstore": [0, 1, 1, 0, 0, 1, 0, 0, 1, 0],

"Professional Development & Training": [1, 0, 1, 0, 0, 1, 1, 1, 0, 0],

"Remarketing": [0, 0, 0, 1, 1, 0, 0, 1, 1, 1],

"Test Preparation & Exam Readiness": [1, 0, 0, 1, 0, 0, 1, 1, 0, 1]

}

# Convert the dictionary to a DataFrame

df = pd.DataFrame(data)

# Pivot the DataFrame (this won't change much since we only have individual rows for each user)

pivot\_table = df.pivot\_table(index="Users", aggfunc="sum")

# Display the pivot table

print(pivot\_table)

import pandas as pd

from sklearn.metrics import jaccard\_score

# Create a dictionary with the data

data = {

"Users": range(1, 11),

"Banking & Finance": [1, 0, 0, 0, 1, 1, 0, 1, 0, 1],

"Digital Subscriptions": [1, 1, 1, 0, 0, 1, 1, 1, 1, 0],

"E-commerce": [1, 0, 1, 1, 1, 0, 1, 0, 1, 1],

"Education & Career Development": [0, 0, 0, 1, 1, 1, 1, 0, 0, 0],

"Interest-Based Targeting": [0, 1, 0, 1, 0, 0, 0, 1, 1, 1],

"Legal & Regulatory": [1, 1, 1, 0, 1, 1, 1, 0, 1, 1],

"Pharmacy/Drugstore": [0, 1, 1, 0, 0, 1, 0, 0, 1, 0],

"Professional Development & Training": [1, 0, 1, 0, 0, 1, 1, 1, 0, 0],

"Remarketing": [0, 0, 0, 1, 1, 0, 0, 1, 1, 1],

"Test Preparation & Exam Readiness": [1, 0, 0, 1, 0, 0, 1, 1, 0, 1]

}

# Convert the dictionary to a DataFrame

df = pd.DataFrame(data)

# Remove the Users column for similarity calculation

df = df.drop(columns=['Users'])

# Initialize a DataFrame to store Jaccard similarity scores

similarity\_matrix = pd.DataFrame(index=df.columns, columns=df.columns)

# Calculate Jaccard similarity for each pair of categories

for col1 in df.columns:

for col2 in df.columns:

similarity\_matrix.loc[col1, col2] = jaccard\_score(df[col1], df[col2])

# Display the similarity matrix

print(similarity\_matrix)

For Cosine Similarity –

import numpy as np

import pandas as pd

# Your dataset - replace this with your actual dataset

data = {

"Users": range(1, 11),

"Banking & Finance": [1, 0, 0, 0, 1, 1, 0, 1, 0, 1],

"Digital Subscriptions": [1, 1, 1, 0, 0, 1, 1, 1, 1, 0],

"E-commerce": [1, 0, 1, 1, 1, 0, 1, 0, 1, 1],

"Education & Career Development": [0, 0, 0, 1, 1, 1, 1, 0, 0, 0],

"Interest-Based Targeting": [0, 1, 0, 1, 0, 0, 0, 1, 1, 1],

"Legal & Regulatory": [1, 1, 1, 0, 1, 1, 1, 0, 1, 1],

"Pharmacy/Drugstore": [0, 1, 1, 0, 0, 1, 0, 0, 1, 0],

"Professional Development & Training": [1, 0, 1, 0, 0, 1, 1, 1, 0, 0],

"Remarketing": [0, 0, 0, 1, 1, 0, 0, 1, 1, 1],

"Test Preparation & Exam Readiness": [1, 0, 0, 1, 0, 0, 1, 1, 0, 1]

}

# Convert the dataset to a pandas DataFrame

df = pd.DataFrame(data)

# Remove the "Users" column

df = df.drop(columns=["Users"])

# Calculate cosine similarity matrix

cos\_sim\_matrix = pd.DataFrame(np.dot(df, df.T) / (np.linalg.norm(df, axis=1)[:, None] \* np.linalg.norm(df, axis=1)))

# Set row and column names as user numbers

cos\_sim\_matrix.index = data["Users"]

cos\_sim\_matrix.columns = data["Users"]

# Print the cosine similarity matrix

print("Cosine Similarity Matrix:")

print(cos\_sim\_matrix)