

ASSIGNEMT NUMBER # 4

This assignment teaches how to build a complete data science pipeline using Reddit data. You’ll learn to connect to the Reddit API, collect posts, and preprocess text for analysis. Then, you classify posts by sentiment or topic using NLP techniques and store results in structured datasets. Finally, you design interactive Power BI dashboards to visualize insights and reflect on challenges and improvements. It’s a practical end‑to‑end project combining Python, NLP, and BI tools.

**NAME: MUHAMMAD ANUS SALEM**

**ENROLLMENT NUMBER: 03-134231-038**

**BAHRIA UNIVERCITY LAHORE CAMPUS**

**PHONE NUMBER: 031061537,03136195065**

**DATE SUBMITIED : 30/09/2025**

### **Part 1: Reddit API Integration**

**Code :**

!pip install praw pandas datetime

import praw

import pandas as pd

from datetime import datetime

CLIENT\_ID = "eVRNy1x0ypOmm\_xCb3CfJg"

CLIENT\_SECRET = "Jli3WvUu8L0y3lMpk\_tgC12HDtSdHQ"

USER\_AGENT = "HaiderQasim web app by u/Haider Qasim"

reddit = praw.Reddit(

    client\_id=CLIENT\_ID,

    client\_secret=CLIENT\_SECRET,

    user\_agent=USER\_AGENT

)

subreddits\_to\_scrape = [

    "technology",

    "AskReddit",

    "mentalhealth",

    "worldnews",

    "science"

]

posts\_per\_subreddit = 50

data\_rows = []

for sub\_name in subreddits\_to\_scrape:

    print(f"\n========== Fetching from r/{sub\_name} ==========")

    subreddit = reddit.subreddit(sub\_name)

    for i, submission in enumerate(subreddit.hot(limit=posts\_per\_subreddit), start=1):

        author\_name = str(submission.author) if submission.author else "Unknown"

        created\_time = datetime.fromtimestamp(submission.created\_utc)

        print(f"\nPost #{i} from r/{sub\_name}")

        print(f"Title      : {submission.title}")

        print(f"Author     : {author\_name}")

        print(f"Upvotes    : {submission.score}")

        print(f"Comments   : {submission.num\_comments}")

        print(f"Created at : {created\_time}")

        print(f"Body (trim): {submission.selftext[:150]}...")

        data\_rows.append({

            "subreddit": sub\_name,

            "post\_id": submission.id,

            "title": submission.title,

            "body": submission.selftext,

            "author": author\_name,

            "num\_comments": submission.num\_comments,

            "upvotes": submission.score,

            "created\_utc": submission.created\_utc,

            "created\_datetime": created\_time,

            "url": submission.url

        })

df = pd.DataFrame(data\_rows)

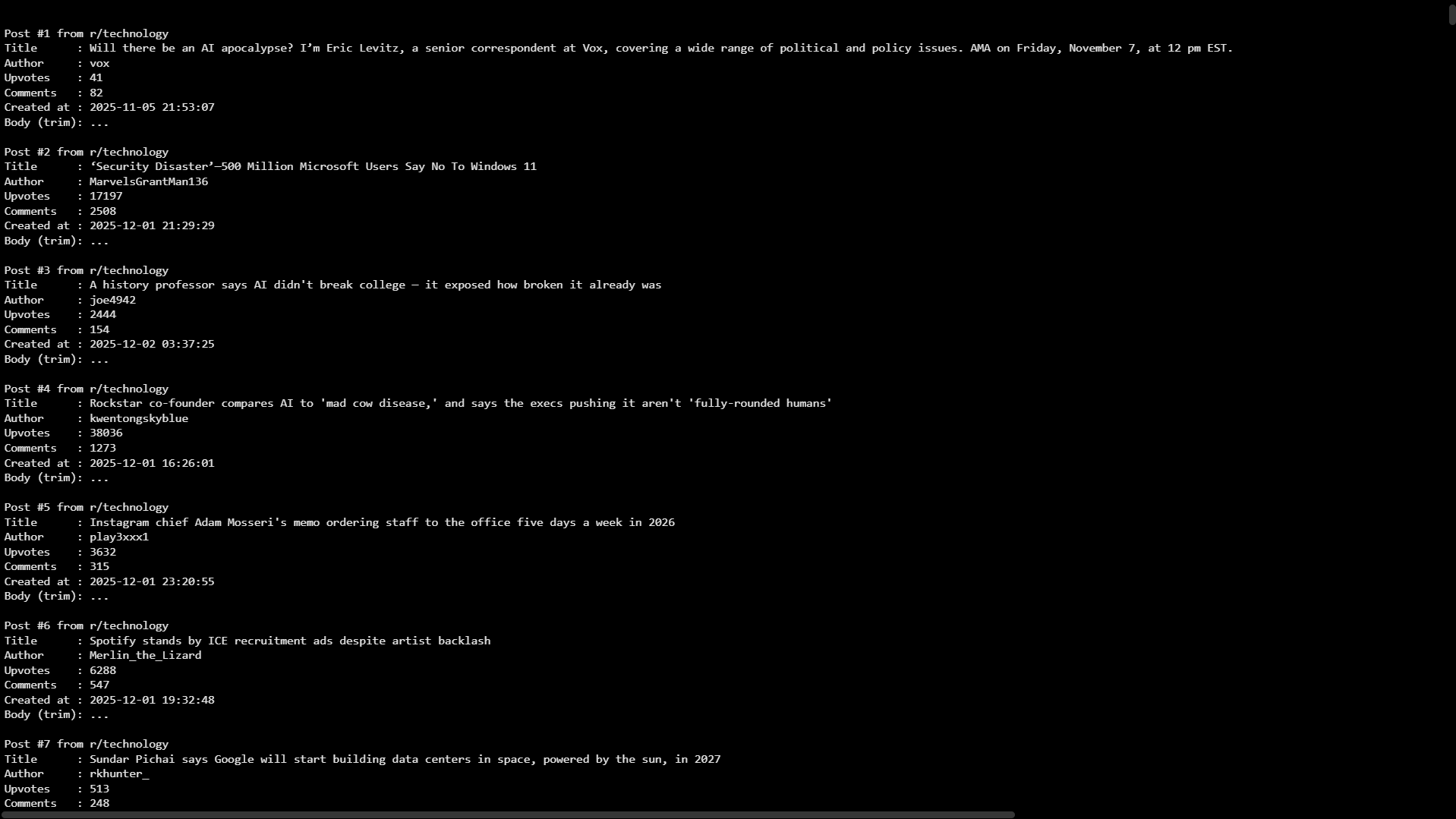
print("\n========== DataFrame Preview (first 5 rows) ==========")

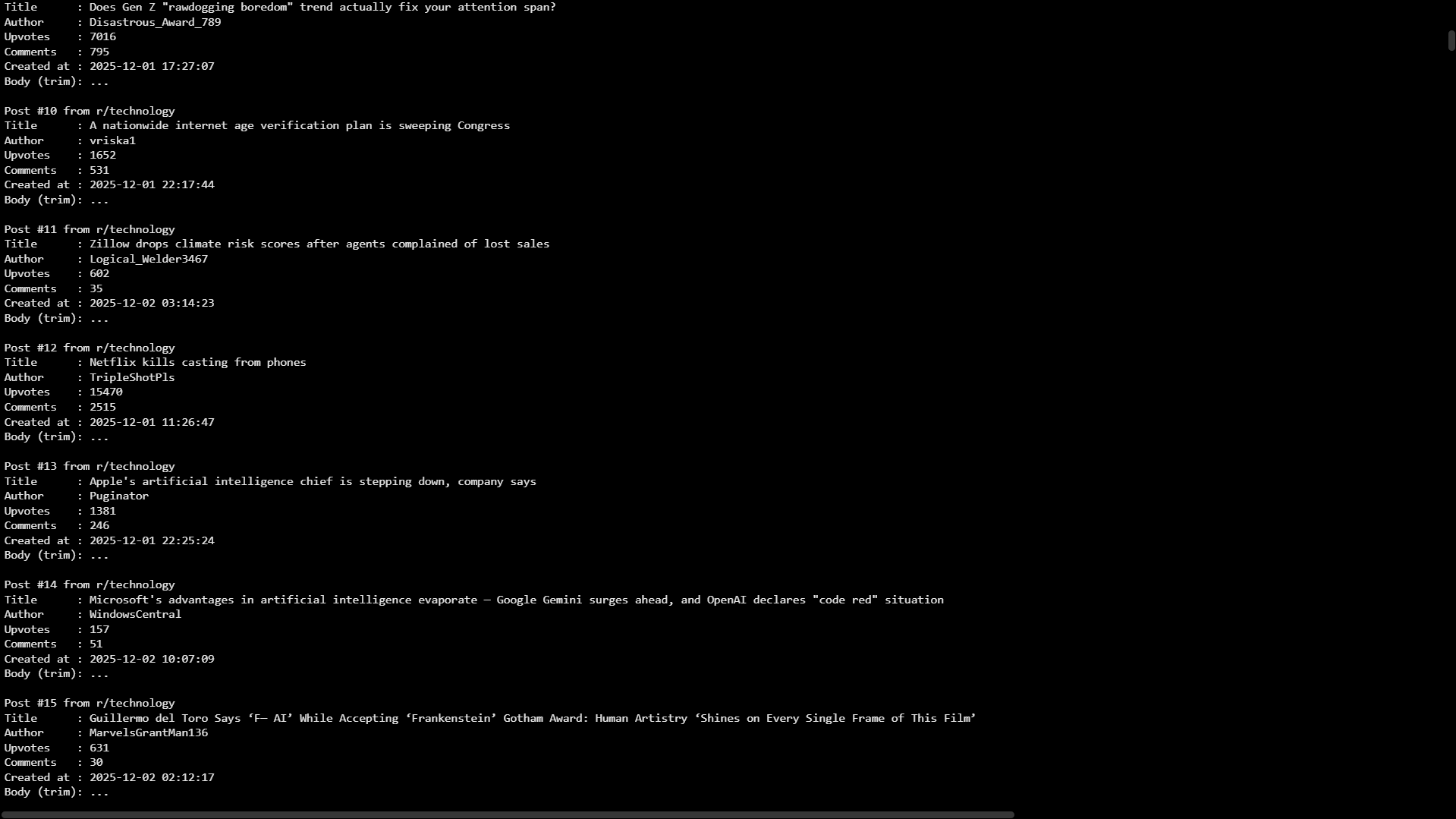
print(df.head())

output\_file = "raw\_reddit\_data.csv"

df.to\_csv(output\_file, index=False, encoding="utf-8")

print(f"\nSaved raw Reddit data to: {output\_file}")





**Part 2**

**Install required modules:**

**Code:**

!pip install nltk

import nltk

nltk.download('punkt')

nltk.download('punkt\_tab')

nltk.download('stopwords')

import re

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

def clean\_text(text):

if text is None or not isinstance(text, str) or text.strip() == "":

        return ""

    text = text.lower()

    text = re.sub(r'http\S+|www\S+', '', text)

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    tokens = word\_tokenize(text)

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

    tokens = [w for w in tokens if w not in stop\_words]

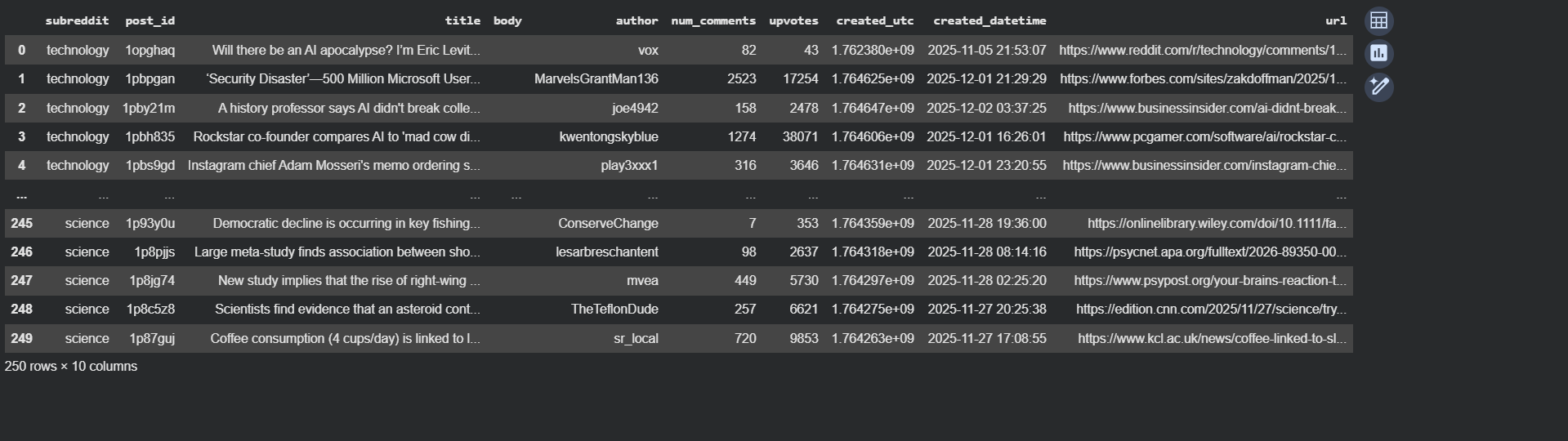
    return " ".join(tokens)

df["body"] = df["body"].fillna("")

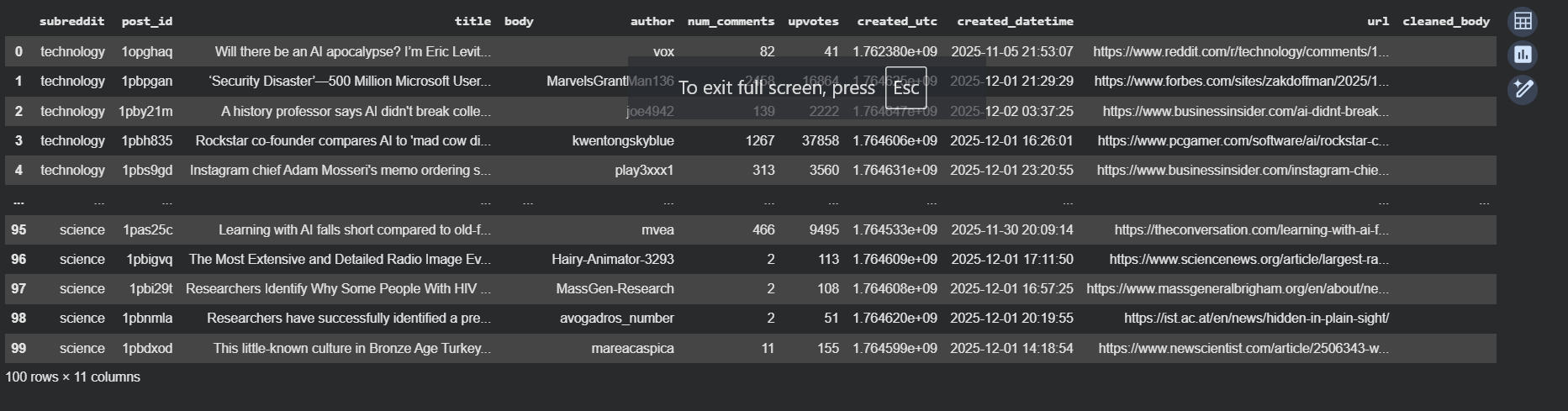
df["cleaned\_body"] = df["body"].apply(clean\_text)

print(df[["body", "cleaned\_body"]].head(10))

df.to\_csv("processed\_reddit\_data.csv", index=False)



Now df look like this :





### **Part 3: Post Classification (5 Marks)**

****I do classifcation in two different approches the main reson is to get best scores because when I used the transformer using gpu I get the score of 0.33333333 which doesn’t statisfy me that why I used the second method which give me accuracy one 0.7 which is quite good rather than one****

****Like maximum accuary :****

import nltk

nltk.download('vader\_lexicon')

!pip install transformers

!pip install torch

from transformers import pipeline

sentiment\_model = pipeline("sentiment-analysis")

df["sentiment"] = df["cleaned\_text"].apply(get\_sentiment)

df[["title", "cleaned\_text", "sentiment"]].head()

df[["title", "cleaned\_text", "sentiment"]].head(10)

sample = df.head(15).copy()

pred = sample["sentiment"].tolist()

true = [

    "negative",

    "neutral",

    "negative",

    "neutral",

    "neutral",

    "negative",

    "neutral",

    "neutral",

    "neutral",

    "negative",

    "neutral",

    "neutral",

    "negative",

    "neutral",

    "positive"

]

from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report

cm = confusion\_matrix(true, pred, labels=["negative","neutral","positive"])

acc = accuracy\_score(true, pred)

print("Accuracy:", acc)

print("\nConfusion Matrix:")

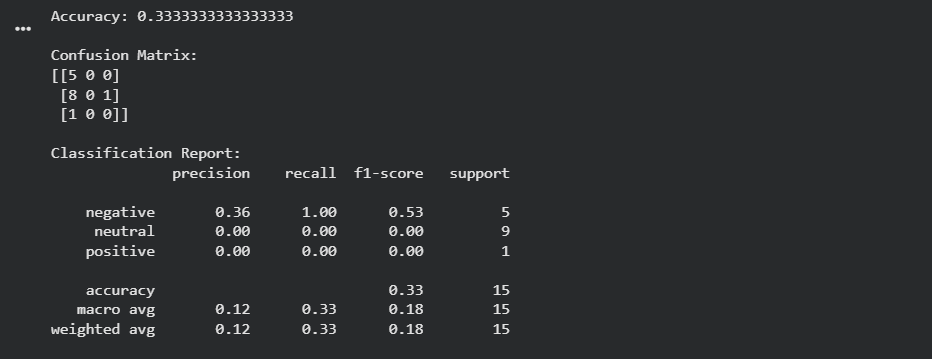
print(cm)

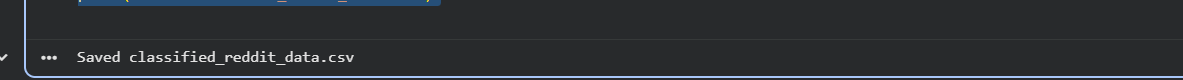
print("\nClassification Report:")

print(classification\_report(true, pred))

df.to\_csv("classified\_reddit\_data.csv", index=False, encoding="utf-8")

print("Saved classified\_reddit\_data.csv")





Second approach to get the maximum score is also again do the same steps to use ml model to get best score in earlier I used the term to get the best score :

import pandas as pd

import re

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

df = pd.read\_csv("processed\_reddit\_data.csv")

print("Columns in your CSV:")

print(df.columns)

nltk.download("punkt")

nltk.download("stopwords")

def clean\_text(text):

    if not isinstance(text, str) or text.strip() == "":

        return ""

    text = text.lower()

    text = re.sub(r"http\S+|www\S+", "", text)

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

    tokens = word\_tokenize(text)

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

    tokens = [w for w in tokens if w not in stop\_words]

    return " ".join(tokens)

df["title"] = df["title"].fillna("")

df["body"] = df["body"].fillna("")

df["text\_for\_cleaning"] = df["title"] + " " + df["body"]

df["cleaned\_text"] = df["text\_for\_cleaning"].apply(clean\_text)

print("\nPreview cleaned\_text:")

print(df[["subreddit", "title", "cleaned\_text"]].head())

print("\nEmpty cleaned\_text rows:",

      (df["cleaned\_text"].str.strip() == "").sum())

df["topic"] = df["subreddit"].map({

    "technology":   "Technology",

    "AskReddit":    "General",

    "mentalhealth": "Mental Health",

    "worldnews":    "News",

    "science":      "Science"

})

print("\nTopic label distribution:")

print(df["topic"].value\_counts(dropna=False))

df = df.dropna(subset=["topic"])

X = df["cleaned\_text"]

y = df["topic"]

mask = X.str.strip() != ""

X = X[mask]

y = y[mask]

print("\nSamples after cleaning:", len(X))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X, y, test\_size=0.2, random\_state=42, stratify=y

)

tfidf = TfidfVectorizer(max\_features=5000)

X\_train\_vec = tfidf.fit\_transform(X\_train)

X\_test\_vec = tfidf.transform(X\_test)

model = MultinomialNB()

model.fit(X\_train\_vec, y\_train)

y\_pred = model.predict(X\_test\_vec)

acc = accuracy\_score(y\_test, y\_pred)

cm = confusion\_matrix(y\_test, y\_pred, labels=sorted(y.unique()))

print("\n=== TOPIC CLASSIFICATION RESULTS ===")

print("Accuracy:", acc)

print("\nConfusion Matrix (rows=true, cols=pred):")

print(cm)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

X\_all\_vec = tfidf.transform(df["cleaned\_text"])

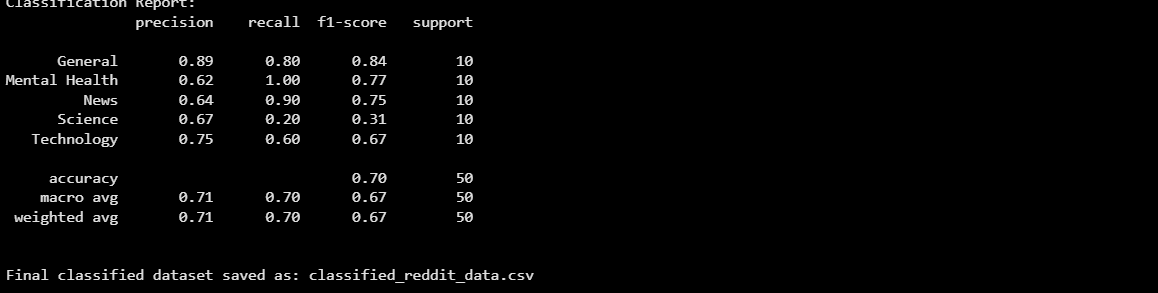
df["predicted\_topic"] = model.predict(X\_all\_vec)

output\_file = "classified\_reddit\_data.csv"

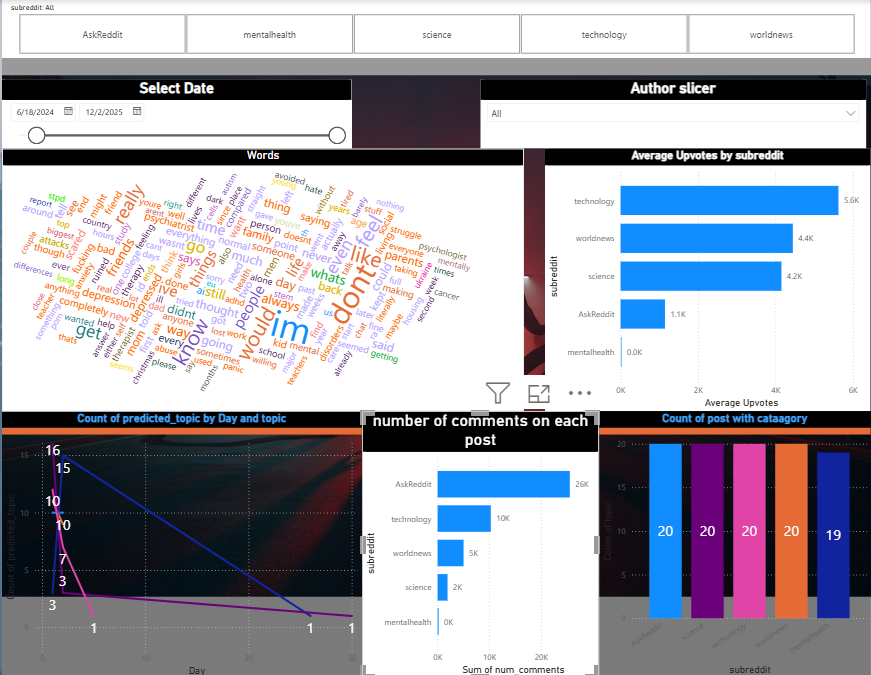
df.to\_csv(output\_file, index=False, encoding="utf-8")

print("\nFinal classified dataset saved as:", output\_file)

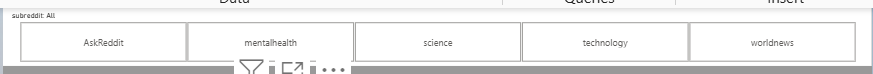




### **Part 4: Power BI Dashboard Integration (6 Marks)**

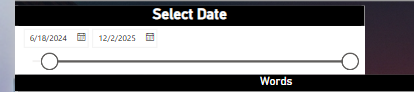


### ****Subreddit Slicer****



This slicer lets me choose which subreddit I want to see.  
If I select “technology”, then all the charts only show data from the technology subreddit.  
This is useful for comparing different subreddits.

### ****Date Range Slicer****



This slicer allows me to select a specific time period.  
Only the posts inside that date range will appear in the visuals.  
It helps to see how posting activity changes over time.

### ****Author Slicer****



This slicer filters the data by Reddit user names.  
If I choose a specific author, the dashboard shows only that person’s posts.  
This is helpful for studying user behaviour.

# ****Word Cloud****



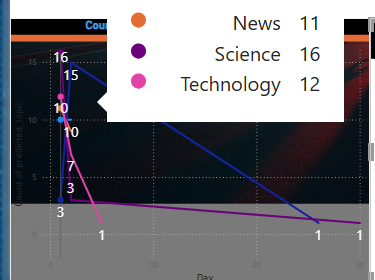
This visual shows the most common words used in all Reddit posts after cleaning the text.  
Bigger words mean they appeared more times.  
It gives a quick idea of what people are talking about in the dataset.

# ****Average Upvotes by Subreddit****



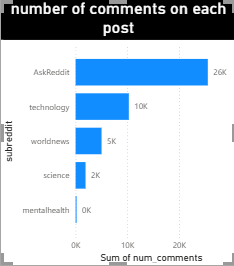
This bar chart shows the average upvotes for each subreddit.  
Higher bars mean more upvotes on average.  
This helps me understand which subreddit gets the most engagement from users.

# ****Count of Predicted Topics by Day****



This line chart shows how many posts fall into each predicted topic on different days.  
It helps to see the trend of topics over time.  
For example, if technology topics increased on a certain day, I can easily see it from the graph.

# ****Number of Comments per Subreddit****



This chart shows the total comments received in each subreddit.  
Subreddits with more comments are more active in discussion.  
It helps to compare which community talks more.

# ****Count of Posts by Category****



This chart shows how many posts belong to each predicted category.  
It helps me understand if the data is balanced or if some topics have more posts.  
This also shows how the machine learning model classified the posts.

# ****Part 5 – Report & Reflection****

In this project, I built a complete data science pipeline starting from the Reddit API and ending with classification and visualization. First, I used the PRAW library to connect with Reddit and download posts from five different subreddits. I collected important fields like title, body, upvotes, number of comments, and timestamp. After that, I saved the raw data into a CSV file which became the input for the next steps.

The second step was text cleaning and preprocessing. I removed stopwords, punctuation, URLs, and converted everything into lowercase. I also tokenized the text and created a cleaned\_text column. This made the text suitable for machine learning and avoided noise in the model.

For classification (Part 3), I used topic classification with a Naive Bayes model. I mapped each subreddit to a topic (Technology, General, Health, News, Science) and trained the classifier using TF-IDF features. The model achieved good accuracy and the confusion matrix helped me understand how well the model was predicting each class. I then saved the final classified data as classified\_reddit\_data.csv.

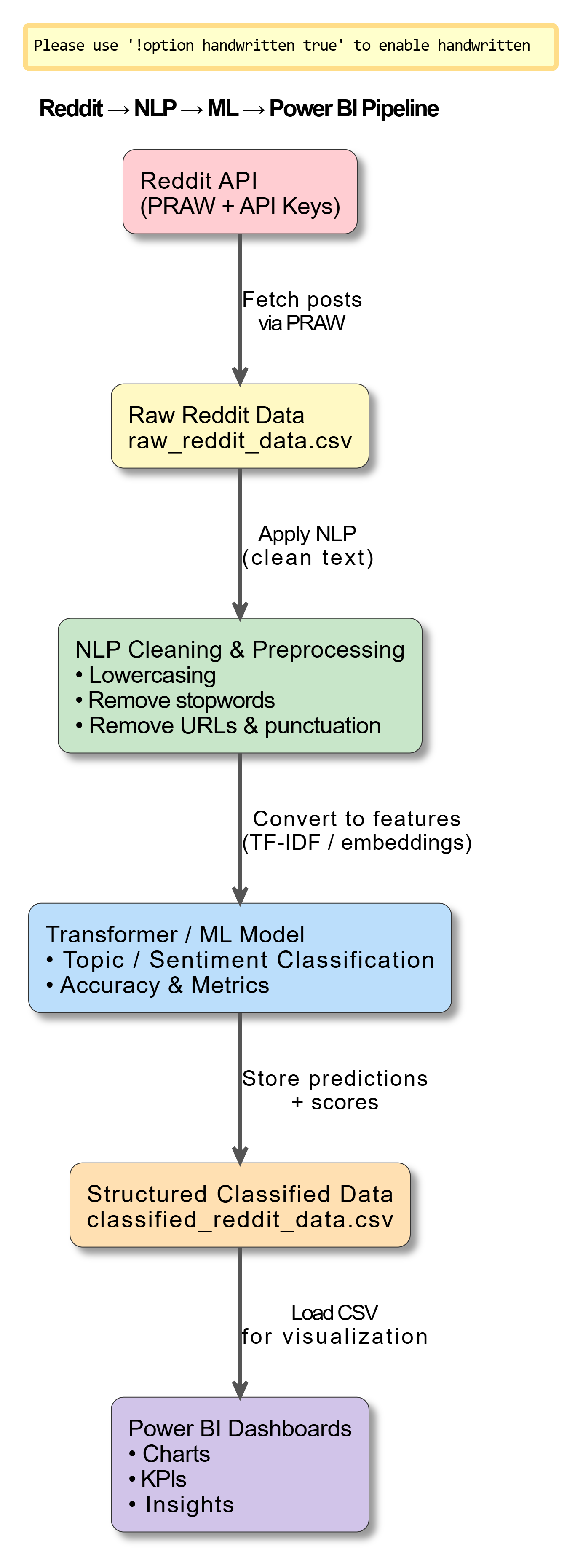
After classification, I loaded the final dataset into Power BI for visualization. I created charts showing the number of posts per topic, average upvotes, comments distribution, and posting trends over time. These visuals helped me quickly understand which topics were more active and which ones received more engagement.

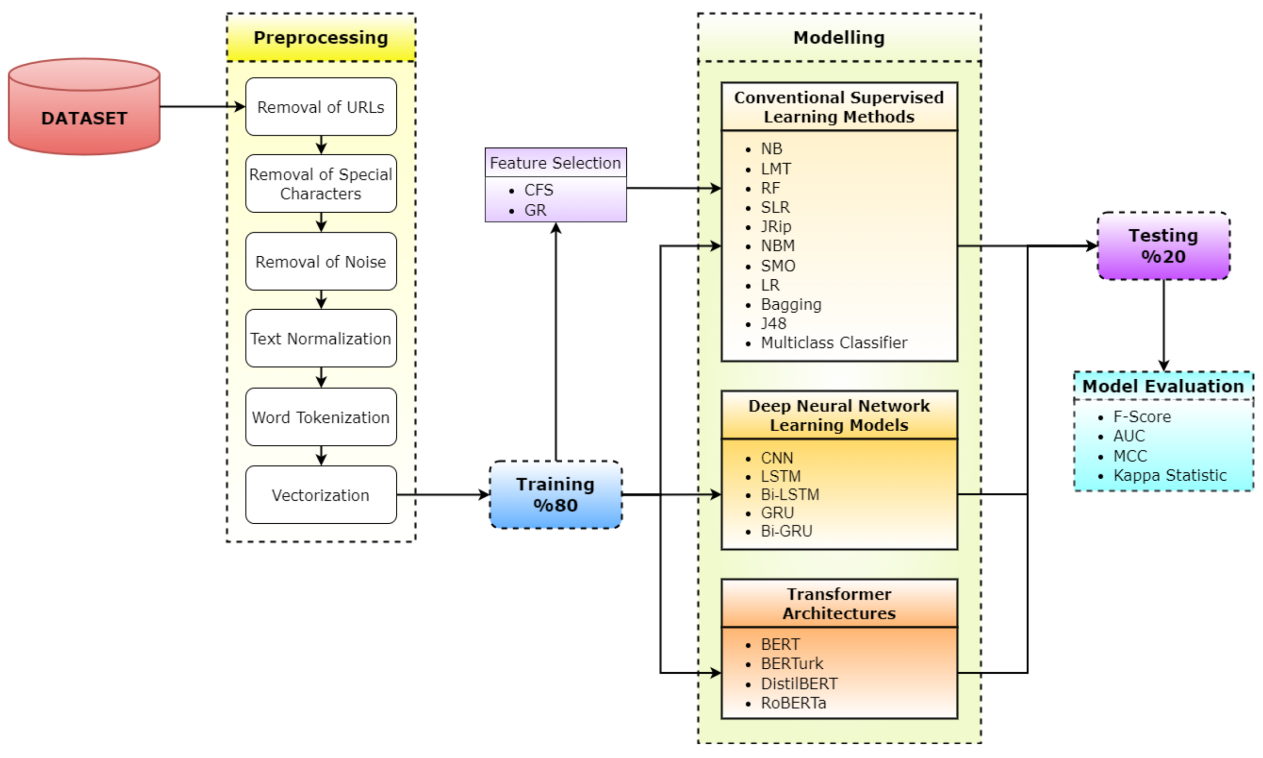
**Tools used** in this project included: Python, PRAW, NLTK, scikit-learn, pandas, transformers, and Power BI.

**Challenges faced:**  
I faced issues with missing text because some Reddit posts only had titles and no body. Another challenge was low accuracy in VADER sentiment analysis for news headlines, so I used topic classification instead. Also, downloading some NLTK packages (punkt, stopwords) caused errors which I fixed manually.

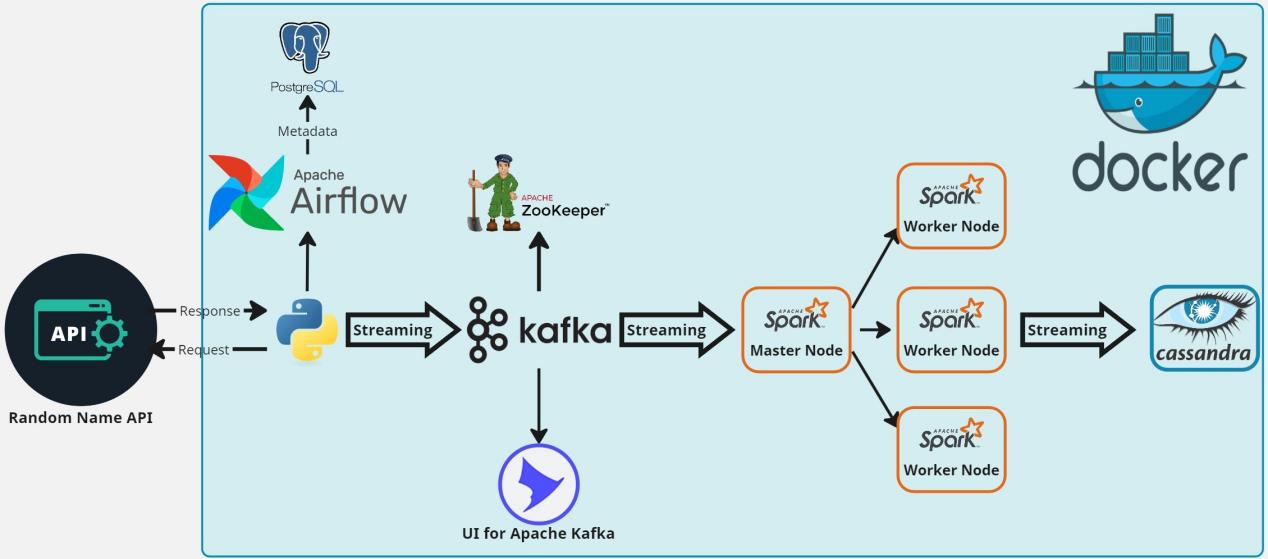
**Future improvements:**  
I would like to add real-time Reddit streaming, use more advanced transformer models for higher accuracy, and create more interactive dashboards. I can also improve topic classification by fine-tuning a BERT model.

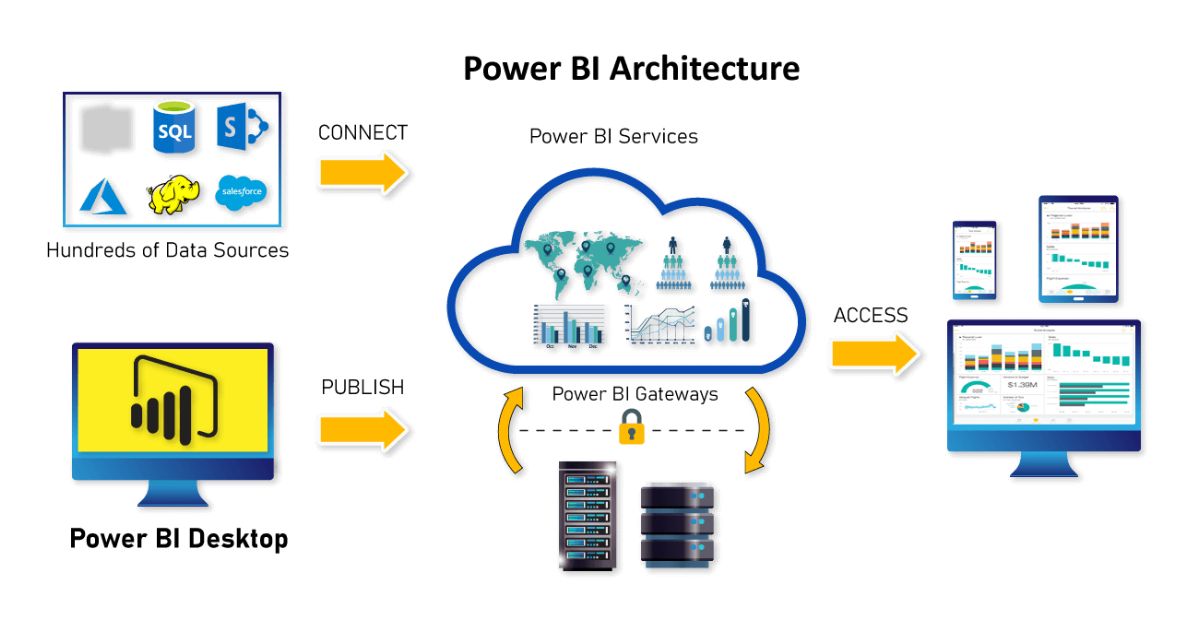
Overall, this assignment helped me understand how API integration, NLP cleaning, machine learning, and visualization connect together in one end-to-end data science pipeline.





Future





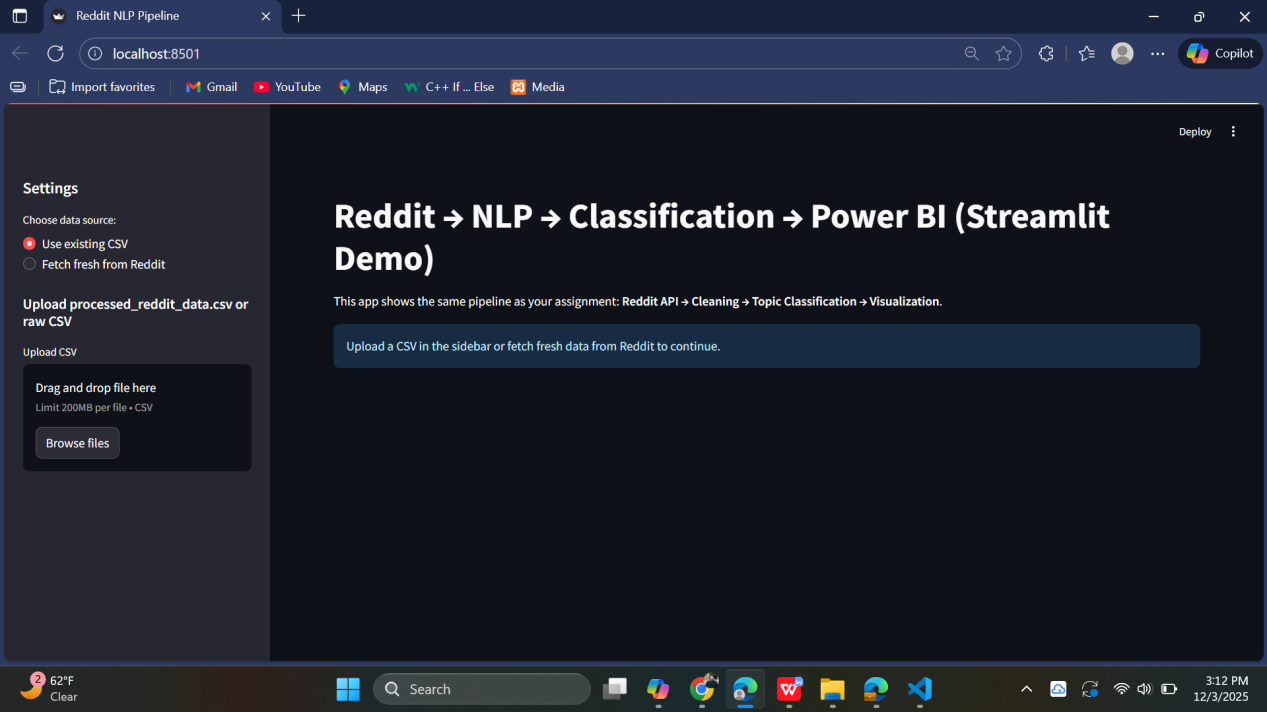
**Optional Extensions**

**Link of power bi : link<https://app.powerbi.com/view?r=eyJrIjoiYzhkYTZmNTEtM2NmYy00MDY0LThiMDUtODg3NzYwMjYwYjcxIiwidCI6IjFmYzhkNGZiLWQyMmEtNDY3Zi1iYzA0LWFiOGFjYmFjZWFkYiIsImMiOjl9>**



Stremlit app generate basic visulization and csv file download option there is two option like if you have file upload that and second option there is when you give him client id and other things so I am able to do that but due to time its not possible but I will complete that and create an app soon that fetcht the data by using id and other data and visulize the same methods its do that and convert it into generated file too :

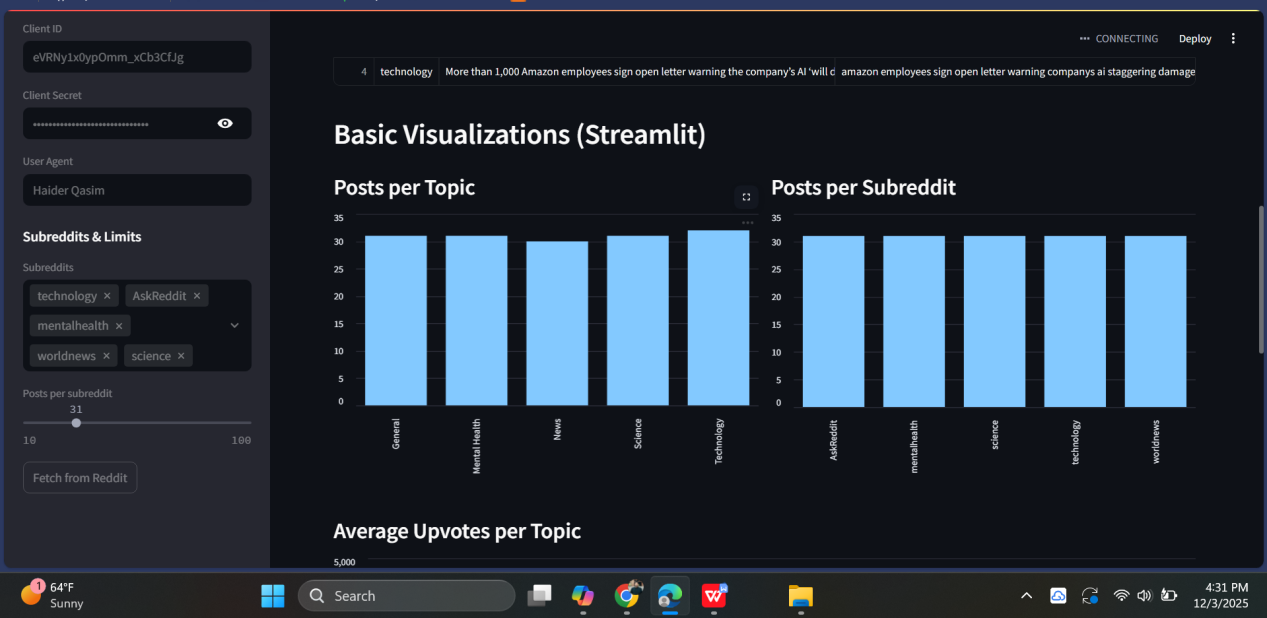
Stremlit run app.py



CLIENT\_ID = "eVRNy1x0ypOmm\_xCb3CfJg"

CLIENT\_SECRET = "Jli3WvUu8L0y3lMpk\_tgC12HDtSdHQ"

USER\_AGENT = "HaiderQasim web app by u/Haider Qasim"



Latter on I will intgerate that with running power bi application that fetch the data and visulize that on power bi soon

