Will You Rewind This?

YouTube Analytics

Ian Aliman, LaRue Brown, Hannah Jordan, Gayathri Sanjeev

2019

Table of Contents

[Introduction 1](#_Toc19112420)

[Analysis and Models 1](#_Toc19112421)

[About the Data 1](#_Toc19112422)

[Data Acquisition & Cleaning 2](#_Toc19112423)

[Cluster Analysis 5](#_Toc19112424)

[Support Vector Machines 6](#_Toc19112425)

[Multinomial Naïve Bayes 10](#_Toc19112426)

[Results 13](#_Toc19112427)

[Cluster Results 13](#_Toc19112428)

[Support Vector Machines Results 13](#_Toc19112429)

[Multinomial Naïve Bayes Results 15](#_Toc19112430)

[Conclusions 16](#_Toc19112431)

[References 16](#_Toc19112432)

# Introduction

When YouTube first made its way onto the internet, few people realized how many hours of video one would be watching years later. It seems the internet never gets bored of watching videos, so now there is an entire library’s worth of YouTube stats. YouTube is the world’s second largest search engine and second most visited site after Google. There are so many measures available from YouTube that can be used to derive various statistical analyses.

Audience retention is one of the measures used by YouTube to rank videos in search. This measure also plays a huge role when YouTube algorithm decides whether to suggest a video to the viewers. Audience retention report to get an overall measure of how well your video keeps its audience. Being able to predict if a retention rate is higher or lower helps to understand what part of the video interests or captures the audience.

# Analysis and Models

The YouTube channel Ink Master has several clips, trailers, and episodes of a popular show by the same name. For the following models and analysis, 10 of the most popular Ink Master videos were randomly selected.

## About the Data

Each of the 10 videos have three components. The first is the actual captions. For each video, a full transcript is recorded. Below is a small sample from one of the video captions.

0:00:00.810,0:00:06.270

[Music]

0:00:03.560,0:00:09.179

welcome only eight of you remain and

0:00:06.270,0:00:11.849

once again the teams are even foreign

…

Notice the transcripts use square brackets (as well as parenthesis) to represent either non-verbal contextual notes or speaking off-screen. Each caption is also broken down into small chunks with a start and end time. Notice the first spoken sentence is broken across several time intervals in the transcript. Each caption consists of three lines. The first line is the start and end time for that caption. The second line is the caption itself. The third line is blank. This pattern repeats for the entire length of the video. The very last caption shows the total length (in HH:MM: SS format) of the video.

The second component is the analytics. The analytics for each video provide useful metrics to label each video. The analytics are in JSON format, and a sample is shown below.

…

"rows": [

[

0.01,

0.9824766871030184,

0.5412873682951259

],

[

0.02,

0.7340995738542272,

0.6187272195344471

],

[

0.03,

0.7772063531784292,

0.6622448317627005

],

…

There are three main metrics provided in each analytics file. The first is the “elapsedVideoTimeRatio”. Since each video is of different length, using the actual moment in time to compare one video to another would not give an accurate comparison. Instead, this measure takes the length of the video and breaks it into a hundred equal parts. Said another way, each of these metrics represent one hundredth segment of a video. The second metric is the “audienceWatchRatio”. This measures the ratio of audience members that watched this segment. The last metric is “relativeRetentionPerformance”. This one looks at how videos overall perform. Comparing the “audienceWatchRatio” to the “relativeRetentionPerformance” will show if a segment of the Ink Master video had a higher audience retention than the average of all other videos.

The third component is the interaction of the viewers with the videos, specifically through comments. For each of the videos, the export of the comments included a comment ID, commenting user, date, timestamp, comment text, number of likes, and replies.

## Data Acquisition & Cleaning

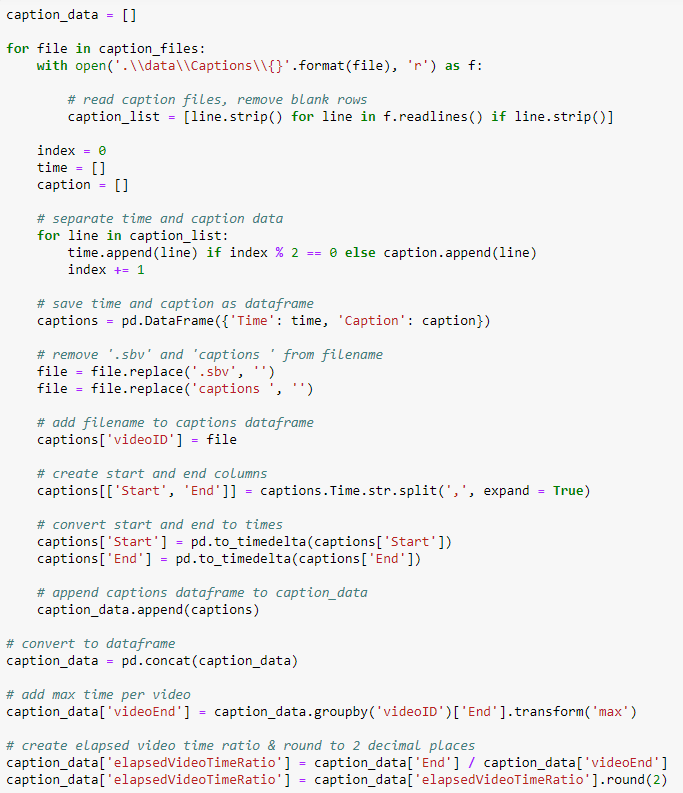
Prior to building predictive models, the data must be acquired and cleaned. Both the captions and analytics are in text files; however, the analytics are in JSON format. The ‘json.load’ method can be used to easily import the analytics from each file. The key is to ensure each file name is preserved. Otherwise, there would be no way to associate the analytics with the caption, nor would there be any way of knowing which analytics are from which video. Assuming all the file names are stored in a list, the following can be used to store the analytics data in a data frame.



*Figure 1.1 Code to store analytics data*

The caption files require a little more prep. Since they are not already in a format that is easily converted, each file must be cleaned and converted to a usable format. As noted earlier, the captions contain a start and end time as well as the actual caption. There is also a blank line. First remove the blank lines. This leaves only two lines per group (time and caption). Knowing this, every odd-numbered line contains the time, and even-numbered line contains the caption. Then, the time itself needs to be further divided into start and end times as well as converted from string to actual time.

Of course, combining the caption data and analytics data only by the file name would not produce the correct results since each analytics file contains 100 rows and the captions can contain any number of rows. A second joining key is needed. The “elapsedVideoTimeRatio” metric in the analytics file will ensure the correct segments for each video are aligned. However, the caption times must be converted to match the same format. The end time for the very last caption has the length of the entire video. With this time, each caption segment’s end time can also be used, and the ratio of these two values will produce the same “elapsedVideoTimeRatio”. Below are the prep steps for the caption data, again assuming all the caption file names are stored in a list.



*Figure 1.2 Code to store caption data*

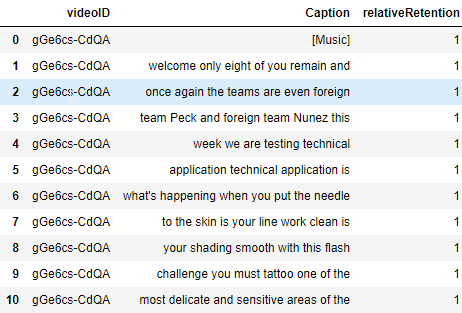
Now both data frames (“caption\_data” and “analytics\_data”) contain a “videoID” (which is the file name) and an “elapsedVideoTimeRatio”. Therefore, the two data frames can be combined with “videoID” and “elapsedVideoTimeRatio” as the joining keys.

Lastly, each of the captions must be labeled. If the “relativeRetentionPerformance” is less than the “audienceWatchRatio”, then the “relativeRetention” is higher (coded at 1). Otherwise, the “relativeRetention” is lower (coded as 0).



*Figure 1.3 Code snippet for the logic behind the creation of label*

The newly combined data used as a starting point for analytics contains the “videoID”, the “Caption”, and the “relativeRetention” label.



*Figure 1.4 Final data to be used for analysis*

## Cluster Analysis

The scraped comments from each of the videos were used for a cluster analysis. To prepare the data, all comments with a manually-compiled list of explicit words and all comments with emojis (which were not readable by the program) were removed. To prevent inclusion of comment texts without context, replies were also removed. From there, in order to have an balanced number of comment texts for each of the videos, the 50 most recent remaining comments for each video were used for analysis. Figure 2.1 displays a word cloud of the comments for analysis.



Figure 2.

Once the comment text was vectorized, the data was cleaned by excluding the NLTK-defined English stopwords, excluding tokens shorter than 3 characters, and combining tenses to reduce the data to 1222 tokens. The cleaned vectorized data was then converted into a matrix for use in sklearn’s K-means clustering function.

While this would normally be the point where an elbow plot could be utilized to identify the ideal number of clusters, this step is unnecessary because it is known that the comment data came from 10 different videos. The matrix of vectorized and cleaned comments were plugged into the K-means clustering algorithm, set to detect 10 clusters. Figure 2.2 displays a summary of how many comments were clustered together.

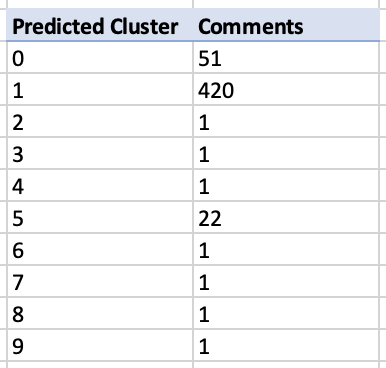


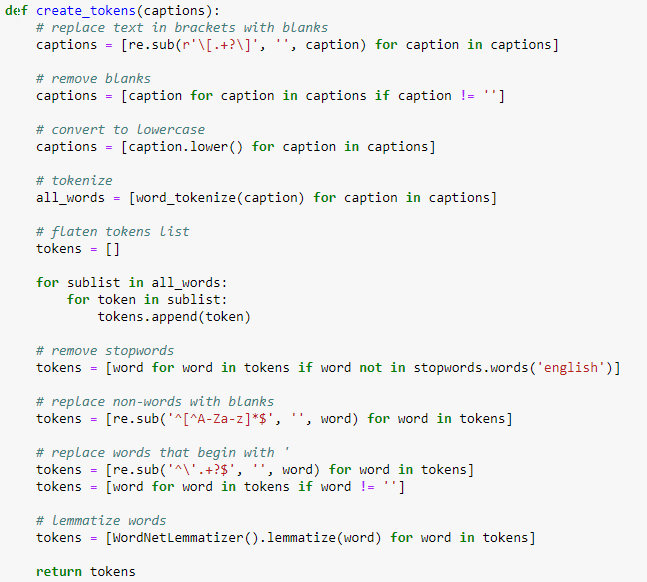
Figure 2.

## Support Vector Machines

Using the “combined\_data” data frame as a starting point, a few additional preparation steps must be performed prior to applying a support vector machine (SVM) model. The following actions must be performed:

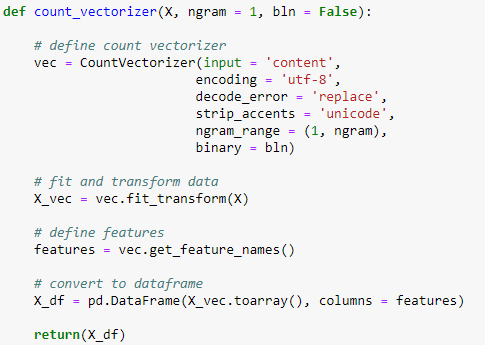
* Remove text in square brackets (which represents the non-verbal contextual notes)
* Convert all captions to lowercase
* Tokenize each caption (split each caption into individual words)
* Remove all English stop words
* Remove all tokens that consist of only numbers or symbols (has no letters)
* Remove the tokens that begin with an apostrophe (from tokenized contractions)
* Remove blank/null tokens

The steps above are self-explanatory. However, one additional step requires some additional explanation. The last step is to lemmatize each of the words. The process of lemmatization groups together various forms of a word to a root word. This helps in analytics as there are several forms of a word, and if each are considered, a new token would be generated and could potentially affect the outcome of the models. In total, all the steps listed above plus the lemmatization are part of tokenizing the captions. Since this process will need to be used multiple times (on all the captions combined, plus each individual row in the “combined\_data” data frame), a function is created to easily tokenize multiple captions.



*Figure 4.1 Code snippet for creation of tokens*

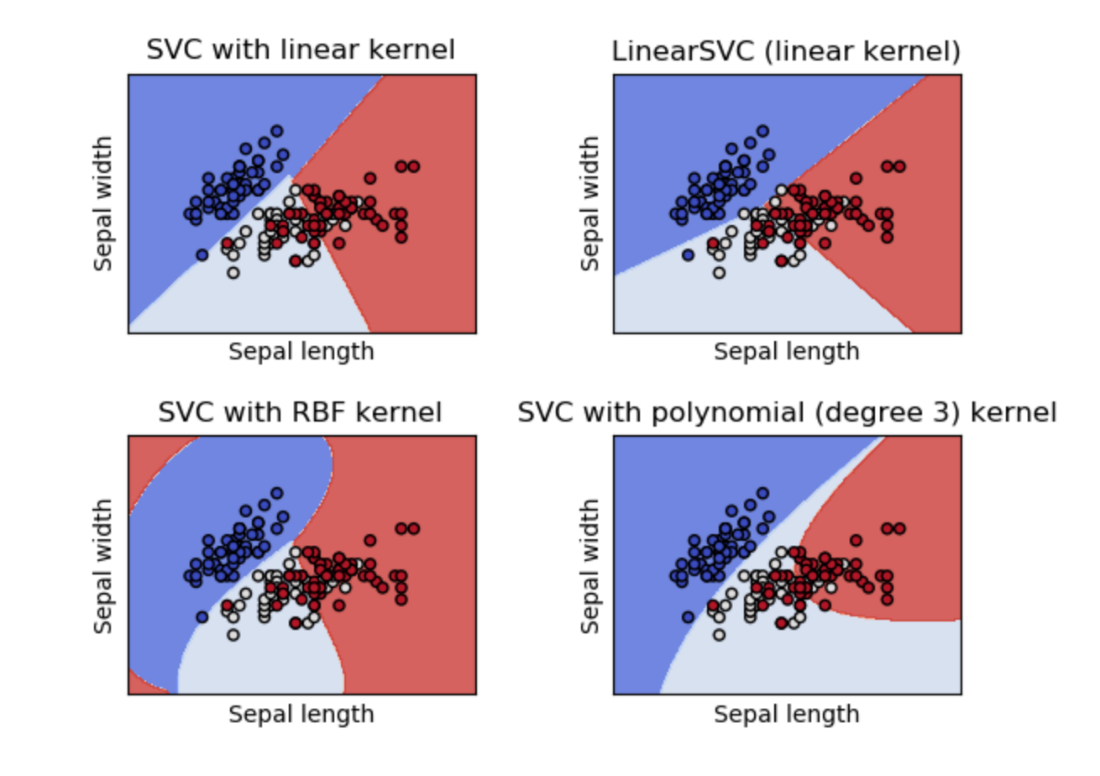
The initial tokenization was to have more granular control. However, the vectorizer has its own tokenizer. So, after all the steps from the “create\_tokens” function are done, each caption is combined back into a single string (for each segment). This will allow the “CountVectorizer” method to properly tokenize and vectorize each caption. Since most of the required steps were performed in the tokenization function, the only things to consider in the vectorizer is which ngram to use and wither to perform a count or simply show presence/absence of a token (binary). Another function is created to perform the vectorization. The function is shown below.



*Figure 4.2 Code snippet for vectorization*

This function will always have a unigram, however, the range can be set so it can include unigrams and bigrams, for example, if the “ngram” parameter is set to 2. It can also be set to either count each occurrence of a token or just show a 1 or 0 for presence or absence (respectively). Since each caption segment is only a few seconds at best and stop words are removed, it is unlikely that using a binary method would have different results than a count; however, both options are available.

The last function is to create and evaluate the SVM models. Since this will be run multiple times, again a function is the best option. Also, there are several parameters the SVM model can take. These parameters are included in a grid search. This will allow the model to test each combination of parameters and select the best one. The two parameters that will be tuned are the kernel and the penalty. The **scikit learn** documentation gives a visual to help better understand the three kernels used.

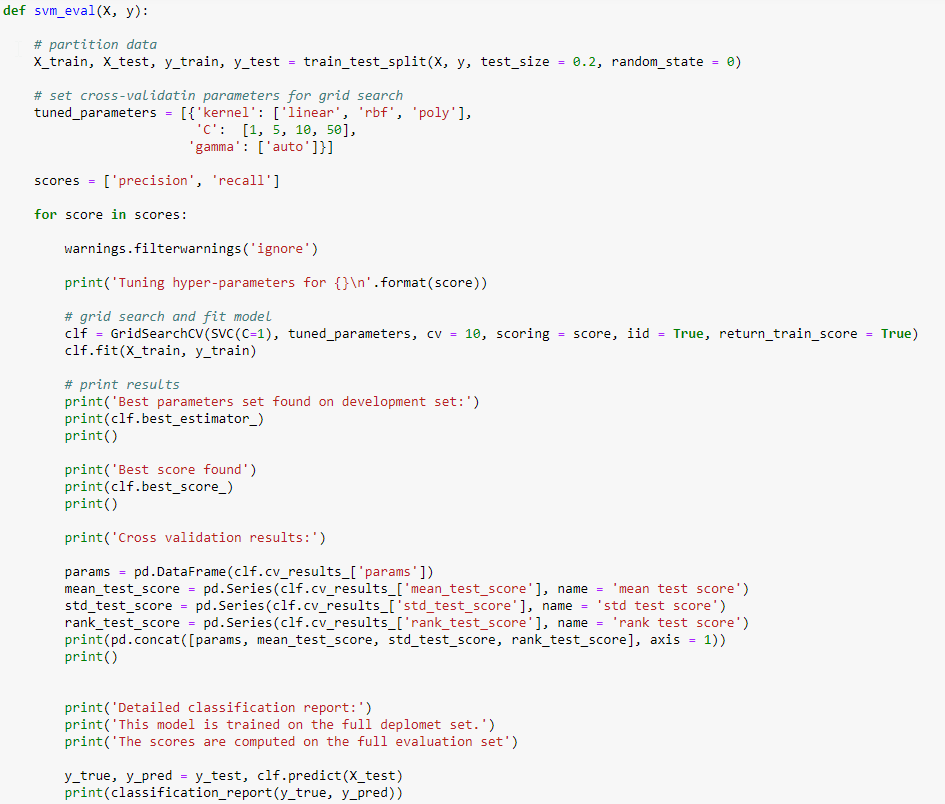


*Figure 4.3 Performance visualization of different SVM kernels*

The “rbf” stands for radial basis function. In total, the three kernels used (linear, poly, and rbf) define how the support vectors are drawn. Typically, linear support vector machines work best to text analytics; however, each should be run.

The second parameter that will be tuned is the penalty. The penalty defines how to handle misclassified items. Setting the penalty too high will result in overfitting the model to a specific data set. Setting it too can result in underfitting.

Lastly, this function needs to evaluate all the combinations of the parameters provided. First, cross validation is used with 10 folds. The evaluation criteria will be both precision and recall. The overall function to evaluate the data using SVM is shown below.



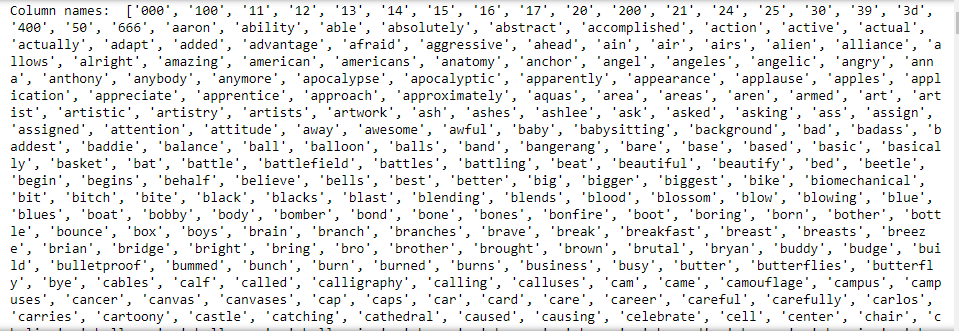
*Figure 4.4 Code snippet for SVM evaluation*

When this function is called, it will perform the cross validation for each combination of tunable parameters and generate a score. This will be done twice, once for precision and once for recall.

## Multinomial Naïve Bayes

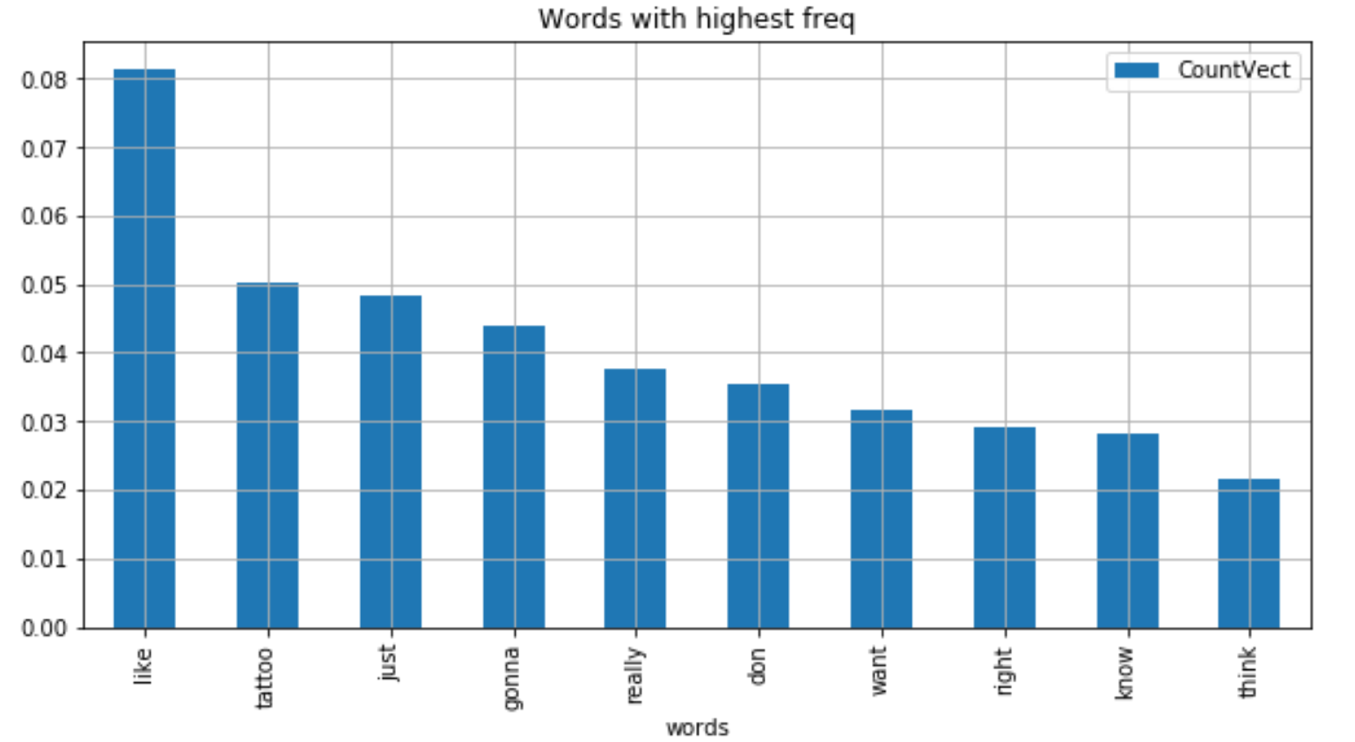
Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values. It is called naive Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value, they are assumed to be conditionally independent given the target value. This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

Using the same data preparation steps as the one outlined in the Support Vector Machine analysis, the Multinomial Naïve Bayes model was prepared. The text data under the Caption column was vectorized using the Count Vectorizer function by removing the English stop words and converting all words into lower case. The resulting vector list was transformed into an array and stored as a data frame with this array of frequencies and words. Fig 5.1 shows the list of words generated from the captions using the vectorization function.

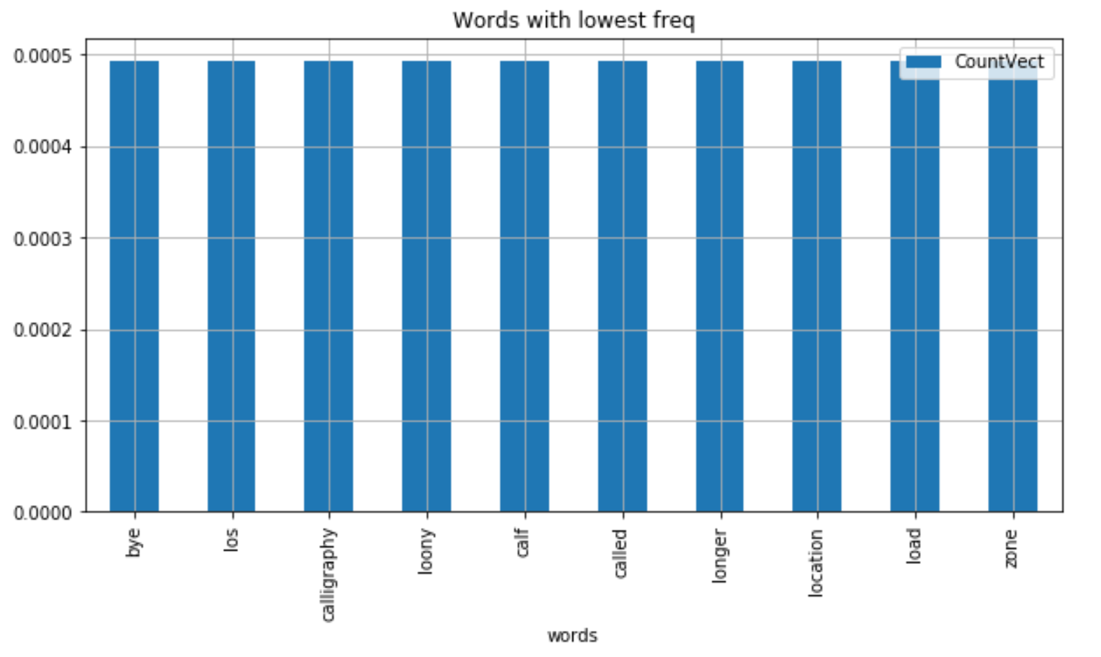


*Figure 5.1 Vectorization result*

The vectorized words were explored (Fig 5.2 & 5.3) to see the words with the highest and lowest frequency in the dataset. Fig 5.2 shows that the words 'like’, ‘tattoo’, 'just’ and ‘gonna’ are some of the most common words in the corpus. Fig 5.3 shows words ‘los’, ’calligraphy’, ’called’ and ‘longer’ are some of the least common words in the captions.

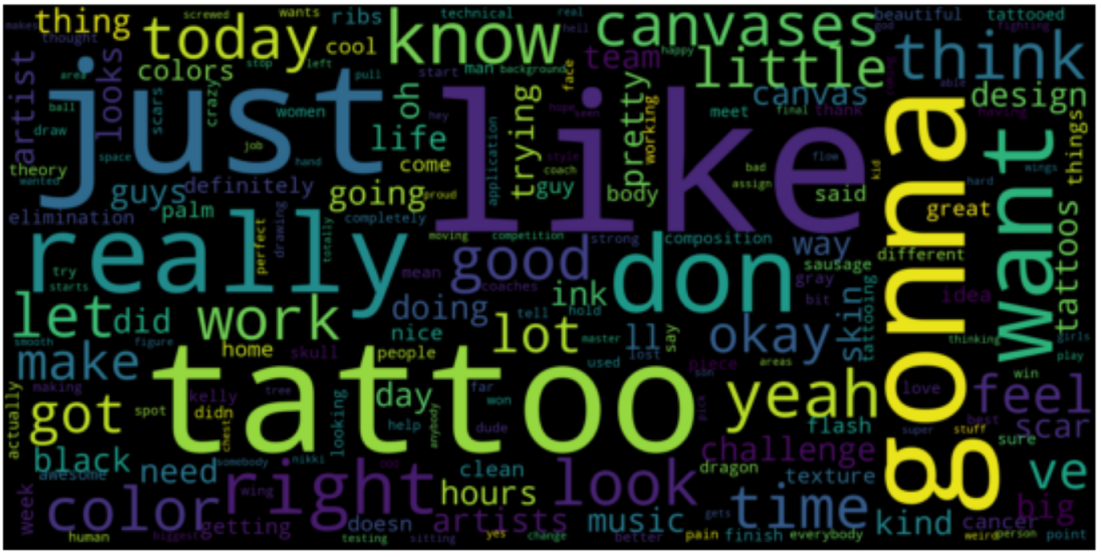


*Figure 5.2 Most common words*



*Figure 5.3 Least common words*

To make the visualization more appealing, a word cloud was generated (Fig 5.4) to see the most and least common words in the data frame.



*Figure 5.3 Word Cloud*

Multinomial naive Bayes function from the library sklearn package is used by holding the label which in this case is the column retention label as the Y variable and use all other words in the training data to create a model. This model is then used to predict the labels of the test data. The results are then compared with the labels of the test data that we have stored in a separate variable. The original cleaned and vectorized dataset was split into training and test set with 70-30 as the ratio for the split.

## Bernoulli Naïve Bayes

The Bernoulli naive Bayes classifier assumes that all our features are binary such that they take only two values (e.g. a nominal categorical feature that has been one-hot encoded). In the multivariate Bernoulli event model, features are independent Booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence features are used rather than term frequencies.

Bernoulli naive Bayes function from the library sklearn package is used by holding the label which in this case is the column retention label as the Y variable and use all other words in the training data to create a model. This model is then used to predict the labels of the test data. The results are then compared with the labels of the test data that we have stored in a separate variable. The original cleaned and vectorized dataset was split into training and test set with 70-30 as the ratio for the split.

The Count Vectorization function is used here to vectorize the reviews. Out of the different parameters that can be used, the first model was generated by only removing the stop words. A second model was developed by removing stop words and using lemmatization. A third model was generated with the above two parameters and removing patterns off the data. These three models were applied to the labelled dataset.

To make the visualization more appealing, a word cloud was generated to see the most and least common words in the data frame after lemmatization. The word cloud shows words like ‘tattoo’, ‘like’ are even more prominent than they are from the Multinomial Naïve Bayes model.

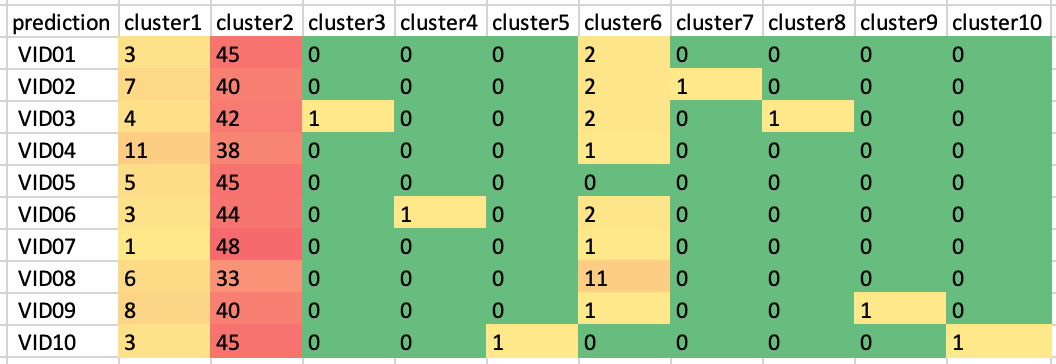


# Results

Results from the various models are analyzed and compared to see which model performed better. Clustering, Multinomial Naïve Bayes, Support Vector Machine results are detailed in this section.

## Cluster Results

After running sklearn’s K-means clustering algorithm on the comments dataset, the following figure displays how the comments from each of the 10 videos were clustered:

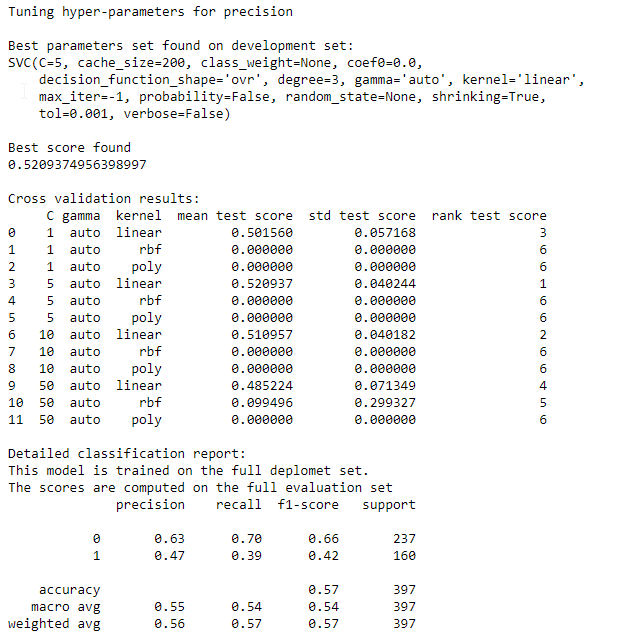


Because all of the videos center around the same television show with the same premise (tattoo challenges), most of the comments’ contents contained similar tokens, resulting in many of the comments being grouped together in cluster2. The other two clusters with higher concentrations were cluster1 and cluster6. Upon further evaluation, these clusters also centered around topics consistent across multiple episodes of *Ink Master*, namely male vs. female challenges (in cluster1) and critical tattooed clients (in cluster6).

The remaining clusters were differentiated by video-specific content, most likely when viewers praised specific tattoos featured in the video.

## Support Vector Machines Results

After running the “svm\_eval” function, a report is generated. Below is a sample of the report. This sample is for the unigram vectors, and the results are tuned for the precision metric. A similar one is produced for recall, which is not shown.



The report provides quit a few key items. First it displays the best parameters for the “SVC” method. It also displays the best precision score found. It then shows results for the 12 combinations of tunable parameters and how they ranked. Lastly, it provides a summary of the precision, recall, F-score, support, and accuracy.

Reviewing the for iterations, the best results for the four are shown in the table below. Note the (L) is for lower and (H) is for higher. These are the two labels for the “relativeRetention”.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Kernel | Penalty | Optimized for | Best Score | Accuracy | Precision (L) | Recall (L) | F-Score (L) | Precision (H) | Recall (H) | F1-Score (H) |
| Linear | 5 | Precision | 0.521 | 0.57 | 0.63 | 0.70 | 0.66 | 0.47 | 0.39 | 0.42 |
| Linear | 50 | Recall | 0.456 | 0.58 | 0.64 | 0.66 | 0.65 | 0.47 | 0.45 | 0.46 |
| Linear | 10 | Precision | 0.500 | 0.56 | 0.62 | 0.47 | 0.65 | 0.45 | 0.40 | 0.42 |
| Linear | 50 | Recall | 0.428 | 0.55 | 0.62 | 0.63 | 0.62 | 0.44 | 0.43 | 0.44 |

Overall, the linear kernel with a penalty of 5 and optimized for precision performed the best. This one performed the best in most of the runs. While some others are close, they do not justify the higher penalty, as this could contribute to overfitting. As expected, the linear kernel was the best of all the runs. However, the accuracy was not very high. The best performing model had an accuracy of 52.1%, which is basically a 50/50 guess.

Overall, the SVM did not provide the best results. One modification would be to include additional videos. While there were 12,113 comments overall, that was from 10 videos. A larger sample of videos may provide better results. Also, including the non-verbal tokens may be beneficial. Perhaps certain audio cues are present during potential rewind moments.

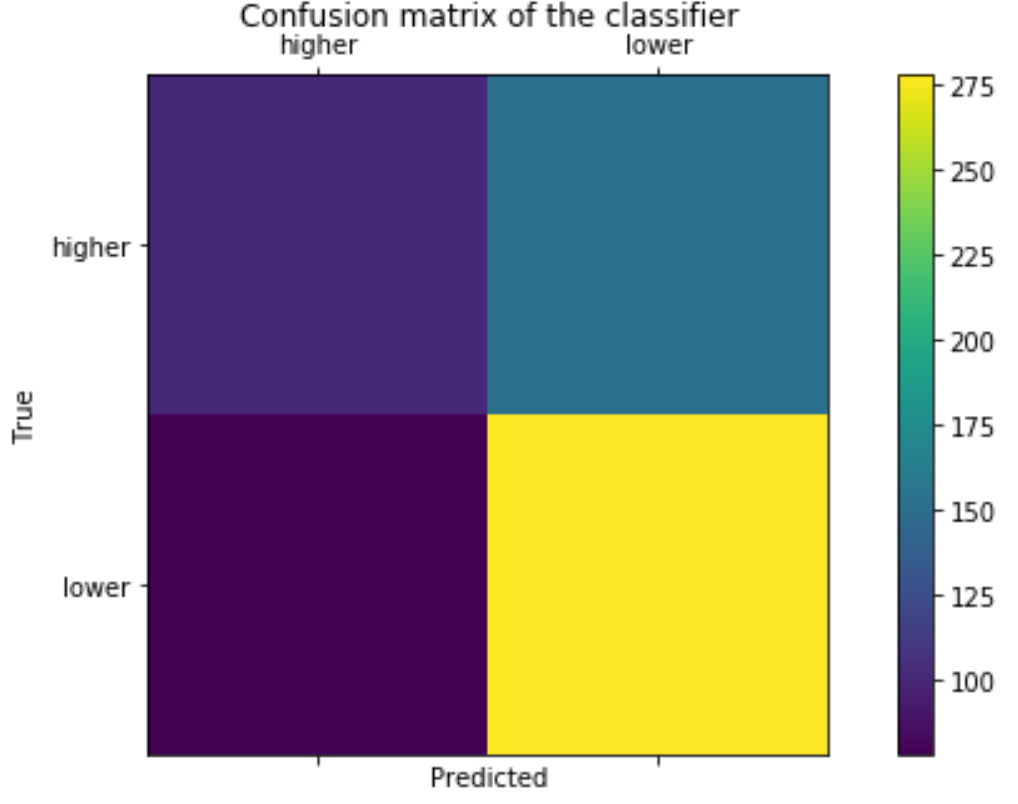
## Multinomial Naïve Bayes Results

A confusion matrix was generated on the results to see how the model performed.

The confusion matrix is:

[[100 152]

[ 78 278]]



*Figure 5.4 Confusion Matrix*

Fig 5.4 shows the number of records that the model correctly and incorrectly classified as a higher or lower retention rate. The model was able to predict the records with lower retention rate with a high accuracy, but it was unable to predict the records with higher retention rate with low accuracy.

The accuracy score of the NB model on predicting the retention label that can detect the higher and lower retention rate words was only 62%. After applying cross validation methods on the dataset with a 6-fold cross validation, the results did not change much. In fact, the accuracy score dropped to 56 from 62% which might be more accurate than the results obtained without cross-validation.

Better tuning of parameters and using cross-validation methods will improve the results of lie detection.

Bernoulli Naïve Bayes Results

For Bernoulli, the accuracy score of the model on the labelled dataset was 59% whereas the 10-fold cross validation score went down to 58%. Subsequent models 2 and 3 (with lemmatization and token pattern removal) yielded better results. The accuracy scores of Model 2 and 3 were better with 61 % accuracy using the 70-30 train test split and went down to 57% with a 10 fold cross validation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Bernoulli** | | | |
|
|  | Model 1 | Model 2 | Model 3 |
| Accuracy Score | 59% | 61% | 61% |
| CV Score - 10 fold | 58% | 57% | 56% |

Figure 13.1 Bernoulli Sentiment Dataset results

So far, comparing the two algorithms on 3 different models, the Bernoulli algorithm fared better on the labelled dataset than the multinomial NB algorithm. And, in terms of which training split model worked, the 70-30 split worked the best to give the best accuracy rate of 61%. Overall, there could be some overfitting of data since the cross validation is yielding lower results than the 70-30 split method. Hence, cleaning the data further by fine tuning the parameters could yield to better results.

# Conclusions

Identifying potential rewind moments from text alone can be a challenge. From the various models used, each showed under 65% accuracy in predicting relative retention, with a few performing just as good as a random guess. One thing that can be understood from this is that the text alone may not indicate a rewind moment. Usually a combination of visual and audio content will entice a viewer to re-watch a specific segment. While it may not be impossible to predict solely off the text, it is much more challenging without seeing what is happening. Many rewind moments have just background music that may be an indication of an event a viewer would want to re-watch.

If analyzing comments for relative retention or rewind moments is to be accomplished only with text, then a much larger labeled sample would be needed. The results were most influenced by the fact that the comments all came from videos from not only the same topic, but compilations from the same television show with the same episode format. Ideally, a much larger sample size would be needed to use captions to measure audience retention.

Lastly, it may be necessary look other types of videos. Like the previous point of a larger sample, a much broader analysis may be needed to see if there are key elements or words that are spoken during segments that are likely to be re-watched or at least have a higher relative retention.

# References

“1.4. Support Vector Machines”, **scikit learn**, available from https://scikit-learn.org/stable/modules/svm.html. Date accessed 8 September 2019.