YouTube Video Categorization

Nikita Amartya
Dept. of Computer Science, NYU
New York
nn899@nyu.edu

Vincent Chabot
Center for Data Science, NYU
New York
vec2241@nyu.edu

Urjitkumar Patel
Center for Data Science, NYU
New York
up276@nyu.edu

Rohit Shankar
Dept. of Computer Science, NYU
New York
rs4737@nyu.edu

DS-GA-1001/GB.3336.11
Fall 2015

ABSTRACT

Predicting the popularity of content on the Web is an important task for supporting the design and evaluation of a wide range of systems, from targeted advertising to effective search and recommendation services. In the case of YouTube videos, correct categorization of a video can play a significant role in gaining popularity. In this project we have classified YouTube videos into different categories using meta-information of YouTube videos. Using text analysis models and decision tree classifiers enabled us to be able to predict the category of a YouTube video with ~0.67 accuracy, based on the analysis done using metadata available for some random 240k videos.

Keywords *– YouTube; Feature Engineering; Text Analysis; Classification Problem; Supervised Learning*

INTRODUCTION

The advent and increasing popularity of Web 2.0 applications brought with it an enormous and ever-growing amount of user generated content. YouTube video sharing system is one such platform and it often figures among the top 3 most popular applications on the Web. It has been reported that YouTube has over a billion users–almost a third of all people on the Internet– every day and YouTube users upload more than 72 hours of video per minute[1]. The total amount of content uploaded to YouTube in 60 days is equivalent to all content that would have been broadcasted for 60 years, without interruption, by NBC, CBS and ABC altogether [2].

Given such a staggering content upload rate, it is unsurprising that the distribution of popularity across different contents on YouTube, as well as on other Web applications is very uneven: most content garners very little attention whereas a small amount of it attracts millions of views [3]. There have been efforts towards building models to predict the popularity of online content using various techniques such as reservoir computing, stochastic models of user behavior and biology-inspired survival analysis techniques [4]. It has been studied and proven that around 80% of the videos watched on a day are older than one month, although the most popular video tends to be one that has been recently uploaded.

BUSINESS UNDERSTANDING

In this context, predicting content popularity is of great Constant waves of new videos and the convenience of the Web are quickly personalizing

the viewing experience, leading to a great variability in user behavior and attention span. In this context, predicting content popularity is of great importance to support and drive the design and management of various services. For example, in online marketing, the information about the expected popularity of a certain type of content can be useful for planning advertising campaigns and estimating costs. Accurately predicting content popularity is also key to support effective information services (e.g., recommendation and searching services)[5]. Category prediction can aid recommendation system and search engines and improving the quality of such services by extending current result ranking strategies. Therefore, it is very important for a service such as YouTube to know to which category each video on its platform belongs.

However, currently YouTube ask to choose a category while uploading a video. It has two major drawbacks:

- The more information a user has to describe when performing an action (here uploading a video), the longer the process to perform that action. It is an additional overhead and hence it is less likely that an average user will do it on a large scale. Yet, content is crucial for a content provider like YouTube for making the service better. In order to maintain an edge against competition, it is important to simplify the process of upload/posting content on the service. Predicting video category based on its title and description for example could enable YouTube to perform category autosuggestion when uploading a video. The user can change it if he feels the need or select another from the suggested categories.
- Also, when a user uploads content to YouTube, there are non-negligible proportions of videos that may be misclassified by the user (mistakenly or because user didn't care and select any random category). Now, think about the auto-playlist option YouTube offers, which we all love. While playing one video, YouTube queues up the next video for you in autosuggestion e.g., it would be a bad user experience to have a documentary starting to play right after a music video. Classification is indeed very crucial for recommendation and personalization. Therefore, based on all the metadata of the video, it could be possible to reclassify all the misclassified videos.

DATA UNDERSTANDING

We successfully obtained the data of random Youtube videos (239225 records) in csv format from a former kaggle competition and it has following columns:

video_category_id, title, description, published_at, viewCount, likeCount, dislikeCount, favoriteCount, commentCount, duration, dimension, definition, caption, licensedContent, topicIds, relevantTopicIds

Our goal was to build a model in order to predict the "video_category_id" based on the values of the other features. The column "video_category_id" has 15 distinct categories, which are:

ID	Category Name
1	Film & Animation
2	Autos & Vehicles
10	Music
15	Pets & Animals
17	Sports
19	Travel & Events
20	Gaming
22	People & Blogs
23	Comedy
24	Entertainment
25	News & Politics
26	Howto & Style
27	Education
28	Science & Technology
29	Nonprofits & Activism

Table 1: YouTube video categories

Now that we had a well defined target variable, the first observation we can make is that any random guess to predict the video category has a 1/15

probability of being correct. So, the "no-model" probability baseline is 1/15 (~ 0.067).

We can also make some further observations about the features (for more details, please refer to Data Preparation section):

- All the features that are not in a numeric format cannot be used as they are and need to be transformed.
- Some features seems to be conveying more than one information and therefore should be split into multiple features.
- The two text features are really different in terms of relevance. In most of the instances of data, 'title' seemed to be most appropriate property describing the video. Even 'description' is not related to the video in some instances (e.g., link to Facebook page) or not existing at all. However, 'description' contains on average much more words than 'title' and therefore could be providing more information on such cases.

Finally, we plotted the correlation matrix to check the 2 following things:

1. The features that were correlated to our target variable
2. The features that were correlated together. Indeed, it means that if two features are highly correlated, they are conveying the same type of information, and we can therefore remove 1 of the 2 in the case of our model is taking too long to run and we want to speed it up.

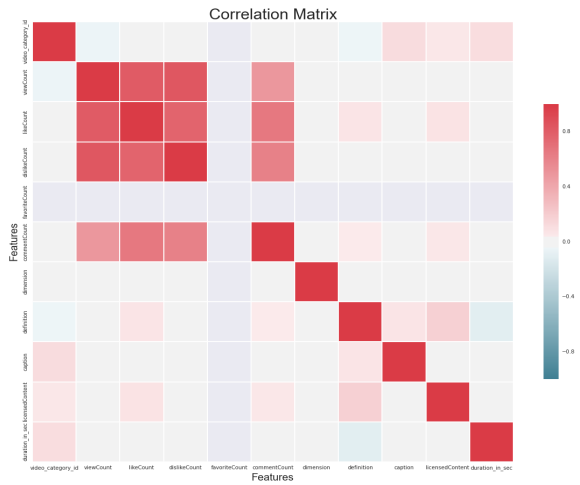

Fig. 1: Correlation Matrix (non numeric features)

DATA PREPARATION

Data preparation was an important part of our work since we had many variables we could not exploit in their original format (such as text data for instance). Also, we had the intuition that some features were containing a lot of relevant information within that one features so we thought we should break these into different new features (e.g., the *published_at* feature was containing a lot of information, such as if the video was published during the week or the weekend, the day or the night, the age of the video)

After analyzing all the features, we observed we needed to work with following catagories of predictor features (independent variables):

- **Text data:** title, description
- **Numeric Data:** viewCount, likeCount, dislikeCount, favoriteCount, commentCount
- **Categorical Data:** dimension, definition, caption, licensedContent
- **Other types:** published_at(date-time), duration, topicIds relevant, TopicIds

See appendix for details on each column.

In order to be able to use different types of values in the meta-information we applied 4 major feature-engineering steps.

a) Text Data Preparation

The text features 'title' and 'description' posed a challenge. Out of 72 MB of data that we were to analyze, almost 18 MB is in languages other than English that use non-ASCII character sets. We needed to translate it to English. We tried ***goslate*** library in python (from google), ***TextBlob*** and **py-translate** libraries and all of them rely on the Google Translate API. After successful attempt on one row translation, we tried to translate 100 rows of data using the script but Google blocked our IP address for using this script for multiple lines of data. After many unsuccessful efforts on translating the data, we decided to write a script to filter out rows containing non-ASCII characters and to work only on ASCII characters data. Our guess is that if

we had been working for a real business situation, we would have had the budget to purchase Google's service in order to explore this option in our model.

b) Categorical Data Preparation

For Categorical Data, we converted below categorical features into binary 0's and 1's.

- dimension: 2d → 0 , 3d → 1
- definition: sd → 0, hd → 1
- caption: true → 1, false → 0
- licensedContent : true → 1, false → 0

c) Other Types Data Preparation

We extracted and transformed two columns in our dataset to gain more details to get more insight about the impact of these features on the target variable.

duration:
The *duration* column was originally in the ISO 8601 format. Value in this column reads as *PT1H8M2S,* which refers to a duration of 1 hour, 8 minutes and 2 sec. Since the duration in such a format is not usable for our purpose, we extracted individual values and then we represented it into only seconds. So, the duration column was transformed into *duration_in_sec* column.

favoriteCount	commentCount	duration	duration_in_sec	dimension	definition	caption	licensedContent
0	26	PT8M2S	482	2d	hd	False	True
0	7	PT2M11S	131	2d	sd	False	False
0	7	PT2M24S	144	2d	hd	False	True
0	193	PT4M33S	273	2d	hd	False	True
0	11	PT5M6S	306	2d	hd	False	True

Fig. 2 DataFrame after transformation

***'published_at'*:**
The feature *'published_at'* was originally in the following format: 2014-11-15T17:00:06.000Z. We extracted the date and time from this format and removed unnecessary values. We converted the format from string to the python datetime format. We did this because the *datetime* has some useful properties like extracting month, day, weekday (boolean value) etc. From this base '*published_at'* feature few extra features can be generated which could have potentially improved the accuracy of the model.
We created additional binary features from *published_at* such as: *winter, summer, spring, fall, is_month_start, is_month_end, late_night, early_morning, day, night, age*. Using these features, we tried to find if these hidden features have any relation with the category of video.
See fig 1 in Appendix for more details.

***'topicids' and 'topicids_relevant'* :-**
The topics from these topics can be retrieved using a Freebase API that had to be integrated with our project code. This was causing few issues as there were a few requisites for the integration and hence these columns were dropped. These features can be used in the future expansion of the project.

d) Removing irrelevant features

After having prepared all these data for modeling, our final step was to drop the features that are not adding any information for our target variable prediction. In order to do that, we computed and plotted the graph showing feature importance of all the features using the basic decision tree. After analyzing the graph we removed those features (*summer, winter, spring, late_night, is_month_end, is_month_start, fall*) which had 0 importance. Results and improvements are discussed in further sections. See figure 6 for reference.

To conclude, we applied some feature engineering on every type of data we have except for numeric data that was clean and ready to mine.

MODELING

Baseline model
First step in our development cycle was to develop a simple baseline model. After some basic initial feature engineering, we decided that the best baseline model we could build quickly to get a first idea without losing too much time in feature

engineering was to train classifiers on the data that was already almost ready to use.

Hence, the first model we trained was a Decision Tree classifier using the features except text data, *duration* and *published_at* (to-be-transformed) features. This allowed us to calculate the importance of the features we used and hence to drop the irrelevant ones. The first accuracy we obtained was as:

Classifier	Decision Tree
Accuracy on train df	1.000
Accuracy on test df	0.174

Table 2: Accuracy with baseline model

We did not set any parameters to obtain this result. As we can see, accuracy on train data is 1 and it is a clear sign of '**overfitting**'. Due to overfitting, we had a poor accuracy of around 0.17 (which is still better than a random guess). Then we played around with hyper parameters to improve this accuracy. We tried to find the best min_samples_split value and best min_samples_leaf value for which we get the best accuracy with the same decision tree classification model. We built the model for all possible combination of different "min_samples_split" and "min_samples_leaf" on a selected value sets and compared the accuracy to get the best configuration

min_samples_split_values :
2,16,32,64,128,256,512,1024,2048,6000
min_samples_leaf_values :
1,16,32,64128,256,512,1024,2048,6000

Below are few plots for accuracies we got for different min_samples_split values and min_samples_leaf values.

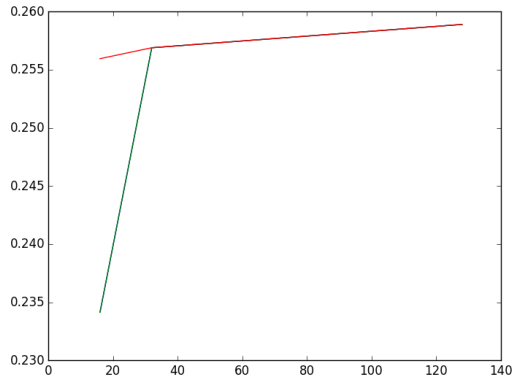

Fig 3.
plot of min_samples_leaf value = 64
x-axis min_samples_split values (20,40,60,80,100,120,140)
Y axis - accuracy for the combination of min_samples_leaf and min_samples_split

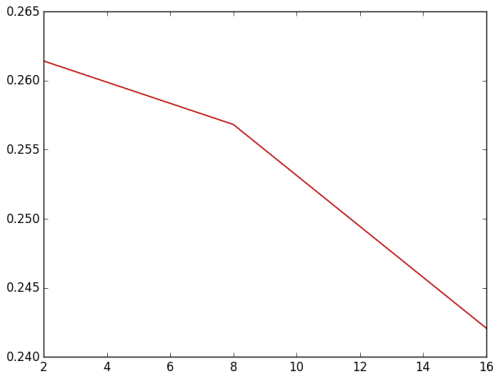

Fig. 4
plot of min_samples_leaf value = 8
x axis min_samples_split values (2,4,6,8,10,12,14,16)
Y-axis - accuracy for the combination of min_samples_leaf and min_samples_split

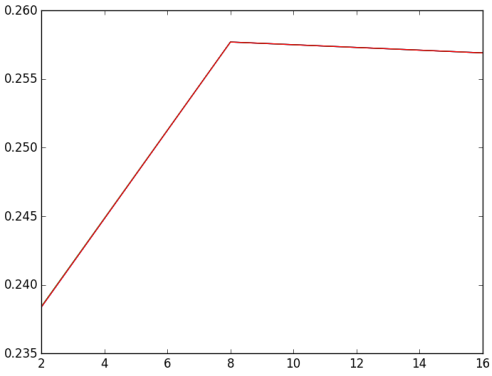

Fig. 5
plot of min_samples_leaf value = 64
x axis min_samples_split values (2,4,6,8,10,12,14,16)
Y axis - accuracy for the combination of min_samples_leaf and min_samples_split

We got the best accuracy for following configuration:

min_samples_split	**2**
min_samples_leaf	**128**
Accuracy	**0.261427526387**

Table 3. Config for best accuracy in DT

To compare different classifiers, we also applied Naive Bayes and Logistic Regression on numeric data. We got a poor accuracy ~ **0.10**.

Hence, we can conclude two things:

1) Our baseline model has an accuracy of 0.26
2) The classifier that performs the best on numeric features is the decision tree classifier.

Therefore, this is the model we continued to use for analysis of numeric features.

After analyzing the feature importances at this stage, clearly "favoriteCount" has 0 importance, so we dropped it from our dataset. The most important numeric features are '*viewCount*', '*likeCount*', *'dislikeCount'*, *'commentCount'*.

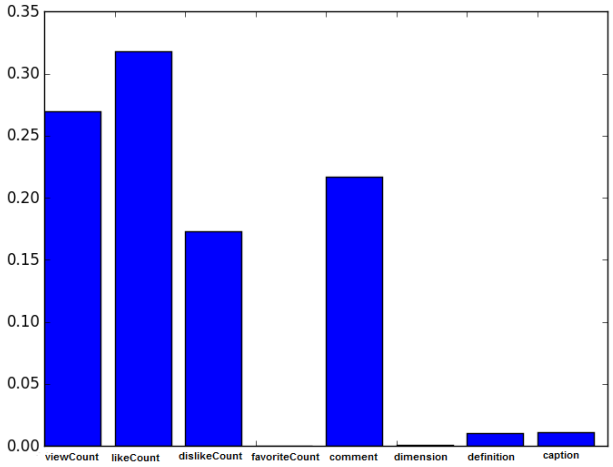

Fig. 6 Feature Importance of numeric columns

At this stage we were able to achieve accuracy ~ 0.27 by analysing a part of numeric data alone. Other columns of data, namely *published_at* and *duration* were yet to be explored. The most important features for our prediction model are the text features: *'title'* and *'description'*. One thing that could really boost our decision tree was if we could somehow convert this other type of data into numeric or binary type.

From this point we started working on:

- Text Analysis
- Decision Tree Enhancement using the other type of data

Text Analysis

As described in the Data Preparation section above, we started by doing some feature engineering on the text data to convert the text into sparse matrix. We also saw in Data Understanding section that 'title' and 'description' features seamed to be conveying very different information, so we decided to train our models on these features separately

After performing feature engineering on text data we had different vectorized form of data for three different techniques: Count-Vectorizer, Binary Count-Vectorizer and TF-IDF. We built the Bernoulli Naive Bayes model on vectorized 'title' and 'description' data separately.

Results we got with different types of vectorized forms are as following:

BernoulliNB() with Count Vectorizer:

Accuracy with only 'title' data	0.5606
Accuracy with only 'description'data	0.5748

BernoulliNB() with Binary Count Vectorizer:

Accuracy with only 'title' data	0.5606
Accuracy with only 'description' data	0.5748

BernoulliNB() with TF-IDF :

Accuracy with only 'title' data	0.5606
Accuracy with only 'description' data	0.5748

Here we also tried to get better accuracy by setting different hyper parametes like stop_words = 'english' and ngram=(1,2), but we did not get any increment in accuracy. We decided not to increase the complexity of the model and rather use the model without any hyper parameters. We decided to move with BernoulliNB() with TF-IDF. Now we had two different prediction columns for our single target variable, one each from 'title' and 'description'. At the end of the modeling section we have described how we got our final prediction using these.

Decision Tree Enhancement using the other type of data

After performing the feature engineering on features '*duration'* and '*published_at'*, we had some new features in our data set that we have already discussed in data preparation section.

So now on this data we built the model using decision tree classifier and the accuracy of the model improved as shown below (as a reminder, accuracy without these new features was of 0.26 on our test set):

Classifier	Decision Tree
Accuracy on train df	0.312
Accuracy on test df	0.287

Table 4: Decision Tree on non-text data

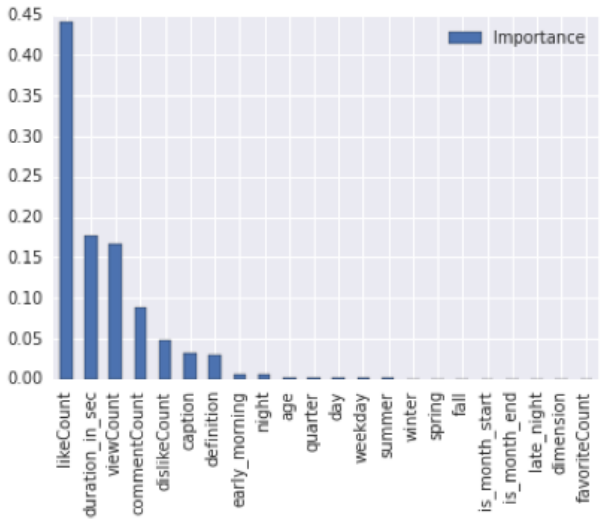

Fig. 7: Features importance (excluding text features)

After having obtained this new accuracy of 0.287 thanks to our feature engineering work on duration and *published_at* features, we wondered if we could improve even more our prediction using all the non text features we had on our videos. We tried one last thing we had heard of, which was Random Forest. We had understood that Random Forest was basically improved Decision Tree, and since our best accuracy was obtained thanks to Decision Tree, we thought this should improve our model. Indeed, we trained Random Forest and obtained the following results:

Classifier	Random Forest
Accuracy on train df	0.327
Accuracy on test df	0.305

Table 5: Random Forest on non-text data

Now we had three different prediction columns for the same one target: one prediction from 'title' feature, one prediction from 'description' feature, and one prediction from all the other features combined. The question that we initially had was how would we know which prediction out of three prediction is correct. We used different classifiers for different type of data during our training and now we had to find a way to merge the result from

different models into one to get the final result. Professor Dalessandro thankfully guided us at this critical point. The solution was 'Ensemble Modeling'. "Ensemble modeling uses multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms. This type of modeling tends to yield better results when there is a significant diversity among the models." [6]

In our case we can think of the Ensemble Model where we can use different classifiers which we used till now and can predict based on the results we got. We deployed the model on test data to get the prediction on the entire data. We had a set of 3 prediction results based on:

a. title
b. description
c. numeric data

We combined these predictions as independent variables into a single pandas dataframe along with *video_category_id* as the target variable. Lastly, we tried the Random Forest classifier. We had learnt that Random Forest was basically an improved Decision Tree, and since our best accuracy was obtained with a Decision Tree, we thought this should improve our model. Indeed, we trained Random Forest and obtained the results as shown below

Accuracy with Decision Tree	**0.669**
Accuracy with Random Forest	**0.667**

Table 6: Final results

In conclusion, we were able to develop a final model which is giving us around 0.67 accuracy on the testing data frame for the target feature *video_category_id* having 15 different classes.

Overall illustration of our Project:

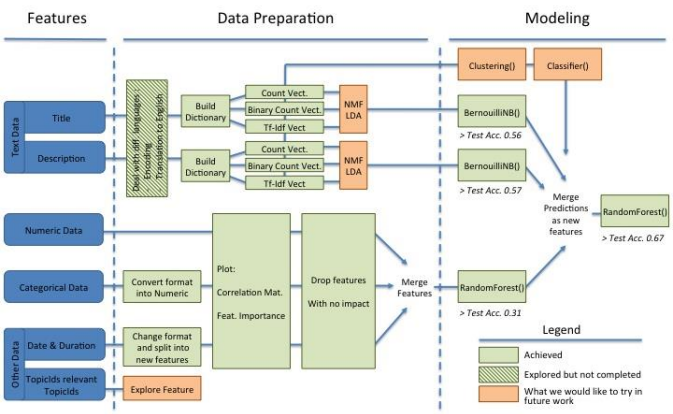

Fig. 8: Overall Illustration of the Project
Large image of this illustration is at the end in the appendix for the reference

EVALUATION

We chose to use **accuracy** as our evaluation metrics (instead of AUC). In the Business Understanding section we mentioned that our first aim was in 'predicting video category based on its title and description which could enable YouTube to perform category auto-suggestion when uploading a video.' In this instance, False-Positives would not be of high cost since it would just require the user to select the right category. Our second aim was that 'based on all the metadata of the video, it could be possible to reclassify all the misclassified videos.' For this particular purpose, it could be a good idea to minimize the false positives and reclassify a video only if we are very confident it was misclassified.

For these above two reasons we felt that accuracy would be the correct evaluation metric to follow.

DEPLOYMENT

When a user uploads a video he/she wants maximum exposure. Often times selecting the right category can go a long way in doing that. Which category a video belongs to, decides how accessible the video is in a search result or in the 'similar videos' section of a video For example if a user has a video entitled 'How to play Tennis', then this

video could either go in the 'How to & Style' category or in the 'Sports'. If our model can be adopted by the content provider (for example Youtube) then we can help the user make the right decision in terms of category selection

With such a model, if we were to improve even more our accuracy, the content service provider could be tempted to simply stop asking the user to validate the video category when uploading it (since it still requires a potential action from him and add complexity). But we recommend, at least in a first development phase, to keep asking for user validation. Indeed, this would act as a safety and validation of the model. For instance if, for some reason, our model was not acting as expected, and no user confirmation was asked, then more and more videos would start to be misclassified. And if we start updating the model with the video that has been (mis)classified by the model itself, then it would lead to a complete deregulation of the classification system. So we should be very careful with that. Hence, in the first phase, we should keep asking user for validation in order to validate our model (and improve it over time.)
Once the model is running and we are extremely confident of its accuracy, we can stop asking user confirmation.

Since all the data obtained is openly available (via the Youtube API), there is no issue of any ethical considerations in that respect. The API data contains only metadata and no personal information of a user is present.

FUTURE WORK

The topics from the columns topicIds and relevantTopicIds could not be used in our project. To extract the topics we needed to use the Freebase API which had to be integrated with our project code. This was causing few issues as there were a few requisites for the integration. With proper integration and extraction these features can be used in the future expansion of the project.

After repeated attempts we were unable to translate videos having non ASCII character set in their title and description. If this project was to be deployed in a real world scenario then we would have the financial budget to purchase Google's translation service in order to explore this option in our model

We would have liked to try to implement NMF / LDA on the text data during feature engineering since we heard that it could improve our accuracy whenever we have text features.

In the project we focused on running classifiers on our text features. But we had an idea which consisted of running a clustering model such as K-mean, then identify to which category each cluster was belonging (i.e. being the closest to), and then identify to which cluster each new input was belonging to and attribute video category that way. This is also something that can be done in the future to improve the model

ACKNOWLEDGMENT

We would like to thank Professor Dalessandro for his guidance and timely feedback. We got stuck in the course of development a few times where his feedback put us on the right track to progress. We would also like to thank Andreas Mueller for guiding us on grid search and cross validation.

CONTRIBUTIONS

Vincent Chabot: Feature Engineering on Text Data (Cleaning and Vectorizing Text Data), Text Data Analysis (classifiers, GridSearchCV, Attempt on clustering, Documentation

Urjitkumar Patel: Analysis of Numeric Data, Baseline Model creation, Feature Engineering for Numeric Data, Developing best model with the best hyper parameter configuration for numeric part, Documentation

Nikita Amartya: Attempt to translate non-English data, cleaning of non-ASCII characters,
Understanding and application of GridSearchCV for best performance, Code merge, final decision tree and random forest analysis, Documentation.

Rohit Shankar: Feature engineering involving transformation of ‘duration’ and ‘published’ into many new and more usable features. Attempted implementing NMF and LDA, Documentation

REFERENCES

[1] http://www.youtube.com/t/press_statistics
[2]http://www.nytimes.com/2009/09/06/magazine/06FOB-medium-t.html
[3]http://conferences.sigcomm.org/imc/2007/papers/imc131.pdf
[4] J. Lee, S. Moon, and K. Salamatian. An approach to model and predict the popularity of online contents with explanatory factors. In Int'l. Conf. on Web Intelligence and Intelligent Agent Technology, 2010.
[5] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes
[6] https://en.wikipedia.org/wiki/Ensemble_learning

APPENDIX

Description of columns

title: Text data
description: Text description of the video as posted by uploader.
viewCount: Number of views.
likeCount: Number of likes the video received.
dislikeCount: Number of dislikes the video received.
favoriteCount: Number of times the video has been marked favorite.
commentCount: Number of comments the video has received.
duration: Duration of video in ISO format e.g., PT2M24S.

Description of Extracted features:

age: The age of the data i.e. number of days it has been uploaded.
weekday: The column has value 1 if it video was published on a weekday and 0 if on a weekend.
quarter: If the video was published in the first quarter of the year (Jan - March) it has a value of 1. 2 for the second quarter and 3 and 4 for the third and fourth quarter respectively.
winter: The column has binary values. If the video was published in the winter it had a value of 1 else 0
summer: If video was published in the summer it has value 1, else 0.
spring: If video was published in the spring it has value 1, else 0.
fall: If video was published in the fall it has value 1, else 0.
is_month_start: The column has binary values, 1 if the video was published at the start of the month, 0 if not.
is_month_end: The column has binary values, 1 if the video was published at the end of the month, 0 if not.
late_night: The column has binary values. The value is 1 if the video was published between midnight and 6am, else 0.
early_morning: The column has binary values. The value is 1 if the video was published between 6 am and midday, else 0.
day: The column has binary values. The value is 1 if the video was published between midday and 6pm, else 0.
night: The column has binary values. The value is 1 if the video was published between 6pm and midnight, else 0.

	video_category_id	title	description	published_at	viewCount	likeCount	dislikeCount
86624	27	How to Draw a Water Drop With Colors	Visit me on FB: https://www.facebook.com/Leona...	2013-06-25T15:01:00.000Z	318850	7011	88

commentCount	duration	definition	...	spring	fall	is_month_start	is_month_end	late_night	early_morning	day	night	new_date	age
718	00	1	...	0	0	0	0	0	0	1	0	897 days 08:59:00	389

Figure 1: Instance of dataframe after feature engineering

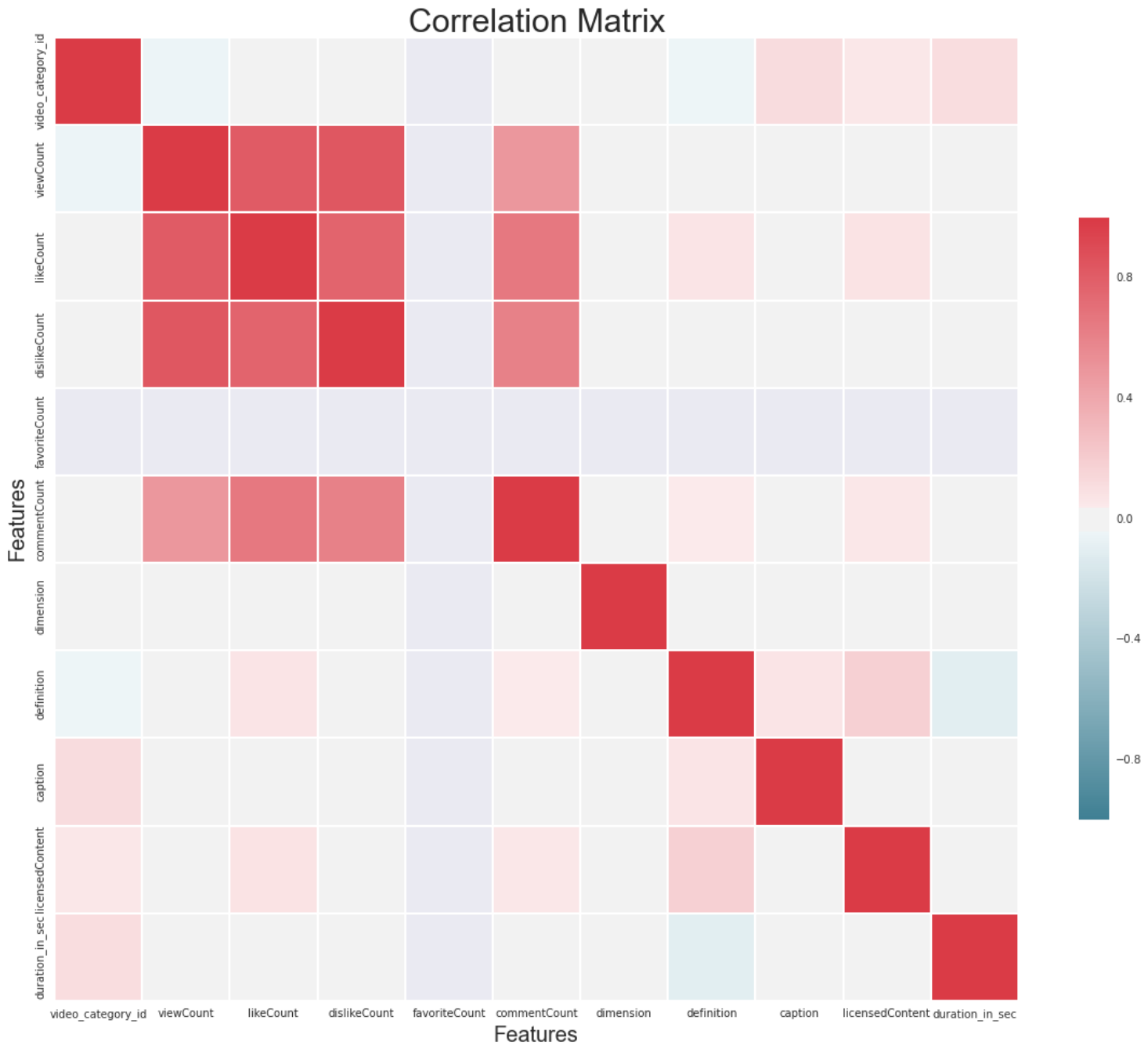

Figure 2: Enlarged image of correlation matrix

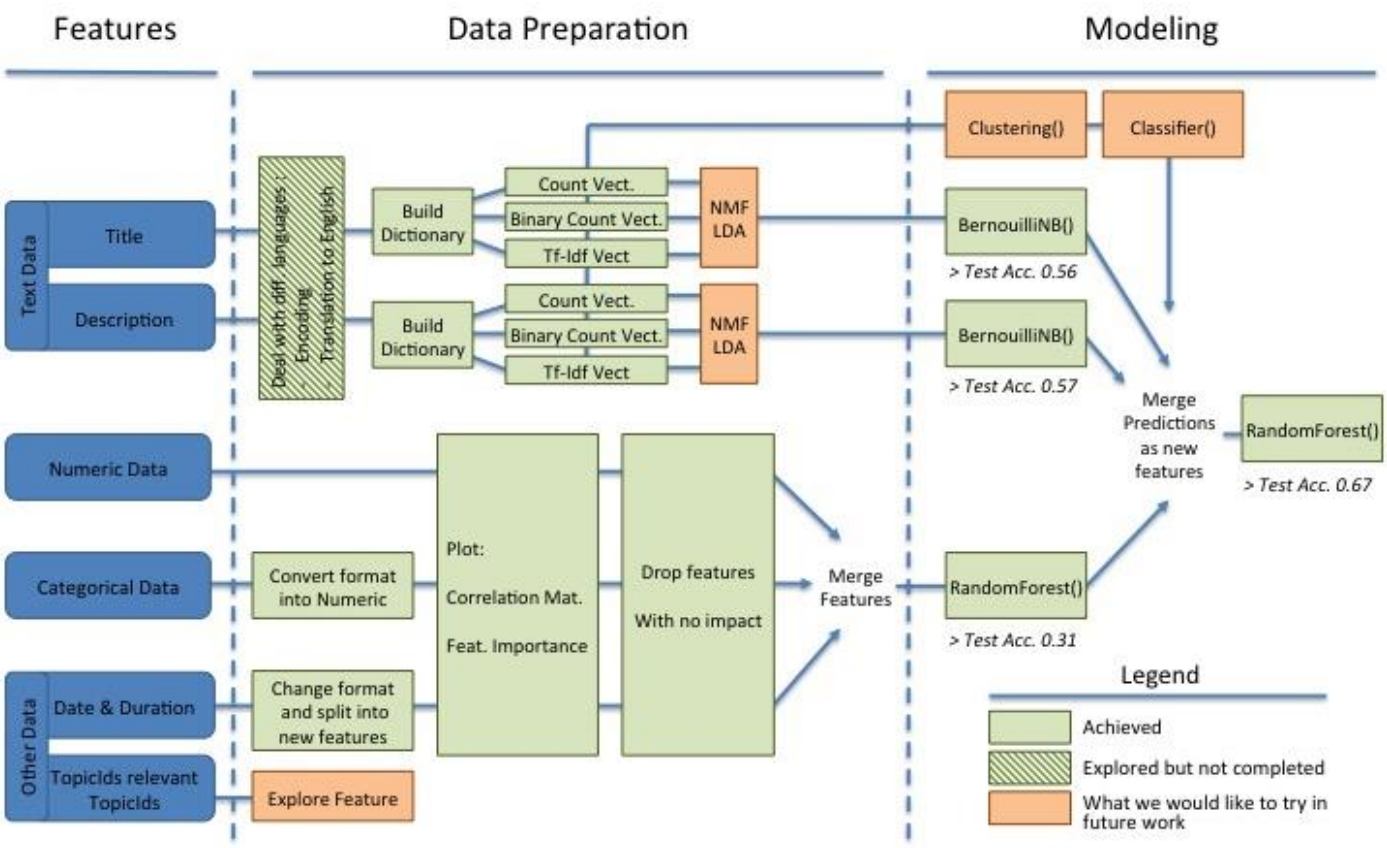

Figure 8: Enlarged image of overall illustration