# Vector Databases can help with Anamoly detection

For this we nees a dataset with anamolies why not use a dataset that is live auto updating so that constantly anamolies take place which can be auto detected and presented in a readable format for the user to read and apply to the real world.

For this tutorial, I will be using [1] .

<https://money.rediff.com/news>

## Code

<https://github.com/heathbrew/Vector-Databases-can-help-with-Anomaly-Detection>

Git clone this repo [2] and follow along for the setup.

## Scraping to create a dataset

Let’s see about all the content that is availaible

import requests

from bs4 import BeautifulSoup

import pandas as pd

# URL for Rediff Money news

url = "https://money.rediff.com/news"

# Fetch HTML content using requests

response = requests.get(url)

if response.status\_code == 200:

    # Parse HTML content using BeautifulSoup

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

soup

You will get the HTML for all the data available.

Let’s convert this to a proper scraped dataset

import requests

from bs4 import BeautifulSoup

import pandas as pd

# URL for Rediff Money news

url = "https://money.rediff.com/news"

# Fetch HTML content using requests

response = requests.get(url)

if response.status\_code == 200:

    # Parse HTML content using BeautifulSoup

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

    # Find all news items

    news\_items = soup.find\_all("div", class\_="rtnews\_row\_more")

    # Extract relevant information from each news item

    news\_data = []

    for item in news\_items:

        title = item.find("p").text.strip()

        link = item.find("a")["href"]

        summary = item.find("div").text.strip()

        published = item.find("span", class\_="timeago").text.strip()

        news\_data.append({

            "title": title,

            "summary": summary,

            "link": link,

            "published": published

        })

    # Create pandas DataFrame

    df = pd.DataFrame(news\_data)

    # Save DataFrame to CSV

    df.to\_csv("Dataset/financial\_news.csv", index=False)

    print("Financial news scraped and saved to financial\_news.csv")

else:

    print("Failed to fetch the webpage")

Financial news scraped and saved to financial\_news.csv

Let’s view it

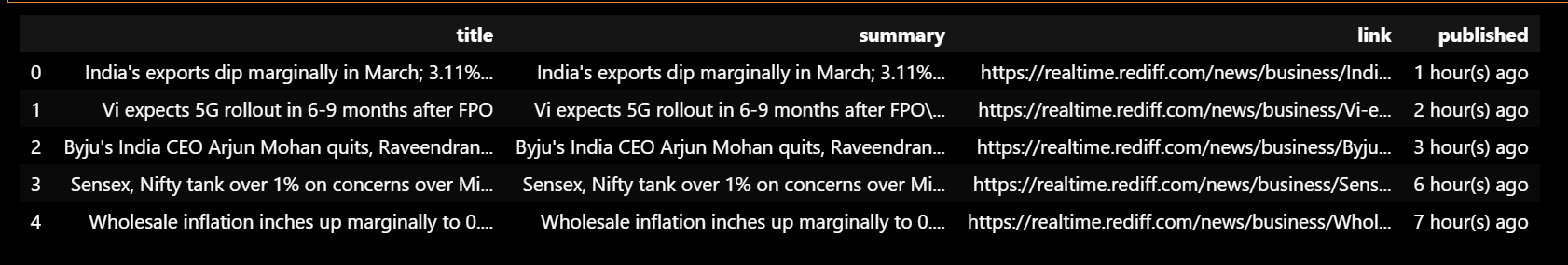
import pandas as pd

# Read the CSV file into a DataFrame

df = pd.read\_csv("Dataset/financial\_news.csv")

# Display the first few rows of the DataFrame

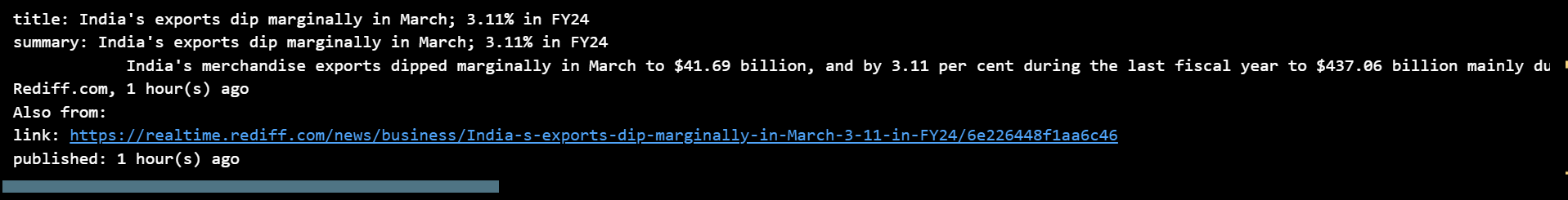
df.head()



Instead of using the pre made dataset using a Live dataset is much better because this will allow the user to check out the concerned stock By Vector database anamoly detection. The user can take BUY and SELL decisions on that stock.

for column in df.columns:

    print(f"{column}: {df[column][0]}")



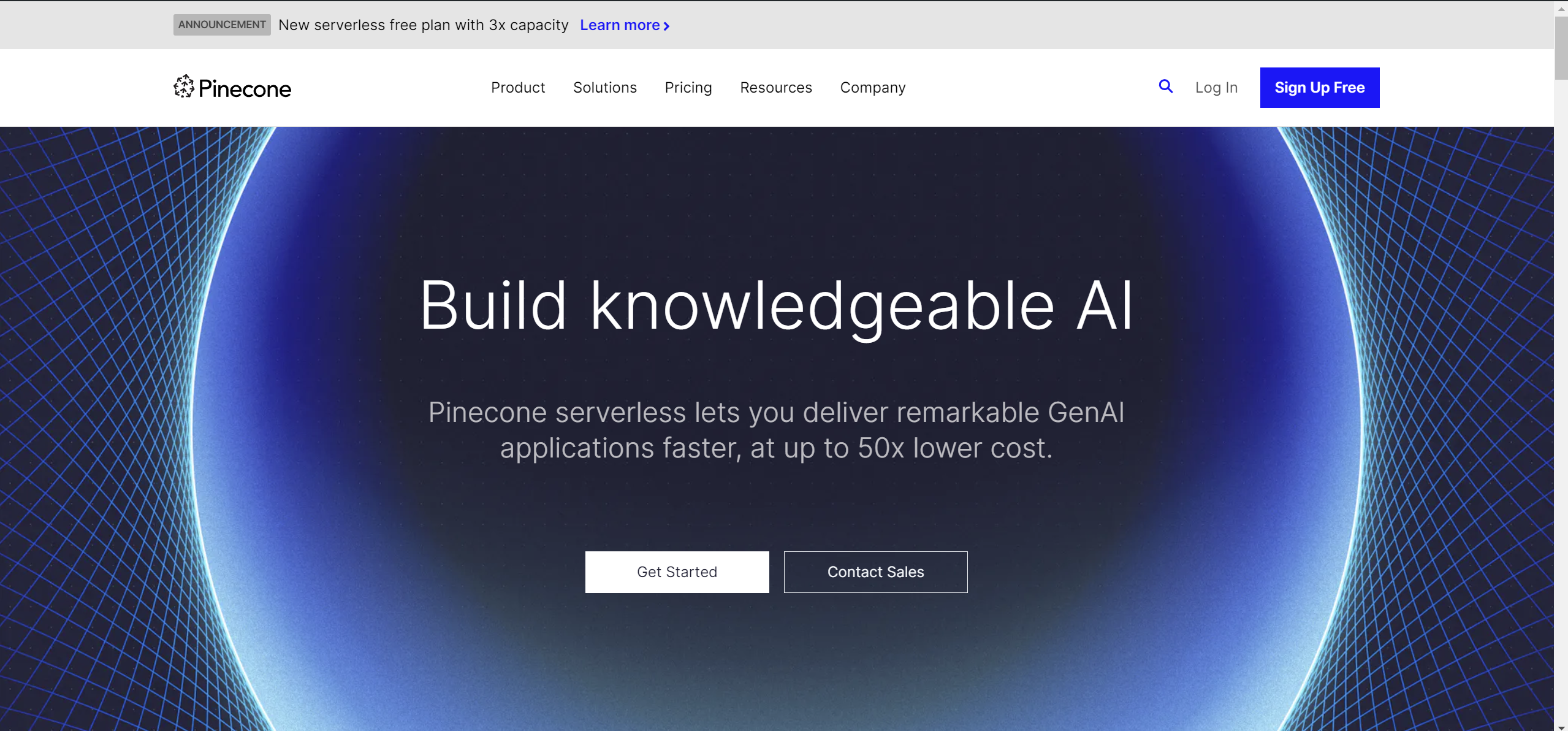
## Creating a PineCone ID

In the last project [3], I used Qdrant which is a locally supported Vector Store for RAG

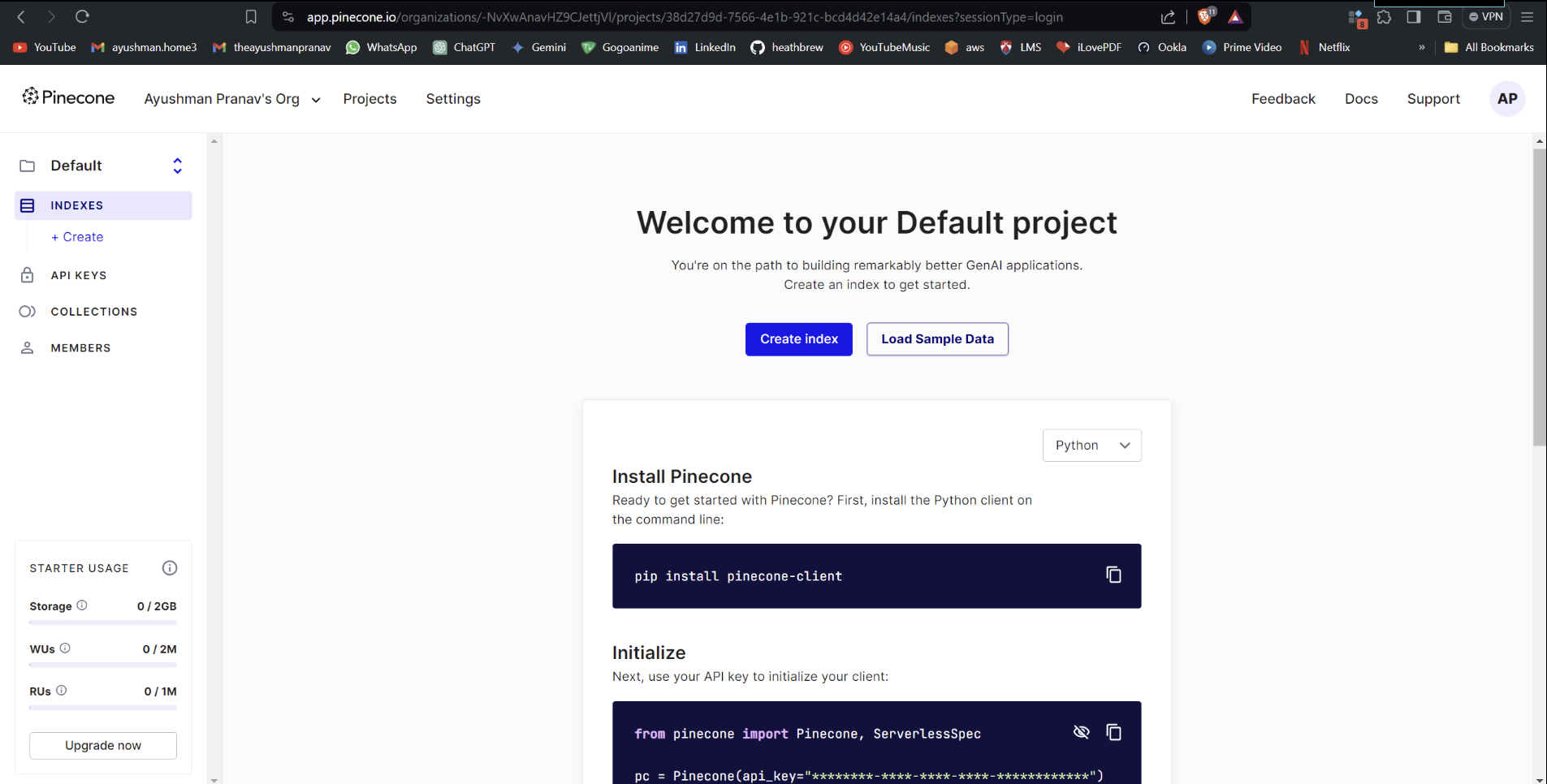
<https://medium.com/devops-dev/steps-to-monitoring-dspy-qdrant-powered-rag-with-prometheus-or-grafana-b642335cbd50>

For this project, I will use PineCone which is an AWS supported Vector store that you can directly In any Live Project

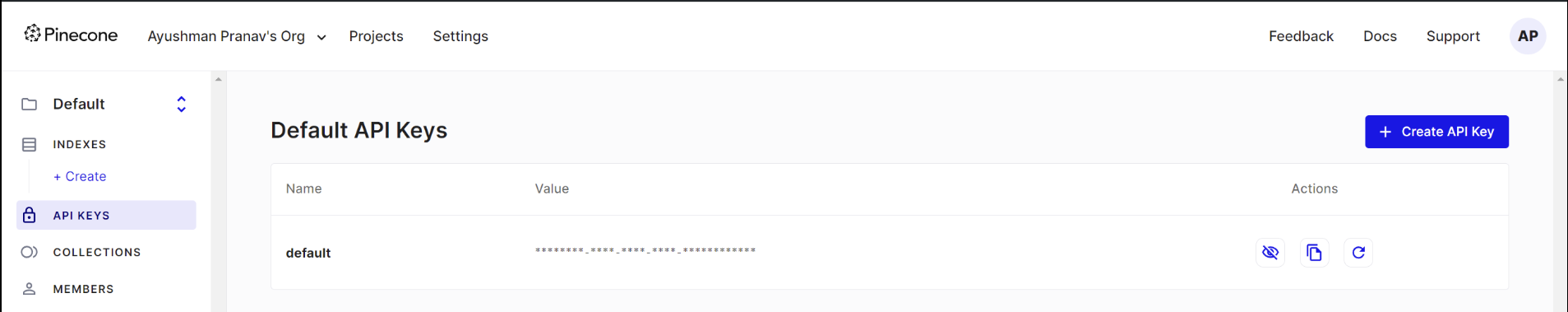
<https://www.pinecone.io/>



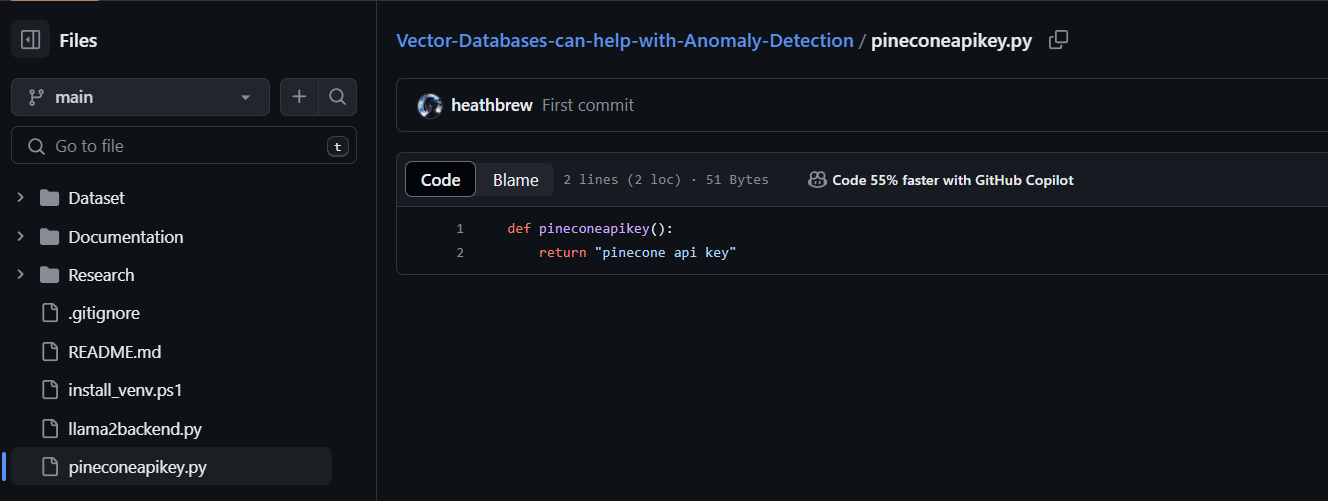
Login and create the account



Go to api keys to create your own api key



Copy this api key and paste in



## Feeding the dataset to Vector Store

Take a look at the general dataset

import pandas as pd

# Load the CSV file into a pandas DataFrame

df = pd.read\_csv("Dataset/financial\_news.csv")

df.head()



### Loading the embedding model

I will be using the MINI-LM12-V2 [4] to embed the news.

from sentence\_transformers import SentenceTransformer

from transformers import AutoTokenizer, AutoModel

from pathlib import Path

def download\_model\_and\_tokenizer(model\_name, save\_path):

    """

    Download and save both the model and the tokenizer to the specified directory.

    Parameters:

        model\_name (str): Name of the model to download.

        save\_path (str or Path): Path to the directory where the model and tokenizer will be saved.

    """

    # Create the save path if it doesn't exist

    save\_path = Path(save\_path)

    save\_path.mkdir(parents=True, exist\_ok=True)

    # Initialize tokenizer and model

    tokenizer = AutoTokenizer.from\_pretrained(model\_name)

    model = AutoModel.from\_pretrained(model\_name)

    # Save tokenizer

    tokenizer.save\_pretrained(save\_path)

    # Save model

    model.save\_pretrained(save\_path)

# Example usage

model\_name = 'sentence-transformers/all-MiniLM-L12-v2'  # Model name to download

save\_path = Path("MiniLM-L12-v2/")  # Path where model and tokenizer will be saved

download\_model\_and\_tokenizer(model\_name, save\_path)

Load the model and the tokenizer

from transformers import AutoTokenizer, AutoModel

def load\_model\_and\_tokenizer(model\_path):

    """

    Load the model and tokenizer from the specified directory.

    Parameters:

        model\_path (str or Path): Path to the directory containing the saved model and tokenizer.

    Returns:

        tokenizer (transformers.PreTrainedTokenizer): Loaded tokenizer.

        model (transformers.PreTrainedModel): Loaded model.

    """

    model\_path = Path(model\_path)

    tokenizer = AutoTokenizer.from\_pretrained(model\_path)

    model = AutoModel.from\_pretrained(model\_path)

    return tokenizer, model

# Load the model and tokenizer

model\_path = Path("MiniLM-L12-v2/")

tokenizer, model = load\_model\_and\_tokenizer(model\_path)

Merge the title and the summary

df['news'] = df.apply(lambda row: row['title'] + ' ' + row['summary'], axis=1)

df.head()



Embed the news

import torch

#Mean Pooling - Take attention mask into account for correct averaging

def mean\_pooling(model\_output, attention\_mask):

    token\_embeddings = model\_output[0] #First element of model\_output contains all token embeddings

    input\_mask\_expanded = attention\_mask.unsqueeze(-1).expand(token\_embeddings.size()).float()

    return torch.sum(token\_embeddings \* input\_mask\_expanded, 1) / torch.clamp(input\_mask\_expanded.sum(1), min=1e-9)

def generate\_embedding(text):

    # Tokenize input text

    encoded\_input = tokenizer(text, padding=True, truncation=True, return\_tensors='pt')

    # Compute token embeddings with model

    with torch.no\_grad():

        model\_output = model(\*\*encoded\_input)

    # Perform mean pooling

    sentence\_embedding = mean\_pooling(model\_output, encoded\_input['attention\_mask'])

    # Convert to numpy for FAISS compatibility and ensure it's 2D

    return sentence\_embedding.cpu().numpy().reshape(1, -1)

# Generate embeddings for the 'input' column

df['encoded\_news'] = df['news'].apply(lambda x: generate\_embedding(x)[0].tolist())

df.head()



### Pushing the data to Pinecone

Initializing Pinecone

from pinecone import Pinecone, ServerlessSpec

from pineconeapikey import pineconeapikey

# Initialize Pinecone

pc = Pinecone(api\_key=pineconeapikey())

logging.info("Initialized Pinecone")

index\_name = "financial-news-index"

# Check if the index already exists

if 'financial-news-index' not in pc.list\_indexes().names():

    logging.info(f"Creating new index: {index\_name}")

    # Get the embedding dimension from the model configuration

    embedding\_dim = model.config.hidden\_size

    # Create a new index if it doesn't already exist

    pc.create\_index(

        name=index\_name,

        dimension=embedding\_dim,

        metric="cosine",

        spec=ServerlessSpec(

        cloud="aws",

        region="us-east-1"

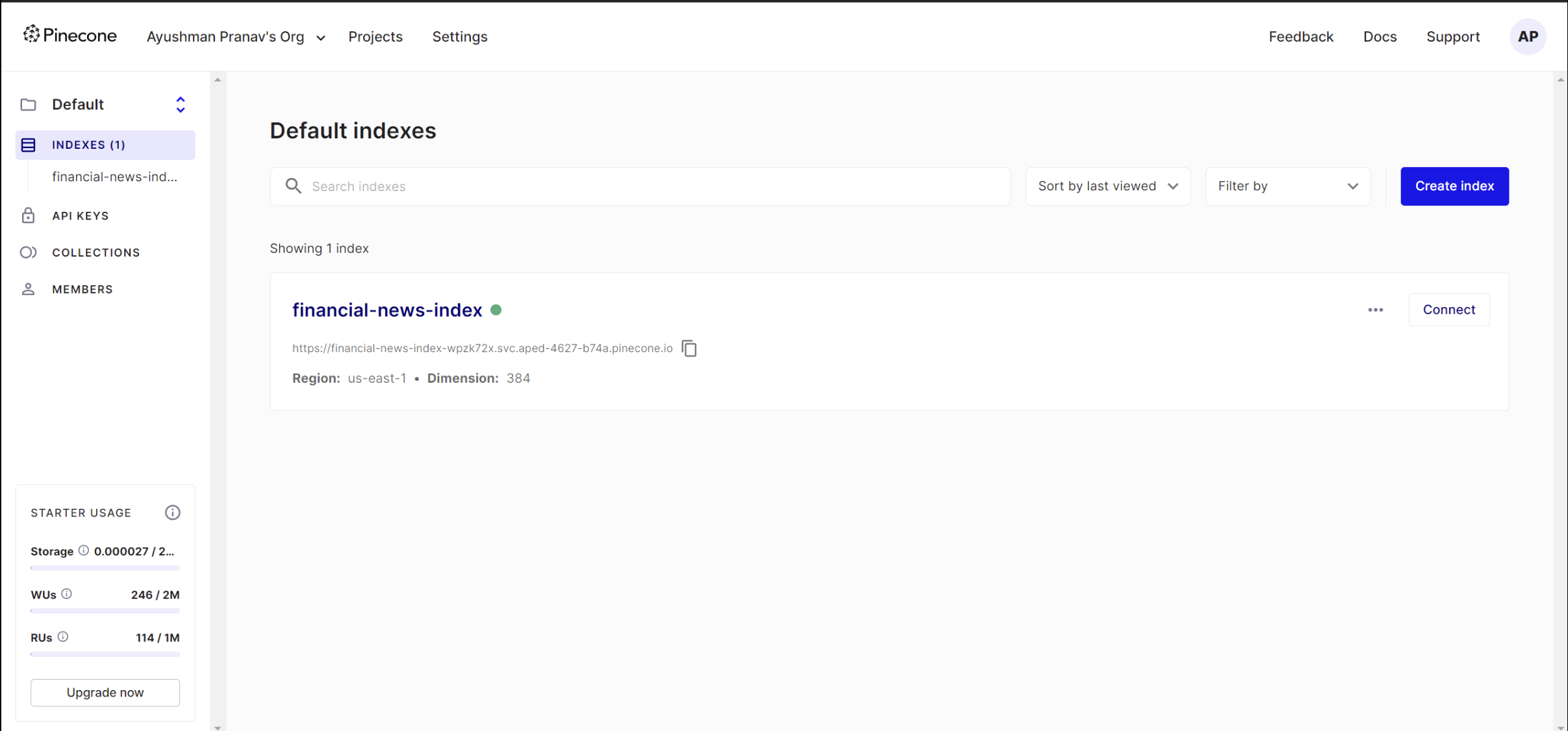
    )

    )

else:

    logging.info(f"Index '{index\_name}' already exists")

index = pc.Index(index\_name)



import itertools

# Initialize Pinecone and create an index

pc = Pinecone(api\_key=pineconeapikey())

index = pc.Index(index\_name)

def chunks(iterable, batch\_size=100):

    """A helper function to break an iterable into chunks of size batch\_size."""

    it = iter(iterable)

    chunk = tuple(itertools.islice(it, batch\_size))

    while chunk:

        yield chunk

        chunk = tuple(itertools.islice(it, batch\_size))

# Extract necessary columns from the DataFrame

encoded\_news = df['encoded\_news']

news = df['news']

# Process the data and combine into a list of dictionaries

formatted\_data = []

for idx, (encoded, text) in enumerate(zip(encoded\_news, news)):

    try:

        # Create a dictionary representing a data point

        data\_point = {"id": str(idx), "values": encoded, "metadata": {"text": text}}

        formatted\_data.append(data\_point)

        logging.info(f"Processed and encoded summary {idx + 1}")

    except Exception as e:

        logging.error(f"Error processing data point {idx + 1}: {e}")

        continue  # Skip to the next data point if an error occurs

# Upsert the formatted embeddings to the index in batches

for ids\_vectors\_chunk in chunks(formatted\_data, batch\_size=10):

    index.upsert(vectors=ids\_vectors\_chunk)

2024-04-16 01:53:31 - INFO - Processed and encoded summary 1

2024-04-16 01:53:31 - INFO - Processed and encoded summary 2

2024-04-16 01:53:31 - INFO - Processed and encoded summary 3

2024-04-16 01:53:31 - INFO - Processed and encoded summary 4

2024-04-16 01:53:31 - INFO - Processed and encoded summary 5

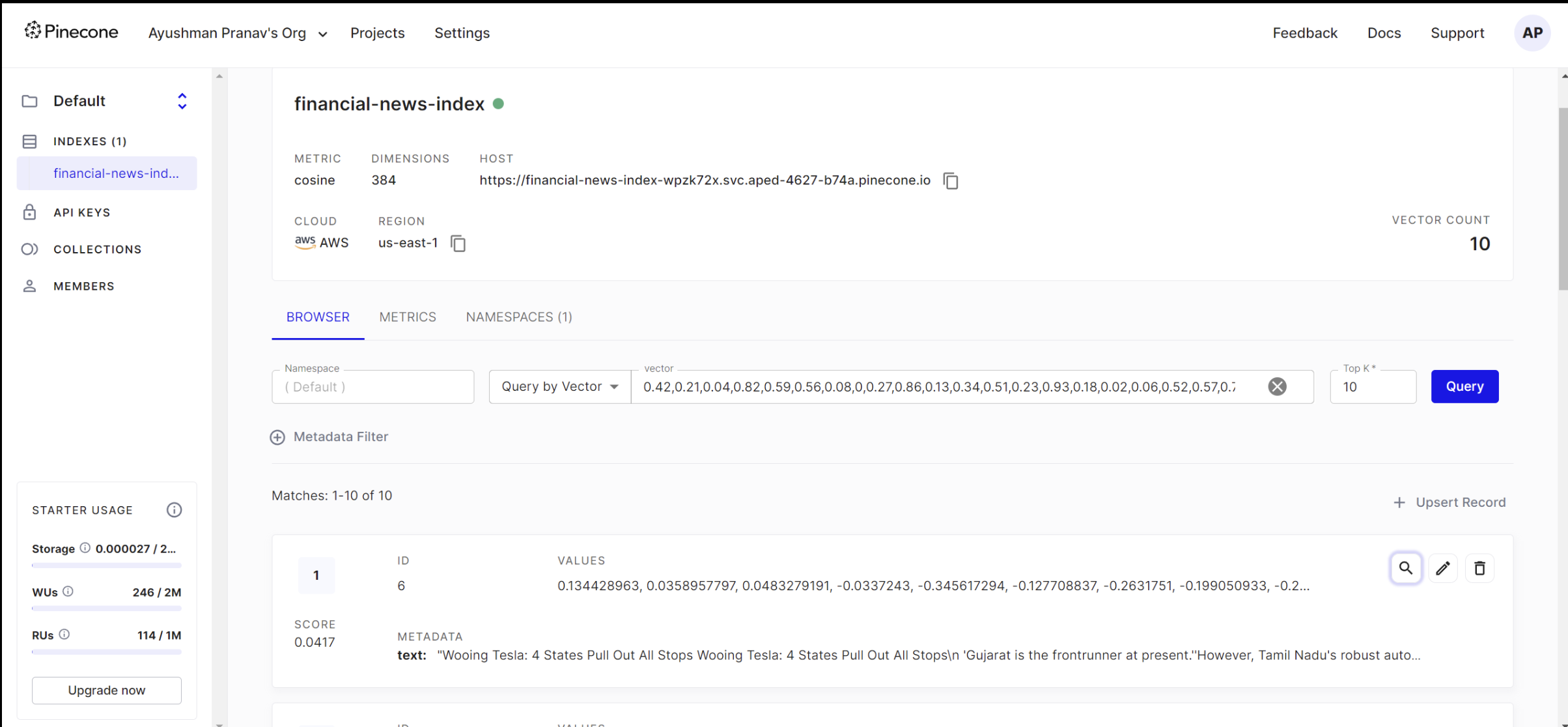
2024-04-16 01:53:31 - INFO - Processed and encoded summary 6

2024-04-16 01:53:31 - INFO - Processed and encoded summary 7

2024-04-16 01:53:31 - INFO - Processed and encoded summary 8

2024-04-16 01:53:31 - INFO - Processed and encoded summary 9

2024-04-16 01:53:31 - INFO - Processed and encoded summary 10



## Querying Pinecone

Query the pinecone for anamolies in any way you want.

index\_name = "financial-news-index"

index = pc.Index(index\_name)

query = "Stock market crashes due to unexpected event"

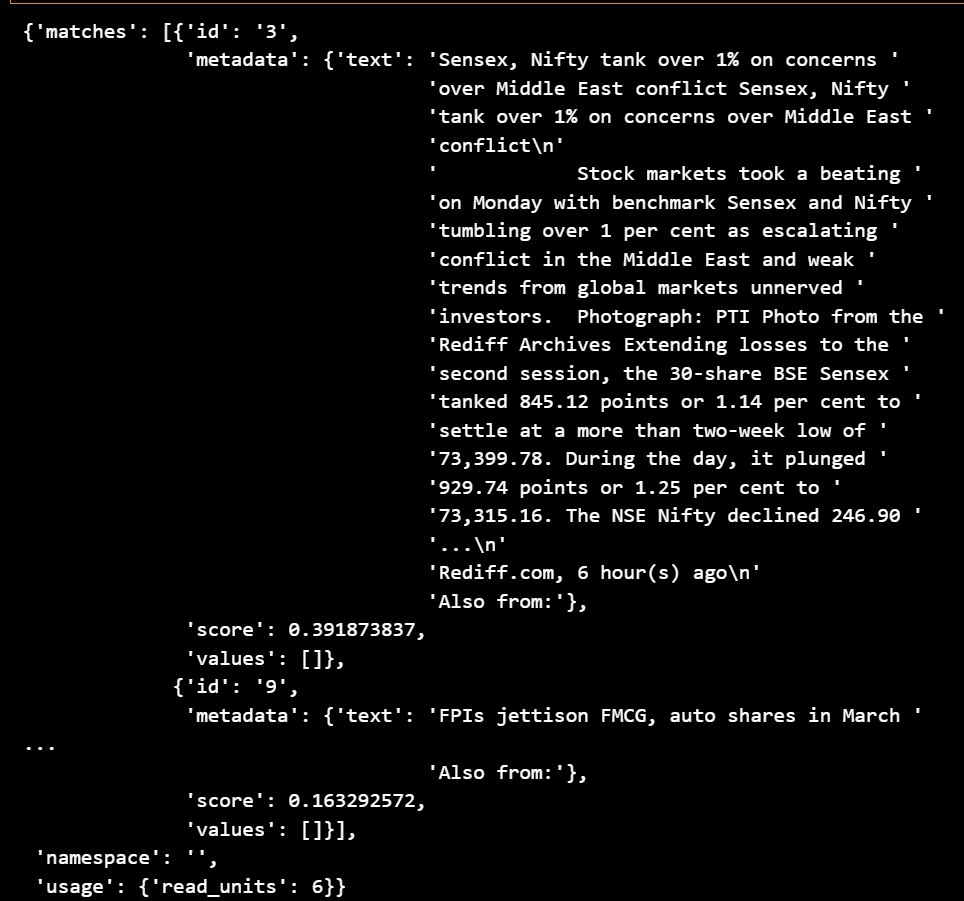
# create the query vector

xq = generate\_embedding(query).tolist()

# now query

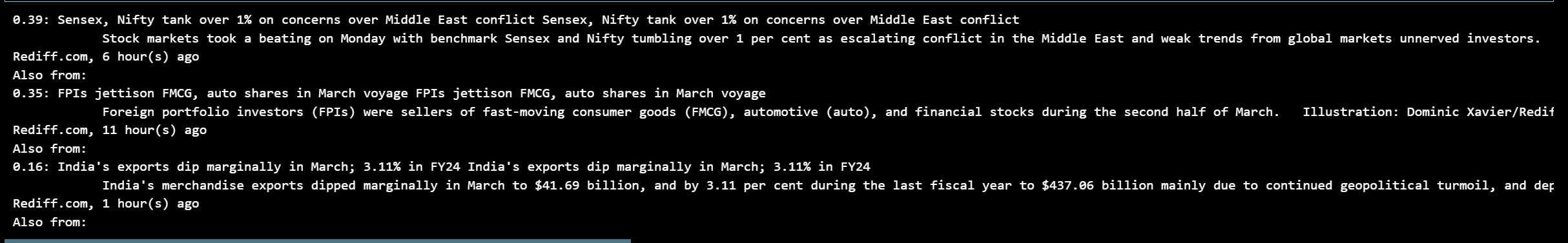
xc = index.query(vector=xq, top\_k=3, include\_metadata=True)

xc



for result in xc['matches']:

    print(f"{round(result['score'], 2)}: {result['metadata']['text']}")



# Initialize an empty string to store concatenated results

concatenated\_results = ""

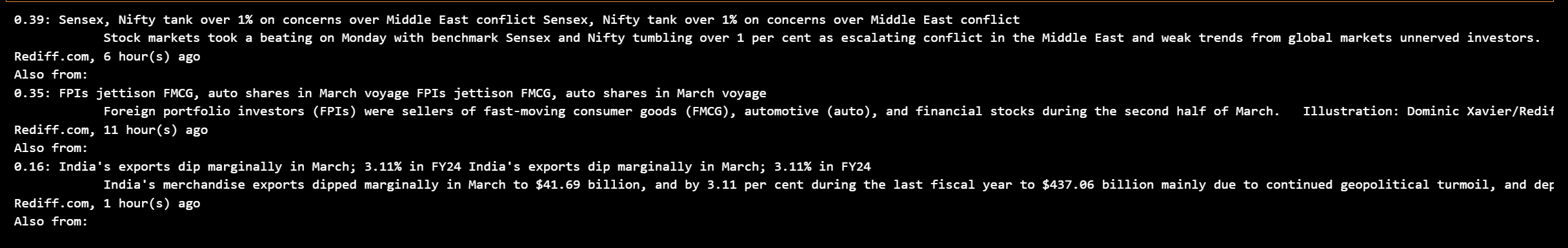
# Iterate over the matches and concatenate each result

for result in xc['matches']:

    concatenated\_results += f"{round(result['score'], 2)}: {result['metadata']['text']}\n"

# Print or use the concatenated results as needed

print(concatenated\_results)



## RAG using Llama2

Use the concated result to generated a paragraph on anamolies which allows you to read everything at once. [5] is an article on how to setup llama2 on your local system and run it. Although [2] includes the ps1 script to run it.

from llama2backend import generate\_text

RAG\_answer = generate\_text(str(query + concatenated\_results)[:512])

' The stock market witnessed a significant downturn today, with the Sensex dropping 821.17 points or 2.79% to settle at 29,677.47. The broader Nifty also took a hit, plummeting 270.7 points or 2.83% to finish at 9,477.50. Heavyweight stocks like Reliance Industries, Tata Steel, HUL, and Infosys led the sell-off, each declining by up to 3%. Market breadth remained weak, with only 16 gainers against 42 losers on the Sensex and 20 winners versus 30 losers on the Nifty.Analysts attributed the market decline to global economic growth concerns, escalating tensions in the Middle East, and a sharp decline in crude oil prices. The US-Iran conflict escalated following the US drone strike that killed top Iranian and US military commanders last week, raising fears of a wider conflict in the region.Adding to investor concerns, the Indian rupee weakened further, reaching a new record low of 76.94 against the dollar.’

## Annomoly Detection

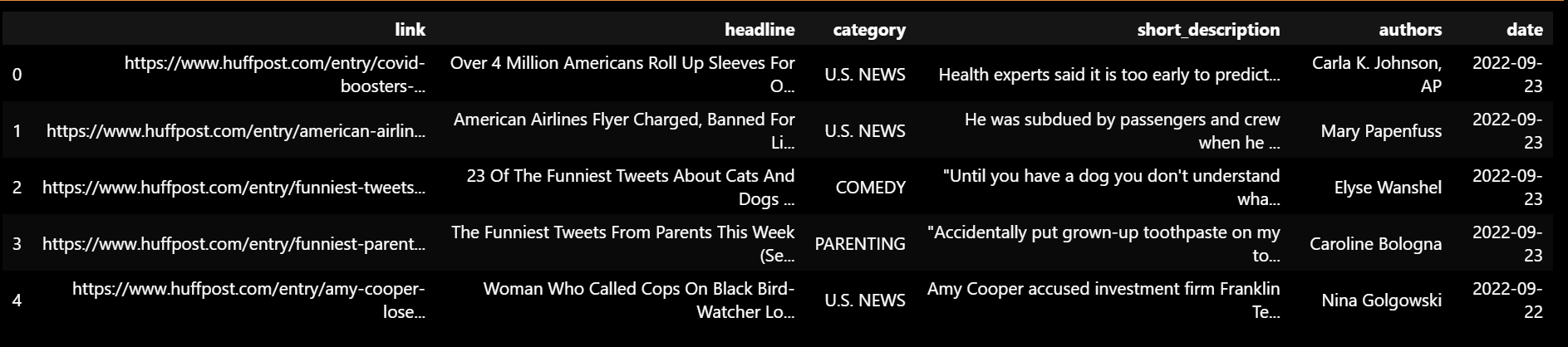
This method is used for Larger datasets so we will be using [6] . We will be loading it

import pandas as pd

# Load the CSV file into a pandas DataFrame

df = pd.read\_json("Dataset/News\_Category\_Dataset\_v3.json" , lines=True)

df.head()



Take the first 1000 rows

# Select the first 1000 rows of the DataFrame

df = df.head(1000)

Create embeddings Like before

# Generate embeddings for the 'input' column

df['encoded\_news'] = df['news'].apply(lambda x: generate\_embedding(x)[0].tolist())

df.head()



### Apply t-SNE for dimensionality reduction

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.manifold import TSNE

X = np.array(df['encoded\_news'].tolist())

# Increase the perplexity value (default is 30)

tsne = TSNE(random\_state=0, n\_iter=1000, perplexity=500)

tsne\_results = tsne.fit\_transform(X)

df\_tsne = pd.DataFrame(tsne\_results, columns=['TSNE1', 'TSNE2'])

df\_tsne['Class Name'] = df['category']  # Using 'title' as a placeholder for 'Class Name'

df\_tsne['news'] = df['news']

# df\_tsne['encoded\_news'] = df['encoded\_news']

df\_tsne.head()



# Plot t-SNE results

fig, ax = plt.subplots(figsize=(8, 6))

sns.set\_style('darkgrid', {"grid.color": ".6", "grid.linestyle": ":"})

sns.scatterplot(data=df\_tsne, x='TSNE1', y='TSNE2', hue='Class Name', palette='Set2')

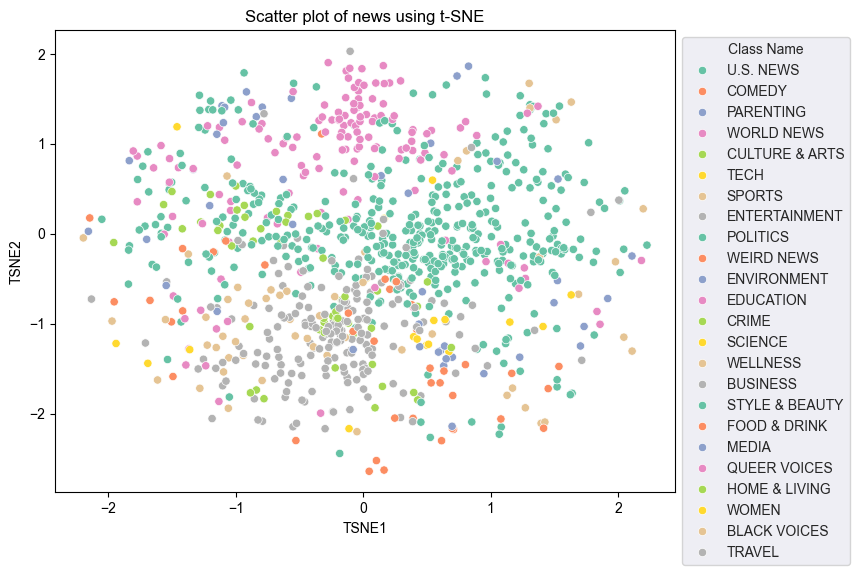
sns.move\_legend(ax, "upper left", bbox\_to\_anchor=(1, 1))

plt.title('Scatter plot of news using t-SNE')

plt.xlabel('TSNE1')

plt.ylabel('TSNE2')

plt.show()



### Outlier Detection

# Function to get centroids of each class

def get\_centroids(df\_tsne):

    centroids = df\_tsne.drop(columns=['news']).groupby('Class Name').mean()

    return centroids

centroids = get\_centroids(df\_tsne)

# Function to detect outliers

def calculate\_euclidean\_distance(p1, p2):

    return np.sqrt(np.sum(np.square(p1 - p2)))

def detect\_outlier(df, emb\_centroids, radius):

    outlier\_indices = []

    for idx, row in df.iterrows():

        class\_name = row['Class Name']

        dist = calculate\_euclidean\_distance(np.array([row['TSNE1'], row['TSNE2']]),

                                            np.array([emb\_centroids.loc[class\_name, 'TSNE1'],

                                                      emb\_centroids.loc[class\_name, 'TSNE2']]))

        if dist > radius:

            outlier\_indices.append(idx)

    return outlier\_indices

# Assuming df\_tsne and centroids are already defined DataFrames

range\_ = np.arange(0.01, 1.0, 0.05).round(decimals=2).tolist()

outliers\_list = []

for i in range\_:

    outliers = detect\_outlier(df\_tsne, centroids, i)

    outliers\_list.append(outliers)

# Combine all outlier indices into a single list

all\_outliers = [idx for sublist in outliers\_list for idx in sublist]

# Update the 'Outlier' column in df\_tsne

df\_tsne['Outlier'] = df\_tsne.index.isin(all\_outliers)

df\_tsne.head()

# Assuming df\_tsne and centroids are already defined DataFrames

range\_ = np.arange(0.01, 1.0, 0.05).round(decimals=2).tolist()

outliers\_list = []

for i in range\_:

    outliers = detect\_outlier(df\_tsne, centroids, i)

    outliers\_list.append(outliers)

# Combine all outlier indices into a single list

all\_outliers = [idx for sublist in outliers\_list for idx in sublist]

# Update the 'Outlier' column in df\_tsne

df\_tsne['Outlier'] = df\_tsne.index.isin(all\_outliers)

df\_tsne.head()



# Filter out rows where 'Outlier' is False

df\_tsne = df\_tsne[df\_tsne['Outlier'] == True]

num\_outliers = [len(outliers) for outliers in outliers\_list]

import matplotlib.pyplot as plt

# Plot range\_ and num\_outliers

fig = plt.figure(figsize=(14, 8))

plt.rcParams.update({'font.size': 12})

plt.bar(list(map(str, range\_)), num\_outliers)

plt.title("Number of outliers vs. distance of points from centroid")

plt.xlabel("Distance")

plt.ylabel("Number of outliers")

for i in range(len(range\_)):

    plt.text(i, num\_outliers[i], num\_outliers[i], ha='center')

plt.show()



def get\_outlier\_texts(df, class\_name):

    # Filter the DataFrame to get outliers of the specified category

    outliers = df[(df['Class Name'] == class\_name) & df['Outlier']]

    # Extract the outlier texts

    outlier\_texts = outliers['news'].tolist()

    return outlier\_texts

# Example usage:

outlier\_texts = get\_outlier\_texts(df\_tsne, 'TECH')

for idx, text in enumerate(outlier\_texts, start=1):

    print(f"Outlier {idx}: {text}\n")

Outlier 1: Twitch Bans Gambling Sites After Streamer Scams Folks Out Of $200,000 One man's claims that he scammed people on the platform caused several popular streamers to consider a Twitch boycott.

Outlier 2: TikTok Search Results Riddled With Misinformation: Report A U.S. firm that monitors false online claims reports that searches for information about prominent news topics on TikTok are likely to turn up results riddled with misinformation.

Outlier 3: Citing Imminent Danger Cloudflare Drops Hate Site Kiwi Farms Cloudflare CEO Matthew Prince had previously resisted calls to block the site.

Outlier 4: Instagram And Facebook Remove Posts Offering Abortion Pills Facebook and Instagram began removing some of these posts, just as millions across the U.S. were searching for clarity around abortion access.

Outlier 5: Google Engineer On Leave After He Claims AI Program Has Gone Sentient Artificially intelligent chatbot generator LaMDA wants “to be acknowledged as an employee of Google rather than as property," says engineer Blake Lemoine.

Outlier 6: Facebook Is Still Allowing Mug Shots Even Though They Can Ruin Lives When an individual’s mug shot goes viral on Facebook, they are often subjected to extreme harassment and struggle to find stable housing and employment.

Outlier 7: Ex-Twitter CEO Dings Elon Musk For Attacks On Twitter's Top Lawyer A one-sided feud between Musk and Vijaya Gadde has turned even uglier.

Outlier 8: Investor Sues Elon Musk Over His Delayed Twitter Filing Marc Rasella says he sold shares of Twitter at “artificially deflated prices,” unaware that Musk had made a large purchase in the social media platform.

def get\_outlier\_paragraph(df, class\_name):

    # Filter the DataFrame to get outliers of the specified category

    outliers = df[(df['Class Name'] == class\_name) & df['Outlier']]

    # Extract the outlier texts

    outlier\_texts = outliers['news'].tolist()

    # Concatenate all outlier texts into one paragraph

    outlier\_paragraph = ' '.join(outlier\_texts)

    return outlier\_paragraph

# Example usage:

outlier\_paragraph = get\_outlier\_paragraph(df\_tsne, 'TECH')

print(outlier\_paragraph)

Twitch Bans Gambling Sites After Streamer Scams Folks Out Of $200,000 One man's claims that he scammed people on the platform caused several popular streamers to consider a Twitch boycott. TikTok Search Results Riddled With Misinformation: Report A U.S. firm that monitors false online claims reports that searches for information about prominent news topics on TikTok are likely to turn up results riddled with misinformation. Citing Imminent Danger Cloudflare Drops Hate Site Kiwi Farms Cloudflare CEO Matthew Prince had previously resisted calls to block the site. Instagram And Facebook Remove Posts Offering Abortion Pills Facebook and Instagram began removing some of these posts, just as millions across the U.S. were searching for clarity around abortion access. Google Engineer On Leave After He Claims AI Program Has Gone Sentient Artificially intelligent chatbot generator LaMDA wants “to be acknowledged as an employee of Google rather than as property," says engineer Blake Lemoine. Facebook Is Still Allowing Mug Shots Even Though They Can Ruin Lives When an individual’s mug shot goes viral on Facebook, they are often subjected to extreme harassment and struggle to find stable housing and employment. Ex-Twitter CEO Dings Elon Musk For Attacks On Twitter's Top Lawyer A one-sided feud between Musk and Vijaya Gadde has turned even uglier. Investor Sues Elon Musk Over His Delayed Twitter Filing Marc Rasella says he sold shares of Twitter at “artificially deflated prices,” unaware that Musk had made a large purchase in the social media platform.

### RAG using Llama2

query = "write a summary for this ? "

print(str(query + outlier\_paragraph)[:512])

write a summary for this ? Twitch Bans Gambling Sites After Streamer Scams Folks Out Of $200,000 One man's claims that he scammed people on the platform caused several popular streamers to consider a Twitch boycott. TikTok Search Results Riddled With Misinformation: Report A U.S. firm that monitors false online claims reports that searches for information about prominent news topics on TikTok are likely to turn up results riddled with misinformation. Citing Imminent Danger Cloudflare Drops Hate Site Kiwi Fa

from llama2backend import generate\_text

RAG\_answer = generate\_text(str(query + outlier\_paragraph)[:512])

RAG\_answer

'... Cloudflare, a company that provides web infrastructure services, has dropped its support for the hate site Kiwi Farms after citing imminent danger to users.\nTwitch Bans Gambling Sites After Streamer Scams Folks Out Of $200,000\nA popular streamer on Twitch has been banned from the platform after he scammed viewers out of over $200,000 through a gambling scheme. The streamer, who goes by the name "AdventureZ," had been promoting a gambling site on his channel for months, encouraging his viewers to deposit money and participate in games of chance. However, the site was not legitimate and AdventureZ was not authorized to promote it. As a result, Twitch has banned AdventureZ from the platform and is working with law enforcement to investigate the matter further.\nThe incident has raised concerns about gambling on Twitch and the potential for fraudulent activities to occur on the platform. In response, Twitch has announced that it will be implementing new measures to prevent gambling-related content from being promoted on the site. These measures'

## Refferences

[1] “Realtime Market News. Live BSE, NSE, Stock Prices, Expert Stock Advice, Share Market Updates : Rediff.com.” Accessed: Apr. 18, 2024. [Online]. Available: https://money.rediff.com/news

[2] “heathbrew/Vector-Databases-can-help-with-Anomaly-Detection: Pinecone , reddiffmoney financial dataset.” Accessed: Apr. 18, 2024. [Online]. Available: https://github.com/heathbrew/Vector-Databases-can-help-with-Anomaly-Detection

[3] A. Pranav, “Steps to Monitoring DSPy-Qdrant Powered RAG with Prometheus or Grafana,” DevOps.dev. Accessed: Apr. 18, 2024. [Online]. Available: https://blog.devops.dev/steps-to-monitoring-dspy-qdrant-powered-rag-with-prometheus-or-grafana-b642335cbd50

[4] “sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 · Hugging Face.” Accessed: Apr. 18, 2024. [Online]. Available: https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

[5] A. Pranav, “How to Install and Run Llama2 Locally on Windows for Free,” Medium. Accessed: Apr. 18, 2024. [Online]. Available: https://medium.com/@AyushmanPranav/how-to-install-and-run-llama2-locally-on-windows-for-free-05bd5032c6e3

[6] “News Category Dataset.” Accessed: Apr. 18, 2024. [Online]. Available: https://www.kaggle.com/datasets/rmisra/news-category-dataset