ChatGPT

YouTube Trending Video Analysis (India)

Abstract

This project analyzes the Kaggle *YouTube Trending Video* dataset for India (`INvideos.csv`), demonstrating a beginner-friendly end-to-end data analysis workflow. We **load and clean** the data (remove duplicates/missing values, parse dates, map category IDs to names), then perform **Exploratory Data Analysis (EDA)** to uncover trends: identifying the most frequent video categories and channels, and studying the distributions of views, likes, and comments. We compute a **correlation matrix** (e.g. between views, likes, comments) to quantify their linear relationships. We also optionally apply **sentiment analysis** on video titles (using TextBlob/VADER) to gauge the emotional tone of trending content. Throughout, we use Python tools (Pandas, Matplotlib, Seaborn, TextBlob/VADER) and produce visualizations (bar charts, histograms, scatter plots, heatmaps) to illustrate findings. This write-up provides code examples and explanations at each step, suitable for beginners. A concise final report (1–2 pages) should be prepared covering Introduction, Tools, Steps, and Key Insights/Conclusions.

Introduction

YouTube's *Trending* list surfaces videos currently popular among viewers. This analysis focuses on trending videos in India, using the Kaggle dataset `INvideos.csv`. The dataset contains daily records of trending videos (2017–18) with fields like `video_id`, `title`, `channel_title`, `category_id`, `publish_time`, `tags`, `views`, `likes`, `dislikes`, `comment_count`, and flags for disabled comments/ratings. Our goal is to clean the data and extract insights such as which categories and channels appear most often, how video metrics (views/likes/comments) are distributed and related, and what the sentiment of video titles/tags is. The project combines Pandas data manipulation, basic statistics (e.g. correlation), natural language processing (sentiment), and visualizations using Matplotlib/Seaborn.

Tools Used

- **Python 3** with the following libraries:
- `pandas` and `numpy` for data loading and manipulation.
- `matplotlib` and `seaborn` for plotting charts and graphs.
- `textblob` and/or **NLTK VADER** for simple sentiment analysis on text.
- (Jupyter Notebook environment for interactive coding and report generation.)

Data Cleaning and Preprocessing

First, we load the CSV and inspect it. We then clean the data by dropping duplicates and missing values, and parsing date/time fields. We also map numeric category IDs to human-readable category names using the provided JSON mapping.

```
import pandas as pd
import numpy as np

# Load the data into a Pandas DataFrame
df = pd.read_csv('INvideos.csv')
print(df.shape)          # e.g. (7095, 16)
print(df.columns.tolist())

# Drop duplicate rows (if any) and missing values
df.drop_duplicates(inplace=True)
df.dropna(inplace=True)  # removes rows with any nulls
print("After cleaning: ", df.shape)

# Convert date columns to datetime type
df['publish_time'] = pd.to_datetime(df['publish_time'])
# The 'trending_date' is in format YY.DD.MM (e.g. '17.14.11'), parse accordingly
df['trending_date'] = pd.to_datetime(df['trending_date'], format='%y.%d.%m')

# Map category IDs to names using the JSON mapping
import json
with open('IN_category_id.json', 'r') as f:
    cat_json = json.load(f)
# Build a dictionary: category_id -> category_name
category_map = {}
for item in cat_json['items']:
    cid = int(item['id'])
    cname = item['snippet']['title']
    category_map[cid] = cname
# Add a new column with category names
df['category_name'] = df['category_id'].map(category_map)

# Preview cleaned data
print(df[['video_id','title','channel_title','category_name','views','likes','comment_count']].he
```

All columns now have appropriate types, and we have a `category_name` field (e.g. *Entertainment*, *Music*, etc.).

Exploratory Data Analysis (EDA)

Most Frequent Categories and Channels

We count how many trending records each category and channel has. This shows which genres and creators dominate the trending list. For example:

```
# Top video categories by count of trending entries
cat_counts = df['category_name'].value_counts()
print("Top categories:\n", cat_counts.head(5))

# Top channels by number of trending videos
chan_counts = df['channel_title'].value_counts()
print("Top channels:\n", chan_counts.head(5))
```

From analyses of YouTube India trends, typical top categories include **Entertainment, Music, Comedy, News**, etc. Indeed, other projects confirm that *Entertainment*, *Music*, *Comedy*, and *News* often dominate the trending list [1]. (Creators like *T-Series*, *WWE*, *Cocomelon*, etc. frequently appear, though actual names depend on the time frame.) These counts can be plotted as bar charts to visualize relative frequencies.

Distribution of Views, Likes, Comments

Trending video metrics are typically **skewed**: most videos have modest views, but a few have extremely high counts. It's common to see a *log-normal* distribution. For example, one analysis found the distribution of views (and likes/comments) to be log-normal, with a median trending-video view count of about 1.14 million [2]. We examine this by plotting histograms (often on a log scale):

```
import matplotlib.pyplot as plt

# Histogram of views (log10 scale)
plt.figure(figsize=(6,4))
plt.hist(np.log10(df['views']+1), bins=50, color='skyblue', edgecolor='k')
plt.title('Log10 Distribution of Views (Trending Videos)')
plt.xlabel('Log10(Views)')
plt.ylabel('Count')
plt.show()
```

Figure: Density of log10(views) for trending videos. The roughly bell-shaped curve indicates a log-normal distribution of views [2].

The plot above (log-transformed) is approximately bell-shaped, confirming that most trending videos have medium view counts, while few are viral outliers. Likes and comment counts show similar skew.

Correlation Matrix (Views vs. Likes/Comments)

We compute pairwise correlations between numeric popularity metrics to see how they relate. In Pandas, `.corr()` yields the Pearson correlation matrix:

```
corr = df[['views','likes','comment_count']].corr()
print(corr)
```

Typically, *views* and *likes* are very strongly positively correlated (e.g. r ≈ 0.9+), since more views usually means more likes. *Comments* also correlate positively with views/likes, though slightly lower (since only a subset of viewers comment). We can visualize this with a heatmap and a scatter plot:

```
import seaborn as sns

# Heatmap of correlation matrix
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation between Views, Likes, Comments')
plt.show()

# Scatterplot of views vs likes (log scale for clarity)
plt.figure(figsize=(5,4))
sns.scatterplot(x='views', y='likes', data=df)
plt.xscale('log')
plt.yscale('log')
plt.title('Views vs. Likes (log-log scale)')
plt.xlabel('Views')
plt.ylabel('Likes')
plt.show()
```

A heatmap shows the Pearson coefficients (values between –1 and 1). High positive values (near +1) indicate strong linear relationships [3]. The scatter plot (on log-log axes) likewise reveals a tight upward trend: more views tend to accompany more likes.

Sentiment Analysis on Titles and Tags *(Optional)*

We can analyze the *sentiment polarity* of video titles (and tags) to see if trending content skews positive or negative. **TextBlob** and **VADER** are simple NLP tools for this. TextBlob returns a polarity score in [-1, 1] (negative to positive) and subjectivity in [0,1] [4]. VADER provides a normalized **compound** score in [-1, +1], plus probabilities for positive/neutral/negative text [5]; commonly, compound ≥ 0.05 indicates positive sentiment, ≤ -0.05 negative [6].

Example using TextBlob on titles:

```
from textblob import TextBlob
# Compute polarity of each title
df['title_polarity'] = df['title'].apply(lambda t:
TextBlob(str(t)).sentiment.polarity)
print(df[['title','title_polarity']].head(3))
```

Or using VADER:

```
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()

df['title_compound'] = df['title'].apply(lambda t: sid.polarity_scores(str(t))
['compound'])
print(df[['title','title_compound']].head(3))
```

These polarity scores can be averaged or binned to label titles as positive/neutral/negative. We could similarly join all tags into strings and score them, although tags often lack sentiment-laden phrases. Overall, such sentiment analysis is exploratory: for example, one study found *Comedy* videos had the most positive tone, while *News* or *Politics* tended negative [1].

Visualizations

We generate clear charts to illustrate our findings:

- **Bar charts** for categorical counts (e.g., top 5 categories or channels).

```
top5_categories = df['category_name'].value_counts().head(5)
top5_categories.plot(kind='bar', color='orchid', title='Top 5 Video
Categories')
plt.ylabel('Number of Trending Videos')
plt.show()
```

- **Histograms** and **density plots** for distributions (as above for views; similarly for likes/comments).
- **Scatter plots** (with log scales) to show relationships (views vs. likes, etc).
- **Heatmaps** for correlation matrices.

For example, a bar chart of top categories might show *Entertainment*, *Music*, etc., each with dozens or hundreds of trending videos. A heatmap of correlation (views, likes, comments) confirms their strong interrelatedness. All charts include axis labels, titles, and legends as needed to be interpretable by beginners.

Key Insights / Conclusion

From our analysis of the Indian trending videos:

- **Popular categories**: Entertainment, Music, Comedy and News are the most frequent genres [1]. (Video platforms report similar trends.) These categories capture typical viewer interests in India.

- **Channel popularity**: A small number of channels (often big media or celebrity channels) contribute many trending videos.
- **Skewed distributions**: The metrics (views, likes, comments) are highly skewed. On a log scale, the distribution of views is roughly normal [2], reflecting that most trending videos have moderate popularity while a few go viral (millions of views).
- **Strong correlations**: Views, likes and comment counts correlate strongly (Pearson $r \gg 0.8$). More views usually means proportionally more likes and comments.
- **Sentiment**: If performed, sentiment analysis often shows that high-engagement genres like comedy or education have more positive-tone titles (fewer dislikes), whereas news/politics titles trend more negative [1], though this depends on current events.
- **Other factors**: A more detailed study (beyond EDA) might consider how publish date affects trending speed, or how tags relate to success. Those require additional steps (time series analysis, NLP on tags, etc.).

Final deliverables: All analysis code should be in a well-commented Jupyter notebook (for reproducibility), with explanations of each step for beginners. A short report (1–2 pages PDF) should be prepared summarizing the **Introduction, Tools**, the **Data Cleaning and EDA steps**, and **Key Insights/Conclusions**. Charts generated by Matplotlib/Seaborn should be included in the report to illustrate findings. These materials can then be organized into a GitHub repository as specified.

Summary: This end-to-end project (loading/cleaning data, EDA, sentiment analysis, visualization) equips a beginner with practical skills in data analysis and storytelling. It demonstrates common tasks (data wrangling, summary statistics, plotting, simple NLP) and guides the learner to draw meaningful conclusions from the YouTube trending dataset.

References

- YouTube Trending Video Dataset documentation [2].
- TextBlob sentiment analysis (polarity range) [4].
- VADER sentiment analysis details and thresholds [7] [6].
- Prior analysis of YouTube India trends for categories and sentiment [1].

---

[1] GitHub - mdash/youtube-popular-videos: Exploration of popular youtube videos in India
https://github.com/mdash/youtube-popular-videos

[2] Data Analysis on The Trending Youtube Videos
https://nycdatascience.com/blog/student-works/data-analysis-on-the-trending-youtube-videos/

[3] Correlation: What is it? How to calculate it? .corr() in pandas
https://data36.com/correlation-definition-calculation-corr-pandas/

[4] Tutorial: Quickstart — TextBlob 0.19.0 documentation
https://textblob.readthedocs.io/en/dev/quickstart.html

[5] [6] [7] Sentiment Analysis using VADER. Understanding Sentiment Analysis in... | by Boula Akladyous | Medium
https://akladyous.medium.com/sentiment-analysis-using-vader-c56bcffe6f24