YouTube Video Popularity Prediction and Engagement Analysis

Description of the Project

The research aims to create machine learning models which will forecast YouTube video popularity through view count and analyze viewer interactions from both web-scraped and API-derived data sources. The research compared two models which used public YouTube search results data from web scraping and YouTube Data API retrieved structured data. The research aims to determine which video characteristics including duration and upload date and engagement metrics and content keywords affect audience numbers the most.

How to Use

1. Ensure python and git are downloaded onto your local machine
2. Clone the repository onto your local machine
3. Connect to the repository through where you locally installed it
4. Setup a python virtual environment if not set up already
5. Install dependencies using `pip install -r requirements.txt`.
6. Create a `.env` file in the project root with your own YouTube API key:

   ```
   YOUTUBE_API_KEY=your_api_key_here
   ```

7. Run the entire pipeline using:

   ```
   python src/run_all.py
   ```

   This executes data scraping, API collection, preprocessing, feature engineering, model training, and visualization automatically. **Note:** The web scraping process in Step 1 will take approximately 12 minutes to complete due to the large keyword set and YouTube page load times.

8. All processed data and output visualizations will be saved in the `data/` directory.

Training

Both datasets (scraped and API) were trained using two regression algorithms:

- Random Forest Regressor
- XGBoost Regressor

The models learned to forecast video view numbers through log transformed data by using engineered features which included engagement rate and duration and days since upload and keyword metrics. The training process used 80% of the data for model development while the remaining 20% served for evaluation purposes.

Inferencing

Once trained, the models can predict expected video popularity for any new input dataset containing the same attributes. The trained models evaluate unseen test data to measure predictive accuracy using R^2 and RMSE (Root Mean Squared Error) metrics.

Data Collection

Data was collected in two primary ways:

1. **Web Scraping:** Using BeautifulSoup and the `scrape_youtube.py` script, 3,000 YouTube video records were gathered across 225 diverse search queries. Attributes extracted include video title, channel, view count, and duration.
2. **API Data Collection:** Using the YouTube Data API (`api_youtube.py`), structured metadata was collected from 15 regions (e.g., US, IN, GB, JP, BR) at approximately 300 videos per region, totaling to about 3000 records.

Used Tools

- Programming Language: Python
- Libraries: pandas, numpy, scikit-learn, xgboost, matplotlib, BeautifulSoup4, requests, dotenv
- APIs: YouTube Data API v3
- Environment: Jupyter/Colab, local Python runtime
- Version Control: GitHub

Collected Attributes

Attribute	Description
title	Video title
channel	Channel name
views	Total view count
likes	Total like count
comments	Comment count
duration	Video length in minutes
upload_date	Date the video was uploaded
tags	Tags and keywords assigned
category_id	Content category

Attribute	Description
description	Video description text

Number of Data Samples

- Scraped dataset: ~3,000 records
- API dataset: ~3,000 records
- Combined total after preprocessing: ~6,000 videos

API Usage

The YouTube Data API retrieved data through HTTPS requests which used region-based queries to fetch trending video information. The API returned JSON data which included metadata stored under three sections named snippet and statistics and contentDetails. The data retrieved from YouTube API was saved to youtube_api_raw.csv before undergoing cleaning and standardization processes.

Sample Data After Preprocessing

title	views	likes	comments	duration_mins	engagement_rate
Taylor Swift – Fortnight	76,924,840	1,020,385	45,321	3.98	0.0138
Madison Beer – Reckless	1,204,058	68,904	3,180	4.48	0.0598

Data Preprocessing

The preprocessing stage involved cleaning and converting view strings like `"76,924,840 views"` into numeric format, normalizing duration from both `"PT5M33S"` (API format) and `"4:29"` (scraped format) into minutes, handling missing titles, tags, and dates with default values, and removing duplicates.

Data Cleaning

The data cleaning included value replacement for irregularities and log transformation of numerical features like views and likes and comments because they showed strong skewness. The model received protection through view count outlier capping at the 99th percentile to prevent model distortion.

Feature Engineering

New derived features were added to strengthen model interpretability:

- Text-based: `title_length`, `word_count_title`, `desc_keyword_count`, `has_music_keyword`

- Engagement-based: `likes_to_views`, `comments_to_views`, `likes_to_comments`, `engagement_rate`
- Temporal: `days_since_upload`, `upload_year`, `upload_month`, `upload_weekday`
- Log-scaled metrics: `log_views`, `log_likes`, `log_comments`

How Data Is Processed and Prepared

After loading from the `data/` directory:

1. Numeric and text columns are converted and standardized.
2. New engineered features are added using the `feature_engineering.py` module.
3. Data is split into training and testing sets (80/20).
4. Features are scaled using `StandardScaler` for consistency before model training.

Model Development and Evaluation

Train/Test Data Partition

- Training set: 80% of data
- Testing set: 20% of data
- Validation metric: R^2 and RMSE

Model 1 – Based on Scraped Data (Updated with 3,000 Samples)

Algorithm: RandomForestRegressor, XGBRegressor
Input: 20 numeric and derived features
Training samples: ~2,400
Testing samples: ~600

Model	RMSE	R^2
Random Forest (Tuned)	123,542	0.9971
XGBoost (Tuned)	1,898,024	0.946

Interpretation:
The Random Forest model achieved near-perfect accuracy (R^2 = .9971), indicating it captured nearly all variance within the training data. While this may suggest slight overfitting, the model accurately represents the expanded dataset's structure. XGBoost performed slightly lower (R^2 = 0.946), demonstrating robust generalization. The expanded dataset from 370 to nearly 2,900 samples significantly improved reliability and overall performance.

Model 2 – Based on API Data (Final Results)

Algorithm: RandomForestRegressor, XGBRegressor
Input: Structured API fields including duration, engagement, and metadata

Training samples: ~2,398
Testing samples: ~600

Model	RMSE	R^2
Random Forest (API)	387,171.71	0.800
XGBoost (API)	360,382.97	0.827

Interpretation:
The two models show excellent predictive capabilities with XGBoost achieving better results than Random Forest for generalization tasks. The R^2 values exceeding 0.8 show that the models effectively explain most of the view count variance. The API data shows higher RMSE values than the scraped dataset because it contains more natural variations and diverse regional data.

Feature Importance

Feature importance was computed using XGBoost's gain-based importance scores. Top predictors included:

1. `log_likes`
2. `engagement_rate`
3. `days_since_upload`
4. `duration_mins`
5. `title_length`

These indicate that user interactions (likes/comments) and upload recency play a stronger role in determining popularity than video length.

Visualization

Generated graphs:

1. Model Comparison: Bar chart comparing R^2 for Random Forest and XGBoost
2. Feature Importance: Bar chart showing top 10 most influential attributes
3. Views vs Duration: Scatter plot showing view count distribution across video lengths

Visuals are saved in `/data` as:

- `model_comparison.png`
- `feature_importance.png`
- `views_vs_duration.png`

Discussion and Conclusions

Findings

- Engagement-based metrics (likes, comments) are the strongest predictors of popularity.
- XGBoost consistently outperformed Random Forest in generalization, but Random Forest achieved near-perfect fitting on the expanded dataset.
- Upload recency and title keyword density moderately affect performance.
- Longer videos do not necessarily gain more views; engagement and recency dominate.

Challenges

- Dynamic loading and JavaScript rendering limited web scraping results.
- Cleaning heterogeneous duration formats (`PT5M33S` vs `4:29`) required multiple regex layers.
- Outliers in viral videos skewed regression results and required log-scaling.

Ethical and Legal Considerations

All scraped and API data was publicly available metadata. No personal or private user information was collected. Data was used strictly for academic research and complies with YouTube's Terms of Service.

Recommendations

- Combine scraped and API data for larger, more balanced datasets.
- Experiment with deep learning models (e.g., LightGBM, Neural Networks).
- Add NLP-based text analysis on titles and tags to better capture content themes.

Deliverables

- **Code:** All source files in `src/` directory including preprocessing, feature engineering, modeling, and visualization scripts.
- **Data:** Cleaned and feature-engineered datasets in `data/` directory.
- **Report:** This document describing the entire workflow and findings.