MA3831- ASSESSMENT 3: NEWS-PPRO

NEWS-PPRO: News Extraction With Sentiment Analysis for Price Prediction and Real-time Outlook

By: Rajesh Shah Punit

INDEX

BACKGROUND

The Indian stock market, often referred to as the Bombay Stock Exchange (BSE) and the National Stock Exchange (NSE), boasts a rich and complex historical background that has evolved over centuries. Its origins can be traced back to the late 18th century when informal trading of stocks and shares began under a banyan tree in Mumbai, eventually leading to the formal establishment of the BSE in 1875. The market's development was significantly influenced by colonial rule, economic reforms, and periods of volatility. In the early 1990s, India initiated economic liberalization, which led to significant changes in the stock market landscape, attracting both domestic and international investors. Today, the Indian stock market is one of the largest in the world, with a diverse array of companies listed and a vital role in the country's economic growth.

The Indian stock market's journey reflects the nation's transition from a closed, state-controlled economy to a dynamic and open marketplace, providing opportunities for investors to participate in India's rapid economic growth and transformation. With a blend of tradition and modernity, the Indian stock market continues to be a hub of financial activity and a barometer of the nation's economic health.

INTRODUCTION

The problem at hand is that traders in the financial industry rely heavily on Microsoft Excel for data analysis, risk management, and reporting, but they lack an efficient tool to monitor news and sentiment analysis of stocks within their portfolios. This gap hinders their ability to make informed financial decisions in a timely manner, potentially leading to missed opportunities or increased risks.

To address this issue, our solution involves the development of an Excel-based app, integrated with Python, that allows traders to effortlessly track news and sentiment for a selected group of stocks. By leveraging RSS feeds from preferred news outlets and incorporating sentiment analysis tools, we aim to provide traders with a comprehensive view of how external factors may impact their portfolios. Furthermore, our solution incorporates visualization tools to facilitate easy interpretation of data trends, making it an invaluable addition to a trader's toolkit. By presenting this user-friendly app and a well-documented report outlining our techniques, challenges, and learnings, we seek to convince traders of the tangible benefits and time-saving advantages our solution offers in monitoring their investments effectively.

SOURCE

In our quest to provide traders with the most reliable and up-to-date information, we've chosen to integrate the MoneyControl website into our Excel-based app for several compelling reasons. It has several advantages such as Comprehensive

coverage, Real-time updates, News Coverage, Historical Data, Market Tools, User-Friendly Interface, Mobile Accessibility, Research Reports, Educational Resources, Community and forums , API Access. MoneyControl is a well-established and trusted financial news platform in India, renowned for its comprehensive coverage of the stock market, economy, and corporate developments. Its user-friendly interface and extensive database of news articles make it an ideal choice for sourcing real-time financial data. By harnessing the power of MoneyControl, we ensure that traders have access to a vast array of news sources and sentiment analysis tools, enhancing the accuracy and relevance of the information they rely upon to make crucial investment decisions.

CODE BRIEF UNDERSTNDING

code descriptions for each of the three Jupyter Notebook files: GroupProject.ipynb, GroupProjectPro.ipynb, and GroupProjectPro1.ipynb. These descriptions should give a high-level overview of what each file does.

1. Code Description for GroupProject.ipynb:
   GroupProject.ipynb is a Jupyter Notebook file that focuses on web scraping and data extraction from the MoneyControl website. It performs the following tasks:
   - Scrapes data related to the NIFTY 500 companies, including their names and financial metrics.
   - Cleans and preprocesses the extracted data.
   - Saves the company names and financial data to CSV files.
   - Merges the data into a final CSV file.
   - Converts the final CSV file into an Excel spreadsheet.
   - Removes unnecessary intermediate files.

   This notebook is an essential part of the data collection pipeline for the research project.

2. Code Description for GroupProjectPro.ipynb:
   GroupProjectPro.ipynb is a Jupyter Notebook file that focuses on news extraction and sentiment analysis from various RSS feeds provided by MoneyControl. It performs the following tasks:
   - Extracts news articles from different categories using RSS feeds.
   - Analyzes the sentiment of each article, categorizing it as positive, negative, or neutral.
   - Generates word clouds to visualize the most common words in articles for each sentiment category.

   This notebook aids in the analysis of news sentiment related to financial markets and stocks.

3. Code Description for GroupProjectPro1.ipynb:

GroupProjectPro1.ipynb is a Jupyter Notebook file that focuses on extracting and displaying article content from the MoneyControl website. It performs the following tasks:

- Retrieves the HTML source code of the main MoneyControl page.
- Identifies and categorizes links to various news categories.
- Navigates through each category page to find and display articles.
- Extracts article titles, content, and publication dates.

This notebook is responsible for retrieving detailed information from selected news articles.

These code descriptions provide a brief overview of the functionality of each notebook, which will be useful for readers to understand the purpose and role of each code component in your research project.

APPROACH/METHODOLOGY

In this section, we describe the overall approach and methodology employed in our research project. Our project involves the development of a comprehensive tool for monitoring news and sentiment analysis for stock price prediction and real-time market outlook. The approach can be summarized as follows:

Data Collection and Preparation

1. Web Scraping for Financial Data (GroupProject.ipynb):
    - We initiate the data collection process by scraping financial data from the MoneyControl website. Specifically, we focus on the NIFTY 500 companies' information, including their names and financial metrics.
    - The data extraction process is facilitated through web scraping techniques using Python libraries such as BeautifulSoup.
    - Extracted data is cleansed and preprocessed to ensure consistency and accuracy.
    - We save the collected data to CSV files and merge them into a final dataset for further analysis.
2. News Extraction from RSS Feeds (GroupProjectPro.ipynb):
    - To provide real-time news updates, we extract news articles from various categories using RSS feeds from MoneyControl.
    - These RSS feeds cover categories such as Latest News, Buzzing Stocks, Technicals, Market Reports, Technology, and International Markets.

- We use Python libraries like feedparser to parse RSS feeds and gather news articles.

Sentiment Analysis

1. Sentiment Analysis (GroupProjectPro.ipynb):
    - Sentiment analysis is performed on the extracted news articles to gauge the sentiment associated with each article.
    - We employ the VADER (Valence Aware Dictionary and Sentiment Reasoner) sentiment analysis tool to determine sentiment polarity.
    - Articles are categorized into positive, negative, or neutral sentiments based on sentiment scores.

Visualization

1. Word Cloud Generation (GroupProjectPro.ipynb):
    - To provide visual insights into the content of news articles, we generate word clouds for each sentiment category (positive, negative, and neutral).
    - Word clouds highlight the most frequently occurring words in articles, aiding in the interpretation of sentiment trends.

Article Content Extraction

1. Article Content Extraction (GroupProjectPro1.ipynb):
    - To provide more detailed information, we extract and display the content of selected news articles from MoneyControl.
    - The content includes article titles, publication dates, and the main article text.
    - We navigate through the website to locate and retrieve this information.

CHALLENGES

While implementing our approach, we encountered several intricate challenges that demanded innovative solutions and careful consideration. These challenges encompassed various aspects of our project, from data collection to sentiment analysis and real-time updates:

1. Web Scraping Complexity:
    - Web scraping, while a powerful technique for data collection, presented challenges due to the dynamic nature of websites. MoneyControl's website structure may change over time, necessitating regular updates to our scraping code to maintain data integrity.

- The website's layout and structure, including changes to HTML elements, required vigilant monitoring to ensure accurate data extraction.

2. Diverse News Sources:
    - MoneyControl's RSS feeds encompass a wide range of news sources, leading to diverse article formats and styles. Parsing and extracting relevant information consistently across different sources posed a significant challenge.
    - Handling variations in article layouts and content structures required flexibility in our parsing algorithms.
3. Accurate Sentiment Analysis:
    - Achieving accurate sentiment analysis in the context of financial news presented challenges due to the nuanced and context-dependent nature of language.
    - Fine-tuning sentiment analysis tools such as VADER to suit financial news terminology was essential to obtain meaningful results.
4. Large Data Volumes:
    - Dealing with a large volume of real-time news data from multiple RSS feeds required efficient data processing and storage solutions.
    - Optimizing code for scalability and performance was crucial to prevent processing bottlenecks.
5. Code Robustness and Scalability:
    - Maintaining code robustness and ensuring it could scale to accommodate future enhancements and additional features was an ongoing challenge.
    - We needed to implement error handling mechanisms and logging to manage unexpected issues gracefully.
6. Latency in Real-Time Updates:
    - Real-time updates from RSS feeds introduced potential latency in data processing, affecting the timeliness of our sentiment analysis and news monitoring.
    - Balancing real-time data retrieval with the need for up-to-date insights required careful system design.
7. Market Complexity and Influencers:
    - Financial markets are influenced by a multitude of factors beyond news sentiment. Our approach primarily focuses on news-related factors but does not account for all market influencers.
    - Acknowledging the limitations of our approach in comprehensively capturing market dynamics was essential.

In summary, our project's success hinged on overcoming these multifaceted challenges, requiring a combination of technical expertise, adaptability, and continuous monitoring. Addressing these challenges allowed us to develop a robust and informative tool for monitoring news and sentiment analysis in the financial domain.

TECHNIQUES USED

Our project employed a blend of modern data science, natural language processing (NLP), web scraping, and data visualization techniques to develop an integrated tool for news extraction, sentiment analysis, and real-time monitoring of financial news. These techniques were pivotal in achieving our project objectives:

1. Web Scraping:
    - Beautiful Soup and Requests: To collect financial news data, we utilized the Python libraries Beautiful Soup and Requests. These libraries allowed us to scrape web pages, parse HTML content, and extract structured data from MoneyControl's website efficiently.
2. Data Parsing and Extraction:
    - HTML Parsing: We parsed the HTML structure of MoneyControl's web pages to extract relevant information such as article titles, descriptions, publication dates, and article URLs.
    - Regular Expressions: Regular expressions were employed to extract stock symbols and company names from news articles, enabling us to identify the entities mentioned in each article.
3. Sentiment Analysis:
    - VADER Sentiment Analysis: We used the VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis tool to assess the sentiment of news articles. VADER is a pre-trained model designed for sentiment analysis, particularly suited for social media and financial text. It provides sentiment scores, including positive, negative, and neutral scores.
4. Named Entity Recognition (NER):
    - spaCy NLP Library: Named Entity Recognition (NER) techniques from the spaCy NLP library were employed to identify and extract company names and stock symbols mentioned in news articles. This facilitated the association of news articles with specific companies.
5. Data Visualization:
    - Word Clouds: To visually represent the most frequently mentioned words in positive, negative, and neutral sentiment

articles, we generated word clouds using the WordCloud library. These word clouds provided an intuitive overview of the key topics discussed in the news.

- Matplotlib: The Matplotlib library was used to create graphical visualizations, including bar charts and line plots, to depict sentiment trends and changes over time.

6. Excel Integration:
    - OpenPyXL: We leveraged the OpenPyXL library to create and manipulate Excel files. This allowed us to compile and present the extracted data, sentiment scores, and company-related information in an easily accessible Excel format.
7. Error Handling and Logging:
    - Python Logging: We implemented error handling and logging using Python's built-in logging module. This ensured that the application could gracefully handle unexpected issues, track the execution flow, and log important events.
8. Real-Time Updates:
    - RSS Feeds: To provide real-time news updates, we integrated multiple RSS feeds from MoneyControl, categorizing news articles into sections such as Latest News, Buzzing Stocks, and Market Reports.
9. Code Modularity and Documentation:
    - Modular Code Structure: We structured our codebase into modular functions and classes to enhance code reusability and maintainability.
    - Documentation: Comprehensive code documentation, including comments and docstrings, was maintained to aid developers and users in understanding the code's functionality.
10. Data Storage and Management:
    - CSV Files: We stored intermediate data in CSV files for easy data management and manipulation.
    - Excel Workbook: The final compiled data, including sentiment scores and company-related information, was saved in an Excel workbook for user-friendly access.

By combining these techniques, we developed a comprehensive tool that extracts, analyzes, and visualizes financial news sentiment in real-time, empowering traders and investors with valuable insights for informed decision-making in the dynamic world of finance.

LIMITATIONS

While our research project offers valuable insights and tools for monitoring financial news and sentiment, it is essential to acknowledge its limitations, which are inherent in the current implementation:

1. Dependency on Data Sources:
    - Our project relies on external data sources, such as the MoneyControl website, for news extraction. Any changes or disruptions to these sources can impact the tool's reliability and functionality.
2. News Coverage:
    - The tool's effectiveness is contingent on the comprehensiveness of the selected news sources. It may not capture news articles from all relevant sources, potentially leading to information gaps.
3. Accuracy of Sentiment Analysis:
    - The sentiment analysis component, while valuable, may not always accurately capture the nuanced sentiment expressed in financial news. Financial language can be complex, and context is crucial.
4. Named Entity Recognition (NER) Challenges:
    - Identifying company names and stock symbols within news articles may not always be precise. NER accuracy can vary, potentially leading to incorrect associations between news and companies.
5. Real-time Delays:
    - The tool's real-time capabilities are subject to delays in data retrieval and processing. Users should be aware that the latest news and sentiment may not always be available instantly.
6. Language Limitations:
    - Currently, the tool primarily supports English-language news. Expanding language support to other languages remains a challenge, as language models and sentiment lexicons need to be adapted.
7. Overreliance on News Sentiment:
    - Relying solely on news sentiment for investment decisions can be risky. Other factors, such as financial fundamentals and market trends, should also be considered.
8. Sentiment Subjectivity:
    - Sentiment analysis is inherently subjective and may not always align with individual investor perceptions. Users should exercise caution and critical thinking when interpreting sentiment scores.

9. User Expertise:
    - Effective use of the tool requires some level of financial knowledge and expertise. Novice investors may find it challenging to interpret news and sentiment data accurately.
10. Security and Privacy Concerns:
    - Integrating the tool with personal financial accounts introduces security and privacy risks. Ensuring robust security measures is essential to protect user data.
11. Scalability:
    - As the user base grows, scalability becomes a concern. Maintaining real-time updates and responsiveness may require significant infrastructure investments.
12. Machine Learning Training Data:
    - If future models incorporate machine learning, obtaining and maintaining high-quality training data can be challenging, impacting the accuracy of sentiment analysis.
13. Regulatory Changes:
    - Financial regulations and compliance standards can change over time. Adapting the tool to stay compliant with evolving regulations is an ongoing challenge.
14. User Customization Complexity:
    - While user customization is desirable, striking the right balance between customization options and usability can be challenging. Too many settings may overwhelm users.
15. Technical Challenges:
    - Implementing advanced features, such as real-time alerts and multilingual support, can pose technical challenges that require continuous development and maintenance.

Understanding these limitations is crucial for users to make informed decisions when using our tool. While we strive for accuracy and reliability, financial markets are inherently unpredictable and complex, and our tool should be considered as one of many sources of information for investment decisions. Continuous improvements and refinements are essential to address these limitations and enhance the tool's capabilities over time.

FUTURE MODELS AND ENHANCEMENTS

Our current project serves as a solid foundation for real-time news extraction and sentiment analysis in the financial domain. However, there are several avenues for future models and enhancements that could further improve the utility and functionality of our tool:

1. Machine Learning-Based Sentiment Analysis:

- One promising avenue for improvement is the implementation of machine learning models for sentiment analysis. Training models on a labeled dataset of financial news articles could enhance sentiment accuracy and adaptability to diverse language patterns.

2. Custom Sentiment Lexicons:
    - Developing custom sentiment lexicons specific to the financial domain can improve the accuracy of sentiment analysis. These lexicons can include financial terms and expressions that carry unique sentiment connotations.
3. Enhanced Named Entity Recognition (NER):
    - Improving NER techniques to identify more precise company mentions and stock symbols can enhance the association of news articles with specific companies. This would provide users with more granular insights into their stock portfolios.
4. Multilingual Support:
    - Expanding language support beyond English can make the tool accessible to a broader international audience. Implementing language detection and translation features can facilitate the analysis of news in different languages.
5. Real-time Alerts and Notifications:
    - Implementing a notification system that alerts users to significant news events or sentiment shifts related to their portfolio holdings can enhance the tool's real-time monitoring capabilities.
6. User Customization:
    - Allowing users to customize the types of news sources, sentiment thresholds, and companies they want to monitor can provide a tailored experience that aligns with their specific investment strategies.
7. Historical Sentiment Analysis:
    - Incorporating historical sentiment analysis can enable users to assess how news sentiment has impacted stock performance over time. This historical perspective can aid in making informed investment decisions.
8. Integration with Trading Platforms:
    - Integrating our tool with popular trading platforms and brokerage accounts can streamline the decision-making process, allowing users to execute trades directly from the tool based on sentiment insights.
9. Machine Learning for Stock Price Prediction:
    - Developing machine learning models for stock price prediction, coupled with sentiment analysis, can provide a

holistic view of potential price movements based on news sentiment.

10. Enhanced Data Visualization:
    - Expanding the range of data visualization options, including interactive dashboards and heatmaps, can improve data interpretation and user experience.
11. Security and Data Privacy:
    - Ensuring robust security measures and data privacy compliance is critical, especially if the tool integrates with user financial accounts.
12. Scalability and Performance Optimization:
    - As the user base grows, optimizing the tool's performance and scalability will be essential to maintain real-time updates and responsiveness.
13. Feedback Mechanism:
    - Implementing a user feedback mechanism can help continuously improve the tool based on user suggestions and requirements.

These future models and enhancements aim to make our tool even more valuable for traders and investors, providing them with cutting-edge capabilities to monitor, analyze, and act upon financial news and sentiment in a rapidly evolving market landscape.

CONCLUSION

In conclusion, our research project represents a significant step toward addressing the critical need for efficient and data-driven tools in the financial industry. The development of an Excel-based application integrated with Python provides traders with a comprehensive solution for monitoring news and sentiment analysis of stocks within their portfolios. By combining the power of real-time news extraction from MoneyControl with sentiment analysis and data visualization, our tool empowers traders to make more informed and timely financial decisions.

Our project underscores the importance of technology-driven solutions in today's dynamic financial landscape. It bridges the gap between traditional financial analysis and modern data analytics, offering a user-friendly interface that streamlines the process of tracking news, understanding sentiment trends, and visualizing data patterns. By embracing automation and data-driven insights, traders can optimize their investment strategies and respond to market developments swiftly.

As we move forward, the continuous evolution of our tool remains paramount. We are committed to enhancing its capabilities, improving sentiment analysis

accuracy, expanding language support, and addressing any limitations identified during its implementation. Moreover, we recognize that the financial markets will continue to evolve, and our tool will adapt accordingly to meet the changing needs of traders and investors. Ultimately, our research project contributes to the ongoing transformation of the financial industry, empowering stakeholders to navigate the complexities of the Indian stock market with greater confidence and efficiency.

REFLECTION

Our journey throughout this research project has been both enlightening and challenging, offering us valuable insights into the intricacies of financial data analysis and sentiment tracking. As a team, we embarked on this endeavor with a shared enthusiasm for leveraging technology to address the pressing needs of traders and investors in the Indian stock market. Throughout the process, we encountered several key takeaways and areas of reflection.

First and foremost, we realized the immense potential that lies at the intersection of finance and technology. The financial industry is undergoing a rapid transformation, where data-driven decision-making is becoming increasingly indispensable. Our project exemplifies the power of technology in streamlining complex tasks, enabling us to automate news extraction, sentiment analysis, and data visualization. This experience reinforced our belief in the importance of staying at the forefront of technological advancements in finance.

Secondly, collaboration played a pivotal role in our project's success. Working as a cohesive team allowed us to pool our diverse skills and expertise, from web scraping to natural language processing, into a cohesive solution. It highlighted the significance of interdisciplinary collaboration and the collective impact it can have on tackling complex challenges.

Lastly, we gained a deeper appreciation for the challenges and opportunities inherent in real-world data analysis projects. We encountered unforeseen hurdles, such as data formatting issues and evolving web structures, which demanded adaptability and problem-solving skills. These challenges taught us the importance of resilience and continuous learning in the face of evolving data landscapes.

In retrospect, our research project not only equips us with technical skills but also fosters a mindset of innovation and adaptability. It reaffirms our commitment to harnessing technology for practical solutions and underscores the significance of staying attuned to the evolving needs of the financial industry.

CODE:

FILE1: GROUPPROJECT.IPYNB

```
from urllib.request import urlopen
from bs4 import BeautifulSoup as soup
import csv
from openpyxl import Workbook
import os

def moneycontrol(mc_url):
    # Fetch the Moneycontrol data
    mc_data = urlopen(mc_url)
    mc_html = mc_data.read()

    # Parse the HTML
    mc_soup = soup(mc_html, 'html.parser')

    # Extract column titles
    headers = mc_soup.findAll('th')

    column_titles = [ct.text for ct in headers]
    column_titles = column_titles[:6]
    # column_titles[6] = column_titles[6][0:17]
    ## Upto Done with Headers

    # Extract company names
    span = mc_soup.find_all('span', class_='gld13')
    company_span_list = [spani.text[:-37] for spani in span]
    # Save company names to a CSV file
    filename = 'mc_NIFTY_500_company.csv'
    with open(filename, 'w', encoding='utf-8', newline='') as f:
        company_name = '\n'.join(company_span_list)
        f.write(company_name)
        f.close()

    ## Now It's time for Valuable Digits
    # First of all we extract the elements which is not neccessory for final
data

    # Remove unnecessary HTML elements
    # Remove p elements
    tdrs = mc_soup.find_all('p')
    for re in tdrs:
```

```
        re.decompose()
    # Remove strongs elements
    tdrst = mc_soup.find_all('strong')
    for re in tdrs:
        re.decompose()
    tws = mc_soup.find_all('div', {'class': 'title2'})
    for re in tws:
        re.decompose()
    tws = mc_soup.find_all('td', {'class': 'vol'})
    for re in tws:
        re.decompose()
    tws = mc_soup.find_all('td', {'class': 'del'})
    for re in tws:
        re.decompose()
    twsas = mc_soup.find_all('td', {'width': '300'})
    for re in twsas:
        re.decompose()

    # Extract valuable data
    ## td_data is Our final Valuable Data
    td_data = mc_soup.find_all('td', {'align': 'right'})

    # Final digit List
    ## Little bit of data cleansing :)
    digit_list = []
    for tt in td_data[0:2505]:
        x = tt.text
        x = x.replace(",", "")
        digit_list.append(x)

    # Save data to a CSV file
    with open('mc_NIFTY_digits.csv', 'w', encoding='utf-8', newline='') as
f:
        writer = csv.writer(f)
        #      writer.writerow(column_titles[1:])
        for i in range(0, 2505, 5):
            writer.writerow([digit_list[i], digit_list[i + 1], digit_list[i + 2],
digit_list[i + 3], digit_list[i + 4]])
        f.close()

    # Clear entire csv file
    with open('final_mc_NIFTY_list.csv', 'w+', encoding='utf-8') as f:
        f.truncate()
        f.close
    #Merge data and company names into a final CSV file
```

```
    with open('mc_NIFTY_digits.csv', 'r', encoding='utf-8') as read_temp,
open('mc_NIFTY_500_company.csv', 'r',encoding='utf-8') as header,
open(
            'final_mc_NIFTY_list.csv', 'a', encoding='utf-8', newline='') as
final_list:
        reader = csv.reader(read_temp)
        writer = csv.reader(header)
        final_obj = csv.writer(final_list)

        final_obj.writerow(column_titles)  # Put Headings on top of list
        for a, b in zip(writer, reader):
            final_obj.writerow(a + b)  # write 50 rows (Company_name+
value)
        read_temp.close()
        header.close()
        final_list.close()

    ## Generate Excel file
    # Read the CSV file
    csv_filename = 'final_mc_NIFTY_list.csv'
    xlsx_filename = 'NIFTY_500_sheet.xlsx'
    # Create a new Excel workbook
    workbook = Workbook()
    worksheet = workbook.active
    # Open the CSV file and write its contents to the Excel worksheet
    with open(csv_filename, 'r', encoding='utf-8') as csv_file:
        csv_reader = csv.reader(csv_file)
        for row in csv_reader:
            worksheet.append(row)
    # Save the Excel workbook
    workbook.save(xlsx_filename)
    print(f'{xlsx_filename} has been created successfully!')


    # Now remove the unnnecessory CSV files
    os.remove('mc_NIFTY_500_company.csv')
    os.remove('mc_NIFTY_digits.csv')
    print('SpreadSheet is Ready for U!')

if __name__ == '__main__':
    mc_url = "https://www.moneycontrol.com/stocks/marketstats/nse-
mostactive-stocks/nifty-500-7/"
    moneycontrol(mc_url)

import pandas as pd
df = pd.read_csv('final_mc_NIFTY_list.csv')
```

```
df
```

FILE2: GROUPPROJECTPRO.IPYNB

```
import feedparser
import spacy
from vaderSentiment.vaderSentiment import
SentimentIntensityAnalyzer
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Define a list of RSS URLs and their respective categories
rss_feeds = [
   {"url": "https://www.moneycontrol.com/rss/latestnews.xml",
"category": "Latest News"},
   {"url": "https://www.moneycontrol.com/rss/buzzingstocks.xml",
"category": "Buzzing Stocks"},
   {"url": "https://www.moneycontrol.com/rss/technicals.xml",
"category": "Technicals"},
   {"url": "https://www.moneycontrol.com/rss/marketreports.xml",
"category": "Market Reports"},
   {"url": "https://www.moneycontrol.com/rss/technology.xml",
"category": "Technology"},
   {"url":
"https://www.moneycontrol.com/rss/internationalmarkets.xml",
"category": "International Markets"}
]

# Initialize spaCy and VADER Sentiment Analyzer
nlp = spacy.load("en_core_web_sm")
analyzer = SentimentIntensityAnalyzer()

# Function to extract stock mentions from text
def extract_stock_mentions(text):
   doc = nlp(text)
   stock_mentions = []
   for entity in doc.ents:
      if entity.label_ == "ORG":
         stock_mentions.append(entity.text)
   return list(set(stock_mentions))

# Function to perform sentiment analysis on text
def analyze_sentiment(text):
   sentiment_scores = analyzer.polarity_scores(text)
```

```
    compound_score = sentiment_scores["compound"]

    if compound_score >= 0.05:
        sentiment = "positive"
    elif compound_score <= -0.05:
        sentiment = "negative"
    else:
        sentiment = "neutral"

    return sentiment

# Initialize word cloud generators
positive_wordcloud = WordCloud(width=800, height=400,
background_color='green', colormap='Greens')
negative_wordcloud = WordCloud(width=800, height=400,
background_color='red', colormap='Reds')
neutral_wordcloud = WordCloud(width=800, height=400,
background_color='gray', colormap='gray')

# Loop through the RSS feeds and their categories
for feed_info in rss_feeds:
    rss_url = feed_info["url"]
    category = feed_info["category"]

    # Parse the RSS feed
    feed = feedparser.parse(rss_url)

    # Lists to store descriptions for each sentiment category
    positive_descriptions = []
    negative_descriptions = []
    neutral_descriptions = []

    # Loop through the feed entries (news articles)
    for entry in feed.entries:

        title = entry.title
        link = entry.link
        description = entry.description
        pub_date = entry.published
        guid = entry.guid

        positive_wordcloud_text = " "
        negative_wordcloud_text = " "
        neutral_wordcloud_text = " "
        # Extract stock mentions
        stock_mentions = extract_stock_mentions(description)
```

```
        # Perform sentiment analysis
        sentiment = analyze_sentiment(description)

        # Print the article details, stock mentions, and sentiment
        print("Title:", title)
        print("Link:", link)
        print("Description:", description)
        print("Publication Date:", pub_date)
        print("GUID:", guid)
        print("Stock Mentions:", stock_mentions)
        print("Sentiment:", sentiment)
        print("\n")

        if sentiment == "positive":
            positive_descriptions.append(description)
        elif sentiment == "negative":
            negative_descriptions.append(description)
        else:
            neutral_descriptions.append(description)

        # Generate word clouds for each sentiment category
        positive_wordcloud_text = " ".join(positive_descriptions)
        negative_wordcloud_text = " ".join(negative_descriptions)
        neutral_wordcloud_text = " ".join(neutral_descriptions)

        # Generate and display word clouds
        if sentiment == "positive":
            positive_cloud =
positive_wordcloud.generate(positive_wordcloud_text)
            plt.figure(figsize=(10, 5))
            plt.title(f"{category} - Positive Sentiment")
            plt.imshow(positive_cloud, interpolation="bilinear")
            plt.axis("off")
            plt.show()

        elif sentiment == "negative":
            negative_cloud =
negative_wordcloud.generate(negative_wordcloud_text)
            plt.figure(figsize=(10, 5))
            plt.title(f"{category} - Negative Sentiment")
            plt.imshow(negative_cloud, interpolation="bilinear")
            plt.axis("off")
            plt.show()

        else:
```

```
            neutral_cloud =
neutral_wordcloud.generate(neutral_wordcloud_text)
            plt.figure(figsize=(10, 5))
            plt.title(f"{category} - Neutral Sentiment")
            plt.imshow(neutral_cloud, interpolation="bilinear")
            plt.axis("off")
            plt.show()
```

FILE3:GROUPPROJECTPRO1.IPYNB

```
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin

# Function to extract and display article title, content, and date
def extract_and_display_article_info(article_url):
    article_response = requests.get(article_url)
    if article_response.status_code == 200:
        article_page_source = article_response.text
        article_soup = BeautifulSoup(article_page_source, 'html.parser')

        # Extract title
        article_title = article_soup.title.string.strip()

        # Extract article content
        article_content = ""
        article_content_div = article_soup.find('div',
class_='article_content')
        if article_content_div:
            article_content = " ".join([p.text.strip() for p in
article_content_div.find_all('p')])

        # Extract article date
        article_date = ""
        article_date_div = article_soup.find('div', class_='article_schedule')
        if article_date_div:
            article_date = article_date_div.find('span').text.strip()

        print(f"Title: {article_title}")
        print(f"Date: {article_date}")
        print(f"Content:\n{article_content}\n")
    else:
        print(f"Failed to retrieve the article page. Status code:
{article_response.status_code}")
```

```
# Step 1: Retrieve the HTML source code of the main page
url = "https://www.moneycontrol.com/company-
article/hdfcbank/news/HDF01"
response = requests.get(url)

if response.status_code == 200:
    page_source = response.text
else:
    print("Failed to retrieve the web page. Status code:",
response.status_code)
    exit()

soup = BeautifulSoup(page_source, 'html.parser')
category_links = soup.find_all('a', href=True)
category_info = {}
skip_categories = [
    'News', 'HOMEPAGE', 'Home', 'Economy', 'Companies', 'Mutual
Funds', 'Personal Finance',
    'IPO', 'Startups', 'SME', 'India', 'World', 'Home', 'Stocks', 'Technical
Analysis', 'Commodity',
    'Currency', 'Gold Rate', 'Silver Rate', 'Trends', 'Latest News', 'Opinion',
'Personal Tech', 'Auto',
    'Fintech', 'Photos', 'Infographics', 'Videos', 'MC Learn', 'Politics',
'Entertainment', 'Travel',
    'Lifestyle', 'Health and Fitness', 'Education', 'Science', 'Books',
'Tech/Startups',
    'Corporate Deposits', 'Upcoming Chat', 'Previous Transcripts', 'All
Schedule', 'Previous Transcript',
    'Videos on Demand', 'Podcast on Demand', 'The Week on Dalal Street',
'Market Minutes',
    'MC Special Podcast', 'Simply Save', 'Policy Talks', 'Gold Rate', 'Silver
Rate', 'Home',
    'Coronavirus', 'Tech/Startups', 'Auto', 'Opinion', 'Politics', 'Personal
Finance', 'EPF Guide',
    'Photos', 'International', 'NEWS', 'All News', 'Business', 'Earnings',
'Management Interviews',
    'Stock Advice', 'Research Reports', 'Business', 'Earnings', 'Mgmt
Interviews', 'Stock Views',
    'Brokerage Reports', 'HDFC Bank News', 'Business News', 'News
Archive', 'Site Map'
]

for link in category_links:
    if '/news/' in link['href'] and link.text.strip() not in skip_categories:
        category_name = link.text.strip()
        category_url = urljoin(url, link['href'])
```

```
            category_info[category_name] = category_url

        print("Category Names and URLs:")
        for category_name, category_url in category_info.items():
            print(f"{category_name}: {category_url}")

            category_response = requests.get(category_url)
            if category_response.status_code == 200:
                category_page_source = category_response.text
                category_soup = BeautifulSoup(category_page_source,
        'html.parser')
                page_title = category_soup.title.string.strip()
                print(f"Page Title: {page_title}\n")

                # Find and display articles within the category page
                article_links = category_soup.find_all('a', href=True,
        class_='g_14bl')
                for article_link in article_links:
                    article_relative_url = article_link['href']
                    article_url = urljoin(category_url, article_relative_url)
                    extract_and_display_article_info(article_url)

            else:
                print(f"Failed to retrieve the web page. Status code:
        {category_response.status_code}")

print("\n" + "-" * 50 + "\n")
```