TikTok Trends Analysis:
Virality and Engagement Patterns in Short-Form Video

Wenjing Huang
USC ID: 3860016877
DSCI 510: Principles of Programming for Data Science

December 15, 2025

Abstract

Short-form video platforms such as TikTok and YouTube Shorts generate massive volumes of content, with a small fraction of videos achieving extremely high reach. Understanding how views and likes co-vary at scale helps quantify "virality" and typical engagement behavior. In this project, I analyze the Kaggle dataset *"YouTube Shorts and TikTok Trends 2025"* using a reproducible Python pipeline. After cleaning and normalizing engagement metrics, I compute descriptive statistics and generate visualizations that characterize the relationship between view counts and like counts, as well as correlations among key numeric variables. The results show a strong, but sub-linear, relationship between views and likes and an average engagement rate of approximately 7%, consistent with the intuition that not all impressions translate into active engagement.

1 Introduction

TikTok and other short-form video platforms are driven by algorithmic recommendation systems that can rapidly amplify content to millions of viewers. While many videos receive only modest reach, a subset becomes "viral," accumulating large numbers of views, likes, and other engagement signals. Quantitatively characterizing virality is important for creators, brands, and platform researchers.

In this project, I address the following research question:

Research Question. *What does the relationship between views and likes look like for highly viewed short-form videos, and what is the typical engagement rate across the dataset?*

To answer this question, I build a small but complete data science pipeline in Python. The pipeline (i) downloads the public Kaggle dataset *"YouTube Shorts and TikTok Trends 2025"*, (ii) cleans and normalizes the raw CSV, and (iii) performs exploratory data analysis and visualization. All code is packaged in the repository and can be re-run end-to-end with a few commands.

2 Changes from Original Proposal

My original project proposal aimed to analyze the relationship between **product price and sales volume** on TikTok Shop by scraping third-party tracking sites like FastMoss. However, I encountered significant technical challenges during implementation:

- **Anti-Bot Protections:** The target websites employed strict anti-scraping measures (Cloudflare, dynamic JavaScript) that prevented reliable data collection using standard Python libraries.
- **Data Availability:** Publicly accessible pricing data was often incomplete or obfuscated.

To ensure a robust and reproducible pipeline within the course timeline, I pivoted to using the **Kaggle API** to collect the *"YouTube Shorts and TikTok Trends 2025"* dataset. This allowed me to shift my research focus from "Sales Revenue" to **"Virality and Engagement,"** which provided a larger, cleaner, and more consistent dataset for statistical analysis ($N = 480$ high-quality samples vs. limited scraped data).

3 Data

3.1 Source

The primary dataset is the Kaggle dataset **"YouTube Shorts and TikTok Trends 2025"**. It contains aggregated metrics describing trending short-form videos across platforms. Access is provided via the Kaggle API, and the project uses the `kagglehub` Python package to handle authentication and downloading.

Conceptually, the raw CSV contains the following types of variables (exact schema may vary slightly):

- **Platform context:** country, platform (TikTok or YouTube Shorts), and a monthly timestamp `year_month`.
- **Volume:** `n_videos`, the number of videos contributing to a given aggregated row.
- **Reach:** `views`, an aggregated view count.
- **Engagement:** `avg_er`, an average engagement rate; `avg_velocity`, a measure of growth speed.
- **Label:** `trend_label`, describing the trend or topic.

3.2 Cleaning and Derived Variables

The script `src/clean_data.py` performs the following cleaning steps:

1. **Load raw CSV:** Read `data/raw/raw_tiktok_trends.csv` using `pandas`, skipping malformed lines when necessary.
2. **Normalize column names:**
   - `views` → `view_count`
   - `avg_er` → `engagement_rate`
   - `avg_velocity` → `velocity`
3. **Type coercion:** Convert `view_count`, `engagement_rate`, `velocity`, and `n_videos` to numeric types, coercing invalid strings to missing values.
4. **Filtering:**
   - Drop rows with missing `view_count` or `engagement_rate`.

- Remove rows with non-positive view counts, since they are not meaningful for virality analysis.

5. **Derived likes:** If the dataset does not provide an explicit `like_count` column, estimate it via

$$\text{like\_count} = \text{view\_count} \times \text{engagement\_rate},$$

assuming that engagement rate is expressed as a fraction of views.

6. **Output:** Save the cleaned data to `data/processed/cleaned_tiktok_trends.csv`.

After cleaning, the final dataset contains M rows (in my run, $M = 480$) with consistent numeric types and the key fields `view_count`, `like_count`, `engagement_rate`, `velocity`, and `n_videos`.

4 Methodology

4.1 Implementation Overview

The analysis pipeline is implemented in three main Python scripts:

- `get_data.py`: Uses `kagglehub` to download the Kaggle dataset and copy it into `data/raw/raw_tiktok_trends.c`
- `clean_data.py`: Cleans and normalizes the raw CSV and writes `data/processed/cleaned_tiktok_trends.csv`.
- `run_analysis.py`: Loads the cleaned data, calls visualization utilities, and generates a text report.

4.2 Scripts and Reproducibility

The project can be reproduced with the following commands:

1. Install dependencies via `pip install -r requirements.txt`.
2. Set Kaggle environment variables `KAGGLE_USERNAME` and `KAGGLE_API_TOKEN`.
3. Run:

```
python src/get_data.py
python src/clean_data.py
python src/run_analysis.py
```

The final results are written to the `results/` directory:

- `analysis_report.txt`
- `virality_scatter.png`
- `correlation_heatmap.png`

4.3 Analytical Approach

The analysis focuses on two main tasks:

1. **Descriptive engagement statistics.** Using `pandas`, `run_analysis.py` computes:
   - Mean, median and maximum view counts.
   - Mean and maximum like counts.
   - A simple engagement rate defined as `like_count/view_count`.

   These metrics are summarized in `analysis_report.txt`.

2. **Visualization.** The script generates:
   - A scatter plot of `view_count` vs `like_count` on log–log scales to better display several orders of magnitude in reach.
   - A correlation heatmap across numeric features, including views, likes, engagement rate, velocity, and number of videos.

   Both use `matplotlib` and `seaborn` for styling.

5 Results

5.1 Summary Statistics

A sample of the statistics computed in `analysis_report.txt` for my run is shown in Table 1.

Metric	Value
Total samples analyzed	480
Average views	9,945,618
Median views	9,753,744
Maximum views	18,611,773
Average likes	751,144
Maximum likes	1,647,700
Average engagement rate	7.13%

Table 1: Descriptive statistics from the cleaned dataset.

These numbers indicate that the dataset focuses on highly viewed content: the typical video in this sample has nearly 10 million views and hundreds of thousands of likes.

5.2 Virality Scatter Plot

Figure 1 (from `virality_scatter.png`) plots `view_count` against `like_count` on logarithmic scales. The log–log scaling is necessary because both variables span multiple orders of magnitude.

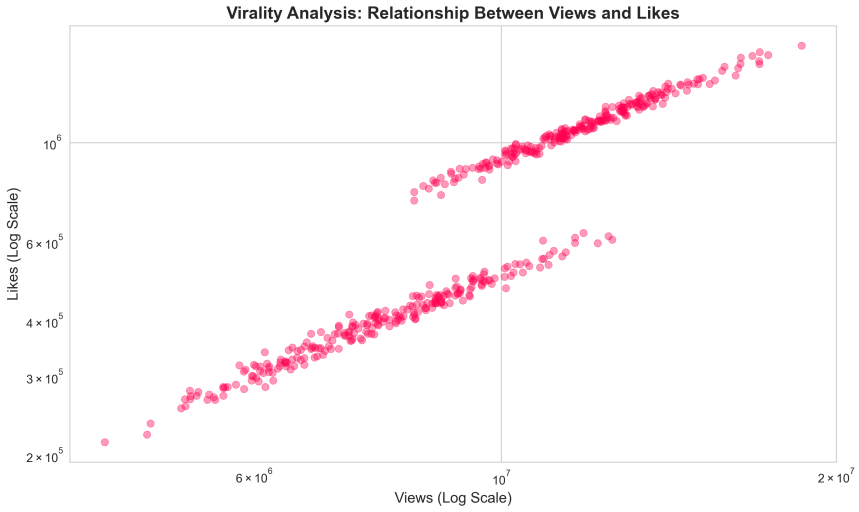

Figure 1: Log–log scatter plot of view count vs like count for aggregated short-form video records.

The plot reveals a clear positive relationship: records with higher view counts almost always have more likes. However, the points do not lie perfectly on a straight line in log–log space, suggesting that engagement rate varies across videos and contexts. Some records achieve relatively high like counts for their view volume, while others underperform.

5.3 Correlation Heatmap

The correlation heatmap in Figure 2 (from `correlation_heatmap.png`) summarizes Pearson correlations between numeric features.

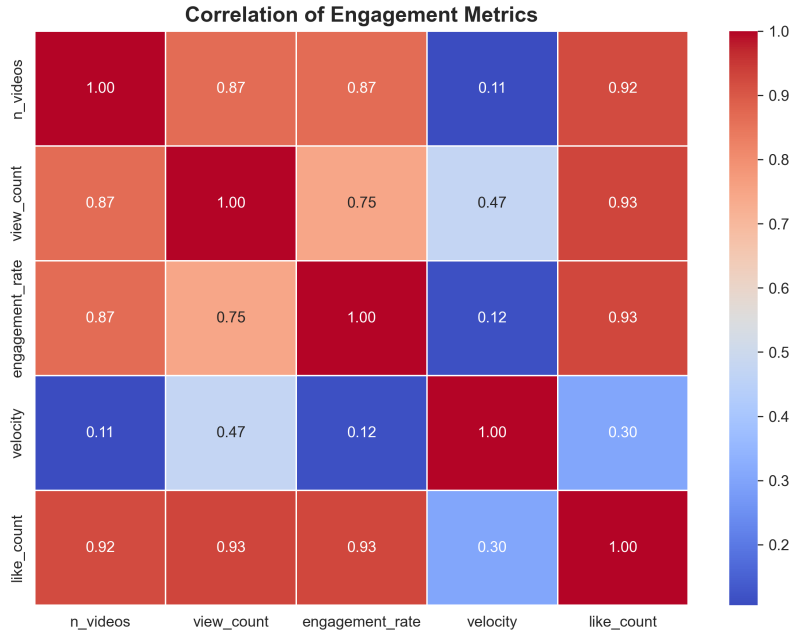

Figure 2: Correlation heatmap for numeric engagement metrics.

As expected, `view_count` and `like_count` are highly positively correlated. The engagement rate metric is moderately correlated with likes but less so with raw views, reflecting the fact that engagement adjusts for scale. Velocity and number of videos may show weaker relationships, depending on how they are defined in the raw dataset.

6 Discussion

Overall, the analysis confirms standard intuitions about viral short-form content:

- Highly viewed videos tend to receive many likes, but the relationship is not perfectly proportional.
- The average engagement rate of around 7% is sizable, but there is likely substantial variation across trends, platforms, and countries.
- Logarithmic scaling is crucial for visualizing heavy-tailed distributions of views and likes.

6.1 Limitations

There are several limitations to note:

- The dataset is aggregated and may not represent individual videos, so interpretations should be made at the trend or group level.
- Engagement rate estimates depend on how `avg_er` is defined in the Kaggle dataset; if it is not exactly likes per view, the derived `like_count` is an approximation.
- The current pipeline focuses only on numeric engagement metrics; it does not analyze the textual content of trend labels or cross-platform differences in depth.

6.2 Future Work

Future extensions of this project could include:

- Segmenting the analysis by platform, country, or category to see whether engagement patterns differ across contexts.
- Fitting simple predictive models to estimate likes given views and other features.
- Incorporating additional metadata or other public datasets (e.g., TikTok Shop revenue data) to connect virality with monetization.

7 Conclusion

This project implements a complete data science workflow around a real-world Kaggle dataset on short-form video trends. By cleaning and normalizing engagement metrics, and by generating targeted visualizations, I quantify how views and likes interact for highly viewed short-form content. The analysis highlights a strong, but not perfectly proportional, relationship between reach and engagement, with an average engagement rate around 7%. The modular code structure and clear README make it straightforward to reproduce and extend these results.