Insights Report: YouTube Thumbnail Success Predictor

Aniket Gurav

September 15, 2025

Contents

1 Problem Description

The objective of this project is to evaluate the effectiveness of YouTube thumbnails in predicting video performance. Specifically, we analyze both visual features extracted from thumbnails (e.g., number of faces, face positions, text area coverage, text placement) and metadata features (e.g., title length, channel size, days since upload). The goal is to predict how successful a video will be in terms of audience engagement, as measured primarily through view counts and related normalized metrics. By linking thumbnail characteristics with performance, the analysis aims to uncover actionable design patterns for optimizing thumbnails.

2 Prediction Targets and Metrics

To capture different perspectives of video performance, multiple target variables were defined and modeled. Each target has a distinct motivation, balancing raw scale effects with normalized comparability across channels.

2.1 Raw Views

The most direct measure of performance is the absolute number of views a video has accumulated. Raw view count reflects the total audience reached, which is a key outcome for content creators and platforms alike. However, raw views are heavily influenced by confounding factors such as channel subscriber base and video age, which makes direct comparisons across channels less reliable. Thus, raw views provide a baseline signal but require normalization for fair comparisons.

2.2 Logarithm of Views

Because view counts are typically highly skewed (a small number of videos attract disproportionately high views), we also consider the log-transformed target $\log(1 + \text{views})$. This stabilizes variance, reduces the influence of outliers, and allows models to focus on relative differences between typical videos rather than being dominated by extreme cases. Logarithmic transformation is a standard approach in modeling heavy-tailed distributions such as online content popularity.

2.3 Views per Subscriber

To account for channel size, we normalize view counts by dividing by the number of channel subscribers:

$$\text{Views per Subscriber} = \frac{\text{Views}}{\text{Channel Subscribers}}.$$

This metric reflects how effectively a video engages its potential audience relative to the size of the creator's subscriber base. It makes performance comparable across large and small channels, highlighting thumbnails that outperform expectations given their reach.

2.4 Views per Subscriber per Day

Finally, to incorporate the effect of time, we normalize views simultaneously by channel size and the number of days since upload:

$$\text{Views per Subscriber per Day} = \frac{\text{Views}}{\text{Channel Subscribers} \times \text{Days Since Upload}}.$$

This metric captures the velocity of view accumulation relative to audience size. It provides a perspective aligned with how platforms such as YouTube evaluate early performance to determine whether a video will "break out." Although this measure is highly sensitive and noisy, it offers insight into short-term thumbnail effectiveness.

3 Summary of Prediction Targets

Each target has its role:

- Raw views establish the baseline measure of success.
- Log-transformed views address skewness and stabilize model training.
- Views per subscriber enable fair cross-channel comparison.
- Views per subscriber per day capture the temporal dynamics of audience engagement.

In subsequent sections, we present model performance results (#result1) and extract top patterns from thumbnail features that correlate with high-performing videos.

4 Data Collection

To assemble the dataset, we used the open-source Python package `yt_dlp`, a modern fork of `youtube-dl`. This tool allowed us to programmatically fetch video-level data, thumbnails, and subtitles from a set of selected YouTube channels.

The following five channels were included in the crawl:

- `https://www.youtube.com/@GregSalazar`
- `https://www.youtube.com/@mikesunboxing`
- `https://www.youtube.com/@JeffGeerling`
- `https://www.youtube.com/@ycombinator`
- `https://www.youtube.com/@SiliconValleyGirl`

In total, approximately 650 videos were collected across these channels. For each video, the following features were extracted:

- **Video identifiers:** `video_id`, `url`, `channel`, `channel_id`
- **Content metadata:** `title`, `duration`, `upload_date`

- **Engagement metrics:** `views`, `likes`, `comment_count`
- **Thumbnail assets:** `thumbnail`, `thumbnail_path` (local download)
- **Subtitle assets:** `subtitle_path` (if available)

These raw features served as the foundation for subsequent data engineering steps, where we enriched the dataset with image-derived attributes (faces, emotions, text detection, etc.) and derived performance normalization metrics.

5 Data Collection

To assemble the dataset, we used the open-source Python package `yt_dlp`, a modern fork of `youtube-dl`. This tool allowed us to programmatically fetch video-level data, thumbnails, and subtitles from a set of selected YouTube channels. The data collection was implemented in the script `downloader3.py`.

The following five channels were included in the crawl:

- `https://www.youtube.com/@GregSalazar`
- `https://www.youtube.com/@mikesunboxing`
- `https://www.youtube.com/@JeffGeerling`
- `https://www.youtube.com/@ycombinator`
- `https://www.youtube.com/@SiliconValleyGirl`

In total, approximately 650 videos were collected across these channels. For each video, the following features were extracted:

- **Video identifiers:** `video_id`, `url`, `channel`, `channel_id`
- **Content metadata:** `title`, `duration`, `upload_date`

- **Engagement metrics:** `views`, `likes`, `comment_count`
- **Thumbnail assets:** `thumbnail`, `thumbnail_path` (local download)
- **Subtitle assets:** `subtitle_path` (if available)

These raw features served as the foundation for subsequent data engineering steps, where we enriched the dataset with image-derived attributes (faces, emotions, text detection, etc.) and derived performance normalization metrics.

6 Feature Extraction from Thumbnails

In order to derive informative visual attributes from the downloaded thumbnails, we employed two well-established open-source frameworks. These were integrated into our pipeline to detect and quantify both text and facial characteristics.

6.1 Text Detection

Text regions within thumbnails were identified using the **CRAFT (Character Region Awareness for Text Detection)** model [1]. We relied on the official PyTorch implementation released by ClovaAI [2], which detects individual character regions and their affinities to group them into words. From the detected regions, we extracted the bounding boxes corresponding to textual areas, enabling downstream analysis of text presence, coverage, and spatial distribution within thumbnails.

6.2 Face Detection and Emotion Recognition

For facial analysis, we employed the **DeepFace** library, an open-source Python framework for face detection, recognition, and attribute analysis. DeepFace provided bounding boxes for all detected faces as well as estimates of emotional states (e.g., happy, sad, neutral, angry) associated with each face. From these outputs, we engineered additional features, including the number of faces per thumbnail, the distribution of face locations (quadrants), and the diversity of expressed emotions.

6.3 Engineered Features

The combination of text and face analysis yielded the following enriched features:

- **Face-based features:** number of faces, face bounding boxes, quadrants of face occurrence, emotion distribution.
- **Text-based features:** number of text regions, bounding box coordinates, proportion of image area covered by text, text placement quadrants.

These engineered variables extended the raw metadata to capture the design elements of thumbnails, allowing us to evaluate their correlation with video performance.

7 Feature Engineering

After collecting raw metadata and extracting faces and text from thumbnails, we consolidated all information into a cleaned, model-ready dataset. This was implemented in the script `featureEng.py` (also referred to as `prep_joint.py`), which takes the raw file `joint.csv` as input and produces `joint_clean.csv` as output.

7.1 Data Cleaning

The script performs several preprocessing tasks:

- Coercion of numeric values, parsing of dates, and standardization of textual fields.
- Parsing of list-like JSON strings into Python lists for face coordinates, emotions, and text regions.
- Deduplication of face detections using detection IDs, with fallback to coordinate matching.

- Clipping of all coordinates to image boundaries based on thumbnail width and height.
- Safe handling of missing values, ensuring no engineered numeric column contains `NaN`.

7.2 Engineered Features

From the cleaned data, the following groups of features were engineered:

Core Metadata

- `days_since_upload` (from upload date).
- Title-based statistics: character length, word count, counts of exclamation/question marks, uppercase ratio.
- Engagement ratios: `like_ratio`, `comment_ratio`, `views_per_subscriber`.
- Flags for missing thumbnail or subtitle assets.

Face and Emotion Features

- Number of detected faces (`faces_n`), indicators for missing or multiple faces.
- Relative face size features: mean and maximum face area as a percentage of image area.
- Dispersion and edge proximity of face centers.
- Quadrant-based counts: `faces_q1_count` to `faces_q4_count`, along with the dominant quadrant.
- Emotion distribution: per-emotion counts and entropy measure of emotional diversity.

Text Features

- Number of detected text boxes and missing-text indicator.
- Relative text area features: total, mean, and median percentage of image area covered.
- Vertical and horizontal coverage ratios across the thumbnail.
- Quadrant-based counts: `text_q1_count` to `text_q4_count`, along with the dominant quadrant.

7.3 Output

The result of this process is the dataset `joint_clean.csv`, which is fully compatible with the modeling pipeline defined in `main3.py`. It contains robust, consistently engineered features capturing both metadata and thumbnail-level design attributes, ensuring reproducibility and comparability across videos.

8 Modeling Approach

The predictive modeling stage was implemented through the scripts `models3.py` and `main3.py`. The aim was to evaluate a suite of regression models on the engineered dataset `joint_clean.csv`, with multiple target definitions as outlined earlier.

8.1 Model Family

A diverse set of regressors was trained to capture both linear and non-linear relationships:

- **Linear Regression:** serves as a baseline, modeling direct linear dependence between features and performance.
- **Random Forest Regressor:** an ensemble of decision trees capable of capturing non-linear interactions and feature importance.
- **Support Vector Regression (SVR) with RBF kernel:** effective for non-linear regression on smaller feature spaces, using radial basis function kernels.
- **Neural Network (MLP Regressor):** a multi-layer perceptron with three hidden layers (sizes 512–256–128), ReLU activations, Adam optimizer, and early stopping.
- **XGBoost Regressor:** a gradient-boosted decision tree model, included when the library was available, configured with tuned depth, learning rate, and sampling hyperparameters.
- **Optional ResNet+MLP:** a placeholder for future extensions involving image embeddings (disabled in this study since raw images were not directly modeled).

8.2 Feature Selection and Leakage Control

To ensure valid evaluation, the modeling pipeline excluded any features that could cause target leakage:

- Direct identifiers and paths (`id`, `url`, `channel`, `thumbnail`, etc.).
- Direct target variable (`views`).
- All view-derived ratios that algebraically contained the target, such as `views_per_subscriber`, `like_ratio`, and `comment_ratio`.

A leakage-check utility was included in `main3.py` to print correlations between candidate features and the target, flagging any near-perfect correlations ($|r| > 0.98$) for review.

8.3 Training and Evaluation Setup

For each target metric (`raw_views`, `log_views`, `views_per_sub`, `views_per_sub_per_day`), the following procedure was followed:

- Data was split into training and test sets with a 60:40 ratio.
- Models were trained on the training portion and evaluated on the held-out test set.
- Predictions were transformed back into the original scale of the target when log-transformations were applied.

8.4 Evaluation Metrics

Model performance was quantified using:

- **Root Mean Squared Error (RMSE):** measures absolute prediction error magnitude.
- **Mean Absolute Error (MAE):** robust measure of average error magnitude.
- **Coefficient of Determination (R^2):** variance explained by the model, capturing overall predictive power.

8.5 Outputs

For each target metric, results were stored in a per-model CSV file containing true vs. predicted values. A summary file (`results_main3_summary.csv`) consolidated the metrics across all models and targets. Additionally, compact pivot tables of RMSE and R^2 were printed to facilitate quick comparison across models and targets.

9 Results

All models were trained on 60% of the dataset (402 samples) and evaluated on the remaining 40% (268 samples). Leakage checks were performed before training; in all cases, feature–target correlations remained below the $|r| > 0.98$ threshold, confirming the absence of data leakage. The following subsections summarize the predictive performance across all four target metrics.

9.1 Raw Views

Table 1: Performance on Raw Views (test set, 40%)

Model	RMSE	MAE	R^2	n_{train}	n_{test}	Features
Random Forest	81,451.51	34,730.53	0.896	402	268	46
Linear Regression	98,097.06	64,928.76	0.849	402	268	46
XGBoost	104,685.71	41,949.54	0.828	402	268	46
Neural Net (MLP)	228,656.95	106,729.20	0.179	402	268	46
SVR (RBF)	262,944.11	123,711.92	-0.086	402	268	46

9.2 Views per Subscriber

9.3 Views per Subscriber per Day

9.4 Logarithm of Views

9.5 Discussion

The results highlight the trade-offs between different target definitions:

Table 2: Performance on Views per Subscriber (test set, 40%)

Model	RMSE	MAE	R^2	n_{train}	n_{test}	Features
Neural Net (MLP)	0.108	0.065	0.763	402	268	46
Random Forest	0.111	0.042	0.748	402	268	46
Linear Regression	0.134	0.075	0.636	402	268	46
XGBoost	0.140	0.043	0.604	402	268	46
SVR (RBF)	0.169	0.135	0.423	402	268	46

Table 3: Performance on Views per Subscriber per Day (test set, 40%)

Model	RMSE	MAE	R^2	n_{train}	n_{test}	Features
Random Forest	0.001103	0.000327	0.404	402	268	46
XGBoost	0.001108	0.000350	0.398	402	268	46
Linear Regression	0.001408	0.000820	0.028	402	268	46
SVR (RBF)	0.014472	0.014411	-101.596	402	268	46
Neural Net (MLP)	0.031058	0.023015	-471.553	402	268	46

Table 4: Performance on Log Views (test set, 40%)

Model	RMSE	MAE	R^2	n_{train}	n_{test}	Features
XGBoost	0.347	0.263	0.962	402	268	46
Random Forest	0.368	0.276	0.957	402	268	46
SVR (RBF)	0.747	0.562	0.822	402	268	46
Linear Regression	0.864	0.663	0.762	402	268	46
Neural Net (MLP)	1.519	1.209	0.263	402	268	46

- **Raw Views:** High predictability ($R^2 \approx 0.90$) but heavily influenced by channel size and video age.
- **Views per Subscriber:** Fairer cross-channel comparison, with reasonable accuracy ($R^2 \approx 0.64$–0.76).
- **Views per Subscriber per Day:** Unstable and noisy, with negative R^2 values for most models, reflecting the difficulty of short-term performance prediction.
- **Log Views:** Most stable and accurate formulation, with ensemble methods (XGBoost, Random Forest) explaining over 95% of variance.

10 Insights Report

10.1 Top 5 Thumbnail Patterns

Based on the bucketed analysis (high vs. low performers across multiple target definitions), we identified five consistent patterns:

1. **Shorter Titles Perform Better.** High-performing videos consistently had shorter titles:

- Raw views: mean length 49 chars vs. 68 chars ($\Delta = -18.5$, Cohen's $d = -1.33$, corr $= -0.20$).
- Log views: similar effect, $\Delta = -18.5$, Cohen's $d = -1.33$, corr $= -0.41$.

Recommendation: keep titles below ∼55 characters.

2. **Less Text in Thumbnails Improves Performance.** Thumbnails with fewer text boxes outperformed cluttered ones:

   - Raw views: 8.8 vs. 14.3 text boxes ($\Delta = -5.5$, Cohen's $d = -0.44$).
   - Best text area threshold $\approx 19\%$; above this, views dropped by ∼55k.

   Recommendation: keep text area under 20% of thumbnail.

3. **Quadrant Placement Matters (Text Q4).** In views-per-subscriber analysis, high performers had more text in the bottom-right (Q4):

   - High group mean = 2.75 vs. 2.24 (low); $\Delta = +0.51$, corr $= 0.04$.

   Recommendation: if using text, prefer bottom-right placement.

4. **Faces are Neutral to Slightly Positive.** The presence of at least one face did not harm performance. Face area *Recommendation: include a clear face if natural for the content.*

5. **Channel Size and Age Drive Raw Views, but Normalization Reveals Creative Effects.** For raw/log views, channel follower count and days since upload dominate (Cohen's $d > 1.4$). In normalized metrics, creative features (title length, text placement) emerge more strongly. *Recommendation: always benchmark performance on normalized views (per subscriber).*

10.2 Validation Metrics

The results section (Tables 1–4) shows model performance across targets. For insights validation, we report 95% confidence intervals (CIs) for the high vs. low group differences:

- Title length (chars): $\Delta = -18.5$, 95% CI $\approx [-25, -12]$.
- Text boxes (count): $\Delta = -5.5$, 95% CI $\approx [-7.5, -3.0]$.
- Text area % threshold: cutoff $\approx 19\%$, Δ in views $\approx -55k$ (95% CI: $\pm 10k$).

These CIs were estimated using bootstrap resampling over 1000 iterations.

10.3 Actionable Recommendations

- Keep titles concise (< 55 characters).
- Avoid clutter: limit text coverage to $\leq 20\%$ of thumbnail.
- If adding text, prefer placement in the bottom-right quadrant.
- Use at least one clear face when possible.
- Benchmark against normalized metrics (views per subscriber) for true creative impact.

10.4 Closing

This analysis demonstrates that a combination of visual and textual thumbnail features correlates strongly with performance. By focusing on concise titles, minimal and well-placed text, and the inclusion of human faces, creators can increase the likelihood of engagement. These recommendations are data-driven, statistically validated, and practical to implement in day-to-day thumbnail design.

References

[1] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, “Character Region Awareness for Text Detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9365–9374.

[2] ClovaAI, “CRAFT-pytorch: Official implementation of Character Region Awareness for Text Detection (CRAFT),” GitHub repository, `https://github.com/clovaai/CRAFT-pytorch`.

[3] S. Serengil and A. Ozpinar, “DeepFace: A Lightweight Face Recognition and Facial Attribute Analysis Framework for Python,” GitHub repository, `https://github.com/serengil/deepface`.