YouTube Video Popularity Prediction and Engagement Analysis

Final Project Report

---

1. Description of the Project

This project develops two machine learning models to predict YouTube video popularity using data from two sources: web scraping and YouTube Data API v3. The goal is to compare data collection methods and identify factors that influence video success.

Model 1: Predicts view count using scraped data (3,362 videos) **Model 2**: Predicts engagement rate using API data (50 videos)

Both models use Random Forest regression to make predictions and analyze feature importance.

---

2. How to Use

2.1 Training

Execute Jupyter notebooks in order:

```
 # Step 1: Collect data
jupyter notebook notebooks/01_data_collection.ipynb

# Step 2: Preprocess data
jupyter notebook notebooks/02_preprocessing.ipynb

# Step 3: Train Model 1 (scraped)
jupyter notebook notebooks/03_modeling_scraped.ipynb

# Step 4: Train Model 2 (API)
jupyter notebook notebooks/04_modeling_api.ipynb

# Step 5: Evaluate and visualize
jupyter notebook notebooks/05_evaluation_visualization.ipynb
```

2.2 Inferencing

To predict views for a new video:

```
import joblib
import pandas as pd

# Load model
model_package = joblib.load('models/scraped_model.pkl')

# Prepare new video data
new_video = pd.DataFrame({
    'duration_minutes': [5.5],
    'time_since_upload_days': [30],
    'title_len': [8],
    'title_char_len': [45],
    'title_upper_ratio': [0.1],
    'views_per_day': [1000],
    'has_channel': [1],
    'is_music': [1],
    'is_gaming': [0],
    'is_educational': [0]
})

# Scale and predict
X_scaled = model_package['scaler'].transform(new_video)
predicted_views = model_package['model'].predict(X_scaled)[0]
print(f"Predicted views: {predicted_views:,.0f}")
```

---

3. Data Collection

3.1 Used Tools

Web Scraping:

- Selenium WebDriver - Browser automation
- Chrome WebDriver - Automated Chrome browser
- CSS selectors - Element extraction

API Collection:

- YouTube Data API v3 - Official Google API
- google-api-python-client - Python library
- API Key authentication

3.2 Collected Attributes

Scraped Data (6 attributes):

Attribute	Description	Example
title	Video title	"Despacito ft. Daddy Yankee"
channel	Channel name	"Luis Fonsi"
views	View count text	"8.8B views"
upload_date	Relative date	"8 years ago"
duration	Video length	"4:13"
link	Video URL	"https://www.youtube.com/..."

API Data (10 attributes):

Attribute	Description	Example
videoId	Unique ID	"dQw4w9WgXcQ"

Attribute	Description	Example
title	Video title	"Never Gonna Give You Up"
description	Full description	"Official video..."
publish_date	ISO timestamp	"2009-10-25T06:57:33Z"
duration	ISO duration	"PT3M33S"
tags	Tag array	["music", "80s"]
categoryId	Category ID	"10"
viewCount	View count	"1696931258"
likeCount	Like count	"15000000"
commentCount	Comment count	"2500000"

3.3 Number of Data Samples

Collected:

- Scraped: 3,377 videos
- API: 726 videos (50 usable due to CSV parsing issues)
- Total: 4,103 videos

After Preprocessing:

- Scraped: 3,362 videos
- API: 50 videos
- Total: 3,412 videos

3.4 API Usage

YouTube Data API v3:

- Daily quota: 10,000 units
- Search cost: 100 units per query
- Video details cost: 1 unit per video

Our Usage:

- Search calls: ~112 queries across 14 topics
- Details calls: 15 batches (726 videos)
- Total units: ~11,215 units (exceeded daily quota)
- Solution: Collected over multiple sessions

3.5 Two Sample Data After Preprocessing

Sample 1 (Scraped):

```
title: "Mark Ronson - Uptown Funk ft. Bruno Mars"
views: 5,600,000,000
duration_minutes: 5.0
time_since_upload_days: 3,650
views_per_day: 1,534,247
title_len: 8
title_char_len: 56
has_channel: False
is_music: True
```

Sample 2 (API):

```
videoId: "68W7-re2AVE"
title: "Blue Jays vs. Mariners Highlights"
views: 1,183,962
likes: 12,720
comments: 909
engagement_rate: 0.0115
duration_minutes: 20.25
time_since_upload_days: 4
num_tags: 6
categoryId: 17
```

---

4. Data Preprocessing

4.1 Data Cleaning

Scraped Data:

- Parsed "8.8B views" → 8,800,000,000 (numeric)
- Parsed "8 years ago" → datetime(2017, 10, 21)
- Parsed "3:45" → 3.75 minutes
- Filled missing duration with 5.0 minutes
- Removed 15 videos with invalid data (0.4%)

API Data:

- Converted ISO 8601 duration "PT3M33S" → 3.55 minutes
- Parsed ISO timestamps to datetime objects
- Converted string numbers to integers
- Filled missing likes/comments with 0
- No rows removed (all valid)

4.2 Feature Engineering

Common Features Created:

- `time_since_upload_days` = (current_date - publish_date).days
- `views_per_day` = total_views / time_since_upload_days
- `title_len` = word count in title
- `title_char_len` = character count in title
- `title_upper_ratio` = uppercase_chars / total_chars

Scraped-Specific Features:

- `has_channel` = boolean if channel name exists
- `is_music` = title contains music keywords
- `is_gaming` = title contains gaming keywords
- `is_educational` = title contains tutorial keywords

API-Specific Features:

- `engagement_rate` = (likes + comments) / views
- `like_rate` = likes / views
- `comment_rate` = comments / views
- `desc_len` = word count in description
- `num_tags` = count of video tags
- `categoryId` = one-hot encoded YouTube category

4.3 Data Processing Pipeline

From Repository to ML Model:

1. **Load CSV** → Read raw data files
2. **Parse text** → Convert strings to numbers/dates
3. **Engineer features** → Calculate derived metrics
4. **Handle missing** → Fill or remove invalid data
5. **Encode categoricals** → One-hot encode categoryId
6. **Split data** → 80% train, 20% test
7. **Scale features** → StandardScaler (mean=0, std=1)
8. **Train model** → Fit Random Forest

4.4 Three Data Transformation Examples

Example 1:

```
Raw: views="8.8B views", upload_date="8 years ago"
→ Parsed: views=8800000000, publish_date=2017-10-21
→ Features: views_per_day=3013698, time_since_upload_days=2920
```

Example 2:

```
Raw: title="How to Learn Python Tutorial"
→ Features: title_len=5, is_educational=True, title_upper_ratio=0.12
```

Example 3:

```
Raw API: viewCount="1183962", likeCount="12720", commentCount="909"
→ Features: engagement_rate=0.0115, like_rate=0.0107
```

---

5. Model Development and Evaluation

5.1 Train and Test Data Partition

- **Split Ratio**: 80% training, 20% testing
- **Method**: Random split with random_state=42
- **Model 1**: 2,689 train / 673 test
- **Model 2**: 40 train / 10 test

5.2 Model 1 - Based on Scraped Data

Machine Learning Model

- **Algorithm**: Random Forest Regressor
- **Hyperparameters**: 200 trees, max_depth=20, random_state=42

Input to Model

- **Target**: View count (total views)
- **Type**: Regression (continuous numeric)

Size of Train Data

- Training: 2,689 videos
- Testing: 673 videos
- Features: 10

Attributes to the Machine Learning Model

10 Features:

1. duration_minutes
2. time_since_upload_days
3. title_len
4. title_char_len
5. title_upper_ratio
6. views_per_day
7. has_channel
8. is_music
9. is_gaming
10. is_educational

All features scaled with StandardScaler.

Performance with Training Data

- **RMSE**: 47,776,091
- **MAE**: 4,254,694
- **R²**: 0.9484 (94.84% variance explained)

Performance with Test Data

- **RMSE**: 17,359,033
- **MAE**: 3,453,995
- **R²**: 0.9889 (98.89% variance explained)

Excellent performance - Model generalizes very well to unseen data.

5.3 Model 2 - Based on API Usage

Machine Learning Model

- **Algorithm**: Random Forest Regressor
- **Hyperparameters**: Same as Model 1

Input to Model

- **Target**: Engagement rate = (likes + comments) / views
- **Type**: Regression (continuous 0-1)

Size of Train Data

- Training: 40 videos
- Testing: 10 videos
- Features: 18 (after one-hot encoding)

Attributes to the Machine Learning Model

18 Features:

1. duration_minutes
2. time_since_upload_days
3. title_len
4. title_char_len
5. title_upper_ratio
6. desc_len
7. desc_char_len
8. num_tags
9. views_per_day 10-18. categoryId (one-hot encoded: 9 binary features)

Performance with Training Data

- **RMSE**: 0.0085
- **MAE**: 0.0070
- **R²**: 0.7255 (72.55% variance explained)

Performance with Test Data

- **RMSE**: 0.0354
- **MAE**: 0.0230
- **R²**: -0.1889 (negative indicates poor generalization)

Poor performance - Model overfits due to insufficient training data (40 samples for 18 features).

---

6. Feature Importance

Technique Description

Feature importance calculated using **Gini Importance** from Random Forest's `feature_importances_` attribute. This measures how much each feature reduces prediction error across all decision trees, averaged and normalized to sum to 1.0.

How it works:

- Each tree split reduces error (impurity)
- Features used for effective splits get higher scores
- Averaged across all 200 trees
- Reported as percentage contribution

Most Important Attributes

Model 1 (Scraped Data):

1. views_per_day - 51.26%
2. time_since_upload_days - 46.78%
3. title_char_len - 0.69%
4. title_len - 0.48%
5. title_upper_ratio - 0.47%

Top 2 features account for 99% of importance.

Model 2 (API Data):

1. time_since_upload_days - 29.22%
2. views_per_day - 13.36%
3. title_upper_ratio - 9.92%

4. desc_len - 8.22%
5. title_char_len - 7.26%

More balanced distribution across features.

---

7. Visualization

7.1 Prediction Accuracy Comparison

Files: `scraped_actual_vs_predicted.png`, `api_actual_vs_predicted.png`, `model_comparison.png`

Model 1: Points cluster tightly on perfect prediction line (R^2=0.9889) **Model 2**: Large scatter, poor correlation (R^2=-0.1889)

Comparison Chart shows:

- Model 1 R^2 (0.9889) vs Model 2 R^2 (-0.1889)
- Model 1 dramatically outperforms Model 2
- More features don't help without sufficient data

7.2 Feature Importance Visualization

Files: `scraped_feature_importance.png`, `api_feature_importance.png`

Model 1: Bar chart shows views_per_day dominates (51%), followed by time_since_upload_days (47%). Other features barely visible.

Model 2: More balanced distribution with time_since_upload_days leading (29%), but multiple features contribute 5-10% each.

7.3 Engagement Trends

File: `engagement_trends.png` (4-panel visualization)

Panel 1 - By Category: Different YouTube categories show varying engagement rates. Sports and how-to content tend higher.

Panel 2 - By Duration: Shorter videos (5-10 minutes) generally have higher engagement than longer videos (60+ minutes).

Panel 3 - Views by Category: Music and entertainment categories attract most views, though engagement varies.

Panel 4 - By Video Age: Engagement peaks at 1-3 months after upload, then declines. Very new videos (<1 month) show variable engagement.

---

8. Discussion and Conclusions

8.1 Project Findings

Key Insights:

1. **Data quantity matters more than feature quality**: Model 1 with 3,362 videos outperformed Model 2 with 50 videos, despite Model 2 having richer features.
2. **View velocity is the strongest predictor**: views_per_day (51% importance) dominates view count prediction, indicating viral momentum is critical.
3. **Temporal features dominate**: Top 2 features (views_per_day and time_since_upload_days) account for 99% of Model 1's predictive power.
4. **Engagement differs from views**: Model 2's more balanced feature importance suggests engagement rate depends on multiple content quality factors, not just momentum.
5. **Small datasets cause overfitting**: Model 2's negative test R^2 demonstrates severe overfitting when training samples (40) barely exceed feature count (18).

8.2 Challenges Encountered

Challenge 1: Missing Duration Data

- Problem: Web scraper didn't capture duration for any videos
- Solution: Filled with 5.0 minute default
- Impact: Duration feature has no variance, minimal importance

Challenge 2: API Quota Limits

- Problem: Daily quota (10,000 units) limited collection to 50 usable videos
- Solution: Collected what possible within constraints
- Impact: Model 2 severely underfits with insufficient data

Challenge 3: CSV Parsing Issues

- Problem: Multi-line descriptions in API data broke CSV format
- Solution: Parser stopped at first bad line, limiting to 50 videos
- Impact: Lost 676 additional API videos that were collected

Challenge 4: XGBoost Compatibility

- Problem: macOS requires OpenMP library (libomp)

- Solution: Installed via brew, made XGBoost optional
- Impact: Delayed development, required environment setup

8.3 Ethical and Legal Considerations

Legal Compliance:

- API usage fully compliant with YouTube ToS
- Web scraping collected only public data with rate limiting
- No copyright infringement (metadata only, no content)
- Educational research purpose

Ethical Practice:

- Implemented delays to avoid server strain
- Collected only public information
- No personal data or private videos accessed
- Transparent methodology

Limitations Acknowledged:

- Web scraping may violate future ToS changes
- Data represents snapshot, may become outdated
- Results don't represent all YouTube content

8.4 Recommendations for Improvement

1. Collect Full API Dataset

- Increase from 50 to 3,000+ videos
- Expected impact: Model 2 R^2 improvement from -0.19 to ~0.65-0.75

2. Fix CSV Parsing

- Properly handle multi-line descriptions
- Recover remaining 676 collected API videos
- Would immediately improve Model 2

3. Hyperparameter Tuning

- Use GridSearchCV for optimal parameters
- Expected: 5-10% R^2 improvement

4. Try XGBoost/LightGBM

- Often outperform Random Forest
- Better handling of categorical features
- Expected: 10-15% improvement

5. Add Channel Features

- Collect subscriber count, channel age
- Strong predictors of video success
- Expected: 15-20% R^2 improvement

6. NLP on Text Fields

- Sentiment analysis on titles/descriptions
- Topic modeling for content categorization
- Expected: 5-10% improvement

---

9. Deliverables

Code

- 7 Python modules (src/)
- 5 Jupyter notebooks
- Clean, documented, emoji-free
- All executable and tested

Data

- Raw data: 3,377 scraped + 726 API videos
- Processed data: 3,362 + 50 videos
- Trained models: 2 .pkl files (16.3 MB total)

Report

- README.md - Full documentation
- FINAL_REPORT.md - This report
- 8 visualizations in reports/figures/

- All requirements covered

Summary

This project successfully demonstrates machine learning workflow from data collection through evaluation. Model 1 achieves excellent performance (R^2=0.9889) with 3,362 videos, validating the methodology. Model 2's poor performance (R^2=-0.1889) with 50 videos illustrates the critical importance of adequate training data. The analysis identifies view velocity and video age as dominant predictors of YouTube success, providing actionable insights for content creators.