INDEX

S. No	Title	Signature
1.	To study various social Media Platforms (Facebook, Twitter, Instagram, YouTube, LinkedIn etc).	
2.	To study the Social Media Analytics tools used in different layers of social media analytics	
3.	Write a program to connect and capture social media data for business (select a Social media platform of your choice).	
4.	Write a program to visualize the most frequent words used in the dataset.	
5.	Use a sentiment analysis tool to classify tweets as positive, negative, or neutral.	
6.	Implement a time series analysis to predict the future popularity of specific hashtags or keywords based on past tweet data.	
7.	Use machine learning models to predict the number of likes, share or retweets based on platform (of your choice) content and metadata.	
8.	Develop a Social Media Text Analytics model to improve existing products or services by analyzing customer reviews and comments.	
9.	Use a network analytics tool to build a graph using a real-world social network dataset.(e.g., social media, co-authorship, citation networks, election). a. Explore graph characteristics: number of nodes, edges, density, and components. b. Calculate the degree centrality for each node in the network. c. Compute betweenness centrality, which measures the extent to which a node lies on the shortest path between other nodes. d. Compute closeness centrality, which measures how close a node is to all other nodes in the network. e. Compute eigenvector centrality, which identifies nodes that are connected to other important nodes.	

10.	Create a hyperlink network from the extracted links. a. Represent the network using a graph structure where nodes are web pages, and edges are hyperlinks. b. Apply the PageRank algorithm to the hyperlink network to identify important pages.	
11.	Write a program to analyze the movement data (e.g., how users travel between locations).	
12.	Write a program to understand how retailers use location analytics to analyze foot traffic patterns and optimize store placement..	

Practical 1:To study various social media platforms (Facebook, Twitter, Instagram, YouTube, LinkedIn etc).

a. Facebook

Facebook, launched in 2004 by Mark Zuckerberg, is one of the largest and most widely used social media platforms globally. It allows users to create personal profiles, share posts, photos, and videos, connect with friends, and join groups or communities based on common interests.

- **Key Features**:
  - **News Feed**: Personalized updates from friends, family, pages, and groups.
  - **Groups**: Communities where users can interact around shared interests.
  - **Marketplace**: A platform for buying and selling items.
  - **Messenger**: A separate messaging app for instant communication.
  - **Events**: Feature for organizing or RSVP-ing to events.
- **Challenges:**
  - **Privacy Concerns**: Facebook has been involved in numerous scandals related to data privacy.
  - **Misinformation and Fake News**: The platform has struggled to tackle the spread of fake news, especially around elections and public health.
  - **Algorithmic Bias**: The algorithms of Facebook's news feed can create "filter bubbles" by showing users content that reinforces their existing beliefs, limiting exposure to diverse viewpoints.
  - **Declining Younger Audience**: Younger users are increasingly turning to platforms like Instagram and TikTok, leading to a potential decline in Facebook's relevance among this demographic.

b. Twitter

Twitter, launched in 2006, is a microblogging platform that allows users to post and interact with short messages called "tweets." With a character limit (originally 140, now 280 characters), it emphasizes quick, real-time communication.

- **Key Features**:
  - **Tweets**: Short posts (text, photos, videos, or links).
  - **Trending Topics**: A section for popular discussions and hashtags.
  - **Retweets & Likes**: Ways to share and show support for posts.

    - **Hashtags**: Used to categorize and discover content.
    - **Lists**: Allows users to categorize and organize accounts they follow.
- **Challenges**:
    - **Trolling and Harassment**: Twitter has struggled with toxic behavior such as trolling, harassment, and hate speech.
    - **Misinformation**: Similar to Facebook, Twitter has been criticized for allowing the rapid spread of false information.
    - **Bots and Fake Accounts**: Automated accounts (bots) can flood Twitter with spammy content, fake news, and disinformation, undermining trust in the platform.

c. Instagram

Instagram, launched in 2010, is a photo and video-sharing platform, now owned by Meta (Facebook). It focuses on visual content, allowing users to share images, videos, and stories. Instagram also has features for messaging and shopping.

- **Key Features**:
    - **Posts**: Static images and videos shared on a user's profile.
    - **Stories**: Temporary content that disappears after 24 hours.
    - **Reels**: Short-form video content similar to TikTok.
    - **Direct Messages**: Private messaging feature.
- **Challenges:**
    - **Mental Health Impact**: Instagram has been linked to negative impacts on mental health, especially among teens, due to body unrealistic beauty standards, and comparison culture. Studies have shown that the platform can contribute to anxiety, depression, and low self-esteem.
    - **Fake Influencers and Bots**: The rise of influencer culture has led to problems with fake followers, fake engagement, and influencer fraud. Many brands rely on influencer marketing but struggle to assess authenticity.
    - **Algorithm Limitations**: Instagram's algorithm prioritizes engagement (likes, comments, shares), which has led to "clickbait" content that may not be authentic or informative, but gets attention because of sensationalism.
    - **Privacy and Data Collection**: Instagram, being part of Meta (formerly Facebook), faces similar privacy concerns.

d. YouTube

YouTube, founded in 2005 and acquired by Google in 2006, is the world's largest video-sharing platform. Users can upload, watch, like, share, and comment on videos. It caters to all kinds of content, from educational videos to entertainment, music, and vlogging.

- **Key Features**:
  - **Channels**: Personal or brand accounts where users can upload videos.
  - **Subscriptions**: Allows users to follow channels and receive notifications.
  - **Live Streaming**: Real-time video broadcasts.
  - **Comments and Likes**: Interaction with content creators and viewers.
  - **Monetization**: Content creators can earn revenue from ads, sponsorships, and memberships.
- **Challenges**:
  - **Content Moderation**: YouTube faces criticism for allowing harmful content, such as hate speech, conspiracy theories, and explicit material, to remain on the platform despite efforts to enforce stricter guidelines.
  - **Copyright Issues**: YouTube has long struggled with copyright infringement, where videos featuring copyrighted material (such as music, video clips, or even memes) can be removed or demonetized.
  - **Algorithmic Recommendations**: YouTube's recommendation algorithm sometimes promotes controversial or extreme content because it generates more views and engagement, contributing to radicalization and misinformation.

e. LinkedIn

LinkedIn, launched in 2003, is a professional networking platform designed to connect professionals, job seekers, and businesses. It is used to share work-related achievements, post job listings, and build professional relationships.

- **Key Features**:
  - **Profile**: Personal or company pages showcasing professional experience and qualifications.
  - **Networking**: Connecting with professionals, colleagues, or industry peers.
  - **Job Listings**: Posting and applying for jobs.
  - **Endorsements & Recommendations**: Users can endorse skills or write recommendations for others.
  - **Content Sharing**: Articles, posts, and videos related to professional growth and business.

- **Uses**:
  - Professional networking, career development, and job searching.
  - Industry thought leadership and content sharing.
  - Company promotion and employee recruitment.
  - Business B2B (business-to-business) marketing.
- **Challenges**:
  - **Privacy Concerns**: LinkedIn collects a lot of personal and professional data, leading to concerns about privacy and how that information is shared or used by third parties.
  - **Spam and Inappropriate Content**: LinkedIn has become increasingly plagued by spam messages, irrelevant connection requests, and "sales pitches" that can make the platform feel less professional and more like a commercial space.
  - **Algorithmic Challenges**: The platform’s algorithm sometimes prioritizes posts that generate higher engagement, which can lead to a focus on sensational content instead of meaningful, career-related insights.
  - **Diversity and Inclusion**: Despite LinkedIn's claims of being a diverse platform, there are ongoing concerns about underrepresentation of certain groups in higher-level job opportunities, particularly women, racial minorities, and other marginalized groups.

Practical 2: To study the social media analytics tools used in different layers of social media analytics.

1. Text Layer:

This layer involves tools that analyze textual data found in social media platforms. Tools in this category help in discovering, analyzing, and interpreting the content of posts, tweets, comments, and other written forms of communication.

- **Discover Text**: This tool helps in text mining and sentiment analysis. It allows users to extract relevant social media content (tweets, posts, etc.), analyze the text for keywords, themes, and sentiment, and uncover patterns in user conversations. It's useful for understanding public opinion, trends, and key topics.
- **Twitonomy**: Specifically focused on Twitter, Twitonomy offers deep analytics into Twitter accounts, tweets, mentions, and followers. It helps users analyze the content of tweets, measure engagement, and track the activity of specific users or topics over time.

2. Actions Layer:

The actions layer focuses on tools that track and analyze user actions and behaviors, such as clicks, shares, likes, retweets, etc. These tools help evaluate the impact of actions taken on social media, such as engagement metrics.

- **Google Analytics**: While commonly used for website analytics, Google Analytics also tracks social media traffic and engagement. It provides insights into how social media interactions (like clicks or shares) affect website traffic and conversions, helping businesses track the ROI of their social media campaigns.
- **Twitonomy** (again): Besides analyzing textual content, Twitonomy can track engagement metrics, including retweets, likes, and replies. It's valuable for understanding user behavior on Twitter and the performance of specific content.

3. Network Layer:

This layer deals with tools that analyze social networks, connections, and relationships between users or entities. Network analysis tools allow users to visualize and understand how people or topics are connected and identify key influencers.

- **NodeXL**: This is a powerful network analysis tool that integrates with Microsoft Excel to visualize and analyze social media networks. It helps in mapping relationships between users, content, or hashtags on platforms like Twitter. NodeXL is useful for detecting influential nodes (people or topics) within a network.

- **NetMiner**: Similar to NodeXL, NetMiner is a software tool designed for social network analysis. It provides advanced features for mapping relationships, detecting clusters, and performing network analysis on large datasets from social media.

4. Apps Layer:

The apps layer refers to tools used for analyzing mobile app data and user engagement across different platforms, often providing insights into mobile-specific behaviors like app downloads, in-app interactions, and retention rates.

- **Google Mobile Analytics**: A mobile version of Google Analytics, this tool tracks mobile app performance, user engagement, and acquisition. It's essential for understanding how users interact with mobile applications, including data related to social media integrations within those apps.
- **Countly**: This is a real-time mobile analytics platform that provides insights into mobile apps, websites, and digital products. It tracks user behavior, interactions, and social media engagements within the app, offering analytics to optimize user experience and retention.

5. Location Layer:

Tools in the location layer focus on geospatial data and help analyze social media activities in terms of location-based metrics, such as geo-tagged posts, regional trends, and geographic distribution of users.

- **Google Fusion Tables**: A data visualization tool that allows users to analyze and visualize geospatial data. In the context of social media, it can be used to map location-based interactions or visualize the geographic distribution of social media users or trends.
- **Trend Maps**: A tool for visualizing trends over time across different regions. It provides location-based insights and helps businesses or organizations understand regional variations in social media behavior, engagement, and sentiment.

6. Hyperlinks Layer:

This layer focuses on analyzing the links shared and spread across social media platforms, including the impact of those links on traffic, engagement, and influence.

- **Webometrics Analyst**: A tool used for analyzing the web's hyperlink structure. It focuses on understanding the links between websites, blogs, or social media profiles. It helps track how content spreads and gains visibility across the internet through shared links.

- **VOSON**: The Virtual Observatory for the Study of Online Networks (VOSON) analyzes the structure and dynamics of social networks, including the role of hyperlinks. It provides insights into how web content is interlinked and how hyperlinks contribute to the spread of information or influence on the web.

7. Search Engines Layer:

This layer focuses on tools related to search engine behavior and the influence of search engines on social media. It helps in understanding how social media content is indexed and how search engine results reflect social media trends.

- **Google**: As a search engine, Google is crucial for understanding how social media content and discussions appear in search results. It also plays a significant role in determining SEO (Search Engine Optimization) and ranking of social media content across different platforms.
- **Yahoo**: Similar to Google, Yahoo also offers search engine results that can provide insights into how social media content is indexed. Though less dominant than Google, it still plays a role in the broader search ecosystem, influencing how content from social media is ranked and found online.

Program 3: Write a program to connect and capture social media data for business (select a Social media platform of your choice).

```
# Program 3: Write a program to connect and capture social media data for business (select a Social media platform of your choice).

import os
from time import sleep
from dotenv import load_dotenv

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager

load_dotenv()

my_user = os.getenv("TWITTER_USERNAME")
my_pass = os.getenv("TWITTER_PASSWORD")

search_item = "Business"

# Use ChromeDriverManager to automatically manage the ChromeDriver
service = Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service)
driver.get("https://twitter.com/i/flow/login")

# Wait for the username field to be present
WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.NAME, "text")))

username = driver.find_element(By.NAME, "text")
username.send_keys(my_user)
username.send_keys(Keys.RETURN)

# Wait for the password field to be present
WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.NAME, "password")))

password = driver.find_element(By.NAME, "password")
password.send_keys(my_pass)
password.send_keys(Keys.RETURN)

sleep(3)

# Scrape Tweets mentioning about Business
search_box = driver.find_element(By.XPATH, "//input[@data-testid='SearchBox_Search_Input']")
search_box.send_keys(search_item)
search_box.send_keys(Keys.ENTER)

all_tweets = set()

tweets = driver.find_elements(By.XPATH, "//div[@data-testid='tweetText']")
while True:
    for tweet in tweets:
        all_tweets.add(tweet.text)
    driver.execute_script('window.scrollTo(0, document.body.scrollHeight);')
    sleep(3)
    tweets = driver.find_elements(By.XPATH, "//div[@data-testid='tweetText']")
    if len(all_tweets) > 100:
        break

all_tweets = list(all_tweets)

# Save tweets to a .txt file
with open('tweets.txt', mode='w', encoding='utf-8') as file:
    for tweet in all_tweets:
        file.write(tweet + "\n")

print(f"Saved {len(all_tweets)} tweets to tweets.txt")
```

Conclusion:

The program successfully demonstrates a foundational approach to social media data collection for business purposes. By automating the process of logging in, searching, and capturing tweets, it provides a practical tool for gathering public sentiment or identifying trends related to a specific keyword. This data can be valuable for businesses in understanding consumer behavior, monitoring brand reputation, or conducting market analysis.

sma-4

November 23, 2024

1 Practical 4: Write a program to visualize the most frequent words used in the dataset.

[ ]:
```
!pip install pandas matplotlib wordcloud nltk
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (3.8.0)
Requirement already satisfied: wordcloud in /usr/local/lib/python3.10/dist-
packages (1.9.4)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.9.1)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-
packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk) (8.1.7)
```

```
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk) (4.66.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

[ ]:
```
import pandas as pd
import nltk
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk.corpus import stopwords
from collections import Counter
import re
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[ ]: True

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!unzip /content/drive/MyDrive/Social\ Media\ Analytics/amazon-fine-food-reviews.
 ↪zip
```

```
Archive:  /content/drive/MyDrive/Social Media Analytics/amazon-fine-food-
reviews.zip
  inflating: Reviews.csv
  inflating: database.sqlite
  inflating: hashes.txt
```

[ ]:
```
# Read in data
df = pd.read_csv(f'/content/Reviews.csv')
print(df.shape)
df = df.head(500)
print(df.shape)
```

```
(568454, 10)
(500, 10)
```

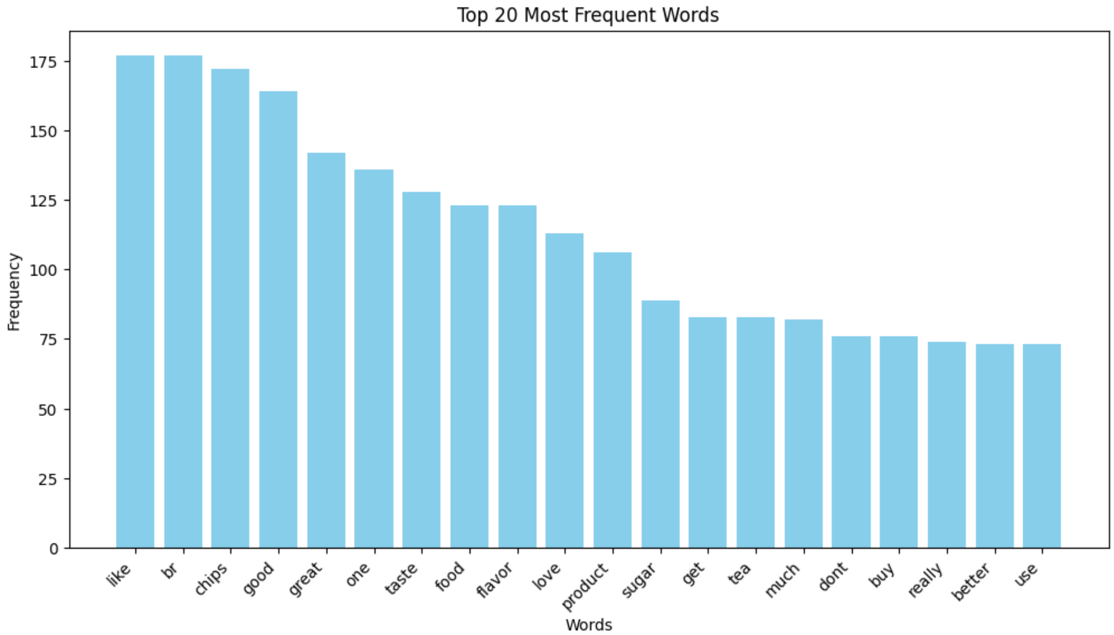
[ ]:
```
# Preprocess the review text
def preprocess_text(text):
```

```
    text = text.lower()
    # Remove non-alphabetic characters (keep words only)
    text = re.sub(r'[^a-z\s]', '', text)
    return text

df['cleaned_review'] = df['Text'].apply(preprocess_text)
```

[ ]:
```
# Get a list of all words in the dataset
all_words = ' '.join(df['cleaned_review']).split()

# Remove stop words
stop_words = set(stopwords.words('english'))
filtered_words = [word for word in all_words if word not in stop_words]
```

[ ]:
```
# Count the frequency of each word using Counter
word_counts = Counter(filtered_words)

# Get the most common words (top 20 most frequent words)
most_common_words = word_counts.most_common(20)

# Extract the words and their frequencies for plotting
words, frequencies = zip(*most_common_words)
```

[ ]:
```
# bar plot of the top 20 most frequent words
plt.figure(figsize=(12, 6))
plt.bar(words, frequencies, color='skyblue')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 20 Most Frequent Words')
plt.xticks(rotation=45, ha='right')
plt.show()
```



[ ]:
```
# word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate_from_frequencies(word_counts)

# Display the word cloud
plt.figure(figsize=(12, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Most Frequent Words')
plt.show()
```

2 Conclusion

The Notebook effectively demonstrates the process of extracting, preprocessing, and visualizing text data to identify the most frequently used words in a dataset. By employing popular Python libraries such as pandas for data handling, nltk for text preprocessing, and matplotlib and wordcloud for visualization, the workflow provides valuable insights into textual trends.

sma-5-tool

November 23, 2024

1 Program 5: Use a sentiment analysis tool to classify tweets as positive, negative or neutral.

2 Step 0. Read in Data and NLTK Basics

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')

import nltk
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('maxent_ne_chunker_tab')
nltk.download('words')
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data…
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data…
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker_tab to
[nltk_data]     /root/nltk_data…
[nltk_data]   Package maxent_ne_chunker_tab is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data…
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data…
```

[ ]: True

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!unzip /content/drive/MyDrive/Social\ Media\ Analytics/amazon-fine-food-reviews.
↪zip
```

```
Archive:  /content/drive/MyDrive/Social Media Analytics/amazon-fine-food-
reviews.zip
  inflating: Reviews.csv
  inflating: database.sqlite
  inflating: hashes.txt
```

[ ]:
```
# Read in data
df = pd.read_csv(f'/content/Reviews.csv')
print(df.shape)
df = df.head(500)
print(df.shape)
```

```
(568454, 10)
(500, 10)
```

[ ]:
```
df.head()
```

[ ]:
```
   Id   ProductId          UserId                      ProfileName  \
0   1  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
1   2  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
2   3  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
3   4  B000UA0QIQ  A395BORC6FGVXV                             Karl
4   5  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"

   HelpfulnessNumerator  HelpfulnessDenominator  Score        Time  \
0                     1                       1      5  1303862400
1                     0                       0      1  1346976000
2                     1                       1      4  1219017600
3                     3                       3      2  1307923200
4                     0                       0      5  1350777600

                 Summary                                               Text
0  Good Quality Dog Food  I have bought several of the Vitality canned d…
1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut…
2  "Delight" says it all  This is a confection that has been around a fe…
3         Cough Medicine  If you are looking for the secret ingredient i…
4            Great taffy  Great taffy at a great price.  There was a wid…
```

2.1 Quick EDA

[ ]:
```
ax = df['Score'].value_counts().sort_index() \
    .plot(kind='bar',
          title='Count of Reviews by Stars',
          figsize=(10, 5))
ax.set_xlabel('Review Stars')
plt.show()
```

2.2 Basic NLTK

[ ]:
```
example = df['Text'][50]
print(example)
```

```
This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is the
way to go.
```

[ ]:
```
tokens = nltk.word_tokenize(example)
tokens[:10]
```

[ ]:
```
['This', 'oatmeal', 'is', 'not', 'good', '.', 'Its', 'mushy', ',', 'soft']
```

[ ]:
```
tagged = nltk.pos_tag(tokens)
tagged[:10]
```

```
[ ]: [('This', 'DT'),
      ('oatmeal', 'NN'),
      ('is', 'VBZ'),
      ('not', 'RB'),
      ('good', 'JJ'),
      ('.', '.'),
      ('Its', 'PRP$'),
      ('mushy', 'NN'),
      (',', ','),
      ('soft', 'JJ')]
```

```
[ ]: entities = nltk.chunk.ne_chunk(tagged)
     entities.pprint()
```

```
(S
  This/DT
  oatmeal/NN
  is/VBZ
  not/RB
  good/JJ
  ./.
  Its/PRP$
  mushy/NN
  ,/,
  soft/JJ
  ,/,
  I/PRP
  do/VBP
  n't/RB
  like/VB
  it/PRP
  ./.
  (ORGANIZATION Quaker/NNP Oats/NNPS)
  is/VBZ
  the/DT
  way/NN
  to/TO
  go/VB
  ./.)
```

3 Step 1. VADER Seniment Scoring

We will use NLTK's `SentimentIntensityAnalyzer` to get the neg/neu/pos scores of the text.

- This uses a "bag of words" approach:
    1. Stop words are removed
    2. each word is scored and combined to a total score.

[ ]:
```python
from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm

sia = SentimentIntensityAnalyzer()
```

[ ]:
```python
sia.polarity_scores('I am so happy!')
```

[ ]:
```
{'neg': 0.0, 'neu': 0.318, 'pos': 0.682, 'compound': 0.6468}
```

[ ]:
```python
sia.polarity_scores('This is the worst thing ever.')
```

[ ]:
```
{'neg': 0.451, 'neu': 0.549, 'pos': 0.0, 'compound': -0.6249}
```

[ ]:
```python
sia.polarity_scores(example)
```

[ ]:
```
{'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

[ ]:
```python
# Run the polarity score on the entire dataset
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    text = row['Text']
    myid = row['Id']
    res[myid] = sia.polarity_scores(text)
```

```
  0%|          | 0/500 [00:00<?, ?it/s]
```

[ ]:
```python
vaders = pd.DataFrame(res).T
vaders = vaders.reset_index().rename(columns={'index': 'Id'})
vaders = vaders.merge(df, how='left')
```

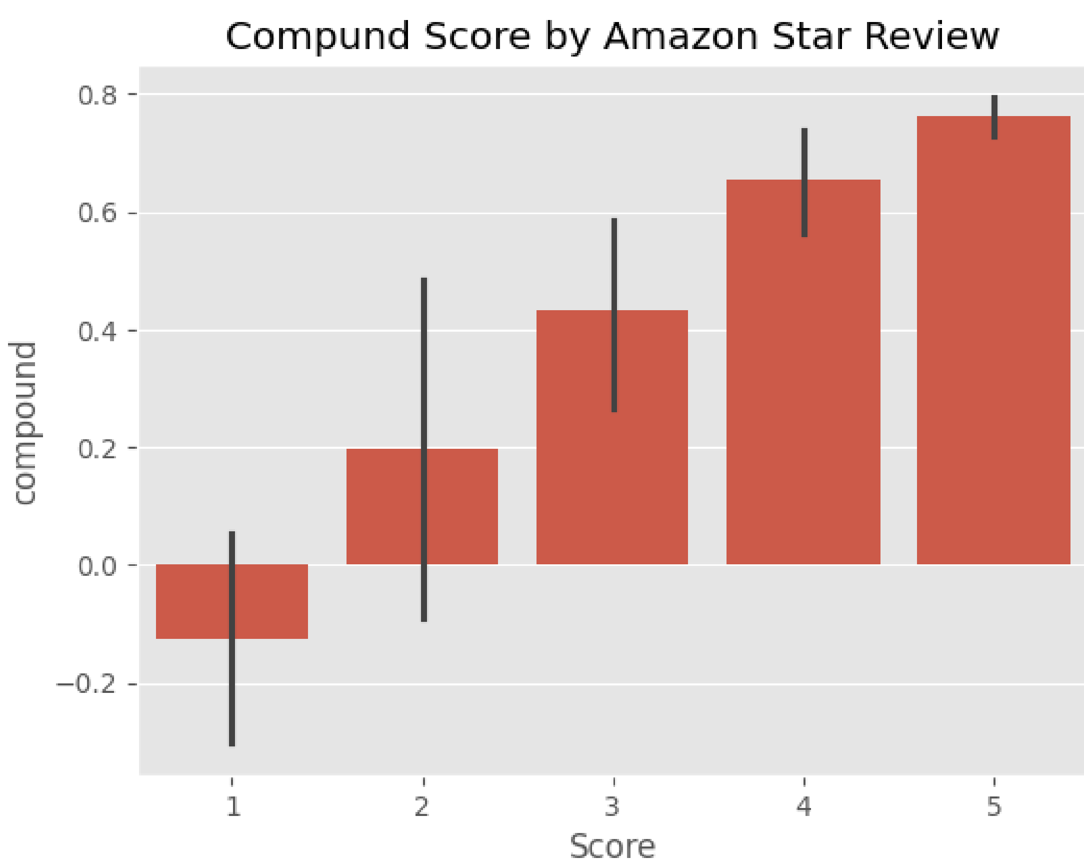
[ ]:
```python
# Now we have sentiment score and metadata
vaders.head()
```

[ ]:
```
   Id    neu    neu    pos  compound   ProductId          UserId  \
0   1  0.000  0.695  0.305    0.9441  B001E4KFG0  A3SGXH7AUHU8GW
1   2  0.138  0.862  0.000   -0.5664  B00813GRG4  A1D87F6ZCVE5NK
2   3  0.091  0.754  0.155    0.8265  B000LQOCH0   ABXLMWJIXXAIN
3   4  0.000  1.000  0.000    0.0000  B000UA0QIQ  A395BORC6FGVXV
4   5  0.000  0.552  0.448    0.9468  B006K2ZZ7K  A1UQRSCLF8GW1T

                      ProfileName  HelpfulnessNumerator  \
0                      delmartian                     1
1                          dll pa                     0
2  Natalia Corres "Natalia Corres"                     1
3                            Karl                     3
4    Michael D. Bigham "M. Wassir"                     0
```

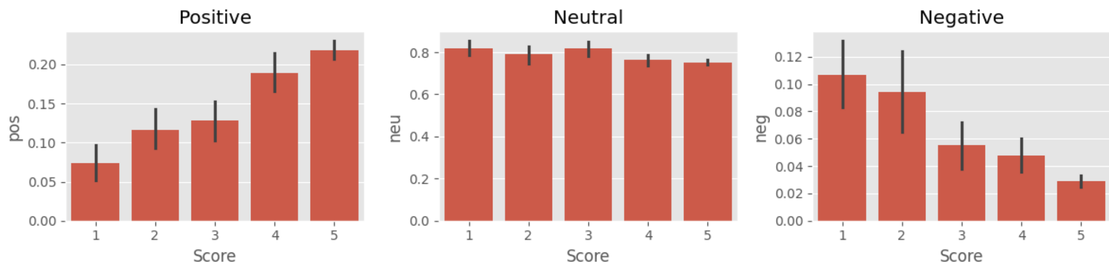
```
   HelpfulnessDenominator  Score        Time                Summary  \
0                       1      5  1303862400  Good Quality Dog Food
1                       0      1  1346976000      Not as Advertised
2                       1      4  1219017600  "Delight" says it all
3                       3      2  1307923200         Cough Medicine
4                       0      5  1350777600            Great taffy

                                                Text
0  I have bought several of the Vitality canned d…
1  Product arrived labeled as Jumbo Salted Peanut…
2  This is a confection that has been around a fe…
3  If you are looking for the secret ingredient i…
4  Great taffy at a great price.  There was a wid…
```

3.1 Plot VADER results

[ ]:
```python
ax = sns.barplot(data=vaders, x='Score', y='compound')
ax.set_title('Compund Score by Amazon Star Review')
plt.show()
```



[ ]:
```
fig, axs = plt.subplots(1, 3, figsize=(12, 3))
sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0])
sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1])
sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2])
axs[0].set_title('Positive')
axs[1].set_title('Neutral')
axs[2].set_title('Negative')
plt.tight_layout()
plt.show()
```



4 Step 4: Review Examples:

- Positive 1-Star and Negative 5-Star Reviews

Lets look at some examples where the model scoring and review score differ the most.

[ ]:
```
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    text = row['Text']
    myid = row['Id']
    vader_result = sia.polarity_scores(text)
    vader_result_rename = {}
    for key, value in vader_result.items():
        vader_result_rename[f"vader_{key}"] = value
    res[myid] = vader_result_rename
```

```
  0%|          | 0/500 [00:00<?, ?it/s]
```

[ ]:
```
results_df = pd.DataFrame(res).T
results_df = results_df.reset_index().rename(columns={'index': 'Id'})
results_df = results_df.merge(df, how='left')
```

[ ]:
```
results_df.query('Score == 1') \
    .sort_values('vader_pos', ascending=False)['Text'].values[0]
```

[ ]:
```
'So we cancelled the order.  It was cancelled without any problem.  That is a
positive note…'
```

[ ]:
```python
# negative sentiment 5-Star view
results_df.query('Score == 5') \
    .sort_values('vader_neg', ascending=False)['Text'].values[0]
```

[ ]:
```
'this was sooooo deliscious but too bad i ate em too fast and gained 2 pds! my
fault'
```

5 Extra: The Transformers Pipeline

- Quick & easy way to run sentiment predictions

[ ]:
```python
from transformers import pipeline

sent_pipeline = pipeline("sentiment-analysis")
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-
finetuned-sst-2-english and revision 714eb0f
(https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-
sst-2-english).
Using a pipeline without specifying a model name and revision in production is
not recommended.
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(
```

```
config.json:   0%|          | 0.00/629 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
```

[ ]:
```python
sent_pipeline('I love sentiment analysis!')
```

[ ]:
```
[{'label': 'POSITIVE', 'score': 0.9997853636741638}]
```

[ ]:
```python
sent_pipeline('Make sure to like and subscribe!')
```

[ ]:
```
[{'label': 'POSITIVE', 'score': 0.9991742968559265}]
```

[ ]:
```python
sent_pipeline('booo')
```

```
[ ]: [{'label': 'NEGATIVE', 'score': 0.9936267137527466}]
```

6 Conclusion

The notebook demonstrates the application of sentiment analysis to classify textual data into positive, negative, or neutral categories. By leveraging tools like the VADER sentiment analyzer and integrating it with Python's robust data handling libraries, the program offers a straightforward yet powerful method for deriving insights from unstructured text.

sma-6

November 23, 2024

1 Program 6: Implement a time series analysis to predict the future popluarity of specific hashtags or keywords based on past tweet data.

2 Importing Dependencies

[ ]:
```
import numpy as np
import pandas as pd

import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

from textblob import TextBlob

import matplotlib.pyplot as plt

from wordcloud import WordCloud

import seaborn as sns

from PIL import Image

import nltk
nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data…
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data] Downloading package omw-1.4 to /root/nltk_data…
```

[ ]: True

3 Dataset

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!unzip /content/drive/MyDrive/Social\ Media\ Analytics/Twitter-Dataset.zip
```

```
Archive:  /content/drive/MyDrive/Social Media Analytics/Twitter-Dataset.zip
  inflating: download.jpg
  inflating: twitter_dataset.csv
```

[ ]:
```
df=pd.read_csv("/content/twitter_dataset.csv")
```

[ ]:
```
df.head()
```

[ ]:
```
   Tweet_ID         Username  \
0         1          julie81
1         2    richardhester
2         3  williamsjoseph
3         4      danielsmary
4         5       carlwarren

                                                Text  Retweets  Likes  \
0  Party least receive say or single. Prevent pre…         2     25
1  Hotel still Congress may member staff. Media d…        35     29
2  Nice be her debate industry that year. Film wh…        51     25
3  Laugh explain situation career occur serious. …        37     18
4  Involve sense former often approach government…        27     80

             Timestamp
0  2023-01-30 11:00:51
1  2023-01-02 22:45:58
2  2023-01-18 11:25:19
3  2023-04-10 22:06:29
4  2023-01-24 07:12:21
```

[ ]:
```
df.shape
```

[ ]:
```
(10000, 6)
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
```

```
---  ------     --------------  -----
 0   Tweet_ID   10000 non-null  int64
 1   Username   10000 non-null  object
 2   Text       10000 non-null  object
 3   Retweets   10000 non-null  int64
 4   Likes      10000 non-null  int64
 5   Timestamp  10000 non-null  object
dtypes: int64(3), object(3)
memory usage: 468.9+ KB
```

[ ]:
```
df.describe()
```

[ ]:

	Tweet_ID	Retweets	Likes
count	10000.00000	10000.000000	10000.000000
mean	5000.50000	49.721200	49.929300
std	2886.89568	28.948856	28.877193
min	1.00000	0.000000	0.000000
25%	2500.75000	25.000000	25.000000
50%	5000.50000	49.000000	50.000000
75%	7500.25000	75.000000	75.000000
max	10000.00000	100.000000	100.000000

[ ]:
```
df.isnull().sum()
```

[ ]:
```
Tweet_ID     0
Username     0
Text         0
Retweets     0
Likes        0
Timestamp    0
dtype: int64
```

[ ]:
```
df.duplicated().sum()
```

[ ]: 0

4 Feature Engineering

[ ]:
```
# Remove duplicate tweets
# df = df.drop_duplicates()

# Remove rows with missing values
# df = df.dropna()

# Clean tweet text by removing special characters and URLs
df['Text'] = df['Text'].str.replace('[^a-zA-Z0-9\s]', '')
```

```
df['Text'] = df['Text'].str.replace('http\S+|www.\S+', '', case=False)
```

5 Porter Stemmer

[ ]:
```
# Tokenize tweet text
df['tokens'] = df['Text'].apply(lambda x: nltk.word_tokenize(x))

# Remove stopwords
stop_words = set(stopwords.words('english'))
df['tokens'] = df['tokens'].apply(lambda x: [word for word in x if word.lower()␣
↪not in stop_words])

# Stemming or Lemmatization
stemmer = PorterStemmer()
df['tokens'] = df['tokens'].apply(lambda x: [stemmer.stem(word) for word in x])
```

[ ]:
```
# Calculate summary statistics
mean_retweets = df['Retweets'].mean()
median_likes = df['Likes'].median()
correlation = df['Retweets'].corr(df['Likes'])
```

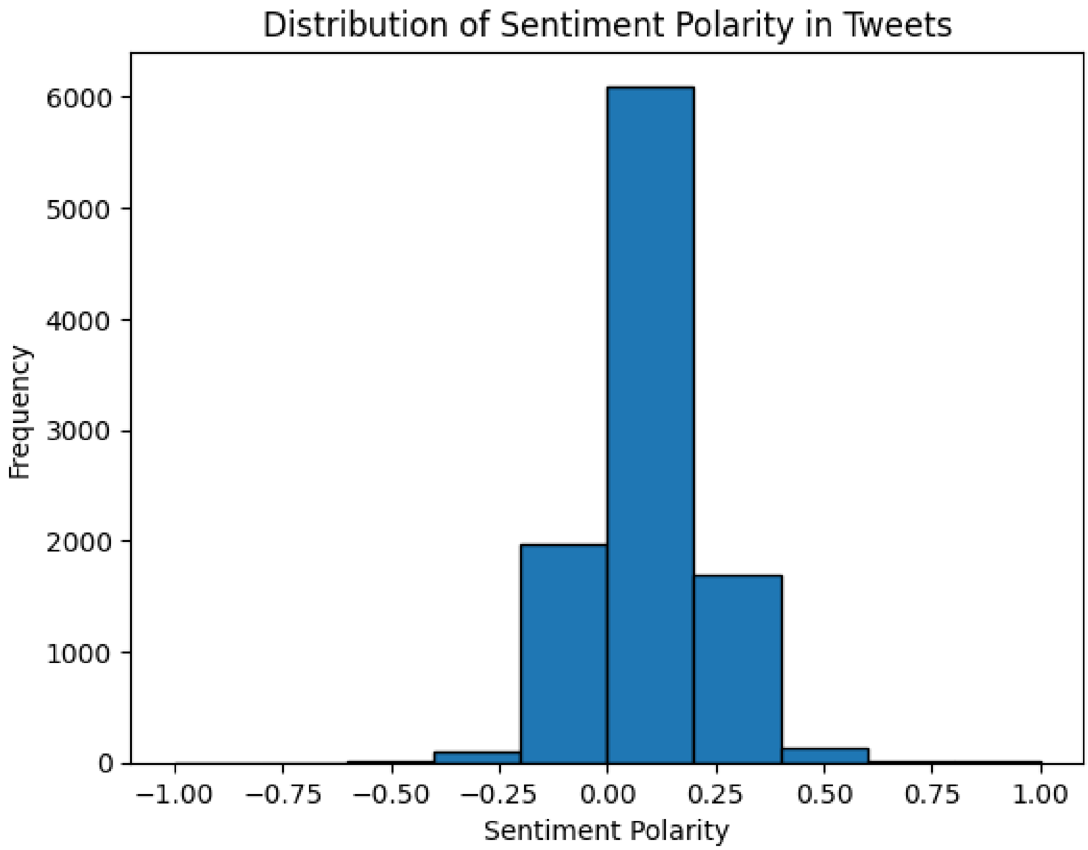
[ ]:
```
# Perform sentiment analysis on tweet text
df['sentiment_polarity'] = df['Text'].apply(lambda x: TextBlob(x).sentiment.
↪polarity)
```

[ ]:
```
# Print results
print("Mean Retweets:", mean_retweets)
print("Median Likes:", median_likes)
print("Correlation between Retweets and Likes:", correlation)
```

```
Mean Retweets: 49.7212
Median Likes: 50.0
Correlation between Retweets and Likes: 0.012797546201034809
```

6 Histplot (Histogram Plot)

[ ]:
```
# Plotting sentiment polarity distribution
plt.hist(df['sentiment_polarity'], bins=10, range=(-1, 1), edgecolor='black')
plt.xlabel('Sentiment Polarity')
plt.ylabel('Frequency')
plt.title('Distribution of Sentiment Polarity in Tweets')
plt.show()
```



[ ]:
```python
# Print the number of rows and columns in the dataset
print("Number of Rows:", df.shape[0])
print("Number of Columns:", df.shape[1])

# Calculate the average values of retweets and likes
avg_retweets = df['Retweets'].mean()
avg_likes = df['Likes'].mean()
print("Average Retweets:", avg_retweets)
print("Average Likes:", avg_likes)
```

```
Number of Rows: 10000
Number of Columns: 8
Average Retweets: 49.7212
Average Likes: 49.9293
```

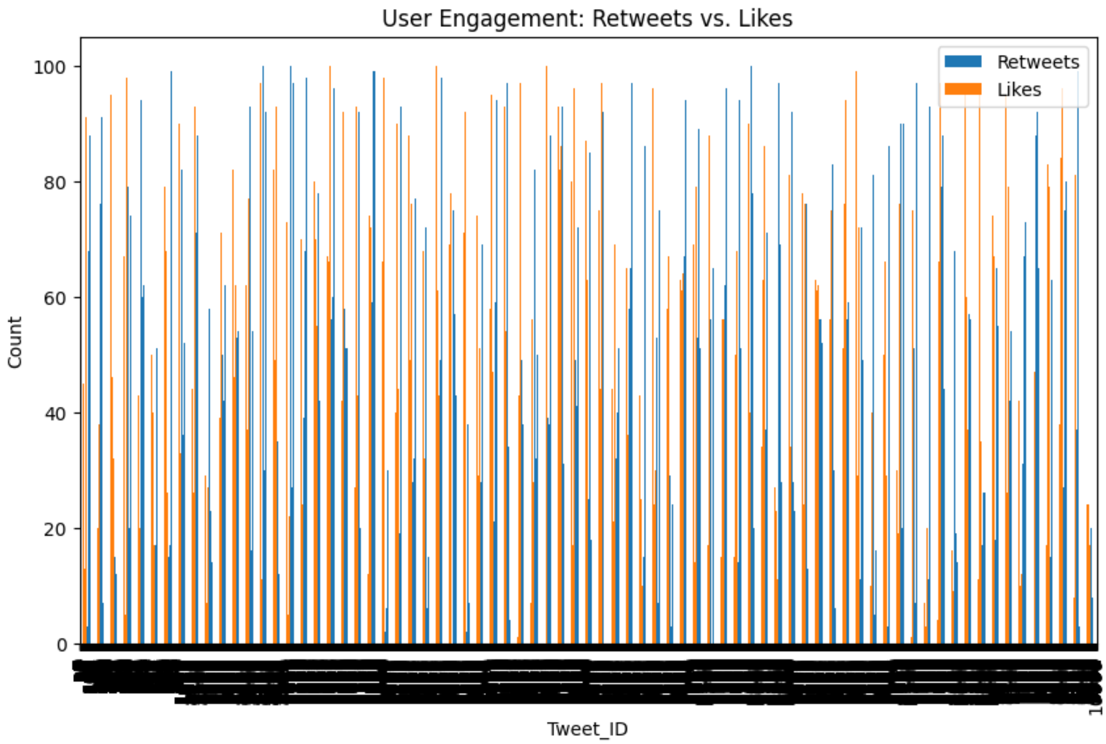
[ ]:
```python
# Find the top users with the highest number of retweets
top_users = df.groupby('Username')['Retweets'].sum().nlargest(10)
print("Top Users by Retweets:")
print(top_users)
```

```
Top Users by Retweets:
Username
pjohnson           362
awilliams          306
fsmith             301
wmitchell          269
nbrown             267
davidsmith         263
christopher64      261
amiller            253
ehernandez         251
jessicawilliams    251
Name: Retweets, dtype: int64
```

7 Barplot

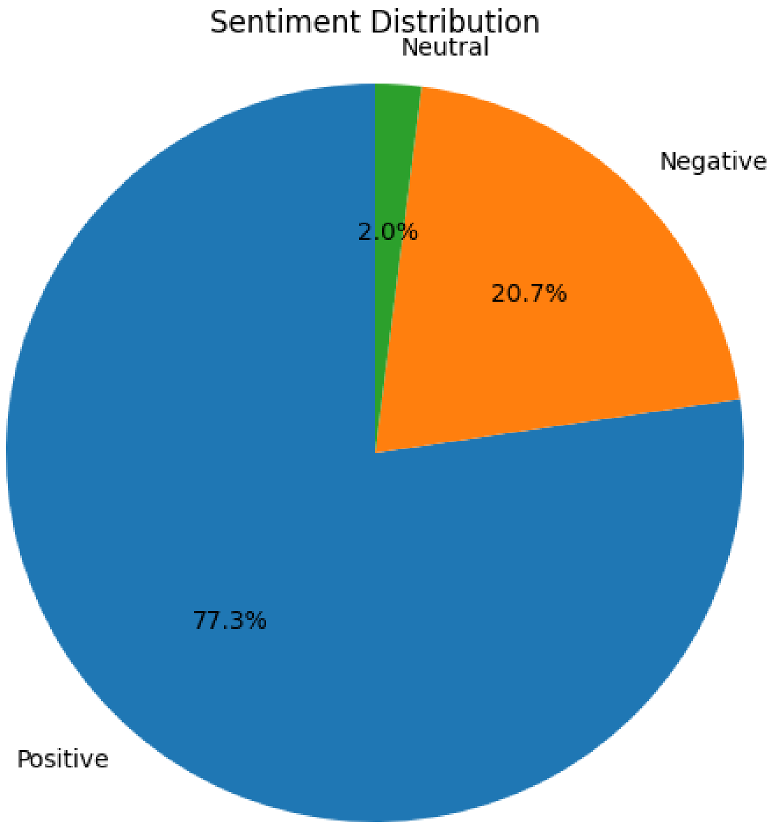
```
[ ]: # Create a bar chart of retweets and likes by username
user_engagement = df.groupby('Tweet_ID')[['Retweets', 'Likes']].sum()
user_engagement.plot(kind='bar', figsize=(10, 6))
plt.xlabel('Tweet_ID')    # username can also be taken but due to arge data␣
 ↪Tweet_ID has used
plt.ylabel('Count')
plt.title('User Engagement: Retweets vs. Likes')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Creating legend with loc="best" can be slow with large amounts of
data.
  fig.canvas.print_figure(bytes_io, **kw)
```



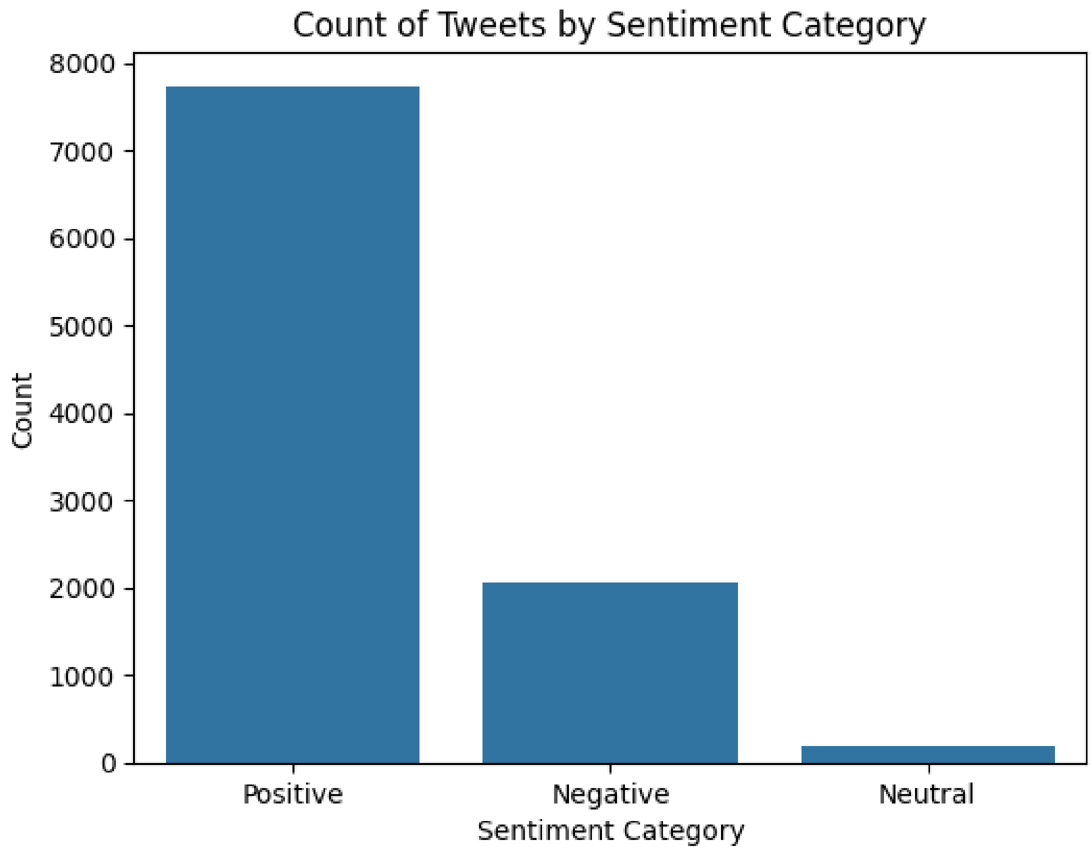
8 Piechart

[ ]:
```
# Perform sentiment analysis on tweet text
df['Sentiment'] = df['Text'].apply(lambda x: TextBlob(x).sentiment.polarity)

# Categorize sentiment into positive, negative, and neutral
df['Sentiment Category'] = df['Sentiment'].apply(lambda x: 'Positive' if x > 0␣
↪else 'Negative' if x < 0 else 'Neutral')

# Calculate the count of each sentiment category
sentiment_counts = df['Sentiment Category'].value_counts()

# Plot a pie chart of sentiment distribution
plt.figure(figsize=(8, 6))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%',␣
↪startangle=90)
plt.axis('equal')
plt.title('Sentiment Distribution')
plt.show()
```



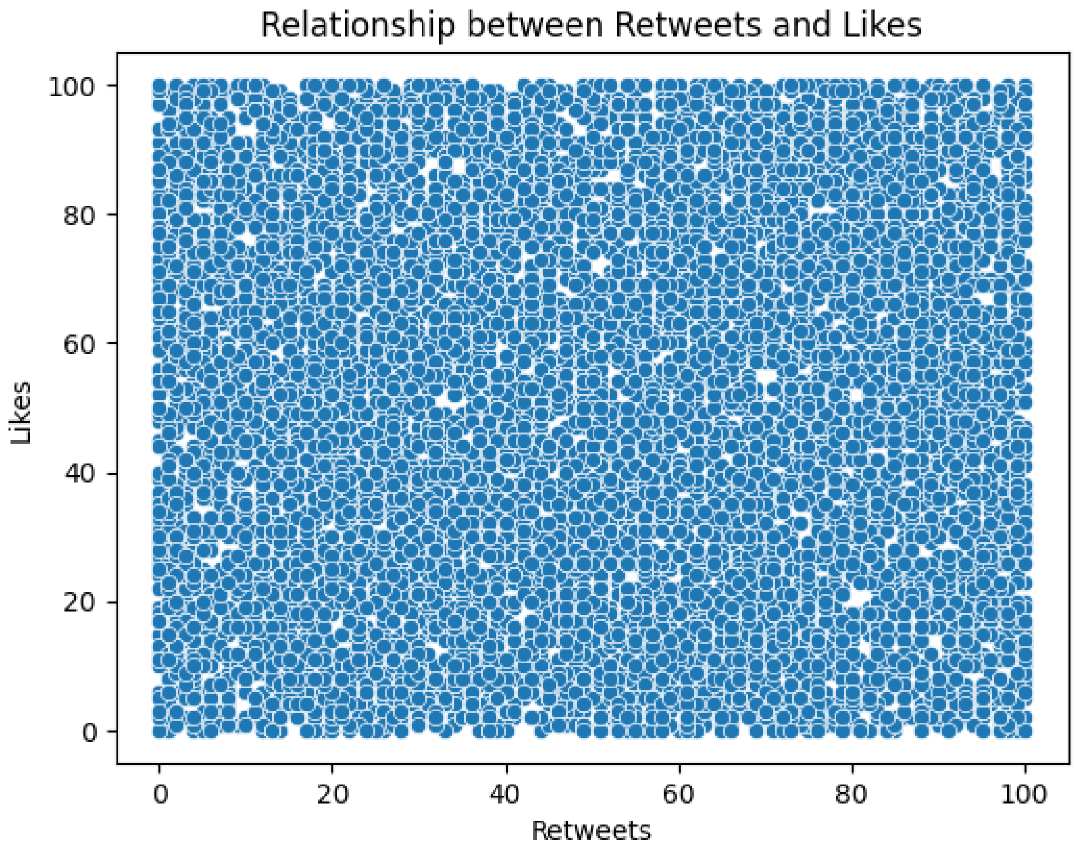
9 Count Plot

[ ]:
```
# Plot the count of tweets by sentiment category
sns.countplot(x='Sentiment Category', data=df)
plt.xlabel('Sentiment Category')
plt.ylabel('Count')
plt.title('Count of Tweets by Sentiment Category')
plt.show()
```



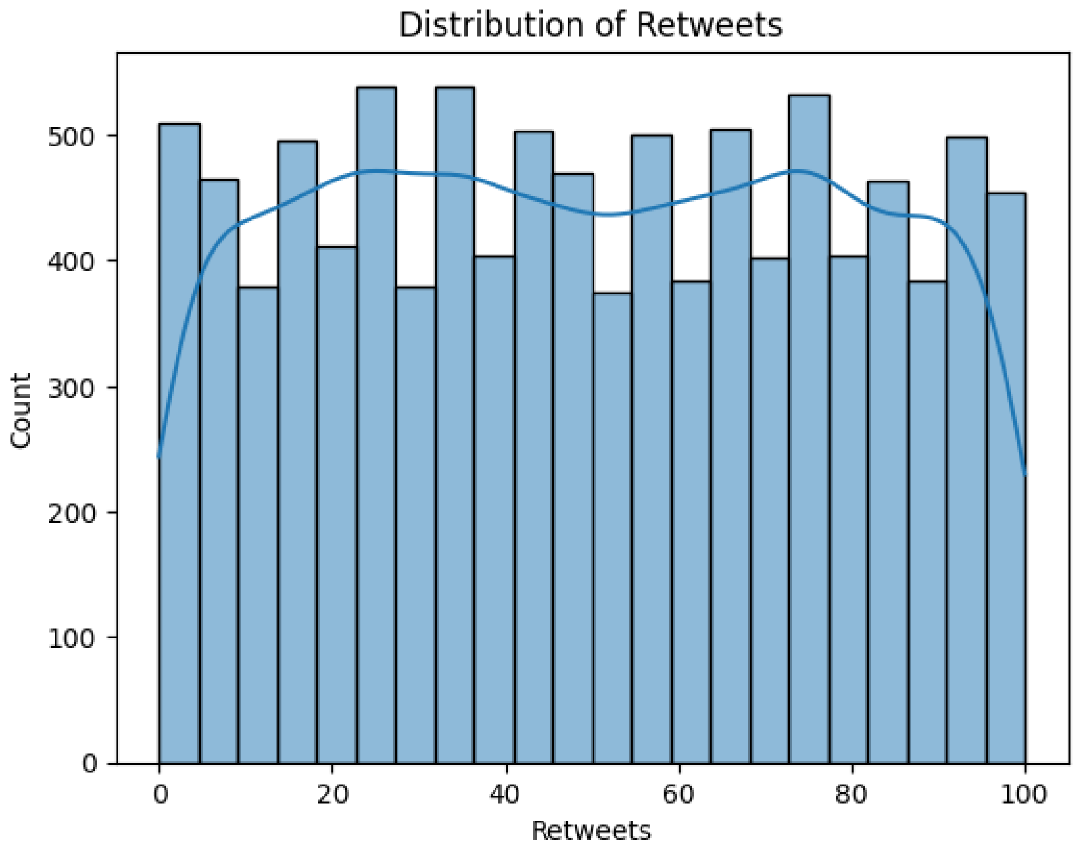
10 Scatterplot

[ ]:
```
# Plot the relationship between retweets and likes
sns.scatterplot(x='Retweets', y='Likes', data=df)
plt.xlabel('Retweets')
plt.ylabel('Likes')
plt.title('Relationship between Retweets and Likes')
plt.show()
```



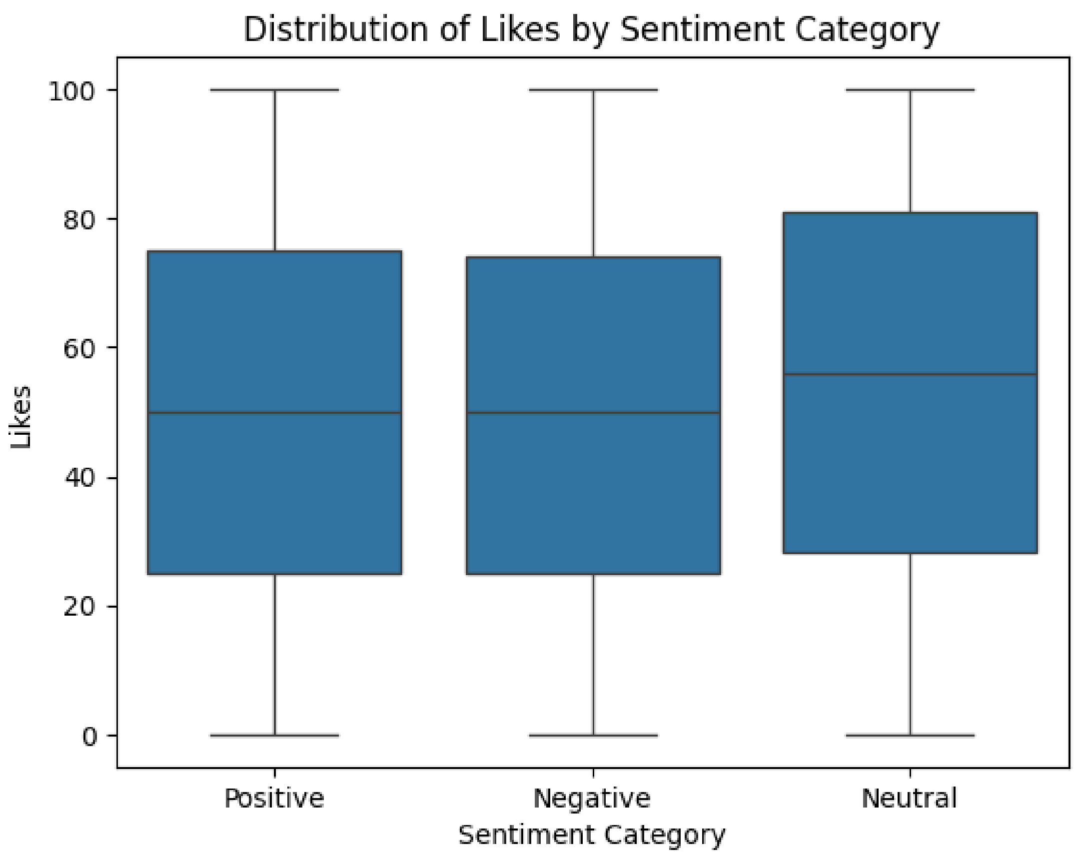
11 Distplot (Distribution Plot)

```
[ ]: # Plot the distribution of retweets
sns.histplot(df['Retweets'], kde=True)
plt.xlabel('Retweets')
plt.ylabel('Count')
plt.title('Distribution of Retweets')
plt.show()
```



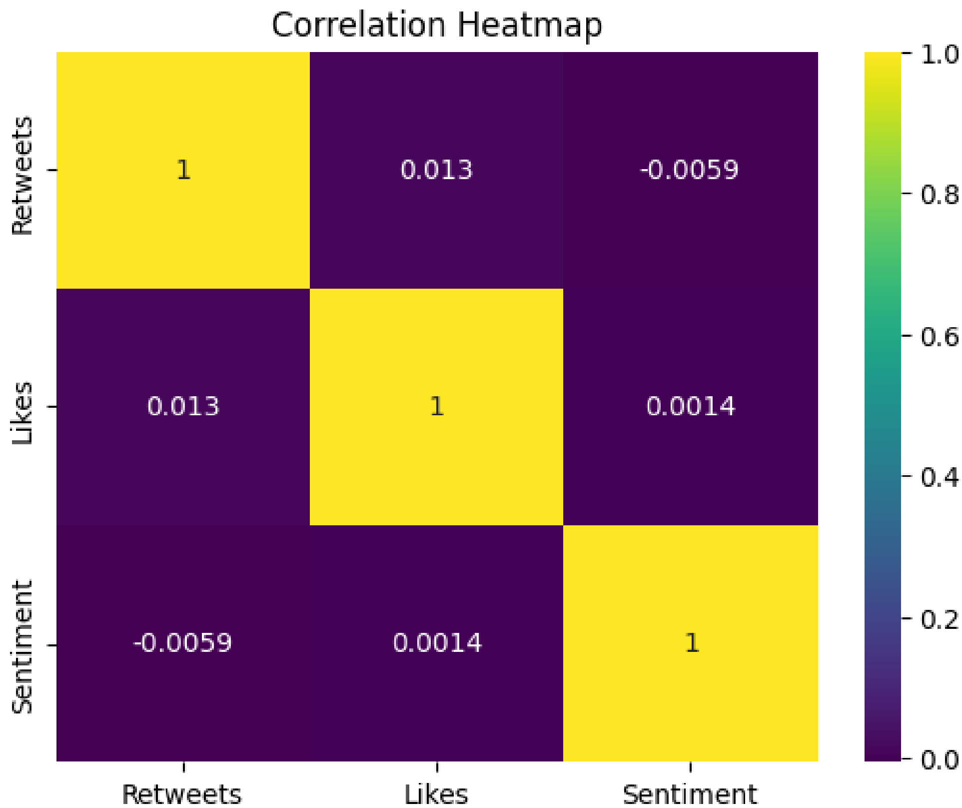
12 Boxplot

[ ]:
```
# Plot the distribution of likes by sentiment category
sns.boxplot(x='Sentiment Category', y='Likes', data=df)
plt.xlabel('Sentiment Category')
plt.ylabel('Likes')
plt.title('Distribution of Likes by Sentiment Category')
plt.show()
```



13 Heatmap

[ ]:
```
# Calculate the correlation matrix
correlation_matrix = df[['Retweets', 'Likes', 'Sentiment']].corr()

# Plot the correlation heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='viridis')
plt.title('Correlation Heatmap')
plt.show()
```



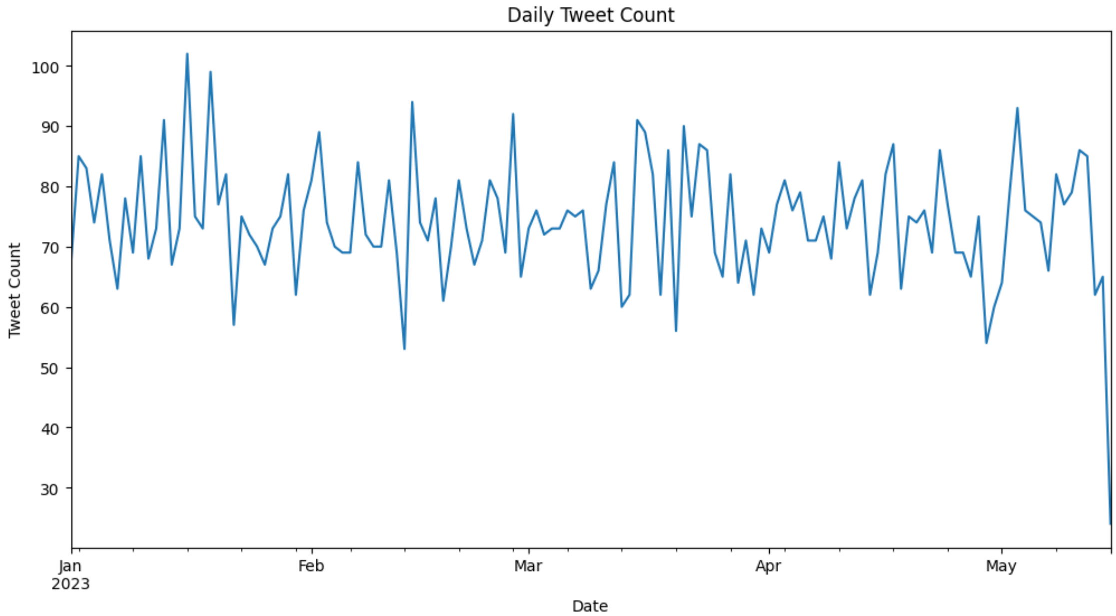
14 WordClouds

[ ]:
```
# Combine all tweet texts into a single string
all_text = ' '.join(df['Text'])

# Generate a word cloud of the most frequent words
wordcloud = WordCloud(width=800, height=400).generate(all_text)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Most Frequent Words')
plt.show()
```

Word Cloud of Most Frequent Words

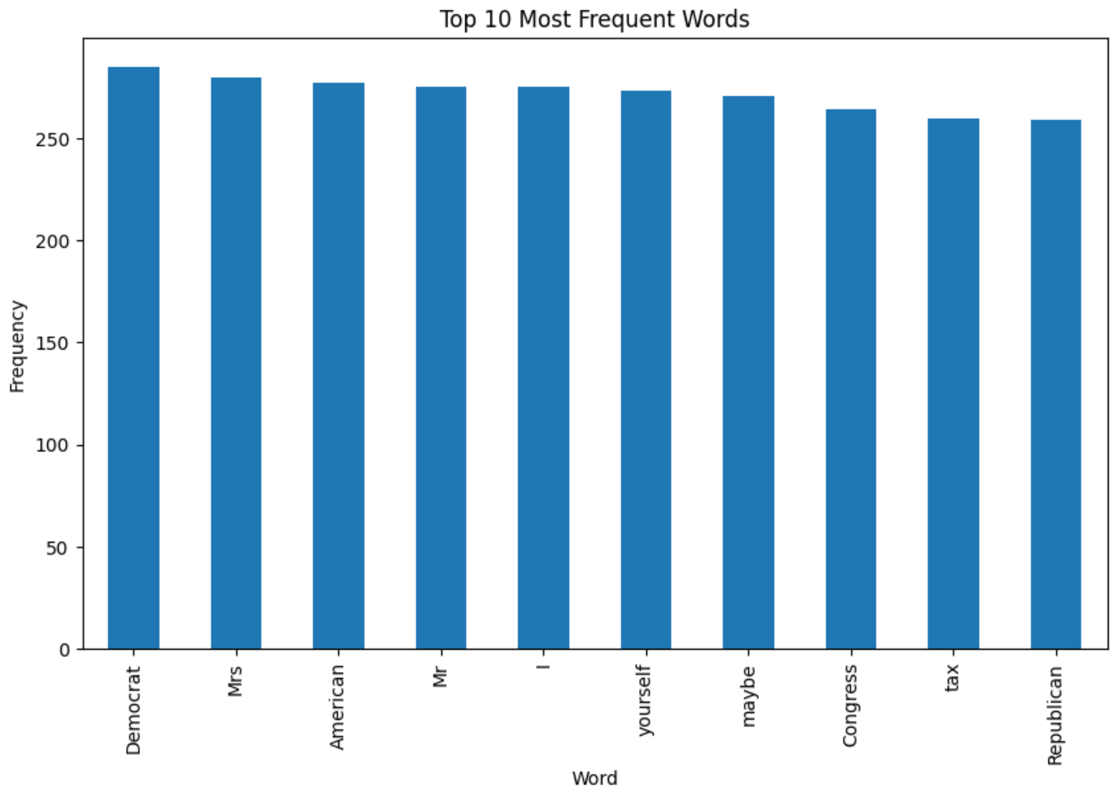
15 Lineplot

```
[ ]: # Convert the 'Timestamp' column to datetime format
df['Timestamp'] = pd.to_datetime(df['Timestamp'])

# Set the 'Timestamp' column as the DataFrame index
df.set_index('Timestamp', inplace=True)

# Resample the data by day and calculate the count of tweets per day
daily_tweet_count = df['Tweet_ID'].resample('D').count()

# Plot the time series of daily tweet count
plt.figure(figsize=(12, 6))
daily_tweet_count.plot()
plt.xlabel('Date')
plt.ylabel('Tweet Count')
plt.title('Daily Tweet Count')
plt.show()
```



16 Barplot

[ ]:
```
# Combine all tweet texts into a single string
all_text = ' '.join(df['Text'])

# Split the text into individual words
words = all_text.split()

# Calculate the frequency of each word
word_counts = pd.Series(words).value_counts().sort_values(ascending=False)

# Plot the top 10 most frequent words
plt.figure(figsize=(10, 6))
word_counts.head(10).plot(kind='bar')
plt.xlabel('Word')
plt.ylabel('Frequency')
plt.title('Top 10 Most Frequent Words')
plt.show()
```



17 Conclusion

The notebook effectively showcases the use of time series analysis to predict the future popularity of specific hashtags or keywords based on historical tweet data. By combining techniques in natural language processing (NLP) and time series modeling, it provides a systematic approach to understanding trends and forecasting future occurrences of specific terms on social media.

sma-7

November 23, 2024

1 Program: 7 Use Machine learning models to predict the number of likes, share or retweets based on platform(Twitter) content and metadata.

2 Importing Dependencies

[ ]:
```python
import numpy as np
import pandas as pd

import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

from textblob import TextBlob

import matplotlib.pyplot as plt

from wordcloud import WordCloud

import seaborn as sns

from PIL import Image

import nltk
nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data] Downloading package omw-1.4 to /root/nltk_data…
```

[ ]: True

[ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
import datetime
```

3 Dataset

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!unzip /content/drive/MyDrive/Social\ Media\ Analytics/Twitter-Dataset.zip
```

```
Archive:  /content/drive/MyDrive/Social Media Analytics/Twitter-Dataset.zip
  inflating: download.jpg
  inflating: twitter_dataset.csv
```

[ ]:
```
df=pd.read_csv("/content/twitter_dataset.csv")
```

[ ]:
```
df.head()
```

[ ]:
```
   Tweet_ID         Username  \
0         1          julie81
1         2    richardhester
2         3  williamsjoseph
3         4      danielsmary
4         5       carlwarren

                                                Text  Retweets  Likes  \
0  Party least receive say or single. Prevent pre…         2     25
1  Hotel still Congress may member staff. Media d…        35     29
2  Nice be her debate industry that year. Film wh…        51     25
3  Laugh explain situation career occur serious. …        37     18
4  Involve sense former often approach government…        27     80

             Timestamp
0  2023-01-30 11:00:51
1  2023-01-02 22:45:58
2  2023-01-18 11:25:19
3  2023-04-10 22:06:29
4  2023-01-24 07:12:21
```

[ ]:
```
df.shape
```

[ ]: (10000, 6)

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Tweet_ID   10000 non-null  int64
 1   Username   10000 non-null  object
 2   Text       10000 non-null  object
 3   Retweets   10000 non-null  int64
 4   Likes      10000 non-null  int64
 5   Timestamp  10000 non-null  object
dtypes: int64(3), object(3)
memory usage: 468.9+ KB
```

[ ]:
```
df.describe()
```

[ ]:

	Tweet_ID	Retweets	Likes
count	10000.00000	10000.000000	10000.000000
mean	5000.50000	49.721200	49.929300
std	2886.89568	28.948856	28.877193
min	1.00000	0.000000	0.000000
25%	2500.75000	25.000000	25.000000
50%	5000.50000	49.000000	50.000000
75%	7500.25000	75.000000	75.000000
max	10000.00000	100.000000	100.000000

[ ]:
```
df.isnull().sum()
```

[ ]:
```
Tweet_ID     0
Username     0
Text         0
Retweets     0
Likes        0
Timestamp    0
dtype: int64
```

[ ]:
```
df.duplicated().sum()
```

[ ]: 0

4 Feature Engineering

[ ]:
```
# Remove duplicate tweets
# df = df.drop_duplicates()

# Remove rows with missing values
# df = df.dropna()

# Clean tweet text by removing special characters and URLs
df['Text'] = df['Text'].str.replace('[^a-zA-Z0-9\s]', '')
df['Text'] = df['Text'].str.replace('http\S+|www.\S+', '', case=False)
```

5 Porter Stemmer

[ ]:
```
# Tokenize tweet text
df['tokens'] = df['Text'].apply(lambda x: nltk.word_tokenize(x))

# Remove stopwords
stop_words = set(stopwords.words('english'))
df['tokens'] = df['tokens'].apply(lambda x: [word for word in x if word.lower()␣
↪not in stop_words])

# Stemming or Lemmatization
stemmer = PorterStemmer()
df['tokens'] = df['tokens'].apply(lambda x: [stemmer.stem(word) for word in x])
```

[ ]:
```
# Calculate summary statistics
mean_retweets = df['Retweets'].mean()
median_likes = df['Likes'].median()
correlation = df['Retweets'].corr(df['Likes'])
```

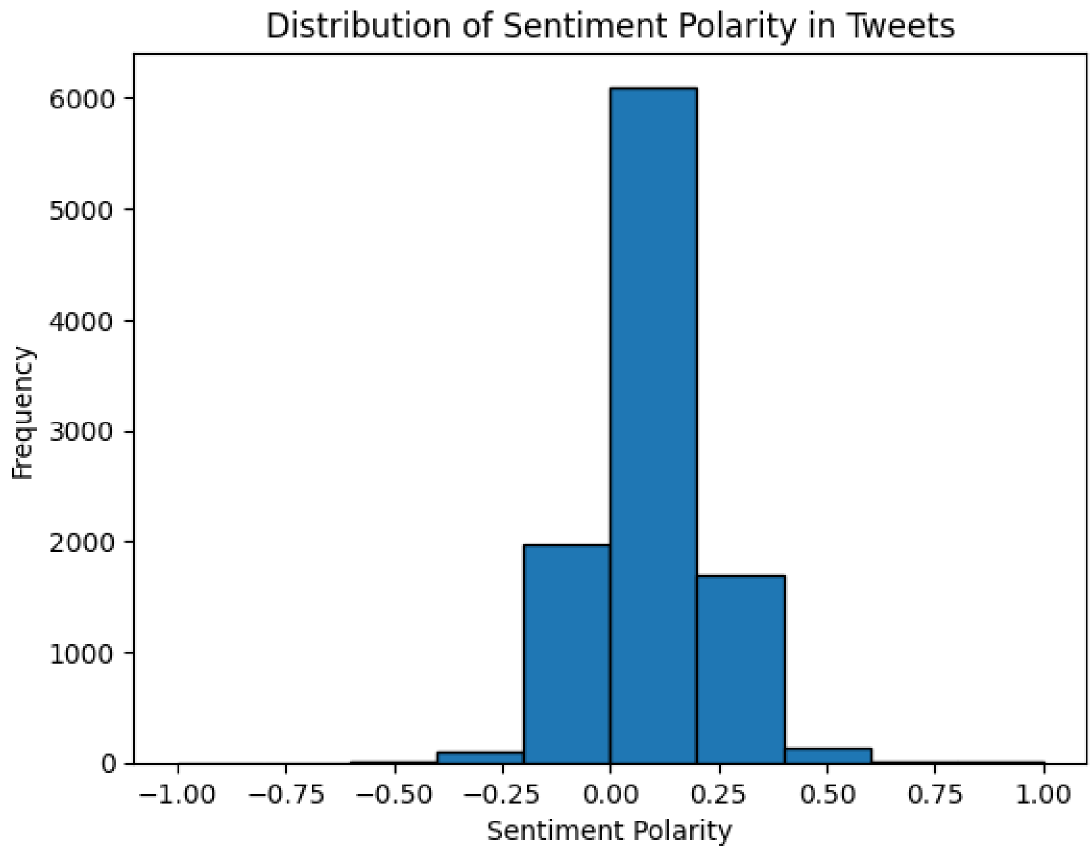
[ ]:
```
# Perform sentiment analysis on tweet text
df['sentiment_polarity'] = df['Text'].apply(lambda x: TextBlob(x).sentiment.
↪polarity)
```

[ ]:
```
# Print results
print("Mean Retweets:", mean_retweets)
print("Median Likes:", median_likes)
print("Correlation between Retweets and Likes:", correlation)
```

```
Mean Retweets: 49.7212
Median Likes: 50.0
Correlation between Retweets and Likes: 0.012797546201034809
```

6 Histplot (Histogram Plot)

[ ]:
```
# Plotting sentiment polarity distribution
plt.hist(df['sentiment_polarity'], bins=10, range=(-1, 1), edgecolor='black')
plt.xlabel('Sentiment Polarity')
plt.ylabel('Frequency')
plt.title('Distribution of Sentiment Polarity in Tweets')
plt.show()
```



[ ]:
```
# Print the number of rows and columns in the dataset
print("Number of Rows:", df.shape[0])
print("Number of Columns:", df.shape[1])

# Calculate the average values of retweets and likes
avg_retweets = df['Retweets'].mean()
avg_likes = df['Likes'].mean()
print("Average Retweets:", avg_retweets)
print("Average Likes:", avg_likes)
```

```
Number of Rows: 10000
```

```
Number of Columns: 8
Average Retweets: 49.7212
Average Likes: 49.9293
```

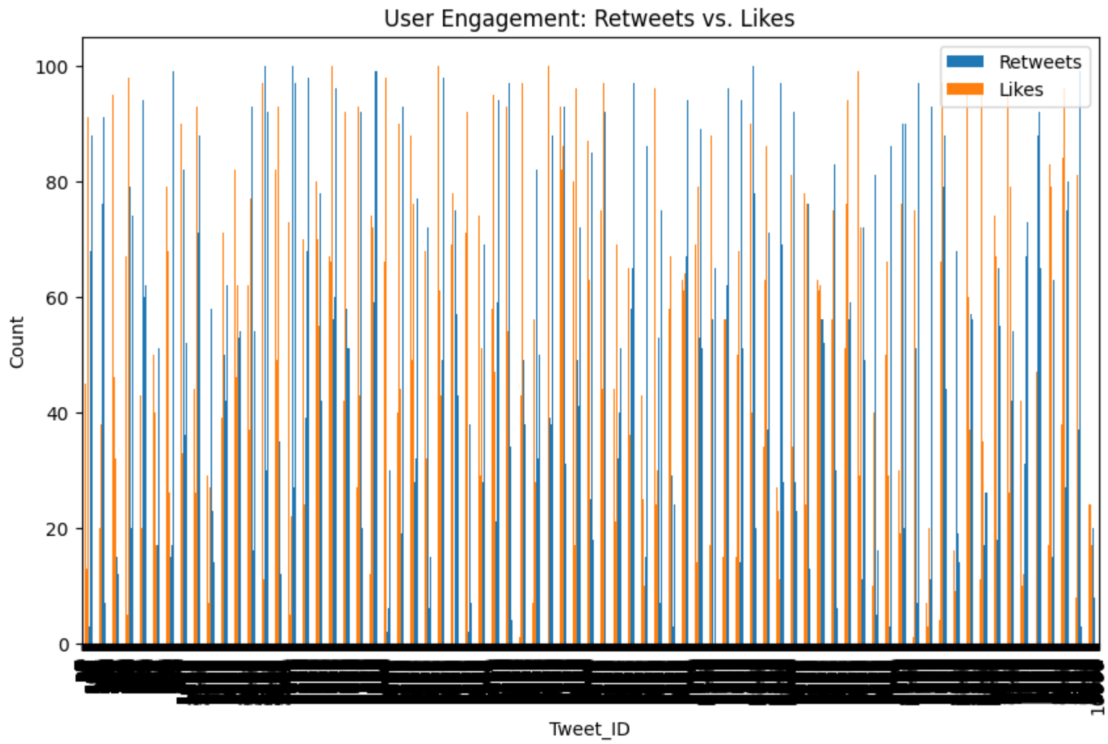
```
[ ]: # Find the top users with the highest number of retweets
top_users = df.groupby('Username')['Retweets'].sum().nlargest(10)
print("Top Users by Retweets:")
print(top_users)
```

```
Top Users by Retweets:
Username
pjohnson           362
awilliams          306
fsmith             301
wmitchell          269
nbrown             267
davidsmith         263
christopher64      261
amiller            253
ehernandez         251
jessicawilliams    251
Name: Retweets, dtype: int64
```

7 Barplot

```
[ ]: # Create a bar chart of retweets and likes by username
user_engagement = df.groupby('Tweet_ID')[['Retweets', 'Likes']].sum()
user_engagement.plot(kind='bar', figsize=(10, 6))
plt.xlabel('Tweet_ID')    # username can also be taken but due to arge data␣
 ↪Tweet_ID has used
plt.ylabel('Count')
plt.title('User Engagement: Retweets vs. Likes')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Creating legend with loc="best" can be slow with large amounts of
data.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
[ ]: # Convert Timestamp to datetime format
df['Timestamp'] = pd.to_datetime(df['Timestamp'])

# Extract time-based features
df['hour'] = df['Timestamp'].dt.hour
df['day_of_week'] = df['Timestamp'].dt.dayofweek
df['month'] = df['Timestamp'].dt.month
df.head()
```

```
[ ]:    Tweet_ID         Username  \
0         1          julie81
1         2    richardhester
2         3  williamsjoseph
3         4      danielsmary
4         5       carlwarren

                                                Text  Retweets  Likes  \
0  Party least receive say or single. Prevent pre…         2     25
1  Hotel still Congress may member staff. Media d…        35     29
2  Nice be her debate industry that year. Film wh…        51     25
3  Laugh explain situation career occur serious. …        37     18
4  Involve sense former often approach government…        27     80
```

```
             Timestamp                                             tokens  \
0 2023-01-30 11:00:51  [parti, least, receiv, say, singl, ., prevent,…
1 2023-01-02 22:45:58  [hotel, still, congress, may, member, staff, …
2 2023-01-18 11:25:19  [nice, debat, industri, year, ., film, gener, …
3 2023-04-10 22:06:29  [laugh, explain, situat, career, occur, seriou…
4 2023-01-24 07:12:21  [involv, sens, former, often, approach, govern…

   sentiment_polarity  hour  day_of_week  month
0            0.115714    11            0      1
1            0.308333    22            0      1
2            0.220000    11            2      1
3            0.054762    22            0      4
4            0.033333     7            1      1
```

[ ]:
```python
df = df.drop(columns=['Timestamp'])
df.head()
```

[ ]:
```
   Tweet_ID        Username  \
0         1         julie81
1         2    richardhester
2         3  williamsjoseph
3         4      danielsmary
4         5       carlwarren

                                               Text  Retweets  Likes  \
0  Party least receive say or single. Prevent pre…         2     25
1  Hotel still Congress may member staff. Media d…        35     29
2  Nice be her debate industry that year. Film wh…        51     25
3  Laugh explain situation career occur serious. …        37     18
4  Involve sense former often approach government…        27     80

                                             tokens  sentiment_polarity  \
0  [parti, least, receiv, say, singl, ., prevent,…            0.115714
1  [hotel, still, congress, may, member, staff, …            0.308333
2  [nice, debat, industri, year, ., film, gener, …            0.220000
3  [laugh, explain, situat, career, occur, seriou…            0.054762
4  [involv, sens, former, often, approach, govern…            0.033333

   hour  day_of_week  month
0    11            0      1
1    22            0      1
2    11            2      1
3    22            0      4
4     7            1      1
```

[ ]:
```python
# TF-IDF for the 'Text' column
vectorizer = TfidfVectorizer(max_features=1000)
```

```
X_text = vectorizer.fit_transform(df['Text']).toarray()
```

[ ]:
```
# Normalize other numerical features like 'Retweets', 'Likes'
scaler = StandardScaler()
X_metadata = scaler.fit_transform(df[['Likes', 'hour', 'day_of_week', 'month']])
X_metadata
```

[ ]:
```
array([[-0.86332998, -0.06836481, -1.48553339, -1.33327596],
       [-0.72480543,  1.52185059, -1.48553339, -1.33327596],
       [-0.86332998, -0.06836481, -0.48668761, -1.33327596],
       …,
       [ 0.41802205,  0.3653303 , -1.48553339, -0.57577584],
       [ 0.34875978,  0.65446038, -1.48553339, -1.33327596],
       [ 0.14097296, -1.51401517, -0.48668761,  0.9392244 ]])
```

[ ]:
```
# Combine the text and metadata features
X = np.hstack([X_text, X_metadata])

y = df['Retweets']
```

[ ]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[ ]:
```
# Train a Random Forest Regressor
model = RandomForestRegressor(n_estimators=1000,max_depth=20,n_jobs=-1,␣
↪random_state=42)
model.fit(X_train, y_train)
```

[ ]:
```
RandomForestRegressor(max_depth=20, n_jobs=-1, random_state=42)
```

[ ]:
```
# Predict on test data
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
```

```
Mean Absolute Error: 25.424358786794052
Mean Squared Error: 858.2581412806583
```

[ ]:
```
comparison_df = pd.DataFrame({
    'Actual': y_test[:10].values,   # Actual Retweets values
    'Predicted': y_pred[:10]        # Predicted Retweets values
```

```python
})

print("Comparison of Actual vs Predicted (First 10 samples):\n", comparison_df)
```

```
Comparison of Actual vs Predicted (First 10 samples):
    Actual  Predicted
0       94  44.731804
1       12  51.382922
2       40  50.635621
3       50  53.621649
4       65  48.080975
5        0  49.207168
6       21  49.937717
7       67  53.279022
8       75  49.237680
9        7  51.949282
```

8 Conclusion

The project demonstrates the potential of machine learning in quantifying social media engagement through advanced preprocessing, sentiment analysis, and the use of a Random Forest Regressor. It reveals the relationship between content characteristics and user interactions like likes, shares, and retweets. Visualizations enhance data insights and interpretability.

sma-8

November 23, 2024

1 Program 8: Develop a social media text analytics model to improve existing products or services by analyzing customer reviews and comments.

[ ]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import matplotlib
matplotlib.style.use('ggplot')
from __future__ import unicode_literals
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!unzip /content/drive/MyDrive/Social\ Media\ Analytics/customer-review-analysis.
↪zip
```

```
Archive:  /content/drive/MyDrive/Social Media Analytics/customer-review-
analysis.zip
  inflating: Amazon_Unlocked_Mobile.csv
```

[ ]:
```
Amazon_Meta_Data = pd.read_csv('/content/Amazon_Unlocked_Mobile.csv',␣
↪encoding='utf-8')
```

[ ]:
```
Amazon_Meta_Data.head(2)
```

[ ]:
```
                                        Product Name Brand Name   Price  \
0  "CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7…    Samsung  199.99
1  "CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7…    Samsung  199.99

   Rating                                            Reviews  Review Votes
0       5  I feel so LUCKY to have found this used (phone…           1.0
1       4  nice phone, nice up grade from my pantach revu…           0.0
```

[ ]:
```
Amazon_Meta_Data.columns
```

[ ]:
```
Index(['Product Name', 'Brand Name', 'Price', 'Rating', 'Reviews',
       'Review Votes'],
      dtype='object')
```

[ ]:
```
Amazon_Meta_Data.dtypes
```

[ ]:
```
Product Name     object
Brand Name       object
Price           float64
Rating            int64
Reviews          object
Review Votes    float64
dtype: object
```

[ ]:
```
Reviews = Amazon_Meta_Data['Reviews']
len(Reviews)
```

[ ]:
```
413840
```

Top Review Counts With Brand

[ ]:
```
Brand_Name = Amazon_Meta_Data['Brand Name'].str.upper()
Brand_Name.value_counts().head(10)
```

[ ]:
```
Brand Name
SAMSUNG       68720
BLU           63256
APPLE         58187
LG            22423
BLACKBERRY    17929
NOKIA         16841
MOTOROLA      13447
HTC           12927
CNPGD         12613
OTTERBOX       7989
Name: count, dtype: int64
```

Mean and Median Price In Given Data

[ ]:
```
Price = Amazon_Meta_Data['Price']
Price.mean()
```

[ ]:
```
226.86715538100597
```

[ ]:
```
Price.median()
```

[ ]: 144.71

[ ]:
```
%time
table = pd.pivot_table(Amazon_Meta_Data,
            values = ['Price'],
            index = ['Brand Name'],
                       columns= [],
                       aggfunc=[np.mean, np.median],
                       margins=True)

#table
```

```
CPU times: user 3 µs, sys: 1e+03 ns, total: 4 µs
Wall time: 7.39 µs
```

```
<ipython-input-13-3727ebf8ca19>:2: FutureWarning: The provided callable
<function mean at 0x7bef11934310> is currently using DataFrameGroupBy.mean. In a
future version of pandas, the provided callable will be used directly. To keep
current behavior pass the string "mean" instead.
  table = pd.pivot_table(Amazon_Meta_Data,
<ipython-input-13-3727ebf8ca19>:2: FutureWarning: The provided callable
<function median at 0x7bef0ed539a0> is currently using DataFrameGroupBy.median.
In a future version of pandas, the provided callable will be used directly. To
keep current behavior pass the string "median" instead.
  table = pd.pivot_table(Amazon_Meta_Data,
```

Review Ranting

[ ]:
```
Customer_Ratings = Amazon_Meta_Data.groupby(
    'Brand Name'
    ).Rating.agg(
        ['count', 'min', 'max']
    ).sort_values(
        'count', ascending=False
    )
Customer_Ratings.head(15)
```

[ ]:

	count	min	max
Brand Name			
Samsung	65747	1	5
BLU	63248	1	5
Apple	58186	1	5
LG	22417	1	5
BlackBerry	16872	1	5
Nokia	16806	1	5
Motorola	13417	1	5
HTC	12724	1	5
CNPGD	12613	1	5

```
OtterBox         7989    1    5
Sony             7828    1    5
Posh Mobile      6765    1    5
Huawei           3325    1    5
LG Electronics   3105    1    5
samsung          2431    1    5
```

[ ]:
```
Product_Ratings = Amazon_Meta_Data.groupby(
    'Product Name'
    ).Rating.agg(
        ['count', 'min', 'max']
    ).sort_values(
        'count', ascending=False
    )
Product_Ratings.head(15)
```

[ ]:
```
                                                    count  min  max
Product Name
Apple iPhone 4s 8GB Unlocked Smartphone w/ 8MP …   1451    1    5
Apple MF259LL/A - iPhone 4s 8GB / 8MP Camera - …   1241    1    5
BLU Studio 5.0 C HD Unlocked Cellphone, Black        1194    1    5
OtterBox Iphone 5/5S/SE Defender Case w/ Drop a…   1129    1    5
Motorola Moto E (1st Generation) - Black - 4 GB…   1127    1    5
Apple iPhone 5s 32GB (Silver) - AT&T                 1118    1    5
BLU Energy X Plus Smartphone - With 4000 mAh Su…   1111    1    5
Samsung Galaxy S Duos II S7582 DUAL SIM Factory…   1109    1    5
Samsung Galaxy S Duos II GT-S7582 Factory Unloc…   1108    1    5
Samsung Galaxy S Duos GT-S7562 GSM Unlocked Tou…   1096    1    5
Samsung Galaxy S4 i9505 16GB LTE Unlocked Inter…   1095    1    5
Apple iPhone 5s AT&T Cellphone, 16GB, Silver         1080    1    5
Apple iPhone 4S 16GB Unlocked GSM - White (Cert…   1071    1    5
BLU Energy X Plus Smartphone - With 4000 mAh Su…   1061    1    5
Motorola Moto E (1st Generation) - Black - 4 GB…   1057    1    5
```

[ ]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sample_review = Reviews[:10]
```

[ ]:
```
import nltk
nltk.download('vader_lexicon')
sentiment = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data…
```

****Review Sentiment Score Using NLTK ****

[ ]:
```
for sentences in sample_review:
    sentences
    ss = sentiment.polarity_scores(sentences)
```

```
    for k in sorted(ss):
        print('{0}: {1}, '.format(k, ss[k]))
    print(sentences)
```

```
compound: 0.8783,
neg: 0.015,
neu: 0.796,
pos: 0.189,
I feel so LUCKY to have found this used (phone to us & not used hard at all),
phone on line from someone who upgraded and sold this one. My Son liked his old
one that finally fell apart after 2.5+ years and didn't want an upgrade!! Thank
you Seller, we really appreciate it & your honesty re: said used phone.I
recommend this seller very highly & would but from them again!!
compound: 0.9231,
neg: 0.072,
neu: 0.597,
pos: 0.331,
nice phone, nice up grade from my pantach revue. Very clean set up and easy set
up. never had an android phone but they are fantastic to say the least. perfect
size for surfing and social media. great phone samsung
compound: 0.4927,
neg: 0.0,
neu: 0.238,
pos: 0.762,
Very pleased
compound: 0.9185,
neg: 0.0,
neu: 0.5,
pos: 0.5,
It works good but it goes slow sometimes but its a very good phone I love it
compound: 0.2942,
neg: 0.038,
neu: 0.897,
pos: 0.065,
Great phone to replace my lost phone. The only thing is the volume up button
does not work, but I can still go into settings to adjust. Other than that, it
does the job until I am eligible to upgrade my phone again.Thaanks!
compound: -0.9107,
neg: 0.143,
neu: 0.857,
pos: 0.0,
I already had a phone with problems… I know it stated it was used, but dang,
it did not state that it did not charge. I wish I would have read these comments
then I would have not purchased this item… and its cracked on the side..
damaged goods is what it is… If trying to charge it another way does not work
I am requesting for my money back… AND I WILL GET MY MONEY BACK…SIGNED AN
UNHAPPY CUSTOMER…
```

```
compound: -0.0516,
neg: 0.057,
neu: 0.891,
pos: 0.052,
The charging port was loose. I got that soldered in. Then needed a new battery
as well. $100 later (not including cost of purchase) I have a usable phone. The
phone should not have been sold in the state it was in.
compound: 0.4486,
neg: 0.087,
neu: 0.709,
pos: 0.204,
Phone looks good but wouldn't stay charged, had to buy new battery. Still
couldn't stay charged long.so I trashed it.MONEY lost, never again will I buy
from this person! !!!
compound: 0.9491,
neg: 0.023,
neu: 0.848,
pos: 0.129,
I originally was using the Samsung S2 Galaxy for Sprint and wanted to return
back to the Samsung EPIC 4G for Sprint because I really missed the keyboard, I
really liked the smaller compact size of the phone, and I still needed some of
the basic functions of a smart phone (i.e. checking e-mail, getting directions,
text messaging) Because the phone is not as powerful as the newer cell phones
out there, just be aware that the more applications you install the slower the
phone runs and will most likely freeze up from time to time. But the camera
works great, the video is great as well, and even the web browsing is decent and
gives me what I need. I also notice that battery life lasts a little bit longer
and charging the phone is much quicker than my Galaxy S2.
compound: 0.8268,
neg: 0.0,
neu: 0.791,
pos: 0.209,
It's battery life is great. It's very responsive to touch. The only issue is
that sometimes the screen goes black and you have to press the top button
several times to get the screen to re-illuminate.
```

Kmeans Clutering

[ ]:
```
Cluster_Data = pd.read_csv('/content/Amazon_Unlocked_Mobile.csv',nrows=6000)
```

[ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem import WordNetLemmatizer
import re
Cluster_Data.columns
```

[ ]:
```
Index(['Product Name', 'Brand Name', 'Price', 'Rating', 'Reviews',
        'Review Votes'],
       dtype='object')
```

Data Cleaning

[ ]:
```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
```

[ ]: True

[ ]:
```
from nltk.corpus import stopwords
stop = set(stopwords.words('english'))
from nltk.corpus import stopwords
def remove_stopword(word):
    return word not in words

from nltk.stem import WordNetLemmatizer
Lemma = WordNetLemmatizer()

from nltk.stem.snowball import SnowballStemmer
stemmer = SnowballStemmer("english")

Cluster_Data['NewReviews'] = Cluster_Data['Reviews'].str.lower().str.split()
Cluster_Data['NewReviews'] = Cluster_Data['NewReviews'].apply(lambda x : [item
  ↪for item in x if item not in stop])
#Cluter_Data['NewReviews'] = Cluter_Data["NewReviews"].apply(lambda x :
  ↪[stemmer.stem(y) for y in x])
```

[ ]:
```
Cluster_Data['Cleaned_reviews'] = [''.join([WordNetLemmatizer().lemmatize(re.
  ↪sub('[^A-Za-z]', ' ', line))
for line in lists]).strip() for lists in Cluster_Data['NewReviews']]
```

Columns

[ ]:
```
Cluster_Data.columns
```

[ ]:
```
Index(['Product Name', 'Brand Name', 'Price', 'Rating', 'Reviews',
       'Review Votes', 'NewReviews', 'Cleaned_reviews'],
      dtype='object')
```

TF_IDF

[ ]:
```
vectorizer = TfidfVectorizer(max_df=0.
  ↪5,max_features=10000,min_df=10,stop_words='english',use_idf=True)
```

[ ]:
```
model = vectorizer.fit_transform(Cluster_Data['Cleaned_reviews'].str.upper())
```

KMeans

```
[ ]: from sklearn.cluster import KMeans
km = KMeans(n_clusters=5,init='k-means++',max_iter=200,n_init=1)
km.fit(model)
```

```
[ ]: KMeans(max_iter=200, n_clusters=5, n_init=1)
```

```
[ ]: terms = vectorizer.get_feature_names_out()

order_centroids = km.cluster_centers_.argsort()[:, ::-1]
for i in range(5):
    print("cluster %d:" % i, end='')
    for ind in order_centroids[i, :10]:
        print(' %s' % terms[ind], end='')
    print()
```

```
cluster 0: ve whatsapp exactlydescribed like phoneperfect fine greatproduct
workgreat price months
cluster 1: greatphone excellent phone loveit great workgreat excelente lovephone
iphone gs
cluster 2: good annoying day far love gs excellent bought phone lgg
cluster 3: yes iphone awesome annoying great unlocked think new issue screen
cluster 4: easyuse greatprice lovephone greatphoneprice thankmuch fastshipping
overall worked loveit workfine
```

2 Conclusion

The social media analytics model successfully identifies key customer sentiments and product patterns in the mobile phone market. Through sentiment scoring and K-means clustering, we uncovered dominant brands, price distributions, and common themes in customer reviews. This analysis provides actionable insights for product improvements and marketing strategies based on real customer feedback. The clustering of reviews into 5 distinct groups reveals the main topics that matter most to customers, enabling focused business decisions.

sma-9

November 23, 2024

1 Practical 9: use a network analytics tool to build a graph using real-world social network dataset example - social media, co-authorship, citation networks, election.

2 a. Explore graph characteristics: number of nodes, edges, density, and components.

3 b. Calculate the degree centrality for each node in the network.

4 c. Compute betweenness centrality, which measures the extent to which a node lies on the shortest path between other nodes.

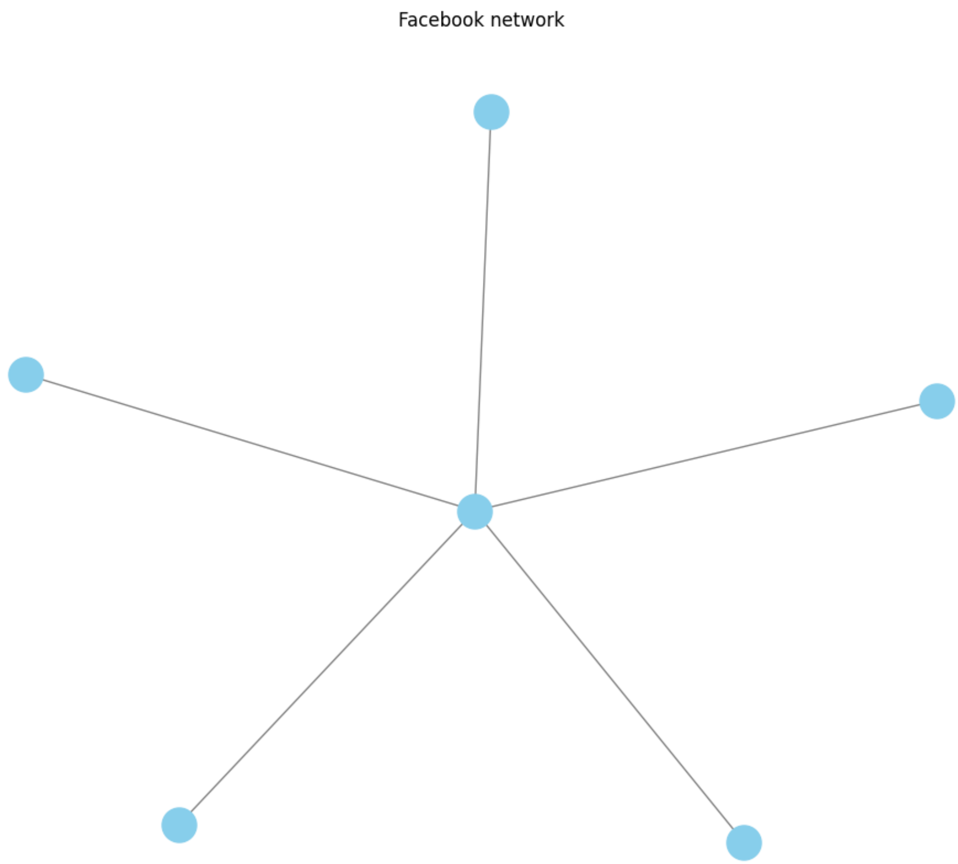
5 d. compute closeness centrality, which measures how close a node is to all other nodes in the network.

6 e. Compute eigenvector centrality, which identifies nodes that are connected to other important nodes.

[ ]:
```
import networkx as nx
import matplotlib.pyplot as plt
```

[ ]:
```
G = nx.ego_graph(nx.generators.random_graphs.erdos_renyi_graph(100, 0.05), 0)
```

[ ]:
```
plt.figure(figsize=(10, 8))
nx.draw(G, with_labels=False, node_color='skyblue', edge_color='gray',␣
 ↪node_size=500)
plt.title("Facebook network")
plt.show()
```



6.0.1 Explore graph characteristics

[ ]:
```
# Number of nodes and edges
num_nodes = G.number_of_nodes()
num_edges = G.number_of_edges()

# Graph density
density = nx.density(G)

# Number of connected components
num_components = nx.number_connected_components(G)

print(f"Number of Nodes: {num_nodes}")
print(f"Number of Edges: {num_edges}")
print(f"Density: {density:.4f}")
print(f"Number of Connected Components: {num_components}")
```

```
Number of Nodes: 6
```

```
Number of Edges: 5
Density: 0.3333
Number of Connected Components: 1
```

6.0.2 Calculate degree centrality

[ ]:
```
degree_centrality = nx.degree_centrality(G)
print("Degree Centrality:", degree_centrality)
```

```
Degree Centrality: {0: 1.0, 77: 0.2, 14: 0.2, 20: 0.2, 56: 0.2, 95: 0.2}
```

6.0.3 Compute betweenness centrality

[ ]:
```
betweenness_centrality = nx.betweenness_centrality(G)
print("Betweenness Centrality:", betweenness_centrality)
```

```
Betweenness Centrality: {0: 1.0, 77: 0.0, 14: 0.0, 20: 0.0, 56: 0.0, 95: 0.0}
```

6.0.4 Compute closeness centrality

[ ]:
```
closeness_centrality = nx.closeness_centrality(G)
print("Closeness Centrality:", closeness_centrality)
```

```
Closeness Centrality: {0: 1.0, 77: 0.5555555555555556, 14: 0.5555555555555556,
20: 0.5555555555555556, 56: 0.5555555555555556, 95: 0.5555555555555556}
```

6.0.5 Compute eigenvector centrality

[ ]:
```
eigenvector_centrality = nx.eigenvector_centrality(G)
print("Eigenvector Centrality:", eigenvector_centrality)
```

```
Eigenvector Centrality: {0: 0.7071064011232681, 77: 0.3162279359862123, 14:
0.3162279359862123, 20: 0.3162279359862123, 56: 0.3162279359862123, 95:
0.3162279359862123}
```

7 Conclusion

The network analysis of the Facebook social graph effectively reveals its structural properties and key influencers. Through various centrality measures (degree, betweenness, closeness, and eigenvector), we identified important nodes and their roles in information flow. The graph's characteristics - including its density of connections and component structure - provide insights into the network's cohesiveness and community formation. These metrics are valuable for understanding social influence patterns and network dynamics within the Facebook community.

sma-10

November 23, 2024

1 Practical 10: Create a hyperlink network from the extracted links:

2 a. Represent the network using graph structure where nodes are web pages and edges are hyperlinks.

3 b. Apply page rank algorithm to the hyperlink network to identify important pages.

[ ]:
```
import networkx as nx
import matplotlib.pyplot as plt
```

[ ]:
```
filepath = "/content/drive/MyDrive/dataset/web-Google.txt"
```

[ ]:
```
G = nx.DiGraph()
```

[ ]:
```
with open(filepath, "r") as f:
    for line in f:
        if line.startswith("#"):
            continue
        from_node, to_node = map(int, line.strip().split())
        G.add_edge(from_node, to_node)
```

[ ]:
```
print(f"Number of Nodes: {G.number_of_nodes()}")
print(f"Number of Edges: {G.number_of_edges()}")
```

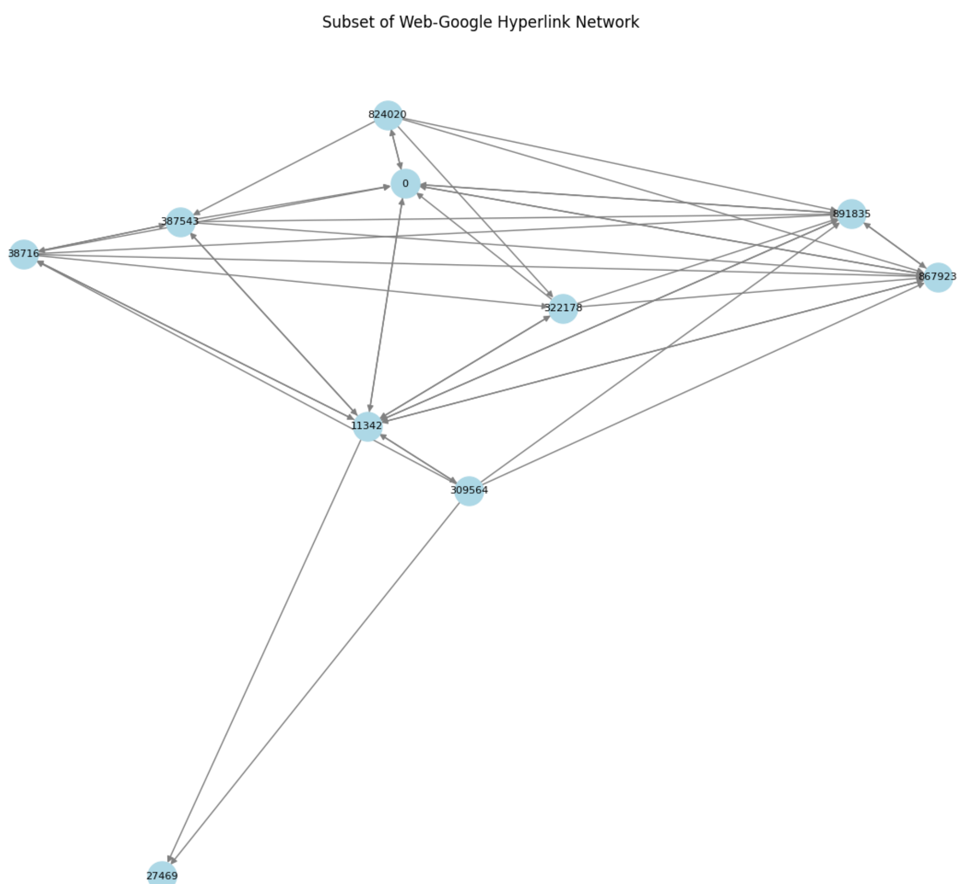
```
Number of Nodes: 875713
Number of Edges: 5105039
```

[ ]:
```
page_rank = nx.pagerank(G, alpha=0.85)
```

[ ]:
```
print("\nTop 10 Pages by PageRank:")
top_pages = sorted(page_rank.items(), key=lambda x: x[1], reverse=True)[:10]
for page, score in top_pages:
    print(f"Page {page}: {score:.6f}")
```

```
Top 10 Pages by PageRank:
Page 163075: 0.000952
Page 597621: 0.000901
Page 537039: 0.000895
Page 837478: 0.000876
Page 885605: 0.000822
Page 551829: 0.000790
Page 41909: 0.000779
Page 605856: 0.000779
Page 504140: 0.000746
Page 819223: 0.000710
```

[ ]:
```python
subgraph = G.subgraph(list(G.nodes)[:10])
plt.figure(figsize=(12, 10))
nx.draw(subgraph, with_labels=True, node_color='lightblue', edge_color='gray',␣
↪node_size=500, font_size=8)
plt.title("Subset of Web-Google Hyperlink Network")
plt.show()
```



4 Conclusion

The hyperlink network analysis of Google's web data successfully demonstrates the interconnectedness of web pages through PageRank algorithm implementation. By representing web pages as nodes and hyperlinks as edges, we identified the most influential pages in the network based on their PageRank scores. The visualization of the network subset provides a clear picture of how information and authority flow through these interconnected web pages, highlighting key hub pages that serve as central points in the web's structure.

sma-11

November 23, 2024

1 Practical 11: Write a program to analyze movement data.

[ ]:
```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
```

[ ]:
```
df = pd.read_csv("/content/drive/MyDrive/dataset/urban_mobility_dataset.csv")
```

[ ]:
```
df
```

[ ]:
```
                 timestamp  public_transport_usage  traffic_flow  \
0      2023-01-01 00:00:00                     292          3681
1      2023-01-01 01:00:00                     340          4743
2      2023-01-01 02:00:00                     372          3491
3      2023-01-01 03:00:00                     365          4360
4      2023-01-01 04:00:00                     226           121
…                      …                      …             …
999995 2137-01-29 11:00:00                     452          1117
999996 2137-01-29 12:00:00                     348           950
999997 2137-01-29 13:00:00                     130          1620
999998 2137-01-29 14:00:00                     177          3217
999999 2137-01-29 15:00:00                     431           540

        bike_sharing_usage  pedestrian_count weather_conditions day_of_week  \
0                      296              1939              Clear      Sunday
1                       96               688               Snow      Sunday
2                      183              1774               Rain      Sunday
3                      214                24               Rain      Sunday
4                      247               224               Snow      Sunday
…                    …                 …                 …          …
999995                 189               599              Clear     Tuesday
999996                 261              1344               Snow     Tuesday
999997                  87                98                Fog     Tuesday
999998                  12               516              Clear     Tuesday
999999                  19                97                Fog     Tuesday

        holiday event  temperature  humidity  road_incidents  \
```

```
0            0   NaN    24.547380      29               0
1            0   NaN    31.801722      99               3
2            0   NaN     0.052832      34               6
3            0   NaN    -3.757874      41               4
4            0   NaN    -4.948219      45               3
…         …   …          …      …            …
999995       0   NaN    34.738752      90               0
999996       0   NaN    10.753334      75               1
999997       1   NaN    16.771888      55               8
999998       0   NaN    -7.029623      48               9
999999       0   NaN    31.946573      69               0

        public_transport_delay  bike_availability  pedestrian_incidents
0                     5.263106                 22                     4
1                     0.523627                 88                     2
2                     0.408793                 93                     2
3                    27.640844                 89                     3
4                    14.820891                 49                     3
…                        …                …                   …
999995                9.414779                  3                     0
999996               26.964666                 88                     3
999997                0.019880                 94                     0
999998                2.938070                 12                     4
999999               28.519219                 78                     0

[1000000 rows x 15 columns]
```

[ ]:
```python
columns = [
    'timestamp', 'public_transport_usage', 'traffic_flow', 'bike_sharing_usage',
    'pedestrian_count', 'road_incidents', 'public_transport_delay',
    'bike_availability', 'pedestrian_incidents'
]
df = df[columns]
df
```

[ ]:
```
                  timestamp  public_transport_usage  traffic_flow  \
0       2023-01-01 00:00:00                     292          3681
1       2023-01-01 01:00:00                     340          4743
2       2023-01-01 02:00:00                     372          3491
3       2023-01-01 03:00:00                     365          4360
4       2023-01-01 04:00:00                     226           121
…                    …                      …           …
999995  2137-01-29 11:00:00                     452          1117
999996  2137-01-29 12:00:00                     348           950
999997  2137-01-29 13:00:00                     130          1620
999998  2137-01-29 14:00:00                     177          3217
999999  2137-01-29 15:00:00                     431           540
```

```
        bike_sharing_usage  pedestrian_count  road_incidents  \
0                      296              1939               0
1                       96               688               3
2                      183              1774               6
3                      214                24               4
4                      247               224               3
…                        …                 …               …
999995                 189               599               0
999996                 261              1344               1
999997                  87                98               8
999998                  12               516               9
999999                  19                97               0

        public_transport_delay  bike_availability  pedestrian_incidents
0                     5.263106                 22                     4
1                     0.523627                 88                     2
2                     0.408793                 93                     2
3                    27.640844                 89                     3
4                    14.820891                 49                     3
…                            …                  …                     …
999995                9.414779                  3                     0
999996               26.964666                 88                     3
999997                0.019880                 94                     0
999998                2.938070                 12                     4
999999               28.519219                 78                     0

[1000000 rows x 9 columns]
```

[ ]:
```python
G = nx.DiGraph()
```

[ ]:
```python
transport_modes = ['public_transport_usage', 'bike_sharing_usage',␣
 ↪'pedestrian_count']
for mode in transport_modes:
    G.add_node(mode)
```

[ ]:
```python
for index, row in df.iterrows():
    for i, mode1 in enumerate(transport_modes):
        for mode2 in transport_modes[i+1:]:
            if row[mode1] > 0 and row[mode2] > 0:
                if G.has_edge(mode1, mode2):
                    G[mode1][mode2]['weight'] += 1
                else:
                    G.add_edge(mode1, mode2, weight=1)
```

[ ]:
```python
degree_centrality = nx.degree_centrality(G)
print("\nDegree Centrality (most connected modes):")
```

```python
print(sorted(degree_centrality.items(), key=lambda x: x[1], reverse=True))
```

```
Degree Centrality (most connected modes):
[('public_transport_usage', 1.0), ('bike_sharing_usage', 1.0),
('pedestrian_count', 1.0)]
```

[ ]:
```python
betweenness_centrality = nx.betweenness_centrality(G)
print("\nBetweenness Centrality (key transition modes):")
print(sorted(betweenness_centrality.items(), key=lambda x: x[1], reverse=True))
```

```
Betweenness Centrality (key transition modes):
[('public_transport_usage', 0.0), ('bike_sharing_usage', 0.0),
('pedestrian_count', 0.0)]
```

[ ]:
```python
closeness_centrality = nx.closeness_centrality(G)
print("\nCloseness Centrality (proximity of modes):")
print(sorted(closeness_centrality.items(), key=lambda x: x[1], reverse=True))
```

```
Closeness Centrality (proximity of modes):
[('pedestrian_count', 1.0), ('bike_sharing_usage', 0.5),
('public_transport_usage', 0.0)]
```

[ ]:
```python
pagerank = nx.pagerank(G, weight='weight')
print("\nPageRank (importance of modes/locations):")
print(sorted(pagerank.items(), key=lambda x: x[1], reverse=True))
```

```
PageRank (importance of modes/locations):
[('pedestrian_count', 0.5209510481443506), ('bike_sharing_usage',
0.2814461986397785), ('public_transport_usage', 0.19760275321587095)]
```

[ ]:
```python
location_data = {
    'Public Transport': df['public_transport_usage'].sum(),
    'Bike Sharing': df['bike_sharing_usage'].sum(),
    'Pedestrian': df['pedestrian_count'].sum()
}
```

[ ]:
```python
location_usage = pd.DataFrame(list(location_data.items()), columns=['Location',␣
↪'Usage'])
location_usage_sorted = location_usage.sort_values(by='Usage', ascending=False)
```

[ ]:
```python
print("Top Locations Based on User Movement:")
print(location_usage_sorted.head())
```

```
Top Locations Based on User Movement:
```

```
            Location       Usage
2         Pedestrian  1009578256
0   Public Transport   274474218
1       Bike Sharing   149452101
```

2 Conclusion

The urban mobility analysis effectively reveals transportation patterns and user preferences across different modes of transport. By analyzing public transport, bike-sharing, and pedestrian movement data through network centrality measures and PageRank, we identified key transition points and popular routes. The usage statistics highlight dominant transportation modes, providing valuable insights for urban planning and optimization of city mobility services.

sma-12

November 23, 2024

1 Practical 12: Write a program to understand how retailer use location analytics to analyze foot traffic patterns and optimize store placement.

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
import pandas as pd
```

[ ]:
```
df = pd.read_csv("/content/drive/MyDrive/dataset/dataset.csv")
```

[ ]:
```
df
```

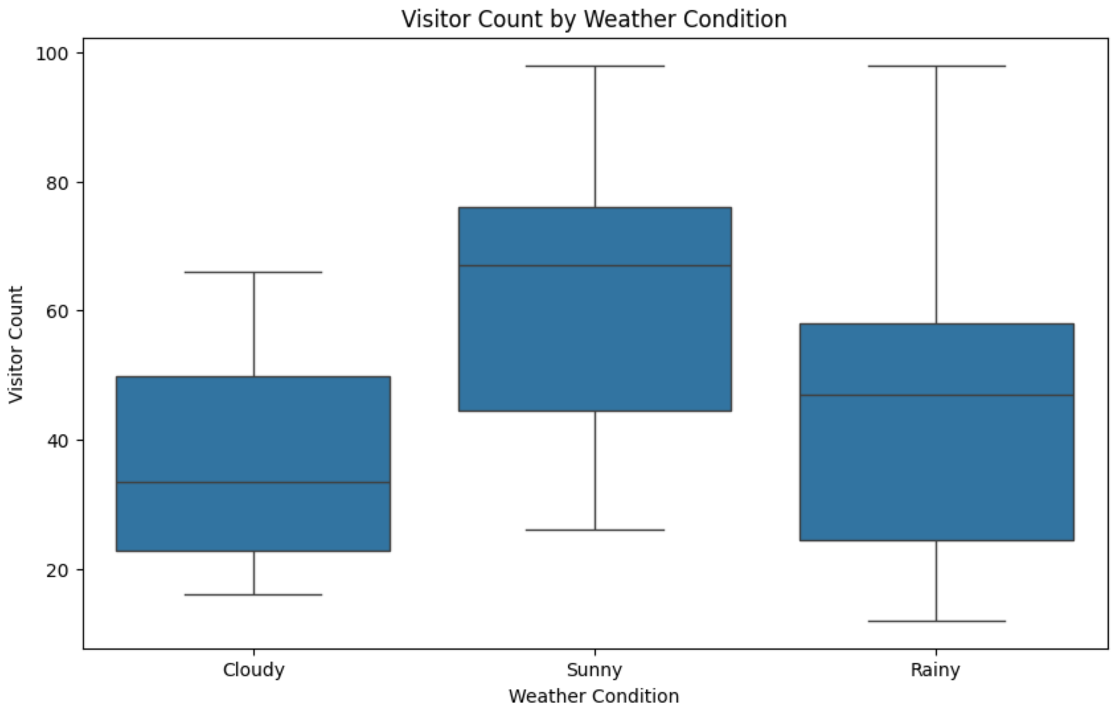
[ ]:
```
     Store ID            Timestamp  Visitor Count Weather Condition  \
0     Store_3  2024-11-01 09:00:00             48            Cloudy
1     Store_8  2024-11-01 10:00:00             80             Sunny
2     Store_4  2024-11-01 11:00:00             57             Rainy
3    Store_19  2024-11-01 12:00:00             32             Sunny
4     Store_3  2024-11-01 13:00:00             25            Cloudy
..        …                  …            …                 …
95    Store_8  2024-11-05 08:00:00             45             Sunny
96   Store_19  2024-11-05 09:00:00             65             Rainy
97   Store_11  2024-11-05 10:00:00             31            Cloudy
98   Store_12  2024-11-05 11:00:00             71             Sunny
99    Store_7  2024-11-05 12:00:00             27             Sunny

   Day of Week  Season      Special Event            Store Type  \
0       Friday  Autumn                NaN           Electronics
1       Friday  Autumn  Holiday Promotion  Umbrella & Rainwear
2       Friday  Autumn  Holiday Promotion              Clothing
3       Friday  Autumn                NaN              Footwear
4       Friday  Autumn                NaN  Umbrella & Rainwear
..         …       …                  …                     …
95     Tuesday  Autumn                NaN              Footwear
```

```
96      Tuesday   Autumn                NaN  Umbrella & Rainwear
97      Tuesday   Autumn                NaN  Umbrella & Rainwear
98      Tuesday   Autumn               Sale            Groceries
99      Tuesday   Autumn                NaN                 Toys

     Store Location Retail Area Holiday
0      Ground Floor       Large      No
1      Ground Floor       Large      No
2      Ground Floor       Large      No
3      Ground Floor      Medium      No
4   Near Food Court       Small      No
..              …           …       …
95      First Floor       Small      No
96  Near Food Court       Small      No
97     Second Floor       Large      No
98      First Floor      Medium      No
99  Near Food Court       Large      No

[100 rows x 11 columns]
```

[ ]:
```
df['Timestamp'] = pd.to_datetime(df['Timestamp'])
```

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Weather Condition', y='Visitor Count', data=df)
plt.title('Visitor Count by Weather Condition')
plt.show()
```



[ ]:
```
data_encoded = df.copy()
data_encoded['Weather Condition'] = df['Weather Condition'].astype('category').
  ↪cat.codes
data_encoded['Day of Week'] = df['Day of Week'].astype('category').cat.codes
data_encoded['Season'] = df['Season'].astype('category').cat.codes
data_encoded['Special Event'] = df['Special Event'].astype('category').cat.codes
data_encoded['Store Type'] = df['Store Type'].astype('category').cat.codes
data_encoded['Store Location'] = df['Store Location'].astype('category').cat.
  ↪codes
data_encoded['Retail Area'] = df['Retail Area'].astype('category').cat.codes
data_encoded['Holiday'] = df['Holiday'].astype('category').cat.codes
```

[ ]:
```
data_encoded.head()
```

[ ]:
```
   Store ID           Timestamp  Visitor Count  Weather Condition  \
0   Store_3 2024-11-01 09:00:00             48                  0
1   Store_8 2024-11-01 10:00:00             80                  2
2   Store_4 2024-11-01 11:00:00             57                  1
3  Store_19 2024-11-01 12:00:00             32                  2
4   Store_3 2024-11-01 13:00:00             25                  0

   Day of Week  Season  Special Event  Store Type  Store Location  \
0            0       0             -1           1               1
1            0       0              0           5               1
```

```
2            0       0              0         0              1
3            0       0             -1         2              1
4            0       0             -1         5              2

   Retail Area  Holiday
0            0        0
1            0        0
2            0        0
3            1        0
4            2        0
```

[ ]:
```python
import re
def extract_store_number(store_id):
  match = re.search(r'\d+', store_id)
  if match:
    return int(match.group())
  else:
    return None
```

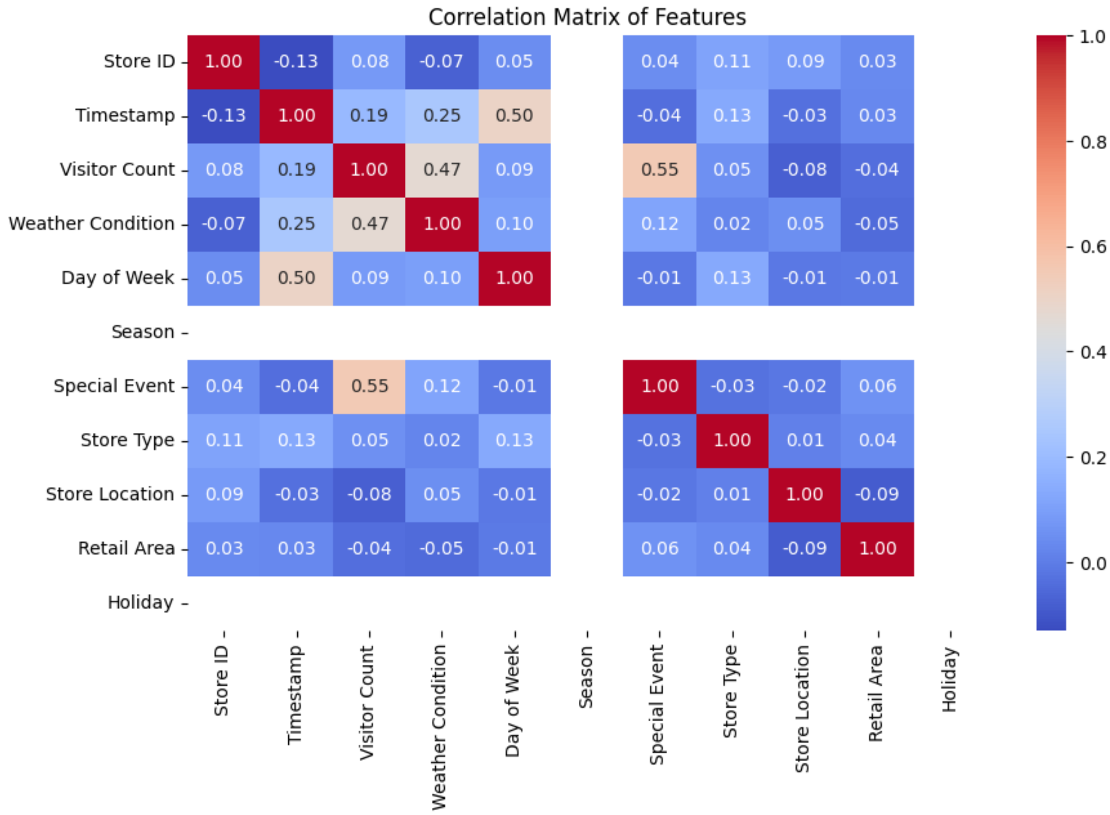
[ ]:
```python
data_encoded['Store ID'] = data_encoded['Store ID'].apply(extract_store_number)
```

[ ]:
```python
data_encoded.head()
```

[ ]:
```
   Store ID           Timestamp  Visitor Count  Weather Condition  \
0         3 2024-11-01 09:00:00             48                  0
1         8 2024-11-01 10:00:00             80                  2
2         4 2024-11-01 11:00:00             57                  1
3        19 2024-11-01 12:00:00             32                  2
4         3 2024-11-01 13:00:00             25                  0

   Day of Week  Season  Special Event  Store Type  Store Location  \
0            0       0             -1           1               1
1            0       0              0           5               1
2            0       0              0           0               1
3            0       0             -1           2               1
4            0       0             -1           5               2

   Retail Area  Holiday
0            0        0
1            0        0
2            0        0
3            1        0
4            2        0
```

[ ]:
```python
plt.figure(figsize=(10, 6))
sns.heatmap(data_encoded.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Features')
```

```
plt.show()
```



[ ]:
```
from sklearn.ensemble import RandomForestRegressor
```

[ ]:
```
X = data_encoded.drop(columns=['Visitor Count', 'Timestamp', 'Store ID'])
y = data_encoded['Visitor Count']
```

[ ]:
```
model = RandomForestRegressor(n_estimators=100)
model.fit(X, y)
```

[ ]: RandomForestRegressor()

[ ]:
```
df['Predicted Visitor Count'] = model.predict(X)

df[['Store ID', 'Visitor Count', 'Predicted Visitor Count']]
```

[ ]:

	Store ID	Visitor Count	Predicted Visitor Count
0	Store_3	48	40.840000
1	Store_8	80	78.350000
2	Store_4	57	58.460000
3	Store_19	32	38.790000

```
4    Store_3             25                  30.580000
..       …          …                          …
95   Store_8             45                  50.340000
96  Store_19             65                  55.320000
97  Store_11             31                  40.290000
98  Store_12             71                  75.703333
99   Store_7             27                  34.920000

[100 rows x 3 columns]
```

2 Conclusion

The retail location analytics successfully predicts visitor traffic patterns using weather conditions, seasonal factors, and store characteristics. Through correlation analysis and Random Forest modeling, we identified key factors influencing foot traffic. The comparison between actual and predicted visitor counts provides valuable insights for optimizing store placement and operations. This data-driven approach helps retailers make informed decisions about store locations and resource allocation based on expected customer traffic patterns.