Predicting YouTube Cooking Video Popularity

Team members: Corbin Cahalan
Alexander Levin-Koopman
Jeffrey Olson

Introduction

From bloggers to influencers, the creator economy is estimated to be worth more than $104 billion and has grown to 50 million entertainers (Flynn 2022). With 2 billion monthly logged-in users and more than a billion hours of videos watched everyday (YouTube 2022), YouTube is a powerhouse for the creator economy and is considered the second most popular website in the world (Similarweb 2022). Some YouTube creators, also known as YouTubers, are able to make millions from advertisement revenue share, brand partnerships, and product placements in their videos (Osborn 2022). Of course, not every video topic generates strong revenue for creators, but cooking is considered a niche on YouTube with one of the highest cost per thousand impressions (Tasty Edits). How do cooking YouTubers gain their viewership, and how do they know whether their videos will be popular? We've developed a model that creators can use to predict whether their cooking video will be popular before they upload it.

Data Extraction

To collect data for our model, we used two sources: YouTube's API and YouTube-DL. To view the code for data extraction and model development, view our Github repository. To view or download our final datasets, check out our Google Drive.

YouTube API

YouTube's API allowed us to extract videos by channel ID and by search query. In our first method, we collected top videos based on a list of cooking keywords. Once we collected the top videos in YouTube search results per keyword, we were able to expand our video list to include more videos from each YouTube channel. In our second method, we gathered a list of ~50 online articles that detail popular cooking YouTube channels. These articles typically based popularity on the channel's total number of views. After extracting the channel IDs and channel usernames from each site, we used YouTube's API to extract all videos from those channels. In total, we were able to collect more than 1.1 million videos through these methods.

YouTube-DL

The youtube-dl extraction tool allowed us to collect more video details, such as subtitles and thumbnail height features, without rate limit issues that are typical of the YouTube API. We also used this tool to extract the highest quality thumbnail possible per video.

Data Considerations

Our team decided that it would make sense to remove YouTube shorts because they are inherently different from the standard video on YouTube. To remove these videos, we filtered out any video that was less than or equal to 60 seconds in length and videos that contained "#shorts" in the title. We also

assured that there were no duplicate videos in our dataset by dropping duplicates on the video ID column. Lastly, we limited our dataset so that all videos have subtitles and a thumbnail.

Overall, it took our team roughly 8-9 weeks to develop and complete data extraction. Although we were able to extract more than 1 million videos from the YouTube API, our final dataset totaled closer to ~500K videos due to time constraints. We ran our YouTube-DL extraction for 7 full days, but had to make a hard cut so that we could move on with the rest of our project. From here, we split our dataset into 80% for training the neural networks and 20% for training the final model. From there, each dataset was split again for training and testing. See Appendix B to learn more about the datasets, columns and features we used, and descriptive statistics.

Defining Popularity

Popularity is a word with many possible definitions, and many consider a video to be popular based on the total number of views that the video accumulates. Some prior research has been made to predict a YouTube video's view count and popularity (Srinivasan 2017 and Li 2019), yet most prior research was able to implement dislike counts into a popularity calculation before YouTube removed the metric from public access.

While many channels have some "viral" videos that garner views far in excess of the channel median (and possibly a healthy payday), we did not want to create a model that detected virality. Instead, we aimed to find which videos generated sincere appreciation among viewers, hopefully leading to ongoing viewership. This would be challenging considering how little nuance is offered by the API's data. This led us to use *likes* as the main proxy for popularity. In brief, our metric reflects how much the video's likes are above or below what is expected (in terms of best linear fit to view and comment counts), and measured in standard deviations. See Appendix A, Description A1 for more details.

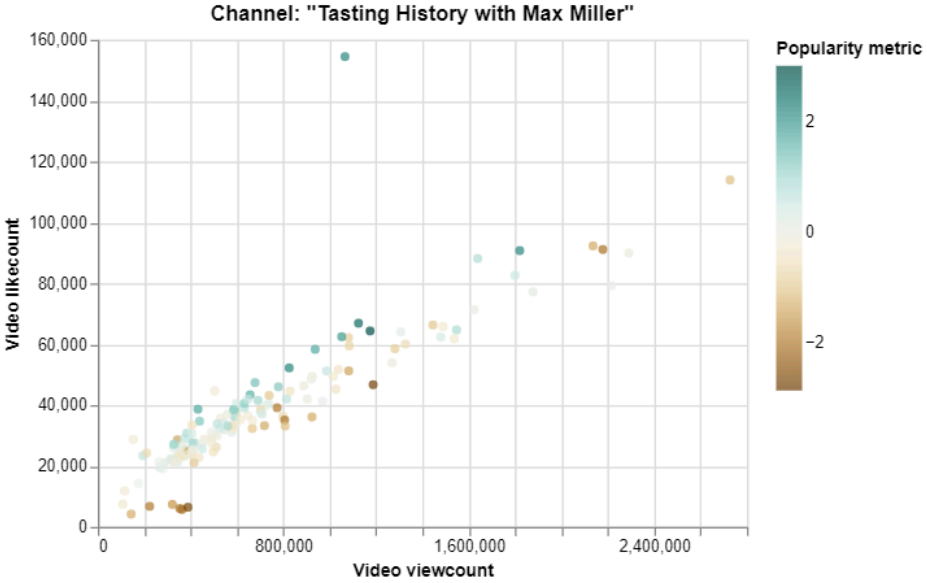


Figure 1: The original popularity metric for one channel.

Although the heuristic case for this metric is strong, our models struggled to converge when training to predict it. For this reason we transitioned to a different metric based on principle component analysis of the log number of views, likes, and comments. As shown in Figure 2, the PCA metric correlates well to

the (log of the) number of likes; in contrast, any relationship between the original popularity metric and number of likes is much harder to see.

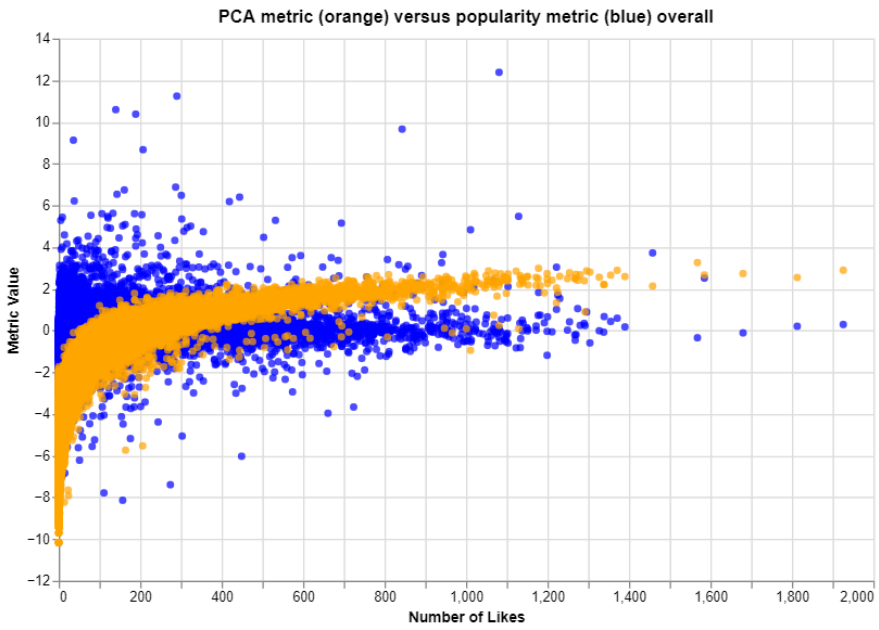


Figure 2: Comparison of original popularity metric and the one based on PCA.

Model

Our final model combines a convolutional neural network (CNN) with an ensemble model to make a prediction on a video's popularity. Whereas other research has been previously made to suggest a thumbnail using CNNs (Arthurs 2017 and Çakar 2021), we use video thumbnails as input for our CNN to predict the log number of video views. The final model uses the output from the CNN and other video attributes to predict the popularity score.

Our original model combined a recurrent neural network (RNN) with a CNN to make predictions, but our RNN failed to converge. Although our RNN didn't work as intended, we were still able to extract meaningful features from the textual data for our final model. See Appendix A, Description A2 for more details on the RNN.

CNN Details

The CNN was used as a column transformer to create a usable feature from the thumbnail for each video. We concluded that the impact of the thumbnail on a video's popularity is most likely related to the video's view counts rather than likes or comments. This proved to be true in the training process. Similar to the RNN, the CNN was not able to find a signal in the original popularity metric we developed, nor was it able to find a signal in the updated PCA metric that was used by the final model. This is why it focused solely on the log transform of the video's view counts.

It is important to note that many of the thumbnail images, especially from prominent cooking channels, have logos in them. We believe that this can influence the CNN, biasing it towards popular cooking channels or celebrity chefs. For more information on the CNN and its performance, see Appendix A, Description A2.

Final Model Details

The goal of this final model was to use the feature created by the CNN and combine it with the rest of the data that was then used to predict the popularity of a given video. The final model is an ensemble of three tree based methods: lightgbm, random forest, and adaboost. Ensembling balances the biases of each model by combining their outputs creating more accurate and stable predictions. This was done using a method called stacking. First, each model produces predictions. Then, a final model, in this case elasticnet, used those models' predictions to make a final prediction. In development, there were other models that were tried; however, it seems that the signal from the data was highly non-linear. All linear models fared the same 0.8 RMSE worse than the tree based methods, and even those had to have fairly deep trees to find the signal.

During the modeling process, we used a random forest model to extract the impurity based feature importance, which is also known as gini importance (Scikit-Learn 0.23.2 Documentation). This indicates which features are important in the model for prediction. At the top, we can see that the log of the video view counts predicted by the CNN are considered the most important. See Appendix B, Table B1 for column descriptions.

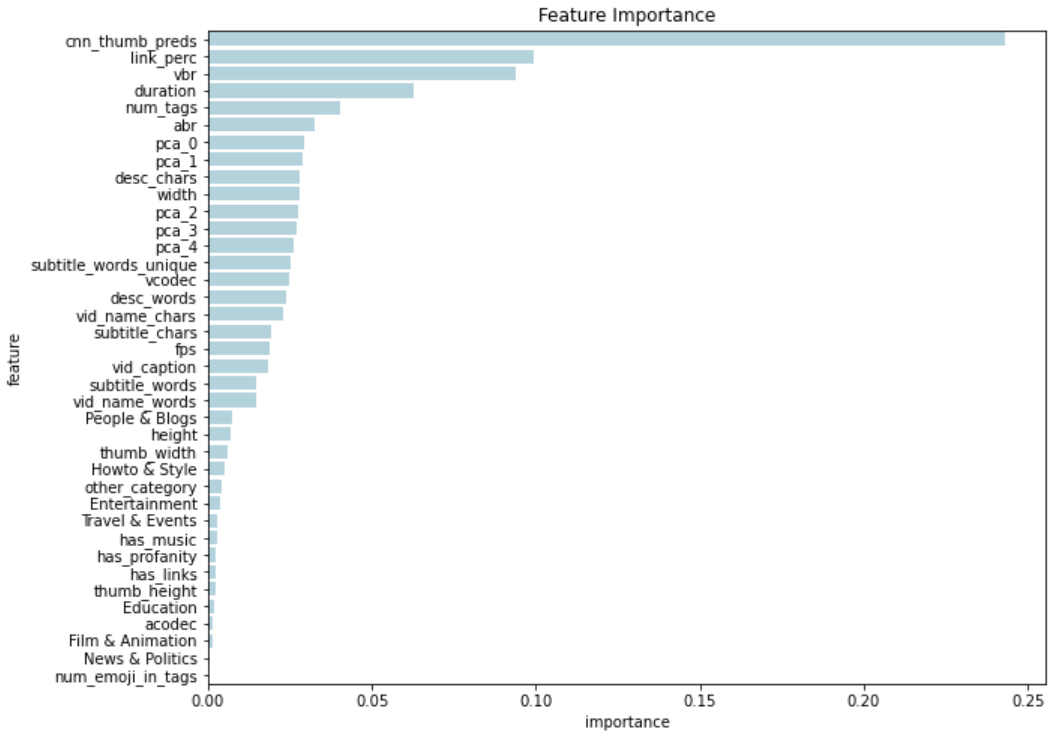


Figure 3: Features' relevance to model performance.

Findings

The combination of models used proved to be fairly accurate at predicting the popularity of a video as we defined it. The CNN added a significant boost in the final results with the thumbnail feature transform, and although the RNN failed to converge we were still able to extract meaningful features from the subtitle and description fields.

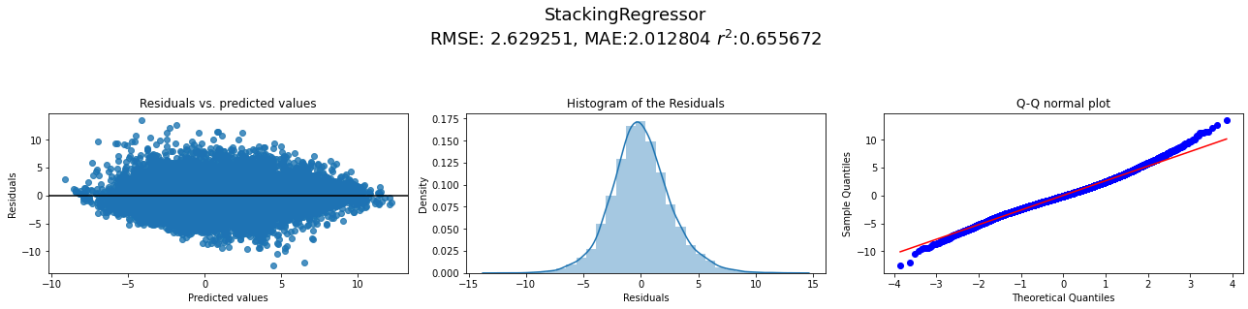


Figure 4: Model performance on holdout data.

Above is the final test result from the tuned and stacked models on holdout data. We can see that the model overcame the few significant outliers from the CNN.

One question we wanted to answer is whether the addition of the CNN feature warrants the extra data and compute resources required in training. In this case, it did add a definitive boost to the final model. To show this, we used a basic linear regression model to predict popularity score with the CNN feature and without it. The addition of this feature provided a reduction of .29 in the RMSE score. With the CNN feature, RMSE=3.462154, whereas RMSE=3.757671 without the CNN feature.

Failure Analysis

RNN fails to converge

In our original plan, a RNN was intended to analyze textual data from the video subtitles and description to predict popularity, but our RNN failed to converge. We attempted various basic modifications to the neural network, yet still did not observe any decrease in the loss function. To double check our process, we used a non-recurrent neural network with TF-IDF embedding and dense layers (which had functioned well for one of us in a NLP task for Milestone II), but still saw no convergence. Most likely the relationship between the subtitles and the popularity metric was too subtle to support a successful RNN without much greater complexity and vastly more data.

The project may have benefited from the use of pre-trained linguistic models. It was probably optimistic to suppose that a fresh RNN could learn enough about the various linguistic modes that these videos exhibit to perform effectively in predicting a concept like popularity.

Examples of failure

For the CNN, there were two large outliers. The CNN predicted that the outliers were very popular with millions of views, but one had a view count of 34 and the other had 4,615. Their thumbnails featured a title on the left and a woman's face on the right. In fact, both of the videos were not about cooking at all and likely came from the search-based data extraction method we used. To combat this, more advanced filtering should occur to ensure that the videos being analyzed are all about the same topic.

The final stacking regression model seemed to take care of the outliers very well and there weren’t any individual failures that pulled the model in one direction or another. Even the failures of the CNN didn’t affect the result of the ensemble model.

Concluding Thoughts, Limitations, & Considerations

One challenge of our project is defining popularity. There are many ways to define popularity, but we were limited to the metrics we can readily extract. More research and testing is likely needed to develop a better version of our metric. Another limitation is that our research only focuses on cooking videos, so more research would be required for other topics to see if our models are generalizable.

One ethical consideration is our web scraping and use of YouTube-DL. While web scraping publicly available data is legal, there are regulations that further research should be aware of when scraping websites. The information we scraped is publicly available and does not include personal data or confidential information, but YouTubers are the copyright owners of their thumbnails. There are also claims that YouTube-DL can be a breach of YouTube's terms of service, depending on how one uses the tool. It is worth noting that one should be cautious if using the tool to download copyrighted video material.

Although our model was effective at predicting popularity, there are many ways to improve upon our efforts. Whether it be developing an improved popularity metric or creating an RNN that converges, more research is necessary to create a model that works not only on cooking videos, but also on other video types. YouTube is considered the Internet's largest hub of videos, and it would be great to make a generalizable model that works for several topics. Lastly, as YouTube changes its video business strategy towards short-form content such as Shorts, it would be useful to develop a model that works for both long-from and short-form videos.

Statement of Work

Alex:

- YouTube-DL extraction
- CNN model
- Great Lakes
- Final ensemble model
- Final dataset construction

Corbin:

- Website scraping method
- Data extraction and cleaning
- Project manager

Jeff

- API extraction method
- Popularity metric definition
- Feature extraction
- RNN Model

Appendix A: Additional Descriptions

Description A1: Popularity Metric.
To find the *pop_metric* of a video *v* from channel *C*:

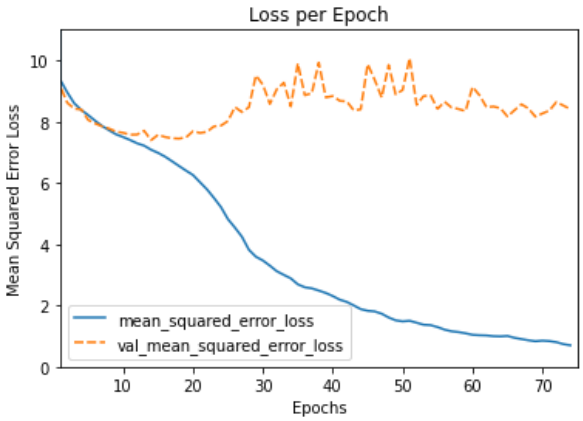
- Calculate *LR*, the best linear fit in *C* for number of likes, given number of views and number of comments
- For a video *v*, if *d(v)* is the difference likes(*v*) - LR(*v*), define *D* as the set of all *d(v)* for *v* in *C*.
- Finally, *pop_metric(v)* is the z-score of *d(v)* in the set *D*.

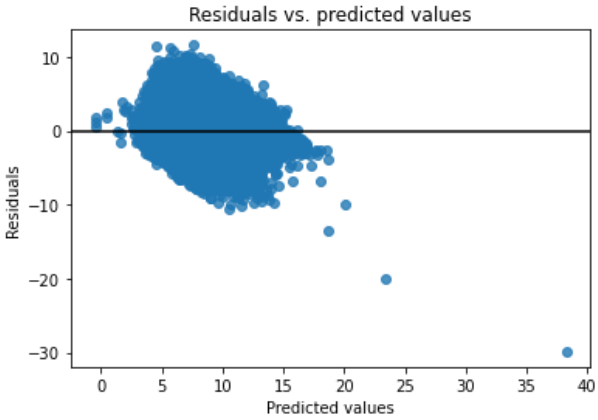
Description A2: RNN Details.
The RNN module trained using the textual data contained in video subtitles. Subtitles are typically a transcription of the video dialog and/or voiceover, and we believed they would reflect the video content well. The subtitles were tokenized, and tokens were embedded into vectors and then used as inputs. The neural network had an initial layer of LSTM cells, followed by a dense layer leading into a final output. Since we are seeking to regress a continuous value (the popularity metric), the final output passed through a linear activation function. We used the Tensorflow library for most aspects of the RNN.

Description A3: CNN Details.
The architecture of the CNN is similar to VGGNet, but we use a linear function to predict the popularity score instead of a softmax function typically used in classification. The VGGNet has twenty layers in total including twelve convolutional layers, five max pooling layers and three dense layers, the last of which is the output layer. This VGNet architecture is preferred here as it has very good performance without being prohibitive in its coding and training time. We could have used Alexnet which is quite a bit smaller with 8 total layers but with its larger kernel and stride size the performance would be significantly reduced.

The Model was trained using the GreatLakes Compute Cluster on three GPUs and took close to 36 hours to complete 75 epochs. Below is the loss per epoch for both thain and val mean squared error loss functions. The CNN was loaded from the checkpoint at epoch 19. And we can see that it did well except for a few large outliers in the residual plot. This can be seen in the cnn_on_test_data.ipynb notebook in our Github. On the test data the scores were:

- Root mean squared error: 2.740465
- Mean absolute error: 2.173752,
- r2 score: 0.243857

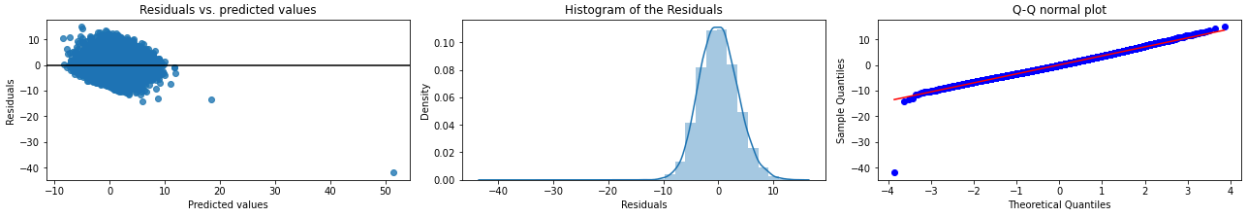




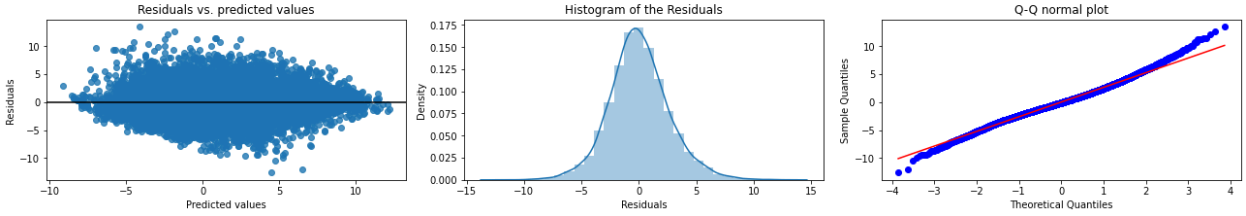
Description A4: Final Model

The ensemble method for the final model seemed to give the best results. This is compared with a basic linear regression model. Seen below was thrown off by a few large outliers.

LinearRegression
RMSE: 3.517129, MAE:2.786998 r^2:0.383852



StackingRegressor
RMSE: 2.629251, MAE:2.012804 r^2:0.655672



As part of the final model process other models were considered for the stacking regressor however even when tuned they didn't fare any better than the basic LinearRegression model. Each of the tree based models were first tuned using the hyperopt library then fed into the stacking regressor. All of the tree methods had the best performance with deep trees. The lightgbm had a max depth of 17, the random forest model had a max depth of 27 and the adaboost had a max depth of 19. The one that is most surprising is the adaboost model. In the default configuration the decision tree base for the adaboost model has a max depth of three. All of this including the parameter search space can be seen in Final_model.ipynb.

Appendix B: Additional Dataset Information

Table B1: Column Directory

Column	Source	Description
chan_query	YouTube API	The query used to extract a given video; none for popular videos via web scraping method
chan_id	YouTube API	YouTube's unique ID per channel
chan_name	YouTube API	Channel name from channel creator
chan_viewcount	YouTube API	Channel's total view count at the moment of extraction
chan_subcount	YouTube API	Channel's total subscriber count at the moment of extraction
chan_start_dt	YouTube API	Channel's creation date
chan_thumb	YouTube API	Channel's thumbnail extracted by YouTube API
chan_vidcount	YouTube API	Channel's total video count at the moment of extraction
vid_id	YouTube API	YouTube's unique ID per video
vid_name	YouTube API	Video Title
vid_publish_dt	YouTube API	Video's publish date
vid_thumb	YouTube API	Video's thumbnail at moment of extraction; note that a YouTuber can change thumbnails several times to A/B test which drives more views
vid_duration	YouTube API	Video length
vid_caption	YouTube API	Boolean whether the video includes captions
vid_viewcount	YouTube API	Video's total view count at moment of extraction
vid_likecount	YouTube API	Video's total like count at moment of extraction
vid_commentcount	YouTube API	Video's total comment count at moment of extraction
vid_seconds	YouTube API	Video's length in seconds based on vid_duration column
description	YouTube DL	Video's description
duration	YouTube DL	Video's length
age_limit	YouTube DL	Whether the video has age restrictions
categories	YouTube DL	Category label for the video
tags	YouTube DL	Keywords that the YouTuber adds to their video
is_live	YouTube DL	If the video was live at the time it went up on youtube
width	YouTube DL	The resolution width of the video in pixels
height	YouTube DL	The resolution height of the video in pixels
fps	YouTube DL	Frames per second
vcodec	YouTube DL	Video codec used
vbr	YouTube DL	Video bitrate used
acodec	YouTube DL	Audio codec used
abr	YouTube DL	Audio bitrate used
thumb_name	YouTube DL	Thumbnail file name
subtitles	YouTube DL	Auto-generated subtitles per video
thumb_width	YouTube DL	Width of the thumbnail image
thumb_height	YouTube DL	Height of the thumbnail image

vid_name_chars	Built Feature	number of characters in the video title
vid_name_words	Built Feature	number of words in the video title
desc_chars	Built Feature	number of characters in the video description
desc_words	Built Feature	number of words in the video description
subtitle_chars	Built Feature	number of characters in the video subtitles
subtitle_words	Built Feature	number of words in the video subtitles
subtitle_words_unique	Built Feature	number of distinct words appearing in the video subtitles
has_profanity	Built Feature	whether the video subtitles contain (redacted) profanity
has_music	Built Feature	whether the video indicates musical accompaniment
has_links	Built Feature	whether the video description has URLs (boolean)
link_perc	Built Feature	how much of the description text is devoted to URLs
num_tags	Built Feature	The number of characters in the video tags
num_emoji_in_tags	Built Feature	The number of emoji in the video tags
cnn_thumb_preds	Built Feature	Output of CNN model

Table B2: Dataset Stats per Model.

	RNN Dataset	**CNN Dataset**	**Ensemble Dataset**
Row Count	293,475	428,023	71,233
Video Views Mean	339,346	299,253	326,419
Video Views SD	3,061,501	2,767,015	2,562,924
Video Views IQR	[541 - 75,300]	[451 - 59,432]	[509 - 66,995]
Video Likes Mean	5,902	4,953	5,547
Video Likes SD	37,437	36,030	31,959
Video Likes IQR	[19 - 189]	[16 - 1,328]	[18 - 1689]
Video Comments Mean	373	311	355
Video Comments SD	2,519	4,284	2,533
Video Comments IQR	[4 - 145]	[3 -105]	[4 - 134]

Table B3: Final Dataset Non-Boolean Feature Stats.

	Mean	**SD**	**IQR**
duration	636.62	1020.04	[251.0 - 733.0]
width	1975.63	737.79	[1920.0 - 1920.0]
height	1143.54	425.97	[1080.0 - 1080.0]
fps	31.53	10.43	[25.0 - 30.0]
vcodec	22.71	2.78	[23.0 - 24.0]
vbr	3895.79	3920.13	[1893.46 - 3964.87]
acodec	1.36	0.48	[1.0 - 2.0]
abr	132.69	12.91	[129.48 - 133.26]
thumb_width	1123.67	352.71	[1280.0 - 1280.0]

thumb_height	633.49	195.49	[720.0 - 720.0]
pca_metric	1.07	4.45	[-2.35 - 4.36]
cnn_thumb_preds	8.83	1.93	[7.55 - 9.94]
vid_name_chars	55.93	22.65	[39.0 - 72.0]
vid_name_words	9.61	4.13	[6.0 - 12.0]
desc_chars	1224.27	1051.51	[441.0 - 1702.0]
desc_words	165.56	151.28	[56.0 - 225.0]
subtitle_chars	5548.49	10060.03	[65.0 - 7136.0]
subtitle_words	1070.08	1939.15	[10.0 - 1380.0]
subtitle_words_unique	271.49	303.26	[7.0 - 406.0]
link_perc	0.2	0.18	[0.04 - 0.32]
pca_0	0.04	0.25	[-0.15 - 0.24]
pca_1	0.12	0.36	[-0.13 - 0.22]
pca_2	0.04	0.21	[-0.13 - 0.18]
pca_3	-0.01	0.16	[-0.13 - 0.09]
pca_4	-0.01	0.14	[-0.13 - 0.08]
num_tags	17.06	12.31	[8.0 - 24.0]
num_emoji_in_tags	0.01	0.13	[0.0 - 0.0]

Figure B1: Visualizations for some column statistics.

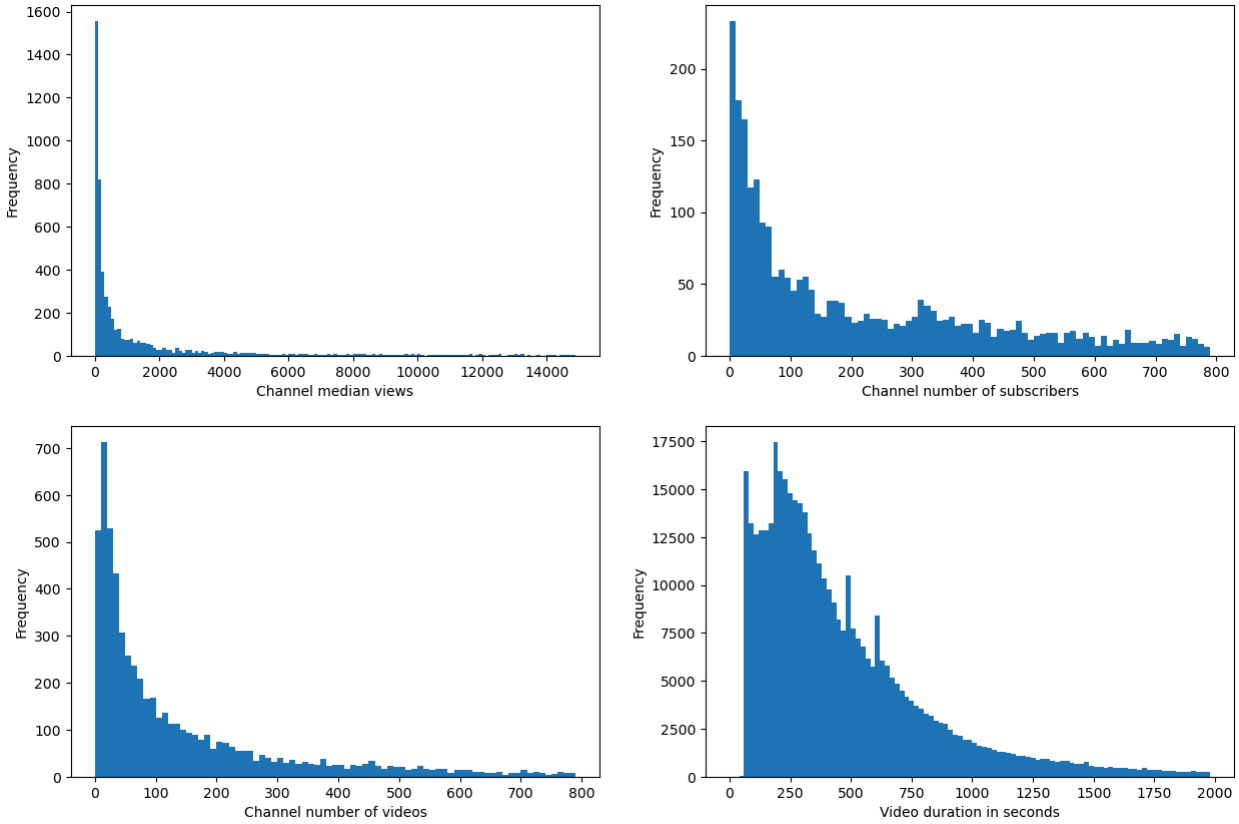


Table B4: Final Dataset Boolean Feature Stats.

	Count	Share
vid_caption	9090	15%
Education	2634	4%
Entertainment	8459	14%
Film & Animation	1073	2%
Howto & Style	30416	50%
News & Politics	804	1%
People & Blogs	13829	23%
Travel & Events	1560	3%
other_category	1822	3%

References

1. Arthurs, N., Birnbaum, S., & Gruver, N., (2017). *Selecting Youtube Video Thumbnails via Convolutional Neural Networks*. Stanford. http://cs231n.stanford.edu/reports/2017/pdfs/710.pdf.
2. Çakar, M., Yildiz, K. & Demir, Ö. (2021, December). *Thumbnail Selection with Convolutional Neural Network Based on Emotion Detection*. International Journal of Advances in Engineering and Pure Sciences. DOI:10.7240/jeps.900561.
3. Flynn, J. (2022, August 18). *20+ INCREDIBLE CREATOR ECONOMY STATISTICS [2022]*. Zippia. https://www.zippia.com/advice/creator-economy-statistics/.
4. Li, Y., Eng, K., & Zhang, L. (2019). *YouTube Videos Prediction: Will this video be popular?* Department of Civil and Environmental Engineering, Stanford University. http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/26647615.pdf.
5. Osborn, J. (2022, February 7). *10 Highest-Paid YouTubers for 2022*. Manofmany. https://manofmany.com/entertainment/highest-paid-youtuber-2022.
6. Similarweb. (2022, October). *Top Websites Ranking*. https://www.similarweb.com/top-websites/.
7. Srinivasan, A., Wang, A., Yee, K., & O'Farrell, R. (2017, December). *Youtube Views Predictor*. https://towardsdatascience.com/youtube-views-predictor-9ec573090acb.
8. Tasty Edits (n.d.). *The 10 Most Profitable YouTube Niches in 2023.* Tasty Edits. https://www.tastyedits.com/most-profitable-youtube-niches/.
9. YouTube. (2022). *YouTube for Press*. https://blog.youtube/press/.
10. YouTube. (n.d.). *API Reference.* YouTube. https://developers.google.com/youtube/v3/docs.
11. "3.2.4.3.1. Sklearn.ensemble.RandomForestClassifier — Scikit-Learn 0.23.2 Documentation." Scikit-Learn.org, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.feature_importances_.