YouTube Views Predictor

Maxence Frenette, Casey Hahn, Michael Lu
GitHub repo: https://github.com/maxencefrenette/youtube-views-predictor

Brief Description

Build and train a machine learning model to predict the number of views a YouTube video will receive based on the video's attributes, such as its title, length, category, comment count, and thumbnail image.

Problem Statement

YouTube is the world's largest video hosting website with more than 2.5 billion monthly users consuming over 1 billion hours of video per day. It is the world's second largest social media platform and the second largest search engine (behind Google, also owned by YouTube's parent company Alphabet). It is estimated that 500 hours of new video content is uploaded to YouTube every minute of every day.

With so much content on offer, how do publishers create videos that will stand out? This is an important question for both content creators and advertisers. According to a report in Variety magazine, the top 10 YouTube stars each earned over $15 million in 2021. In fiscal 2022, YouTube's advertising revenue totaled $29.2 billion.

Are there features and/or attributes of a video that make it more likely to garner attention from viewers and gain views?

Objective

Create a machine learning model to predict the number of views a video will receive based on the video's attributes and features. Utilize image and natural language processing techniques to analyze the video's thumbnail image, title, and description and leverage them as features in our model.

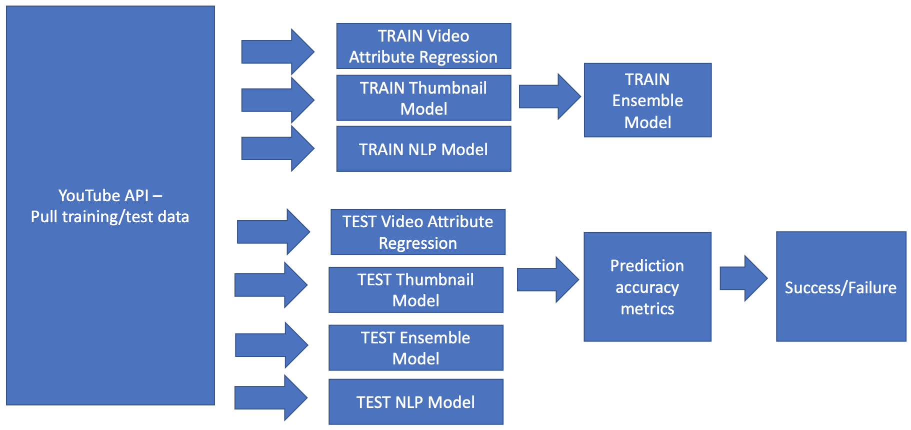
Approach/Methodology

We used multi-class classification to analyze YouTube video views. We split the views into 10 buckets based on every 10th percentile of views. Using this approach, we built a classification model to predict the view counts based on a video's features/attributes, including the thumbnail image that is displayed in the video suggestions and listings on the YouTube website.

In our analysis of YouTube video views, we used three different approaches to predict the number of views: 1) We built models using the video's attributes such as video length, category, time since release, likes-to-views ratio, and comments-to-views ratio. 2) We used a Convolutional Neural Network (CNN) and other deep learning models to analyze video thumbnail images and identify features that may predict the video's number of views. 3) We built Natural Language Processing (NLP) models to consider the video's title and description as additional predictive features.

Finally, we combined all three types of data into an ensemble model or hybrid machine learning model to predict the number of views.

Block Diagram


YouTube API – Pull training/test data
TRAIN Video Attribute Regression
TRAIN Thumbnail Model
TRAIN NLP Model
TRAIN Ensemble Model
TEST Video Attribute Regression
TEST Thumbnail Model
TEST Ensemble Model
TEST NLP Model
Prediction accuracy metrics
Success/Failure

Datasets

YouTube has a publicly available data API that developers can utilize to programmatically access the site's functionality and to obtain data about the videos that are hosted on the site.

To ensure that all users have access to the API, YouTube meters access using a credit system. Developers are given 10,000 daily API credits per day at no charge. Each API call costs a certain amount of credits based on the complexity of the API call. The API has further constraints to limit the type and quantity of data that can be retrieved. For example, it is not possible to request data on all videos for a particular country or category.

Working within these constraints, the team devised a strategy for downloading a representative dataset of sufficient size for this project: Download data on the top 50 most viewed videos in the United States. Then download data about 25 related videos (per YouTube's algorithm) for each of the top 50 videos. This strategy would give us data on 1,250 videos.

Our team pulled video data on two separate occasions 30 days apart: on March 4, 2023, and again on April 3, 2023. After de-deduplication of the data based on unique alphanumeric video identifiers, we ended up with a dataset consisting of 2,324 records with 24 fields (Data Dictionary available in Appendix). We randomized the data and applied a 70/10/20 split to create our train, validation, and test datasets, respectively.

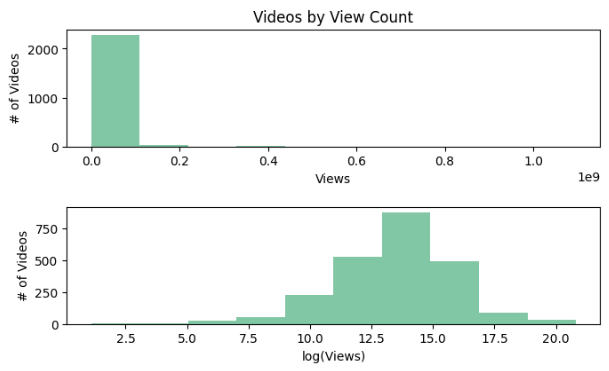
We additionally downloaded the thumbnail images for each of the 2,324 videos in our dataset. The URL for each video's thumbnail image is contained in the dataset retrieved via the YouTube data API. The image files were retrieved using a standard hypertext transport protocol (HTTP) request. All images were full color, of the same file format (JPG), and had consistent dimensions (small: 120x90 [10,800 pixels]; medium: 320x180 [57,600 pixels]).

Following exploratory data analysis (see next section), we performed logistic transformation on numeric columns that had wide data ranges, such as a video's number of views and a channel's number of subscribers. This de-skewed the distribution of data and, as a result, minimized the impact of outlier data points on our models.

Exploratory Data Analysis Graphs

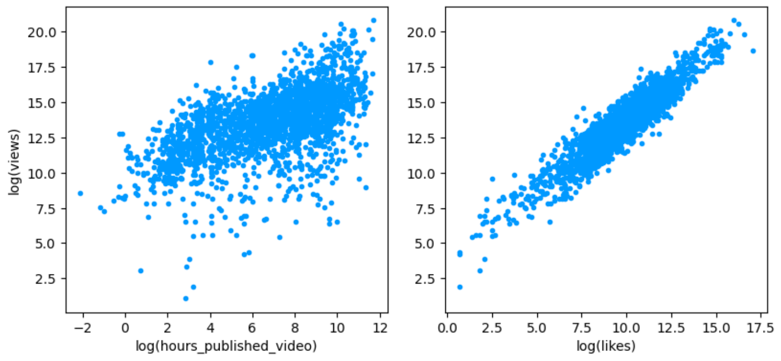
After downloading the data, the team performed exploratory data analysis on the dataset.

First and foremost, we looked at the range of values and the distribution of our output variable of interest: number of views. The graph to the right shows that the distribution of views is highly right skewed. Performing a natural log transformation on the variable spreads the values at the lower end of the range out more and results in a near normal distribution.

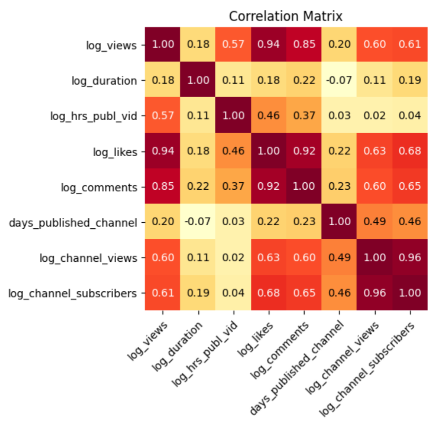


The log transformation was applied to other features that had a similarly wide range and skewed distribution of values, such as channel subscribers, channel views, and number of comments.

Next, we created a set of scatter plots to identify the relationship between video features and the output variable of interest (video views). A few of the scatter plots are shown below.



Linear relationships appear to exist between the natural log of views and the natural logs of (i) the hours since the video was published, (ii) the number of likes, and (iii) the number of comments. The full set of scatter plots are in the Appendix.



	log_views	log_duration	log_hrs_publ_vid	log_likes	log_comments	days_published_channel	log_channel_views	log_channel_subscribers
log_views	1.00	0.18	0.57	0.94	0.85	0.20	0.60	0.61
log_duration	0.18	1.00	0.11	0.18	0.22	-0.07	0.11	0.19
log_hrs_publ_vid	0.57	0.11	1.00	0.46	0.37	0.03	0.02	0.04
log_likes	0.94	0.18	0.46	1.00	0.92	0.22	0.63	0.68
log_comments	0.85	0.22	0.37	0.92	1.00	0.23	0.60	0.65
days_published_channel	0.20	-0.07	0.03	0.22	0.23	1.00	0.49	0.46
log_channel_views	0.60	0.11	0.02	0.63	0.60	0.49	1.00	0.96
log_channel_subscribers	0.61	0.19	0.04	0.68	0.65	0.46	0.96	1.00

Finally, the team created a correlation matrix to understand how the data features are related to each other. Channel subscribers and channel views have a correlation coefficient of 0.96, which means the variables move very closely together; thus, we can safely eliminate one of these features without losing much information. Similarly, the number of likes and comments a video receives are highly correlated with each other. In addition, both features are highly correlated with the number of views.

What is considered success/failure?

The primary goal of this project is to predict a video's performance on YouTube using its attributes, thumbnail, and title/description. We plan to build separate models for each type of data available. Given the noise present in our dataset, we do not anticipate achieving high performance metrics. However, we aim to demonstrate that our ensemble model outperforms the models only using video attributes. This would indicate that using a video thumbnail and/or the title/description can improve the model's predictive accuracy for a video's performance on YouTube. Ultimately, our objective is to develop a reliable and accurate model that can predict a video's success on the platform.

Evaluation Parameters

Accuracy, precision, F1 score, and recall will be our evaluation metrics.

Experiments

Baseline Models

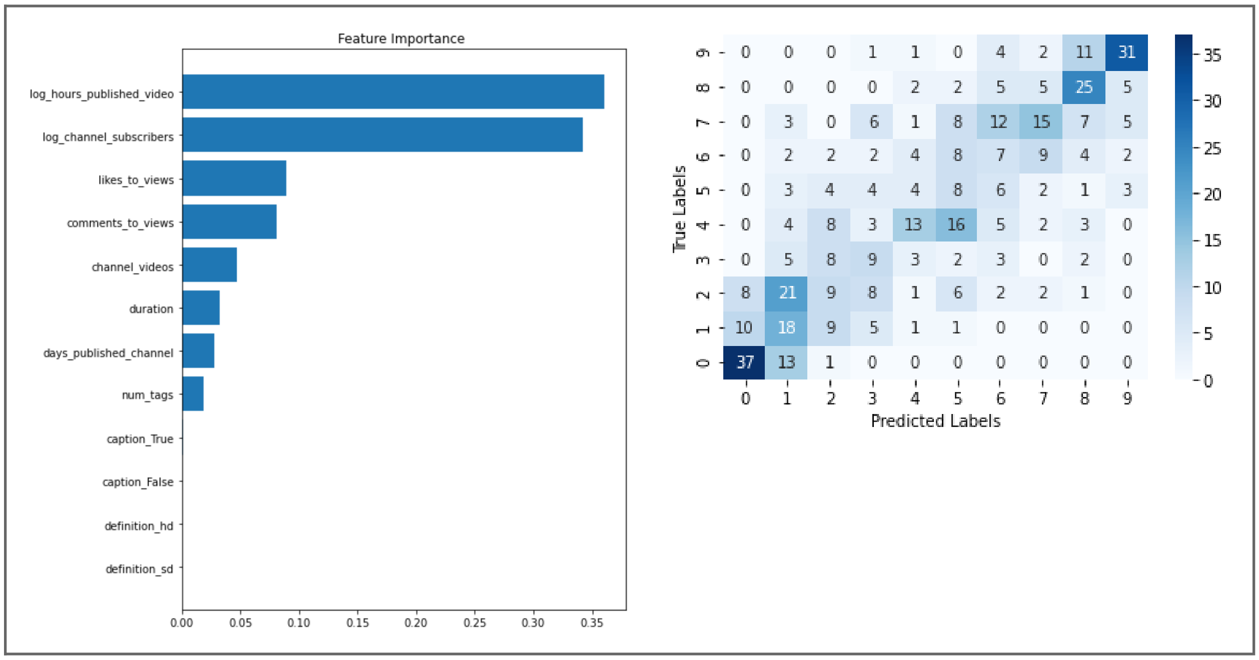
Results

In our study, we established a baseline by analyzing only the video attribute data to determine the best predictors of video views. We found that the number of channel subscribers and hours published were the most influential factors in predicting views. We will use this baseline to evaluate whether additional information from thumbnail images and text in the titles and descriptions can improve predictive accuracy.

Approach	Baseline: Random Guess	Logistic Regression	Gradient Boosted Decision Trees	Random Forest (TPOT Model)
Accuracy	0.10	0.11	0.37	0.37
Precision	0.10	0.13	0.37	0.35
Recall	0.10	0.11	0.37	0.36
F1 Score	0.10	0.07	0.36	0.35

Tests/Graphs/Discussion

The highest performing baseline model, the gradient boosted decision tree model had the highest precision of 67% for videos with lowest and highest views but struggled more at predicting videos with views in the middle categories. There were 8 features that helped higher importance in the model but hours published and channel subscribers were by far the most important.



Natural Language Processing Models

Results

Each video in our dataset was categorized into 10 classes, each representing the decile in which the video's view count falls into. We employed three Natural Language Processing (NLP) techniques - Bag of Words, Embeddings, and Transformers - on the text of the video titles and descriptions (separately) to train classification models. We then used each model to make predictions based on the video title or description. The evaluation metrics for each model are summarized in the table below:

Text Analyzed	**Video Title**			**Video Description**		
Technique	Bag of Words	Embeddings	Transformers	Bag of Words	Embeddings	Transformers
Accuracy	0.10	0.10	0.15	0.20	0.18	0.14
Precision	0.17	0.09	0.15	0.28	0.20	0.09
Recall	0.10	0.10	0.15	0.20	0.18	0.14
F1 Score	0.11	0.05	0.12	0.22	0.16	0.10

The confusion matrices comparing predicted labels to actual labels for each NLP model can be found in the Appendix.

A baseline model that predicts the same class regardless of input would achieve an accuracy score of 10% (1 in 10 chances of correctly predicting the video's view count decile). As you can see in the results table above, most of the NLP models performed no better than this baseline model. We suspect that the results of these NLP models are poor because the videos in our dataset are too varied and we did not have sufficient data points for any meaningful patterns to be found in the text. We believe that accuracy can be improved with (i) more data points for training the NLP models or (ii) focusing the NLP models on specific categories of videos so that text patterns can more easily be found.

Tests/Graphs/Discussion

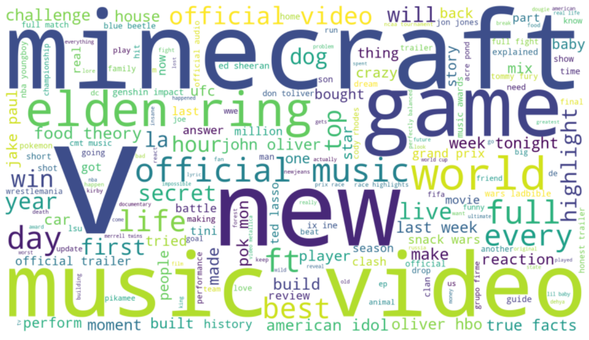
YouTube video descriptions contain a multitude of textual elements that do not actually describe the video content, including hyperlink URLs, social media user handles, emojis, etc. The first step the team took in implementing NLP techniques was to clean and normalize the data. We converted all characters to lowercase and we removed hyperlink URLs, non-alphabetic characters, single-character words, and common stop words (using the Natural Language Toolkit's English stop words set) from the video titles and descriptions. The table below shows a few examples of text in our dataset before and after cleaning.

Before Cleaning	**After Cleaning**
Bo Nickal learns the GOAT still has a few tricks up his sleeve 😏	bo nickal learns goat still tricks sleeve
Prosecutor Presses Alex Murdaugh on Video of Him with Slain Family at Dog Kennel	prosecutor presses alex murdaugh video slain family dog kennel
Hes trying to steal my job how dare he Everything is streamed live @ https://www.twitch.tv/flats before being uploaded to YouTube. ►Second Youtube Channel: https://www.youtube.com/channel/UCpXyxLahbn7vZfR5fexenuw…	hes trying steal job dare everything streamed live uploaded youtube second youtube channel…

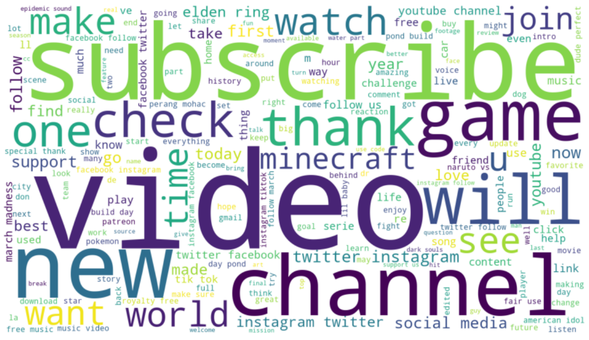
After cleaning the titles and descriptions in our training dataset had 26,640 and 12,656 unique words, respectively. Each title had an average of 6 words and each description had an average of 80 words. These metrics were used to guide us in hyperparameter tuning of our Embeddings model.

We created word clouds to visualize the most common words found in video titles and video descriptions:

Video Titles



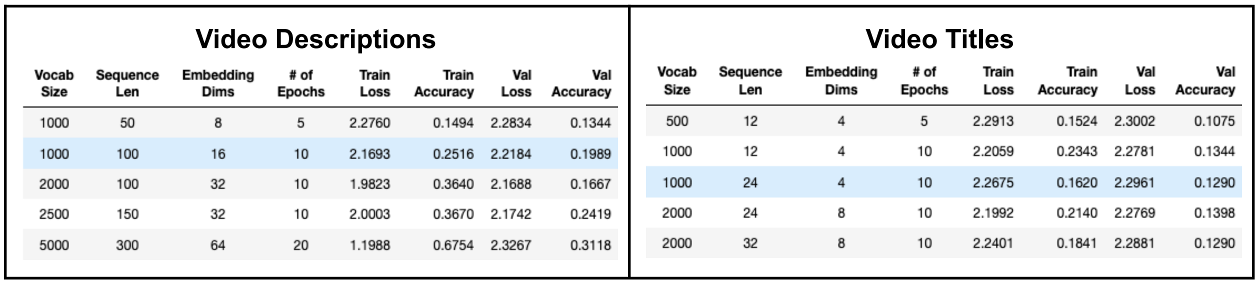
Video Descriptions



Bag of Words
We leveraged scikit-learn's CountVectorizer to generate the one-hot encodings for the model. scikit-learn's TfidfTransformer was used to adjust the "importance" of the words based on the term frequency-inverse document frequency algorithm (though some of that work was already done by filtering out common stop words in our data cleaning step). Finally, we used the multinomial Naive Bayes classifier to classify the videos into the target labels.

Embeddings
To build our Embeddings model, we used the TensorFlow Keras preprocessing package to tokenize and pad the video titles and descriptions to the desired length. The TensorFlow consisted of an embedding layer, a convolution layer, a global average pooling layer, and a dense layer for the output. The model was trained repeatedly with different hyperparameter values for the vocabulary size, the sequence length, the embedding dimensions, and the number of epochs. The initial values for vocabulary size and sequence length were selected based on the number of unique words and the average number of words in the input texts (titles and descriptions).

The table below shows the loss and accuracy for each set of hyperparameters. The hyperparameter sets highlighted in blue were selected for the final model, as they provided the highest validation accuracy without overfitting the model to the training data.



Video Descriptions

Vocab Size	Sequence Len	Embedding Dims	# of Epochs	Train Loss	Train Accuracy	Val Loss	Val Accuracy
1000	50	8	5	2.2760	0.1494	2.2834	0.1344
1000	100	16	10	2.1693	0.2516	2.2184	0.1989
2000	100	32	10	1.9823	0.3640	2.1688	0.1667
2500	150	32	10	2.0003	0.3670	2.1742	0.2419
5000	300	64	20	1.1988	0.6754	2.3267	0.3118

Video Titles

Vocab Size	Sequence Len	Embedding Dims	# of Epochs	Train Loss	Train Accuracy	Val Loss	Val Accuracy
500	12	4	5	2.2913	0.1524	2.3002	0.1075
1000	12	4	10	2.2059	0.2343	2.2781	0.1344
1000	24	4	10	2.2675	0.1620	2.2961	0.1290
2000	24	8	10	2.1992	0.2140	2.2769	0.1398
2000	32	8	10	2.2401	0.1841	2.2881	0.1290

Transformers
We used a pre-trained BERT (Bidirectional Encoder Representations from Transformers) language representation model, specifically the case insensitive BERT base model (bert-base-uncased). The model has a vocabulary of approximately 30,000 English words trained on the English-language edition of Wikipedia and over 11,000 unpublished books. Recall that the cleaned video descriptions in our dataset contained 26,640 unique words, so we felt that this model was appropriate for this project.

The BERT model became the initial layer in the classifier model, which we built using PyTorch's neural network (torch.nn) library. The model also includes a dropout layer, a linear transformation layer, and a Softmax layer.

Image Models

Results

We experimented with several machine learning models to predict the number of views based on the YouTube thumbnail images. However, our models consistently performed worse than random guessing. To improve our predictions, we also tried a pre-trained image object detection model and a NSFW (Not-Suitable-For-Work) model. Only the NSFW model proved slightly better than random guessing. In general, the complexity of the YouTube thumbnail images may be contributing to poor model performance, and may require a large number of images for training to produce accurate predictions.

Approach	Deep learning	Feature extraction with object detection	Feature extraction with NSFW Scores
Best performing model	CNN	Random Forest	Gradient Boosted Decision Tree
Accuracy	0.10	0.08	0.12
Precision	0.03	0.01	0.12
Recall	0.08	0.10	0.12
F1 Score	0.03	0.01	0.12

Tests/Graphs/Discussion

Deep Learning
The deep learning approach involved using images of YouTube thumbnails as input to train deep learning models. The images were pre-processed to standardize the inputs and then fed into a convolutional neural network (CNN) and a neural network.

During the model training, parameter tuning was completed on the deep learning models, including epoch size, batch size, optimizers, activation functions, filter size and the number of layers. To improve the CNN's performance, the training data was augmented by adding more training samples with augmented contrast, and flipped images.

However the CNN model's performance was poor, with only 10% of predictions being accurate in the test data set. We tried pulling higher resolution thumbnail images, standardizing the pixels by subtracting the mean and dividing by the standard deviation and reviewed the train and test data set for balance, but these did not explain or help the low performance of the models.

There could be several reasons for the poor performance of the models. One possible reason could be the lack of sufficient training data. Another possible reason could be the complexity of the data and nuanced relationship between thumbnail images and views. For future work, pulling only 1 specific category of youtube videos could help isolate patterns that the models could learn.

Feature extraction with Object Detection
In an attempt to improve the accuracy of our image classification models, we explored the use of feature extraction. We pre-processed our image data and fed them into four pre-trained object detection models: VGG16, Xception, YOLO, and MRCNN. The objects detected by these models were then passed into different predictive models, including logistic regression, gradient boosted decision trees, and random forests.

However, we found that the performance of these models using the object detection labels was poor across all four object detection models. While VGG16 showed slightly better results, we still sought to improve our models. We experimented with converting the detected objects to embeddings using the GloVe pre-trained model, and with using the Natural Language Toolkit (NLTK) to find the hypernym of each object detection label for a simpler category. Unfortunately, none of these additional efforts resulted in any significant improvements in our models.

Feature extraction with NSFW Scores
To further improve the accuracy of our thumbnail analysis, we experimented with using the pre-trained Open-NSFW2 model to detect "Not-Suitable-For-Work" (NSFW) content in the images. We pre-processed the images and passed them through the NSFW detector model, and then used the NSFW score as a feature in our predictive models, including logistic regression, gradient boosted decision trees, and random forests.

While the performance of these models using the NSFW score was poor, it was still better than random guessing, suggesting that there is some predictive power in the NSFW score. However, the majority of thumbnails that received a high NSFW score were actually wrestling images and other false positives for nudity, indicating that some fine-tuning of the model would be beneficial. Despite this limitation, the NSFW score is the one area in our thumbnail analysis that showed some potential for improving predictive accuracy.

Ensemble Models

Results

In order to maximize accuracy, we integrated all the most successful approaches into one single model. First, we added the image features to our existing best performing baseline models: the Gradient Boosted Decision Tree and the Random Forest. Then, we made an ensemble model that combined the output of those two models with a logistic regression. Finally, we added the bag of words models for the video title and description, as discussed in the NLP section. Here are the results:

Model	Gradient Boosted Decision Tree with Image Features	Random Forest with Image Features	Basic Data Ensemble Model	Full Ensemble (Basic Data and NLP)
Accuracy	0.34	0.36	0.38	0.39
Precision	0.35	0.37	0.39	0.39
Recall	0.34	0.36	0.38	0.39
F1 Score	0.34	0.35	0.37	0.37

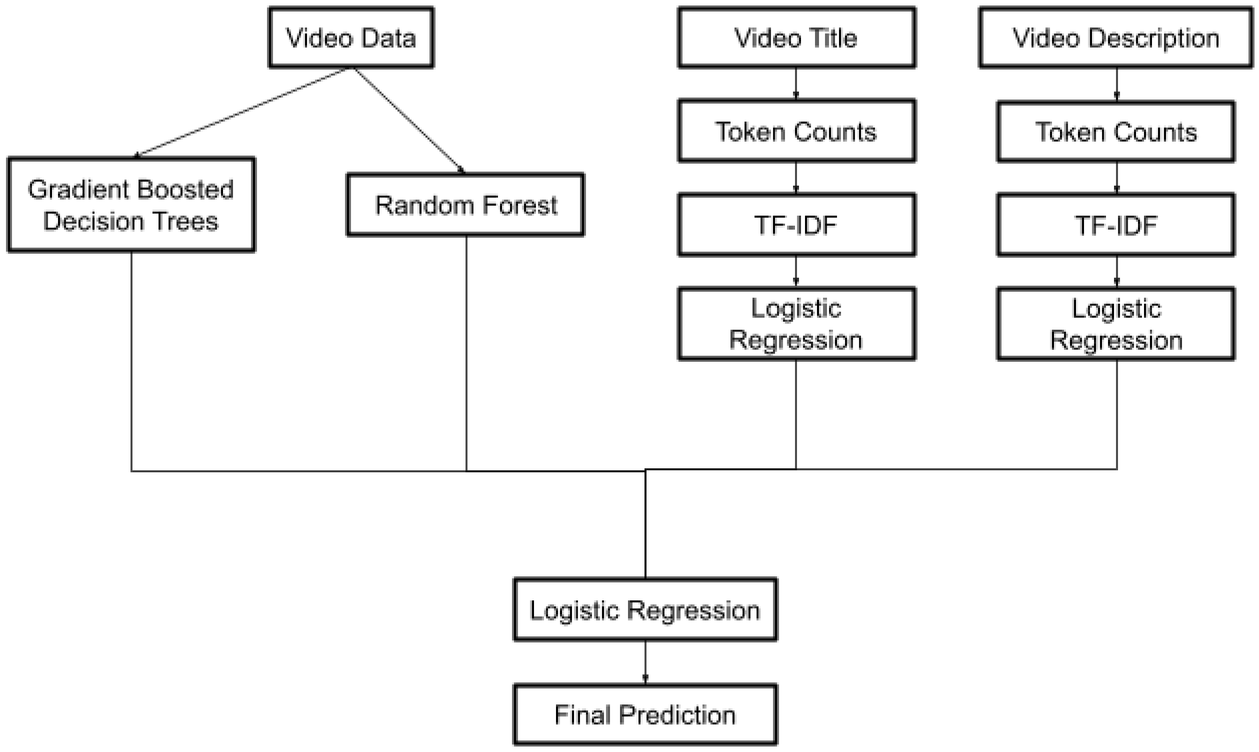
Tests/Graphs/Discussion

Baseline Models with Image Features
Contrary to the initial expectation, the results revealed that the inclusion of image features did not improve the performance of the models; rather, it led to a decrease in performance compared to the baseline models. As a result, it was decided not to include image features in the subsequent iterations of the models. This finding highlights the importance of carefully assessing the contribution of additional features to the performance of predictive models before incorporating them.

Basic Data Ensemble Model
Since Ensemble models made up of many weak predictors generally perform better than their parts, the two best performing baseline models were combined in an ensemble model. The outputs of the Gradient Boosted Decision Trees model and the Random Forest models were combined using a logistic regression. This resulted in an accuracy increase from 0.37 to 0.38.

Full Ensemble Model
Finally, the bag of words models for both the video title and description were added since they were the best performing models. At this stage, we also realized that the bag of words model performed slightly better with a logistic regression instead of a naive bayes classifier as their final output layer. The final

model gave an accuracy of 0.39, the best that we have achieved in this project. Here is a diagram of the final model:



Constraints and Limitations

In the course of conducting our research project, we encountered significant constraints which had an impact on the data collection process and analysis. The first limitation was the constraints imposed by the YouTube API, which hampered our ability to gather the volume of data we would have preferred. Specifically, we were unable to extract as many data points as we desired from the API, which presented a considerable challenge for our research.

Furthermore, we faced the issue of not obtaining a true random sample from the API. The only viable approach that we found for data collection involved pulling the most popular videos of the day in the United States and then gathering related videos to those, which led to a non-random sample. This approach limits the representativeness of our data, which is a significant concern in research methodology.

Another major constraint that we faced was the limited computing power available to us. This was a critical limitation that we had to work around while conducting our research project. Even if we had access to more data points, the image processing models that we used for our analysis required more would have taken too long to train with our limited computing power. This constrained our ability to use more sophisticated models, and we had to rely on simpler, less powerful models to avoid overfitting.

Taken together, the limitations that we faced in terms of data collection and analysis had a significant impact on the quality and representativeness of our research findings. Although we attempted to mitigate these constraints as much as possible, we recognize that our results may be biased due to the non-random nature of our sample and our use of simpler models. Therefore, we urge caution in interpreting our findings and recommend that future research seeks to overcome these constraints through more sophisticated data collection methods and increased computing power.

Standards

Our project leveraged many generally available open source libraries and packages:

NLP Libraries & Packages

- Natural Language Toolkit (NLTK)
    - Stop words for filtering out frequently used words that have little meaning on their own
    - Hypernyms for object detection labels
- scikit-learn (sklearn)
    - LabelEncoder - converting categorical values to numeric values for ML
    - CountVectorizer - creating one-hot encodings used in the Bag of Words model
    - TfidfTransformer - calculating term frequency and inverse document frequency for finding words that are not frequently used but carry lots of meaning
    - MultinomialNB - multinomial naive Bayes classifier
    - GridSearchCV - for parameter tuning
- TensorFlow
    - Tokenizer - creating padded tokenized representations of text strings
    - Keras - building multi-layer models for text embeddings and thumbnail images
- Pytorch & HuggingFace Transformers
    - BertModel & BertTokenizer - pre-trained bi-directional transformer model for language representation
- TPOT
    - Hyperparameter tuning and pipeline optimization for scikit-learn models
    - AutoML
- AutoKeras
    - Hyperparameter tuning and architecture optimization for keras neural network models
    - AutoML

Image Processing Libraries & Packages

- VGG16, Xception, YOLO, Mask R-CNN - pre-trained models for detecting objects in images
- Open-NSFW2 - pre-trained model for detecting Not-Safe-For-Work content in images and videos
- TensorFlow - Keras for building the Convolutional Neural Network and dense neural networks

Core Python Libraries & Packages

- NumPy, Pandas, Matplotlib, Seaborn

Comparison

A similar project to predict YouTube views through machine learning was undertaken in 2017 by Allen Wang, Aravind Srinivasan, Kevin Yee and Ryan O'Farrell. A discussion of their project can be found here. They, too, had hoped to predict video views given a video's title and thumbnail image. Unlike our project, they focused on a single category: health & fitness videos. Their dataset consisted of data on over 115,000 videos; far greater than the 2,324 videos in our dataset.

Like our project, they trained neural networks on the video titles and thumbnails to directly predict video views. And although they do not discuss the specific results they got from these neural networks, they said in their discussion that they "did not get very promising results". Our findings are generally in concurrence with theirs.

They chose to instead focus on extracting features from the titles and thumbnails that they would then feed into a model along with other video features, such as number of comments, number of likes, number of channel subscribers, etc. This is very similar to the NLP, image, and ensemble models we built.

With video titles, they focused on how “clickbait-y” the title was. Clickbait is text written to attract attention and entice readers. They found the clickbait titles were not limited to videos with the highest views, so it was not much of a predictor of video views. We tried several NLP methods – bag of words, embeddings, and transformers – on the video titles and descriptions in the hopes of finding patterns (of words, of context/meaning, of structure) that could be predictive of video views. All of our NLP models failed to produce significant results, similar to this team’s findings.

With video thumbnails, they focused on how “not-safe-for-work” (NSFW) the image was. This team found that videos with high views had a higher NSFW score than videos with low views. We too tried a NSFW scoring approach in our models. We concur that NSFW content appears to be a predictor of video views: Of the different image processing models we tried, the NSFW scoring model produced the highest accuracy scores.

Ethics & Fairness

Much has been written in the press about the algorithms social media sites use to recommend content to their users. Social media companies are often faulted for promoting violent and dangerous content in order to keep user engagement (and thus, advertising revenues) high. The number of views any particular video receives is due in large part to YouTube’s recommendation engine surfacing the video to its audience.

The models in our project were trained with data that is essentially an end-product of YouTube’s algorithms. Thus, our models are susceptible to the biases that might be present in those algorithms. Our models could potentially lead to more undesirable content as creators craft content based on our models recommendations to chase more views. To mitigate these risks, we could selectively exclude undesirable content from our dataset.

The details of how YouTube’s algorithms work are closely guarded secrets. Using machine learning to identify patterns in the video features, titles, descriptions, and thumbnail images and understand how they impact views helps shed a little light on the inner workings of the algorithm. Viewed in this light, this project can help creators and advertisers overcome any biases in YouTube’s algorithms and level the playing field a bit.

Our training dataset includes data on just over 1,600 YouTube videos. That is but a tiny fraction of the estimated 800 million videos available on the website worldwide. Our data gathering started with the top 50 videos viewed in the United States, then fanned out to 25 videos related to each of them. As a result of this data gathering strategy, the resulting dataset is made up of predominantly English-language content. Additionally, our NLP transformers model used a BERT model that was pre-trained on English-language content. As a result, our models are useless for foreign-language content. Massively increasing the size of the dataset, including data on foreign-language content, and leveraging foreign-language vocabularies in our models would overcome this problem.

Besides the pre-trained BERT model for NLP, we also used pre-trained models for object and NSFW content detection. Any biases present in these pre-trained models would also impact our’s. Fortunately, these models are generally open-source and subject to public review. Similar transparency about our model would greatly increase confidence in its fairness.

Future Work

There are lots of future avenues to improve the performance of these models.First, there are ways to get better model performance without pulling more data. Although we performed hyperparameter optimization on our sklearn models using the TPOT library to get out baseline models, we didn’t rerun the same process on our state of the art model at the end of the project. This could have resulted in additional accuracy.

Similarly, we still have lots of ways to tweak our image models in order to extract some signal out of them. Since our issue is probably overfitting, we could try making a CNN with less trainable parameters in hopes of recognizing basic patterns in the thumbnails. We could also reduce the number of groups of image detection labels in our feature extraction in order to have a simpler model.

Then, there is the obvious approach of pulling additional data. This could be done by automating the script so that it pulls ~1000 youtube videos every day from the API. This could give us the ability to train more complex models.

We previously tried training a neural network on the basic video data and used the autokeras library to tune the hyperparameters. This didn't beat our baseline models in terms of accuracy, so we abandoned the approach. Perhaps that with more data, it would become a viable approach.

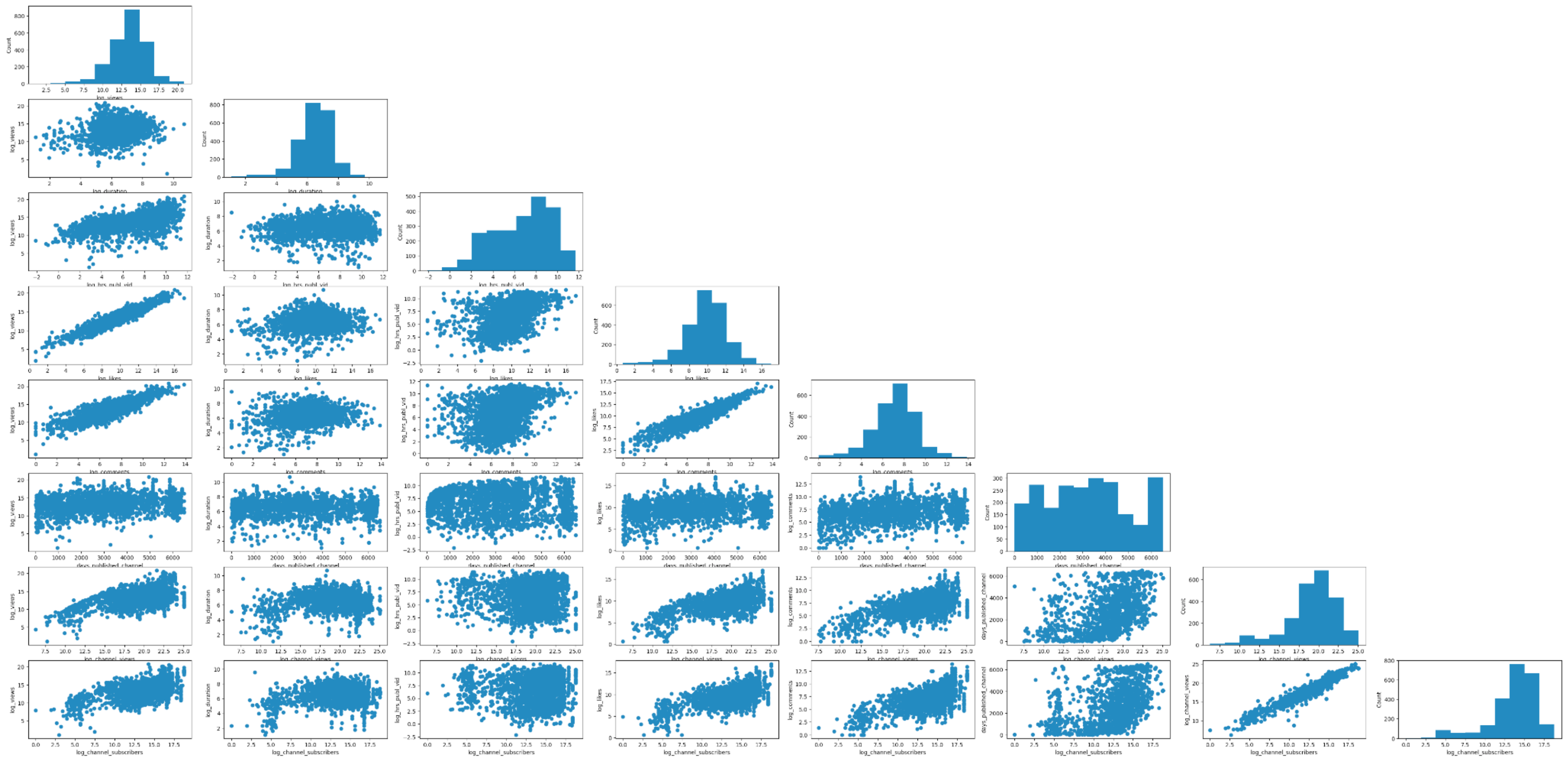
Another opportunity that would be available with more data would be to train a model on the actual video. This would also require much more computing power than we had available as well as advanced machine learning techniques to get a model that can derive useful information from a raw video.

Appendix

Data Dictionary

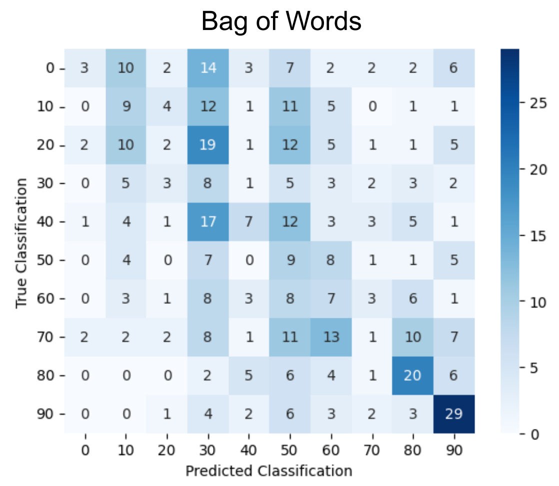
Column Name	Data Type	Description
video_id	Text	Unique alphanumeric identifier for the video
title	Text	Title of the video
description	Text	Description of the video
video_published_at	DateTime	Date and time video was published
hours_published_video	Numeric	Number of hours since video was published
num_tags	Numeric	Number of text tags assigned to the video
thumbnail_url	Text	URL for retrieving the video's thumbnail image
thumbnail_width	Numeric	Width of the thumbnail in pixels
thumbnail_height	Numeric	Height of the thumbnail in pixels
duration	Numeric	Duration of the video in number of seconds
definition	Categorical	Video's resolution (hd = 1920x1080 pixels)
caption	Boolean	Presence of text captions in the video
views	Numeric	Number of times the video has been viewed
likes	Numeric	Number of likes a video has received
favorites	Numeric	Number of times a video has been added to a viewer's favorites list
comments	Numeric	Number of comments posted to the video's page on the YouTube website
channel	Text	Name of the channel to which the video was published
channel_id	Text	Unique alphanumeric identifier for the channel
channel_description	Text	Description of the channel
channel_published_at	DateTime	Day and time the channel was published
days_published_channel	Numeric	Number of days since the channel was published
channel_views	Numeric	Total number of views of the channel's videos
channel_videos	Numeric	Total number of videos published to the channel
channel_subscribers	Numeric	Number of users who have subscribed to the channel

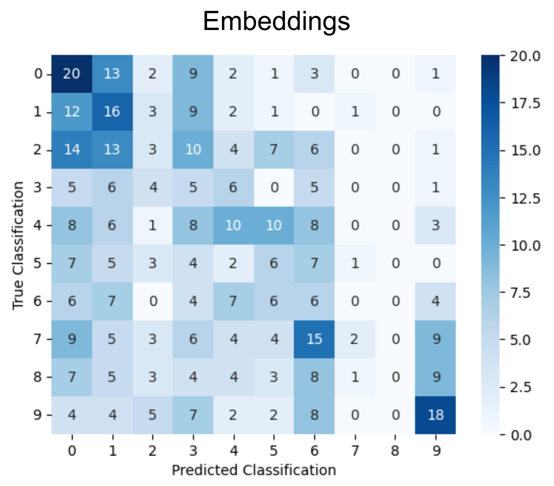
Scatter Plots of Video Features and Video Views

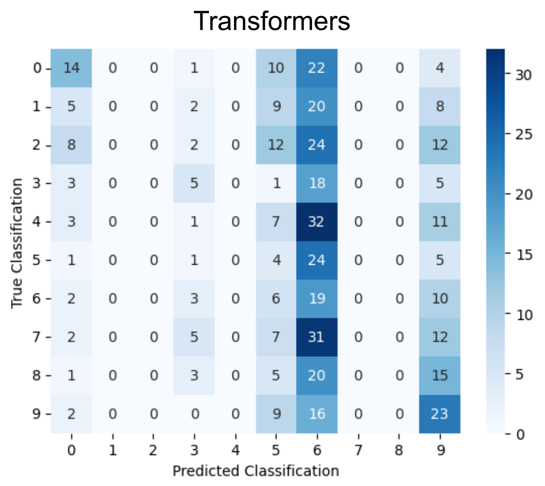


Confusion Matrices for NLP Models

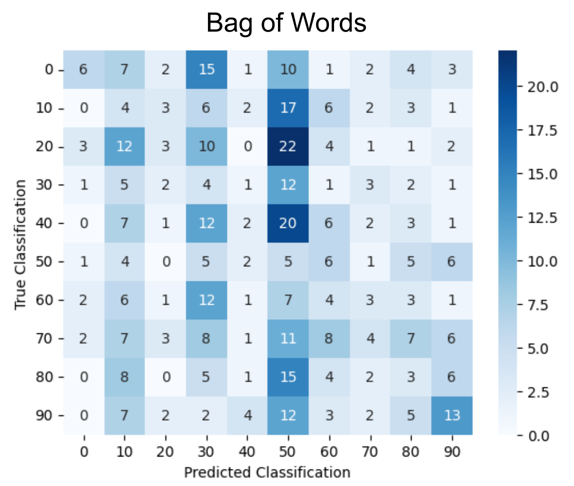
Input Data: Video Descriptions

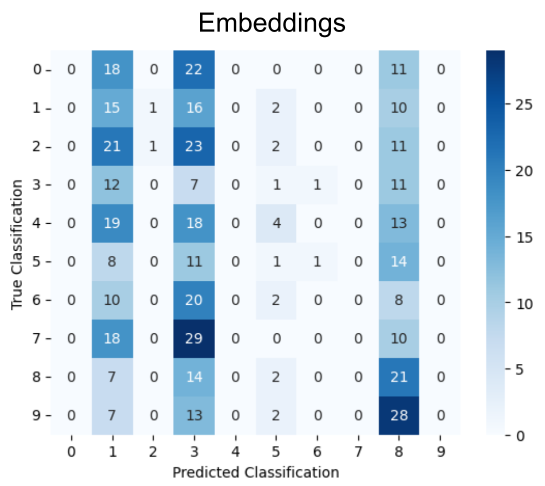


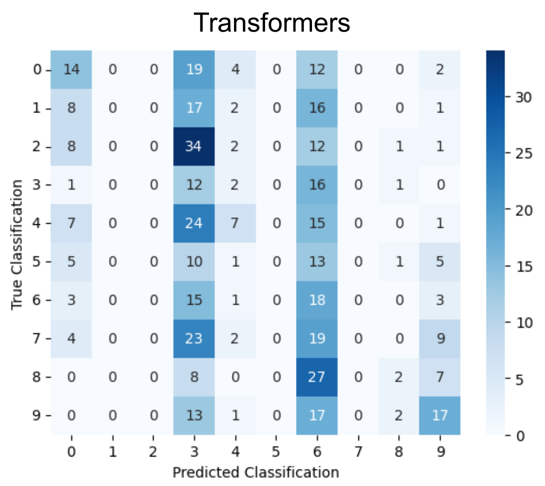




Input Data: Video Titles

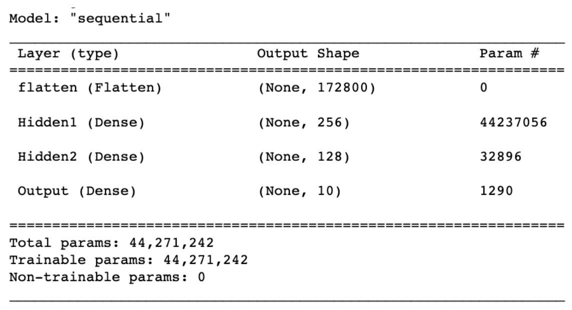




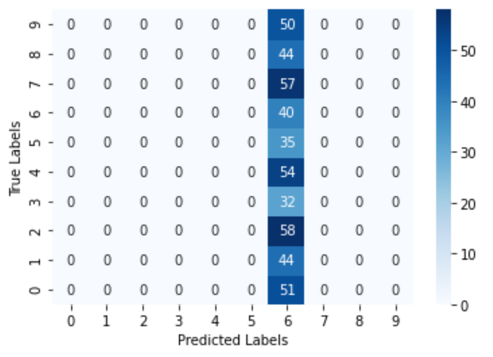


Thumbnail deep learning model approach

Neural Network:

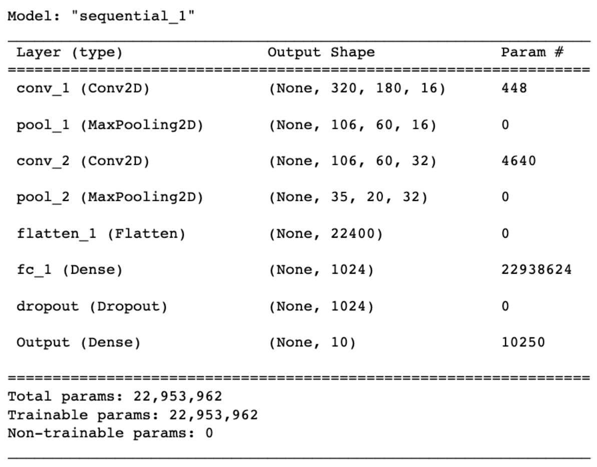


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 172800)            0

 Hidden1 (Dense)             (None, 256)               44237056

 Hidden2 (Dense)             (None, 128)               32896

 Output (Dense)              (None, 10)                1290

=================================================================
Total params: 44,271,242
Trainable params: 44,271,242
Non-trainable params: 0
_________________________________________________________________
```

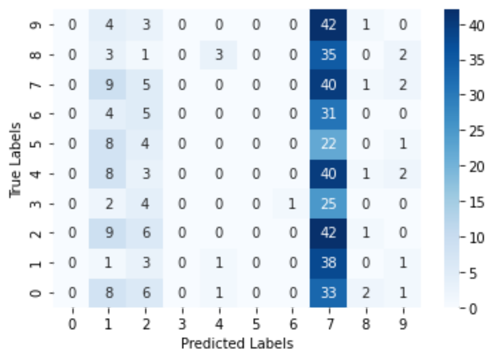


	0	1	2	3	4	5	6	7	8	9
9	0	0	0	0	0	0	50	0	0	0
8	0	0	0	0	0	0	44	0	0	0
7	0	0	0	0	0	0	57	0	0	0
6	0	0	0	0	0	0	40	0	0	0
5	0	0	0	0	0	0	35	0	0	0
4	0	0	0	0	0	0	54	0	0	0
3	0	0	0	0	0	0	32	0	0	0
2	0	0	0	0	0	0	58	0	0	0
1	0	0	0	0	0	0	44	0	0	0
0	0	0	0	0	0	0	51	0	0	0

CNN model:



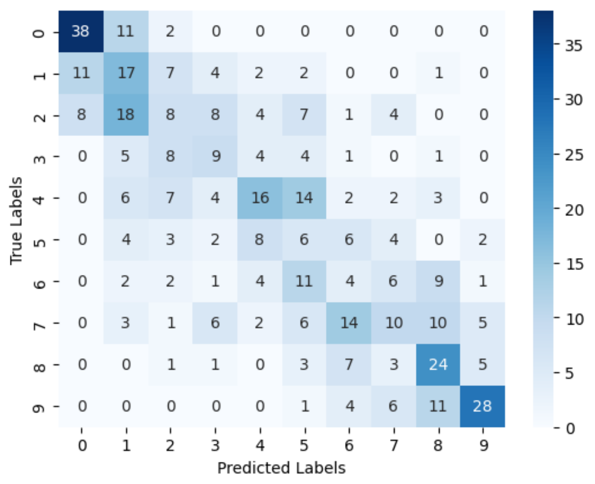
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_1 (Conv2D)             (None, 320, 180, 16)      448

 pool_1 (MaxPooling2D)       (None, 106, 60, 16)       0

 conv_2 (Conv2D)             (None, 106, 60, 32)       4640

 pool_2 (MaxPooling2D)       (None, 35, 20, 32)        0

 flatten_1 (Flatten)         (None, 22400)             0

 fc_1 (Dense)                (None, 1024)              22938624

 dropout (Dropout)           (None, 1024)              0

 Output (Dense)              (None, 10)                10250

=================================================================
Total params: 22,953,962
Trainable params: 22,953,962
Non-trainable params: 0
_________________________________________________________________
```



	0	1	2	3	4	5	6	7	8	9
9	0	4	3	0	0	0	0	42	1	0
8	0	3	1	0	3	0	0	35	0	2
7	0	9	5	0	0	0	0	40	1	2
6	0	4	5	0	0	0	0	31	0	0
5	0	8	4	0	0	0	0	22	0	1
4	0	8	3	0	0	0	0	40	1	2
3	0	2	4	0	0	0	1	25	0	0
2	0	9	6	0	0	0	0	42	1	0
1	0	1	3	0	1	0	0	38	0	1
0	0	8	6	0	1	0	0	33	2	1

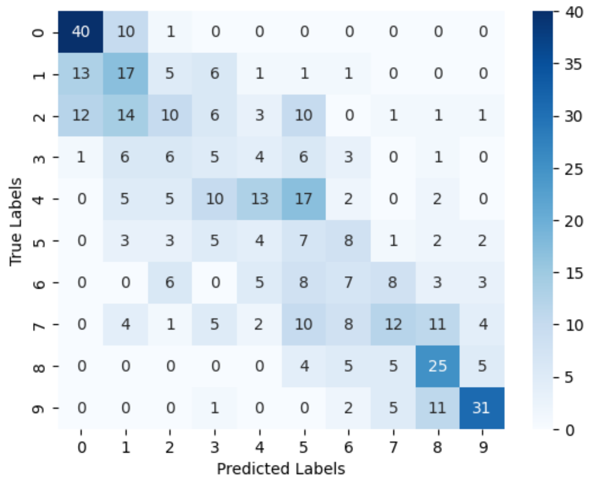
Confusion Matrices for Ensemble Models

Gradient Boosted
Decision Tree with Image Features



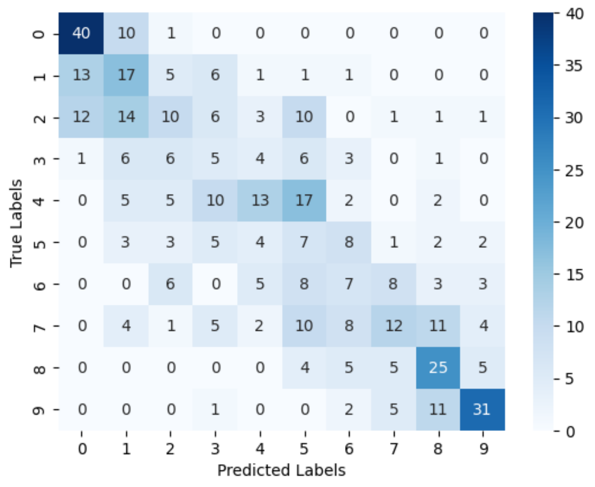
True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	38	11	2	0	0	0	0	0	0	0
1	11	17	7	4	2	2	0	0	1	0
2	8	18	8	8	4	7	1	4	0	0
3	0	5	8	9	4	4	1	0	1	0
4	0	6	7	4	16	14	2	2	3	0
5	0	4	3	2	8	6	6	4	0	2
6	0	2	2	1	4	11	4	6	9	1
7	0	3	1	6	2	6	14	10	10	5
8	0	0	1	1	0	3	7	3	24	5
9	0	0	0	0	0	1	4	6	11	28

Random Forest with Image Features



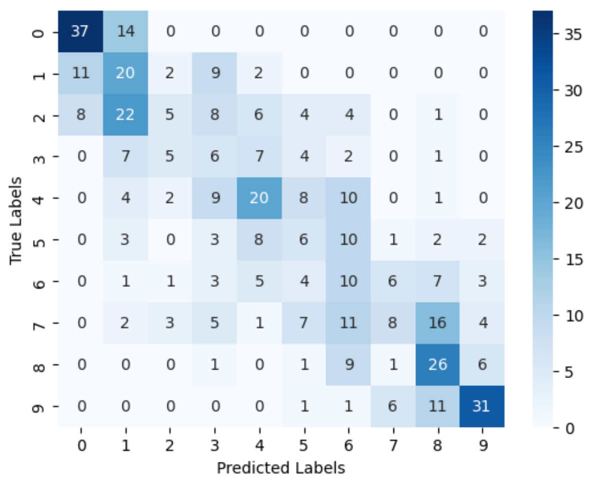
True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	40	10	1	0	0	0	0	0	0	0
1	13	17	5	6	1	1	1	0	0	0
2	12	14	10	6	3	10	0	1	1	1
3	1	6	6	5	4	6	3	0	1	0
4	0	5	5	10	13	17	2	0	2	0
5	0	3	3	5	4	7	8	1	2	2
6	0	0	6	0	5	8	7	8	3	3
7	0	4	1	5	2	10	8	12	11	4
8	0	0	0	0	0	4	5	5	25	5
9	0	0	0	1	0	0	2	5	11	31

Basic Data Ensemble Model



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	40	10	1	0	0	0	0	0	0	0
1	13	17	5	6	1	1	1	0	0	0
2	12	14	10	6	3	10	0	1	1	1
3	1	6	6	5	4	6	3	0	1	0
4	0	5	5	10	13	17	2	0	2	0
5	0	3	3	5	4	7	8	1	2	2
6	0	0	6	0	5	8	7	8	3	3
7	0	4	1	5	2	10	8	12	11	4
8	0	0	0	0	0	4	5	5	25	5
9	0	0	0	1	0	0	2	5	11	31

Full Ensemble (Basic Data and NLP)



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	37	14	0	0	0	0	0	0	0	0
1	11	20	2	9	2	0	0	0	0	0
2	8	22	5	8	6	4	4	0	1	0
3	0	7	5	6	7	4	2	0	1	0
4	0	4	2	9	20	8	10	0	1	0
5	0	3	0	3	8	6	10	1	2	2
6	0	1	1	3	5	4	10	6	7	3
7	0	2	3	5	1	7	11	8	16	4
8	0	0	0	1	0	1	9	1	26	6
9	0	0	0	0	0	1	1	6	11	31