Tweet and Image Analysis of NSE Top 20 Firms

Objective

The objective of this project is to collect and analyze the latest tweets from the top 20 companies listed on the NSE. The goal is to create a clean and structured dataset that includes the tweet text, engagement details (likes and reshares), and information about any images attached to the tweets—such as how many images are present, whether they contain humans, and the gender of the people in the images. This dataset can be useful for studying how companies communicate on social media and how their content is received.

Company and Twitter Handle Compilation

To begin the project, I first compiled a list of the top 20 companies listed on the National Stock Exchange (NSE) of India. For this, I referred to the official NSE India website (https://www.nseindia.com/), where companies are ranked based on their market capitalization.

Once the company names were finalized, the next step was to identify their official X (formerly Twitter) handles. I manually verified each company's official presence on X by checking their websites, press releases, and verified social media links. Care was taken to ensure that the selected handles were authentic, actively maintained, and reflected the company's official communications.

The finalized list was structured as a Python dictionary with each entry containing the company name and its official X handle. This list served as the foundational input to the tweet scraping script. Here's a sample of the structure used:

```
COMPANIES = [

    {"name": "Reliance Industries Ltd", "handle": "RIL_Updates"},

    {"name": "HDFC Bank Ltd", "handle": "HDFC_Bank"},

    ...

]
```

Tweet Scraping with Selenium

To collect the most recent tweets from the top 20 NSE-listed firms, I developed an automated scraping script named selenium_twitter_scraper.py. This script uses the Selenium library to control a Chrome browser and simulate human-like interactions on the X (formerly Twitter) platform. This was necessary to handle dynamic content loading and bypass anti-scraping protections.

A. Purpose

The main aim of the script is to extract the latest ~100 tweets from each company's official X handle and collect structured information including:

- Tweet text
- Tweet URL and ID
- Timestamp
- Engagement metrics (likes, retweets, replies)
- Media content (image URLs and locally downloaded copies)

B. Workflow and Code Explanation

1. Company Input

The script starts with a predefined list of the top 20 NSE firms and their corresponding official X handles. This list drives the scraping process by looping over each handle.

2. WebDriver Setup (setup_driver())

This function configures the Chrome WebDriver with options to make the browsing session appear more human-like. Automation flags are disabled, and a real user-agent string is used to avoid detection by Twitter's anti-bot systems.

3. Optional Login (login_to_twitter(driver))

The script offers the option to log into Twitter using credentials provided at runtime. Logging in allows access to more tweets and richer data, especially in cases where public content is limited.

4. Scraping Tweets (scrape_tweets(driver, twitter_handle))

This function navigates to a company's tweet timeline and scrolls through the page to load more tweets. For each visible tweet:

- The HTML structure is analyzed using Selenium to extract:
    - Tweet ID and permalink (used to construct the full tweet URL)
    - Tweet text
    - Timestamp
    - Number of likes, retweets, and replies

    - Image URLs (excluding profile or avatar pictures)

- The script attempts up to 30 scrolls per company to collect the required number of tweets.

5. Extracting Tweet Content (extract_tweet_data(article))

Each tweet (<article> element) is parsed to extract structured information. Engagement metrics are parsed carefully to handle various notations (e.g., 1.2K, 3M), and images are collected using multiple selector paths to ensure all media are captured.

6. Downloading Images (download_image(url, folder, filename))

If a tweet includes images, they are downloaded using the requests library. Each image is saved locally in a folder named after the Twitter handle and timestamp, and file names include the tweet ID for easy mapping.

7. Saving Output (save_to_csv(tweets, twitter_handle))

All tweets for a given company are saved to a structured CSV file. Each row contains:

- Tweet_ID, Tweet_Text, Date, Likes, Retweets, Replies, Image_URLs, and Local_Images

8. Orchestration (main())

This function controls the overall process:

- Iterates over the 20 companies
- Sets up a new browser instance for each company
- Logs in if enabled
- Scrapes tweets and downloads images
- Writes a company-specific CSV
- Appends a summary entry to a master CSV including the company name, number of tweets, number of images, and file path

To avoid detection, the script introduces random wait times between companies.

C. Output

For each company, the script generates:

- A timestamped CSV file containing up to 100 tweets
- A folder with locally saved images
- A master summary CSV file named company_tweets_summary_<timestamp>.csv, which lists:

    - Company Name
    - Twitter Handle
    - Path to the tweet CSV file
    - Number of tweets scraped
    - Number of tweets containing images

D. Significance

This script forms the foundation of the data pipeline. It not only automates tweet collection but also structures the data for downstream analysis. The inclusion of image downloading is particularly important, as it enables later steps like detecting human presence and estimating gender from tweet images.

Its modular design and resilient scraping logic make it adaptable for scaling to additional companies or timelines, and it ensures reliable data extraction under dynamic web conditions.

Merging Tweet Files into a Master Dataset

After individually collecting and storing tweets for each of the top 20 NSE companies, the next step was to consolidate these files into one unified dataset. The script merge_company_tweets.py performs this task by systematically reading all individual CSV files, standardizing the data, and merging them into a single, organized CSV file suitable for further analysis.

A. Purpose

- Combine all company-specific tweet files into one master dataset.
- Append company name and handle for each tweet to ensure source traceability.
- Ensure uniform formatting across datasets collected at different times.
- Generate a single CSV file that simplifies analysis and visualization.

B. Workflow and Code Explanation

1. Directory and File Setup

- The script looks for tweet CSV files in the company_tweets/ directory.
- It filters only files matching the pattern *_tweets_*.csv, excluding any summary files.

2. Reading and Processing Files

- For each matching file:
    - Extracts the Twitter handle from the filename.

    - Uses the function get_company_name(handle) to retrieve the corresponding company name from the most recent summary file.
    - Read the tweet data using Python's csv.DictReader().

3. Row Standardization

- For each tweet, a standardized dictionary is created with the following fields:
    - Company_Name
    - Company_Handle
    - S.No (Serial Number from original file)
    - Tweet_ID, Date, Tweet_Text
    - Engagement metrics: Likes, Retweets, Replies
    - Image metadata: Image_URLs, Local_Images
- If column headers differ slightly (e.g., tweet_id vs Tweet_ID), the script accounts for this using .get() with fallback keys.

4. Sorting and Output

- After collecting all tweet rows across companies, the script sorts tweets by date in descending order to prioritize the most recent content.
- It reassigns a fresh serial number (S.No) for the final dataset.
- Writes all merged records to a CSV file named:
  all_company_tweets_<timestamp>.csv in the root directory.

C. get_company_name(company_handle) Function

- This function ensures every tweet row includes the full company name (not just the Twitter handle).
- It reads the latest summary CSV file generated during scraping.
- Matches the handle to the company name listed in the summary.

D. Outcome of This Script

The script produces one clean, consolidated file with the following columns:

Company_Name, Company_Handle, S.No, Tweet_ID, Date, Tweet_Text, Likes, Retweets, Replies, Image_URLs, Local_Images

This unified dataset:

- Combines tweet data from all 20 companies
- Is sorted by recency
- Is ready for content and image analysis in the next stage

E. Importance

This step was essential to:

- Eliminate the complexity of working with 20 separate files
- Provide a centralized dataset for downstream processing (e.g., human detection, gender classification)
- Ensure consistency and traceability in the data pipeline

Image and Content Analysis

To complete the data enrichment process, I developed the script analyze_tweet_images.py, which analyzes images attached to tweets in order to identify the presence of humans and estimate their gender. This step adds a valuable layer of information to the textual and engagement data collected earlier, making the final dataset suitable for visual content analysis and behavioral studies.

A. Purpose

The primary goal of this script was to enhance the previously merged tweet dataset by:

- Counting the number of images associated with each tweet.
- Detecting the presence of human faces using face detection techniques.
- Estimating the gender of individuals present in images using a basic heuristic.
- Creating a structured output that combines all this enriched information.

B. Workflow and Code Explanation

1. Input and Output

- Input: A merged tweet CSV file from the earlier step (e.g., all_company_tweets_20250517_155120.csv).
- Output: A new file (enhanced_tweets_<timestamp>.csv) that includes the original tweet data along with new image analysis fields.

2. Image Source Handling

Using the function get_image_paths(), the script attempts to:

- First locate and use local image files listed in the Local_Images column.
- If not available, download images from URLs listed in the Image_URLs column.
- All newly downloaded images are temporarily stored in a temp_images/ directory.

3. Human and Gender Detection

The analyze_image() function performs the following:

- Loads and converts the image to grayscale.
- Applies OpenCV's Haar Cascade classifier to detect faces.
- Based on the face(s) detected:
  - No face: human_present = "no", gender = "unknown"
  - One face: uses a simple heuristic (face width vs. height) to guess gender as "male" or "female".
  - Multiple faces: sets gender = "both".
- Although basic, this approach provides a placeholder for real-world ML models.

4. Per-Tweet Processing

Using the process_tweet() function:

- The script constructs a proper tweet URL using the company handle and tweet ID.
- It records:
  - The number of images in the tweet.
  - Whether any image contains a human face.
  - An overall gender estimate based on all images.
  - Paths of images used for analysis.
- The result is a structured row ready for inclusion in the final output.

5. Overall Execution (main() function)

- Reads the input merged CSV using Pandas.
- Filters tweets with image content for full processing.
- Adds a random sample of 100 image-less tweets for completeness.
- All tweets are passed through process_tweet() and stored in a results list.
- The final DataFrame is sorted by date and saved as the enriched dataset.

C. Final Output Structure

Each tweet in the final dataset includes:

- tweet_url
- text
- likes_count, reshares_count
- num_images
- human_present (yes/no)
- detected_gender (male/female/both/unknown)
- image_file_paths (if applicable)
- Original metadata: company_name, company_handle, tweet_id, date

D. Significance

This script transforms the dataset from a basic tweet log into a multimodal dataset—combining text, engagement metrics, and image-based insights. It enables research questions related to:

- Visual representation in corporate tweets
- Human presence and audience engagement
- Gender diversity in brand imagery

Extended Image and Content Analysis (YOLO + DeepFace)

Building on the initial image analysis module—which used OpenCV's Haar Cascade and heuristic methods for human and gender detection— I further enhanced the pipeline using more robust deep learning techniques. This step was aimed at improving the accuracy and reliability of insights extracted from visual content in corporate tweets.

A. Purpose

To significantly improve the accuracy of human presence and gender detection in tweet images using modern deep learning models:

- YOLOv8 (You Only Look Once) for real-time object detection of humans.
- DeepFace for gender classification using facial analysis.

B. Workflow and Code Explanation

The enhanced script (analyze_tweets_new_model.py) extends the previous image analysis by incorporating deep learning-based methods as follows:

1. Input and Output

- Input: A merged tweet dataset (e.g., all_company_tweets_20250517_155120.csv) that includes tweet text, metadata, and image paths.
- Output: A new enriched dataset (Test_enhanced_tweets_<timestamp>.csv) containing advanced image-based insights.

2. Image Retrieval

- Attempts to use Local_Images first.
- Falls back on downloading images from Image_URLs if local files are missing.
- Temporary images are stored in a temp_images/ directory.

3. YOLOv8 for Human Detection

- Loads a YOLOv8n model (yolov8n.pt) via the ultralytics package.
- Detects objects in each image, and filters for class 0 (person) to confirm human presence.

- Extracts bounding box coordinates of detected humans for further gender analysis.

4. DeepFace for Gender Classification

- Crops each detected human region and passes it to DeepFace.analyze() for gender prediction.
- The gender with the highest confidence is selected.
- Gender classifications returned: "male", "female", "both", or "unknown".

5. Tweet-Level Aggregation

Each tweet record includes:

- Total number of images.
- Whether a human was detected (human_present = "yes"/"no").
- Overall detected gender based on all valid images.
- Path of images analyzed.
- Preserved tweet metadata (text, likes, reshares, etc.).

C. Final Output Structure

In the extended version of the project, the same output structure is retained, but the values in the human_present and detected_gender columns are now derived using YOLOv8 and DeepFace.

D. Significance

The integration of YOLO and DeepFace into the image analysis pipeline marks a significant advancement in the analytical depth and precision of this project. By employing two widely recognized deep learning models—YOLOv8 for object detection and DeepFace for facial analysis—the dataset now offers reliable, automated insights into the visual components of corporate tweets.

This enhancement brings several key advantages:

- Improved Accuracy: The heuristic-based detection used previously has been replaced with robust, pre-trained models, substantially increasing the reliability of human and gender identification.
- Gender Representation Analysis: The use of DeepFace supports more nuanced studies into gender portrayal and diversity in brand imagery across top NSE-listed companies.
- Scalability: The modular design of the new script, along with the use of scalable pre-trained models, allows for seamless extension of this methodology to other firms, industries, or social media platforms.

Overall, this enriched dataset enhances the project's relevance for academic research, policy evaluation, and strategic communication analysis, making it both comprehensive and practically applicable.

Summary of Tools & Libraries Used

To successfully complete this project, I utilized a range of Python tools and libraries. These tools were chosen for their robustness, flexibility, and suitability for handling tasks like browser automation, web scraping, image processing, and data manipulation. Below is a breakdown of the key libraries used and their specific roles in the project:

1. Selenium

- **Purpose:** Automated browser control for scraping dynamic web content from Twitter/X.
- **Usage:** Used in selenium_twitter_scraper.py to simulate human interactions with Twitter, including login, scrolling, and extracting tweets.

2. WebDriver Manager

- **Purpose:** Automatically manages browser driver compatibility.
- **Usage:** Ensured the correct version of ChromeDriver was installed and used with Selenium, reducing setup friction.

3. OpenCV (cv2)

- **Purpose:** Image processing and face detection.
- **Usage:**
  - Used in analyze_tweet_images.py to detect human faces in tweet images.
  - Applied Haar Cascade classifier (haarcascade_frontalface_default.xml) for real-time face detection.

4. Pandas

- **Purpose:** Data loading, manipulation, and transformation.
- **Usage:**
  - Used in both analyze_tweet_images.py and other steps to read CSV files, filter rows, and generate final structured outputs.
  - Played a key role in combining tweet-level metadata with image analysis results.

5. CSV & Glob

- **CSV:**

    - Used across all scripts to read and write structured data in a lightweight, portable format.
    - Helped build and maintain consistency between interim and final datasets.

- **Glob:**
    - Used in merge_company_tweets.py to dynamically identify and load relevant tweet CSV files.

6. Requests

- **Purpose:** Image downloading over HTTP.
- **Usage:**
    - Downloaded images when local versions were not available using URLs from tweets.
    - Ensured that each image could be analyzed offline for human and gender detection.

7. Regular Expressions (re)

- **Purpose:** Pattern matching and string parsing.
- **Usage:** Extracted tweet IDs, cleaned metric values, and interpreted URL patterns in the scraping process.

8. OS and Path Utilities

- **Purpose:** File handling and path management.
- **Usage:**
    - Ensured images and CSVs were saved in properly structured directories.
    - Checked the existence of local files before re-downloading.

9. Datetime

- **Purpose:** Timestamp management for filenames and sorting.
- **Usage:**
    - Timestamped all generated output files for reproducibility.
    - Enabled proper chronological sorting of tweets in merged datasets.

10. Numpy

- **Purpose:** Numerical operations and array management (used minimally).
- **Usage:** Integrated by OpenCV for internal image processing tasks.

11. Ultralytics (YOLOv8)

- **Purpose:** Real-time object detection, specifically to detect humans in tweet images.

- **Usage:** Used to load a YOLOv8n model (yolov8n.pt) and identify human figures (class 0) in images by drawing bounding boxes around them for further gender analysis.

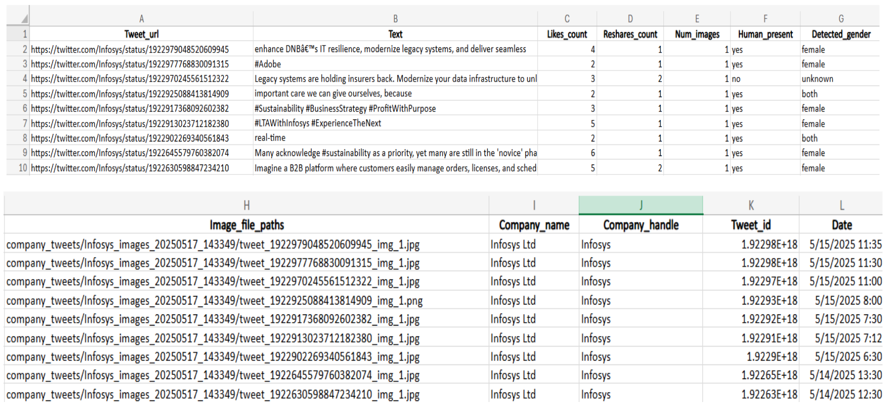
12. DeepFace

- **Purpose:** Facial attribute analysis using deep learning models.
- **Usage:** Applied to cropped human regions detected by YOLO to predict gender. The model returns gender probabilities, which are aggregated across images to classify tweets as featuring male, female, both, or unknown.

Final Output

The final output of this project is a clean and structured dataset that brings together tweet-level information and image analysis results for the top 20 companies listed on the National Stock Exchange (NSE). Each row in the dataset corresponds to a single tweet and includes details such as the tweet text, tweet URL, number of likes and reshares, number of images (if any), whether a human is present in those images, and the inferred gender(s) of the individuals identified.

Below is a sample preview of the final dataset format:



	A	B	C	D	E	F	G
1	Tweet_url	Text	Likes_count	Reshares_count	Num_images	Human_present	Detected_gender
2	https://twitter.com/Infosys/status/1922979048520609945	enhance DNBâ€™s IT resilience, modernize legacy systems, and deliver seamless	4	1	1	yes	female
3	https://twitter.com/Infosys/status/1922977768830091315	#Adobe	2	1	1	yes	female
4	https://twitter.com/Infosys/status/1922970245561512322	Legacy systems are holding insurers back. Modernize your data infrastructure to unl	3	2	1	no	unknown
5	https://twitter.com/Infosys/status/1922925088413814909	important care we can give ourselves, because	2	1	1	yes	both
6	https://twitter.com/Infosys/status/1922917368092602382	#Sustainability #BusinessStrategy #ProfitWithPurpose	3	1	1	yes	female
7	https://twitter.com/Infosys/status/1922913023712182380	#LTAWithInfosys #ExperienceTheNext	5	1	1	yes	female
8	https://twitter.com/Infosys/status/1922902269340561843	real-time	2	1	1	yes	both
9	https://twitter.com/Infosys/status/1922645579760382074	Many acknowledge #sustainability as a priority, yet many are still in the 'novice' pha	6	1	1	yes	female
10	https://twitter.com/Infosys/status/1922630598847234210	Imagine a B2B platform where customers easily manage orders, licenses, and sched	5	2	1	yes	female

H	I	J	K	L
Image_file_paths	Company_name	Company_handle	Tweet_id	Date
company_tweets/Infosys_images_20250517_143349/tweet_1922979048520609945_img_1.jpg	Infosys Ltd	Infosys	1.92298E+18	5/15/2025 11:35
company_tweets/Infosys_images_20250517_143349/tweet_1922977768830091315_img_1.jpg	Infosys Ltd	Infosys	1.92298E+18	5/15/2025 11:30
company_tweets/Infosys_images_20250517_143349/tweet_1922970245561512322_img_1.jpg	Infosys Ltd	Infosys	1.92297E+18	5/15/2025 11:00
company_tweets/Infosys_images_20250517_143349/tweet_1922925088413814909_img_1.png	Infosys Ltd	Infosys	1.92293E+18	5/15/2025 8:00
company_tweets/Infosys_images_20250517_143349/tweet_1922917368092602382_img_1.jpg	Infosys Ltd	Infosys	1.92292E+18	5/15/2025 7:30
company_tweets/Infosys_images_20250517_143349/tweet_1922913023712182380_img_1.jpg	Infosys Ltd	Infosys	1.92291E+18	5/15/2025 7:12
company_tweets/Infosys_images_20250517_143349/tweet_1922902269340561843_img_1.jpg	Infosys Ltd	Infosys	1.9229E+18	5/15/2025 6:30
company_tweets/Infosys_images_20250517_143349/tweet_1922645579760382074_img_1.jpg	Infosys Ltd	Infosys	1.92265E+18	5/14/2025 13:30
company_tweets/Infosys_images_20250517_143349/tweet_1922630598847234210_img_1.jpg	Infosys Ltd	Infosys	1.92263E+18	5/14/2025 12:30

Challenges and Limitations

During the execution of this project, a few challenges were encountered that are important to acknowledge. Since Twitter/X loads tweets dynamically, scraping them required simulating human-like scrolling using Selenium, which made the process slower and more complex. In addition, the platform has anti-scraping mechanisms, so care had to be taken to avoid triggering rate limits or blocks. There were also cases where images could not be downloaded or were too unclear to analyze properly. Finally, to keep the scraping efficient and avoid access issues, the number of tweets collected per company was capped at around 100. Despite these limitations, the overall data collection and enrichment process was successful and produced a valuable dataset for further analysis.

Conclusion

This project successfully demonstrated the end-to-end process of collecting, merging, and enriching social media data from the top 20 NSE-listed companies. By combining browser automation, structured data handling, and basic image analysis techniques, a comprehensive dataset was created that captures both the textual and visual content of corporate tweets. The final output includes not only tweet text and engagement metrics but also detailed information about image presence, human detection, and estimated gender. Despite some limitations in face detection and the simplified classification methods used, the project delivers a valuable foundation for further research.

Appendices

All Python scripts, outputs, and supporting files used in this project are available in the following GitHub repository:

Mallika3105/Data-Extraction-and-Image-Analysis-from-Tweets-of-NSE-Top-20-Companies