CZ4032 Data Analytics & Mining

Group 10

Ang Jun Liang (U1722336E) 16.67%

Bryan Leow Xuan Zhen (U1721837L) 16.67%

Nigel Ang Wei Jun (U1721087C) 16.67%

Wang Daini (U1721818F) 16.67%

Wang Xiaoyu (U1721992H) 16.67%

Yee Wei Min (U1720243E) 16.67%

Table of Content

1 Abstract

YouTube has brought about evolutionary changes on how people watch videos around the globe. Millions of videos have been created up to date and these have come along with an enormous amount of information. This report aims to describe, visualise and analyse YouTube data to give both descriptive and predictive insights for more efficient and effective use of the information carried with the videos. More specifically, we aim to design a classifier to predict the video category and conduct an association analysis on the video tags.

We first obtained our data which consists of YouTube video information across a range of dates and regions, and then examined and visualised the general statistics of these categorised videos. After experimenting with different classification methods, we derived a final ensemble model (with top 9 tuned models), achieving an accuracy of 73.8% in predicting video category. Using FP-Growth algorithm, we obtained a list of maximal frequent itemsets and analysed the associations between video tags.

By visualisation, pattern observation and model application, we suggested predictive measures and derived conclusions for better management of YouTube videos.

2 Problem Description

2.1 Background and Motivation

YouTube was founded in 2005 by three former PayPal employees - Chad Hurley, Steve Chen, and Jawed Karim. Through YouTube, users are allowed to upload, view, rate and share videos, add videos to playlists, comment on videos as well as subscribe to other users [1]. YouTube now has 1.9 million active users worldwide and is one of the world's largest video sharing platforms, despite only having a short history of 14 years. Everyday, people around the world watch over 1 billion hours of videos on YouTube and leave millions of comments [2].

YouTube's wide variety of functionalities and its large viewership have made it one of the best mediums to study the watching behaviours and video preferences of people together with their social networks. For example, using the large amount of data available, we would be able to easily generate insights about the watching preferences of different demographics. By analyzing popular videos, we can dive into the elements that make a video viral, and perhaps provide suggestions to YouTubers on how to make popular videos.

Even though there are many possible areas of analysis and research, we decided to focus on the following problems due to limited time and resources.

2.2 Problems Identified and Significance

Prediction of video category

With its large user base, 500 hours of video are uploaded to YouTube every minute worldwide. This translates to 720,000 hours of video uploaded every day. Among all the features of the videos (such as title, description, category and tags), category is one of the important features used by YouTube to classify the videos and recommend to its users.

While categories can be manually selected by the users, inexperienced users may be confused by the large variety of categories. Hence, they may choose the wrong category when uploading their videos. When this happens, videos may be wrongly recommended to a group of audience who have no interest in them. This can negatively affect the number of views, which potentially leads to loss of advertisement income for the uploader. Additionally, the audience who received the recommendation would view YouTube recommendation as poor, thus having a poor impression of YouTube.

Therefore, we decided to build a classifier to predict the video category based on other information about the video, such as title, tags and video comments. The algorithm would be able to recommend a suitable category for the video, so that the aforementioned mis-categorisation can be reduced.

Association analysis of video tags

When watching a video, users can click on a tag to search for more videos related to the tag. For example, if a user is following videos on a new iPhone release event, he can simply click on a tag that he thinks is the most relevant in order to search for all related videos. Simply put, tags are a useful way for the audience to search for related videos, and for the uploaders to increase their video views.

Similar to above, correct tags can improve the income for uploaders and give a better impression for audience users. Therefore, we decided to conduct an association rule analysis to determine what are the common tags that go together. From there, we can recommend suitable tags for the uploaders.

2.3 Related Work

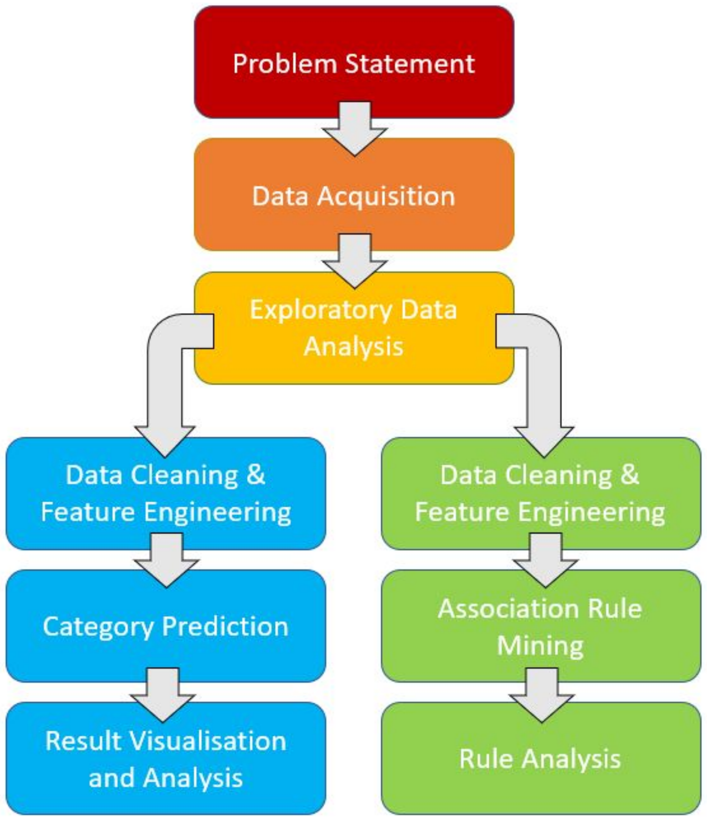
Prior to the actual start of the analysis, we have done a research on the related work in terms of YouTube video data mining tasks.

From our research, we realised that there are quite a number of approaches and algorithms for category prediction. Each of these approaches have their own advantages and disadvantages. Therefore, we decided to try out different algorithms and conduct a model ensemble in hope for getting the best performance. These will be elaborated in the following sections.

Apart from algorithms, we also referenced the method used by *DataSnaek* [3] to scrape the data from YouTube API. The detailed data scraping process will be elaborated in the following sections.

3 Approach and Methodology

In this project, our flow of analysis is as follows:


Problem Statement
Data Acquisition
Exploratory Data Analysis
Data Cleaning & Feature Engineering
Category Prediction
Result Visualisation and Analysis
Data Cleaning & Feature Engineering
Association Rule Mining
Rule Analysis

We started with our problem definition, so that we could have a focused analysis on the problem at hand. This has helped us find a suitable dataset, as well as in conduct relevant exploratory analysis. After that, we moved on to data acquisition, where we tried to make and/or find a dataset which could aid us in our analysis. Next, we conducted an exploratory data analysis, so as to get a deeper understanding of our data and derive some inspirations in terms of feature engineering.

For the data mining algorithms, we divided into two sections because we have two different problem statements. We decided to conduct separate data cleaning and feature engineering processes because the two problems require different handlings of features.

For the first problem (boxes in blue), we started with data cleaning and feature engineering of the unstructured data (e.g. text), before feeding the generated features into the different classification algorithms. We compared the accuracies of the different algorithms and came up with an ensemble model with 9 different classification methods. Finally, we plotted the confusion matrices and analysed the prediction results.

For the second problem (boxes in green), we also started with data cleaning and feature engineering of the tags, which is also text data. The cleaned features are then used to conduct association rule mining with our own implementations. Finally, we analysed some top rules we obtained from the association rules mining.

This approach will be elaborated in the following sections in detail.

4 Data Acquisition

4.1 Data Scraper

To prepare the datasets required for our analysis, we have attempted to scrape the desired data via an API - YouTube Data API. To do so, one must first prepare a ‘api_key.txt’ with an API key obtained here.

We ran a version of the Trending YouTube Video Scraper by *DataSnaek* with some slight modifications, and obtained the data on videos trending on the particular day. Detailed comments provided by *DataSnaek* can be found in the notebook titled ‘*0A - data_scraper.ipynb*’ in ‘*data_scraper*’ folder.

The Data Scraper prepares the columns of the dataset to be built, builds a link based on the input region and API key to be fed to the API, and formats the JSON data to be added to the dataset. The data returned by the API are the current day’s trending videos in a particular region.

Further details of the Data Scraper and instructions on how to obtain an API key can be found in Section 9.1 of the Appendix.

Now, due to the limitations of the scraper being that only the present day’s trending videos are available, we utilise the datasets compiled by DataSnaek as our basis for analysis.

4.2 Kaggle API

The datasets we use for analysis will be obtained via the Kaggle API, and kaggle must be installed via *pip* or *conda*. After which API credentials must be properly initialised.

After the credentials are prepared, downloading of the datasets can be done via a command prompt, or within a Jupyter Notebook. We have prepared a Jupyter Notebook, *“0B - kaggle.ipynb”* in ‘*data_scraper*’ folder for this purpose. Further details on how to install kaggle, authenticate and both methods to download the dataset can be found in Section 9.2 of the Appendix.

4.3 Datasets Acquired

The datasets we acquired from *DataSnaek* are the information of daily trending videos in the US and UK from 13th September 2017 to 22nd October 2017 (i.e: for 40 days). The datasets contain almost 200 trending videos everyday each for the US and UK.

Data files we downloaded are *GBcomments.csv, GBvideos.csv, UScomments.csv, USvideos.csv, GB_category_id.json, US_category_id.json*. These files are output into a folder ‘kaggle’, inside the ‘data_scraper’ folder. Alternatively, they have been downloaded and zipped into a *.7z* file; Simply extract the files into the same location.

GBcomments.csv and *UScomments.csv* are comment data files that contain 4 columns and their corresponding columns are video_id, comment_text, likes (contain the number of likes of the comment) and replies (contain the number of replies of the comment). *GBcomments.csv* contains 718,458 rows of data and *UScomments.csv* contains 691,408 rows of data.

GBvideos.csv and *USvideos.csv* are video statistic files that contain 11 columns and their corresponding columns are video_id, title (video title), channel_title, category_id (Separated by '|', [none] is displayed if there are no tags), tags, views (number of view of the video), likes (number of likes), dislikes (number of dislikes), thumbnail_link, date (Formatted in [day].[month]). Video_id is the common id field in both comment and video statistic files and can act as the attribute to merge the comment and video statistic files. *GBvideos.csv* contains 7,996 rows of data and *USvideos.csv* contains 7,999 rows of data.

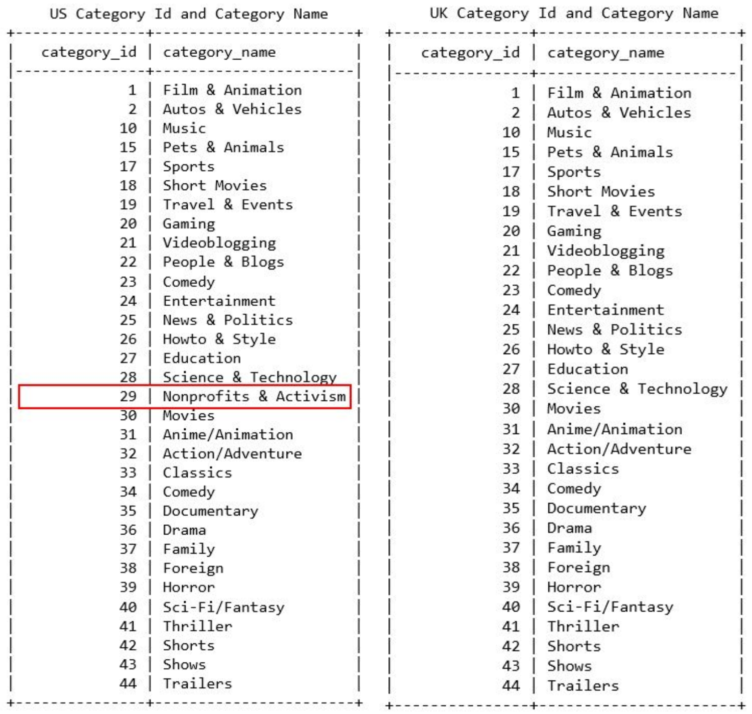
GB_category_id.json and *US_category_id.json* are the files that contain the category id with its corresponding category name. However, the category name and category id may be varies in different countries. Hence, detection of discrepancy (i.e. same category id but different category name) will be performed in EDA.

5 Exploratory Data Analysis

In this section, we tried to use bar charts, word maps and other visualisation tools to reveal the trends of the dataset. Due to the report page limit, we will only include the visualisation and explanation about the EDA that are related to our data mining task in this section. For the code implementation and more EDA, please refer to "*1 - EDA.ipynb", "7 - Word Map Part 1.ipynb"* , "*7 - Word Map Part 2.ipynb*" and "*6 - Analyze Relationship between Like or Dislike Ratio and Sentiment.ipynb*".

5.1 Exploring GB_category_id.json and US_category_id.json files

As the category name and category id may be varies in different countries, we will first check if there are any discrepancies (i.e. same category id but different category name) between the two files.



```
     US Category Id and Category Name
+---------------+------------------------+
|  category_id  | category_name          |
|---------------+------------------------|
|             1 | Film & Animation       |
|             2 | Autos & Vehicles       |
|            10 | Music                  |
|            15 | Pets & Animals         |
|            17 | Sports                 |
|            18 | Short Movies           |
|            19 | Travel & Events        |
|            20 | Gaming                 |
|            21 | Videoblogging          |
|            22 | People & Blogs         |
|            23 | Comedy                 |
|            24 | Entertainment          |
|            25 | News & Politics        |
|            26 | Howto & Style          |
|            27 | Education              |
|            28 | Science & Technology   |
|            29 | Nonprofits & Activism  |
|            30 | Movies                 |
|            31 | Anime/Animation        |
|            32 | Action/Adventure       |
|            33 | Classics               |
|            34 | Comedy                 |
|            35 | Documentary            |
|            36 | Drama                  |
|            37 | Family                 |
|            38 | Foreign                |
|            39 | Horror                 |
|            40 | Sci-Fi/Fantasy         |
|            41 | Thriller               |
|            42 | Shorts                 |
|            43 | Shows                  |
|            44 | Trailers               |
+---------------+------------------------+
```

```
     UK Category Id and Category Name
+---------------+------------------------+
|  category_id  | category_name          |
|---------------+------------------------|
|             1 | Film & Animation       |
|             2 | Autos & Vehicles       |
|            10 | Music                  |
|            15 | Pets & Animals         |
|            17 | Sports                 |
|            18 | Short Movies           |
|            19 | Travel & Events        |
|            20 | Gaming                 |
|            21 | Videoblogging          |
|            22 | People & Blogs         |
|            23 | Comedy                 |
|            24 | Entertainment          |
|            25 | News & Politics        |
|            26 | Howto & Style          |
|            27 | Education              |
|            28 | Science & Technology   |
|            30 | Movies                 |
|            31 | Anime/Animation        |
|            32 | Action/Adventure       |
|            33 | Classics               |
|            34 | Comedy                 |
|            35 | Documentary            |
|            36 | Drama                  |
|            37 | Family                 |
|            38 | Foreign                |
|            39 | Horror                 |
|            40 | Sci-Fi/Fantasy         |
|            41 | Thriller               |
|            42 | Shorts                 |
|            43 | Shows                  |
|            44 | Trailers               |
+---------------+------------------------+
```

Figure 5.1: The YouTube category IDs and category names, for US and UK respectively

From Figure 5.1, we can see that there are no discrepancies in category id and name between the two countries. However, the US has a category id 29 named Nonprofit & Activism but UK does not.

5.2 Exploring GBcomments.csv, GBvideos.csv, UScomments.csv, USvideos.csv files

First of all, we concatenate the two video-statistics .csv files. Some of the rows with errors are removed when we read in the csv files. After concatenation, the video statistic file has 15,985 rows and 11 columns.

5.2.1 Number of Unique Values for each attribute

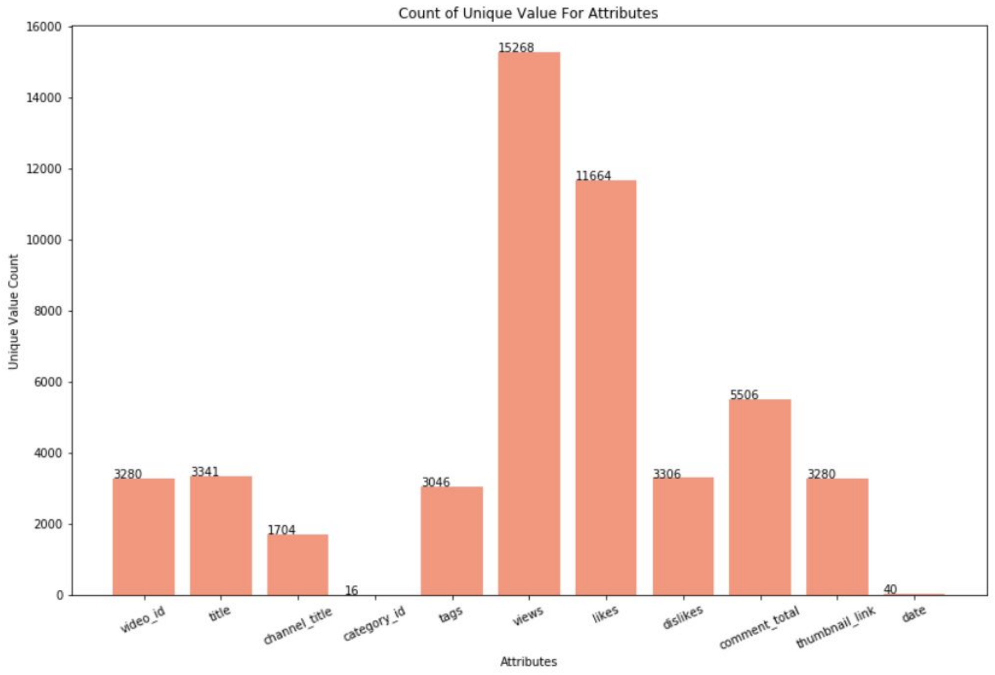


Figure 5.2.1: Number of unique values for each attribute

From Figure 5.2.1, we realised that although there are 15985 rows of data but there are only 3280 unique **video_id**. This means that there are videos that trend more than one day in the 40 days period. Next, we also noticed that the number of unique **title**s are slightly more than the number of unique **video_id**. This means that some of the video titles are changed when the video is on trending. Besides, there are only 16 unique **category-id**s in the video statistic files but there are 32 categories in *GB_category_id.json* file and 31 categories in *US_category_id.json* file. To confirm that our dataset consists of the correct information, we went to youtube upload page and realised there are only 15 categories available for the user to choose from. Hence, we assume that Youtube changes the number of categories overtime to group the subset of categories together into bigger category and our dataset that contains only 16 unique categories is correct.

Realising that our dataset containing duplicated **video_id**s as there are videos that trend more than one day in the 40 days period, we decided to remove the duplicated **video_id** before we continue on further analysis. This is because we are not doing analysis on time series and we do not want one video that trends on multiple days to affect our analysis.

5.2.2 Number of trending videos in each Category

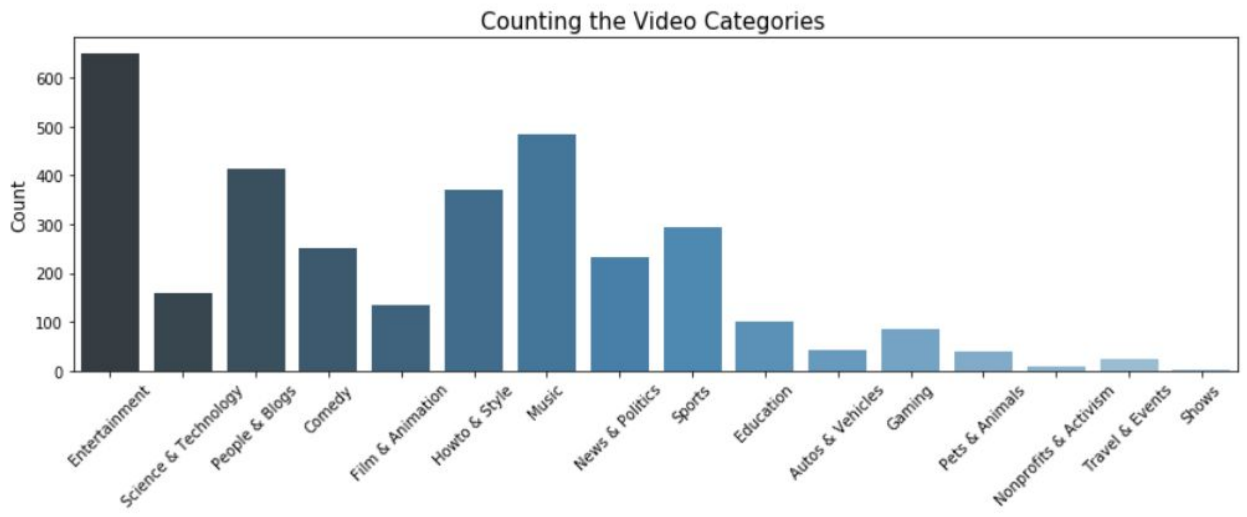


Figure 5.2.2: Count of Trending Videos of each Category

From Figure 5.2.2, we can see that **Entertainment** is the most popular category in Youtube trending videos followed by **Music, People & Blogs and Howto & Style**. Besides, the number of videos in **Nonprofits & Activism** and **Shows** are significantly low and will affect the quality of video category prediction and hence the videos in these two categories will be removed when doing video category prediction.

5.2.3 Distribution of number of comments in each Category

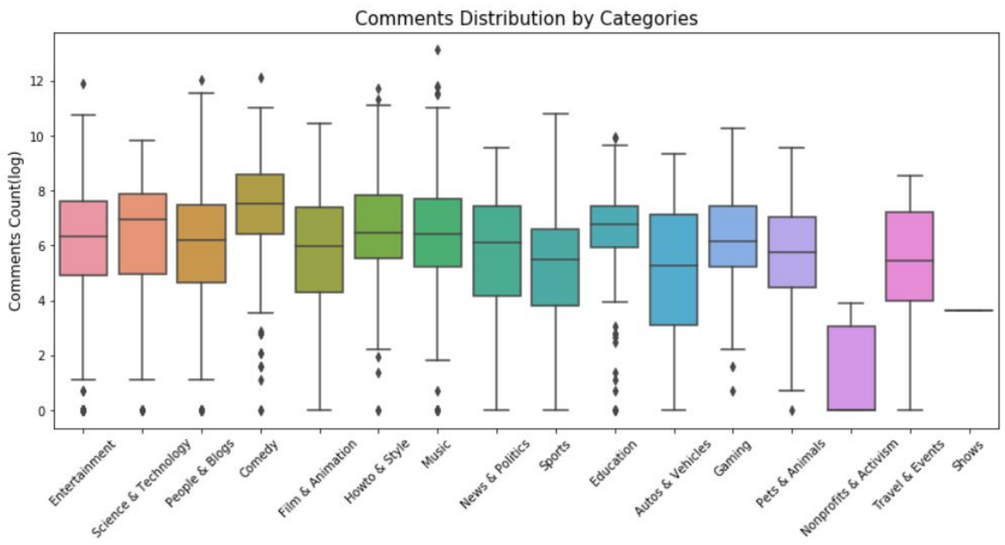


Figure 5.2.3: Distribution of comment counts(log) in each Category

From Figure 5.2.3, we can see that video with highest number of comments is in **Music** category but **Comedy** has the highest median in number of comments. The comments counts are in log so we can see the distribution clearly. We also concluded that there are sufficient number of comments in each category for us to do category prediction by using comments after excluding the two categories **Nonprofits & Activism** and **Shows** that have lesser number of comments.

5.2.3 Most Common Words

To find out the most common words based on category, we plotted 2 word clouds, respectively for titles and tags, and comment. We first combined the raw data frames to include targeted text. Next, we cleaned the text by making text lowercase, removing text in square brackets, removing punctuation and removing words containing numbers. After that, by applying *CountVectorizer*[7], we will recognize the top words for each category. In the process, we found that after generating a document-term matrix, words like ‘likes’, ‘im’, ‘love’, ‘just’ and so on, appear in the comments of almost all the categories. Thus, we added these words into the stop words list , removed all the stop words and created a word cloud for each category.

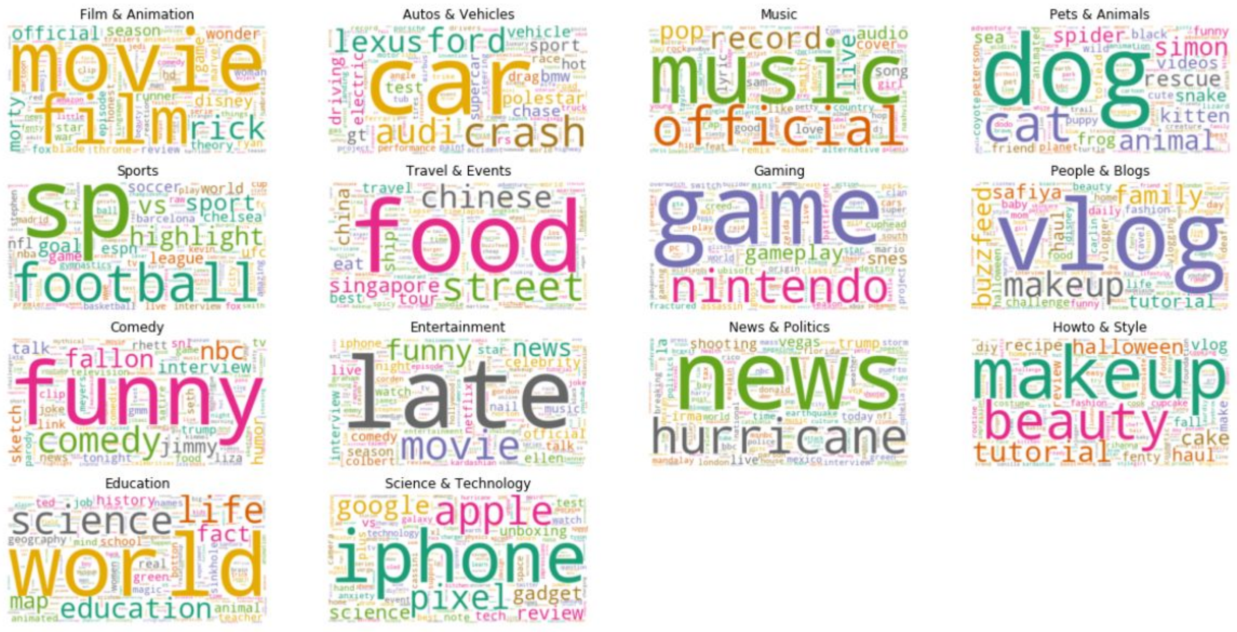


Figure 5.2.3.1: Word Clouds for Titles and Tags

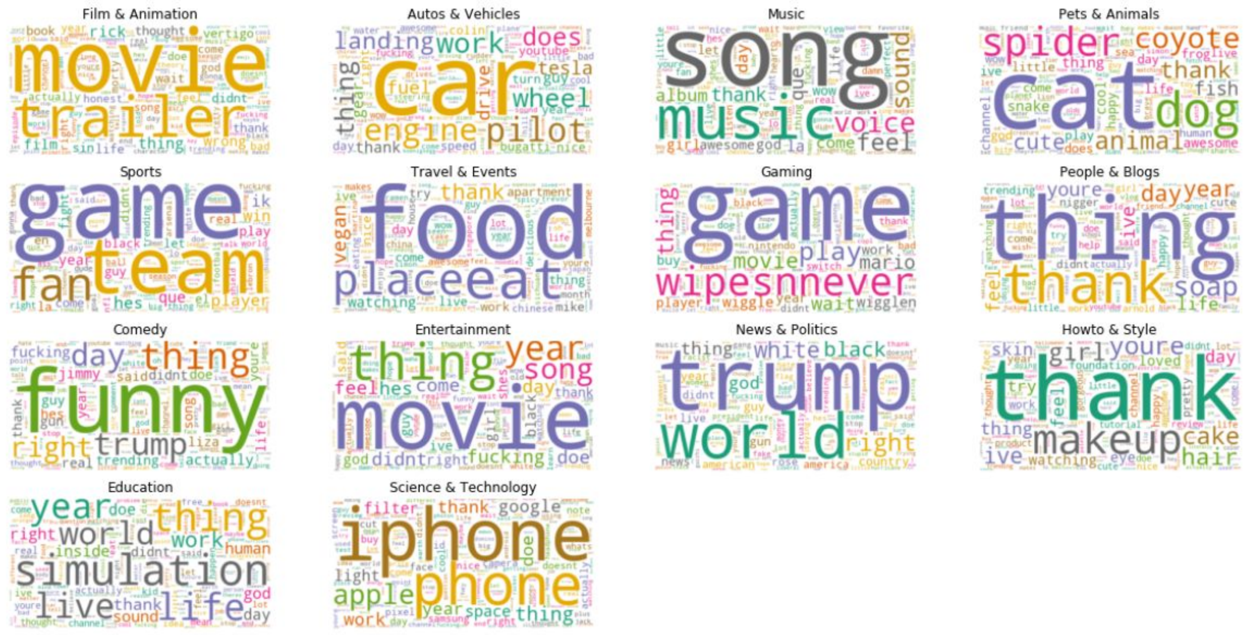


Figure 5.2.3.2: Word Clouds for Comments

Figure 5.2.3.1 shows the word cloud for the titles and tags of each category. Figure 5.2..2 show the word cloud for comments of each category. Generally, the words that are used in the titles, tags and comments are different for different categories. This shows that title, tags and comments of the videos are the main attributes that we can use to predict the video category.

We noted that in the category **Science & Technology**, both show 'iphone' is the most common word people talk about, and 'food' is most mentioned in **Travel & Events** category. This might suggest that prediction on the category based on titles and tags, and comments may give similar results for these categories.

However, we also found out that in **Pets & Animals**, 'dog' is mentioned the most in titles and tags whereas 'cat' is mentioned the most in comments. This shows that the prediction of these categories based on titles and tags, and comments may not give the same result. This leads us to predict the video category by using both titles and tags, and comments separately.

5.3 Analyzing relationship between sentiment and likeability

We hypothesized that the likeability of a video can be determined from its comments' sentiment and performed some experiments. We used TextBlob's API for sentiment analysis. We determined the likeability and dislikeability of a video through its like_rate (i.e number of likes per 100 views) and dislike_rate (i.e number of dislikes per 100 views).

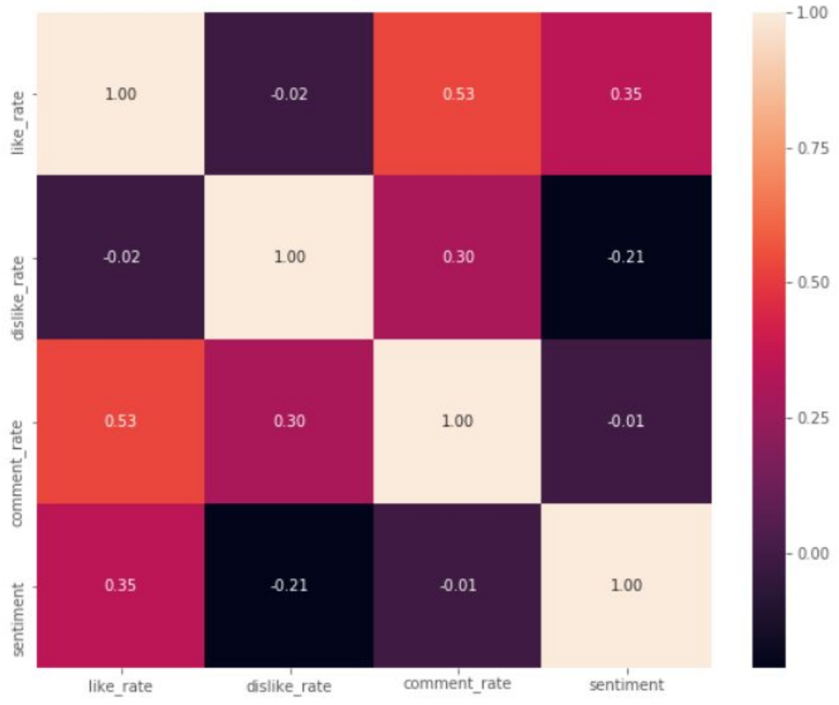


Figure 5.3: Correlation between like_rate, dislike_rate, comment_rate and sentiment

We found a weak positive correlation of 0.35 between sentiment and likeability and a weak negative correlation of -0.21 between sentiment and dislikeability. Additionally, we found no correlation (-0.01) between sentiment and comment rate (number of comments per 100 views). Hence, we deduce that our hypothesis is only partially right.

6 Algorithms and Experiments

6.1 Category Prediction

We consider 2 approaches to predict the category of a video. The first approach uses the title and tags of the video, while the second approach uses the comments for the video. Experiments with different models are conducted for each approach, and the selected models are then combined (i.e. model ensemble) to improve the accuracy.

6.1.1 Implementation

Using video title and tags
In this section, we will first expound on the main ideas behind the first approach - category prediction using video title and tags. For the code implementation, please refer to *"3 - Category prediction using video title and tags Part 1.ipynb"*.

After reading in the *"videos.csv"* file as a pandas dataframe, we encode the label *category_id* with an encoder. A dictionary *category_id_to_word* is created with the encoded *category_id* as the key, and the respective *category_name* as the value. This dictionary allows us to look up what a particular *category_id* corresponds to quickly. A preliminary inspection allows us to observe that *category_id 14 - 'Nonprofits & Activism' and category_id 15 - 'Shows'* have way too few videos. As such, we decide to remove them from our analysis.

We define a function named *clean_and_join_title_and_tags*. In this function, we first combine the title of the video and the different tags into one single string. The string is then tokenized into individual tokens in a list with lowercase characters only. Next, we remove the tokens that are not alphanumeric as we hypothesize that special characters are not meaningful in providing insights to the context of the videos, but rather add noise to the classification process. We then proceed with filtering out the stop words such as "the", "to", "for", etc. since they do not aid us in finding the context of the video as well [8]. Next, we use PorterStemmer, which is known for its simplicity and speed[9], to reduce the inflection in the tokens to their root forms.The stemmed tokens are then combined and a stemmed sentence is returned from the function.

Since some of the videos in the training data could be trending for some time, there could be duplicates. We treat the videos as duplicates if the stemmed sentence for the videos are the same, and remove them. The remaining stemmed sentences are then converted to a matrix of TF-IDF[10] features with the *TfidfVectorizer.* These TF-IDF features will serve as our features (X-variables) while the *category_id* will serve as the corresponding label (Y). The dataset is then shuffled before it is used for training and testing.

To get a general sense of how well the features can be used to predict the label, we get the baseline accuracies of the models built from the following algorithms: **Logistic Regression**, **Multinomial Naive Bayes**, **Random Forest**, **Support Vector Machines**, **K-Nearest Neighbours**, **Decision Tree** and **Feed-forward Neural Network**. We find that even without tuning, the baseline models have an average accuracy of 50%. Next, 5-fold stratified cross validation is used for the evaluation of the models and hyperparameter selection/tuning. After fine-tuning and finding the best parameters for each of the models, we found that **Multinomial Naive Bayes** algorithm gave us the best accuracy at 71%.

Using video comments

Category prediction using comments is similar to that using title and tags. The only difference is that we now use comments instead of title and tags. Since it is similar, this will not be elaborated on. For category prediction using comments, we found that **logistic regression** gave us the best accuracy of 66%. For code implementation, please refer to *"2 - Category Prediction using Comments.ipynb"*.

Model ensemble

From the 2 approaches, we have built a total of 11 models. We used a simple voting strategy between the models to determine the output of the ensemble. We ensemble all 11 models first, and then try to remove the models with the worst accuracy and ensembling again. Eventually, we found that the top 9 models ensembled gave the best accuracy of 73.8%. Please refer to *"4 - Category prediction with Ensemble Modelling.ipynb"* for code implementation.

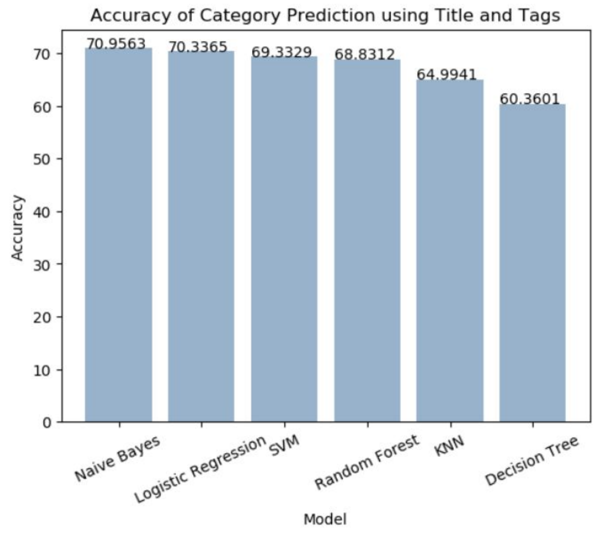
6.1.2 Extra Implementation

Additionally, we tried using a multi layered perceptron network in *"3 - Category prediction using video title and tags Part 2.ipynb"* to predict video category using video title and tags. After trying various combinations of layers, we found a 3 hidden layer network with nodes 768, 384, 64 gave the best accuracy of 0.68. We used Adam optimizer and employed Early Stopping and Dropout to prevent overfitting. We tried both feature vectors from GloVe embeddings and TF-IDF and found TF-IDF gave superior results.

6.1.3 Comparison Schemes

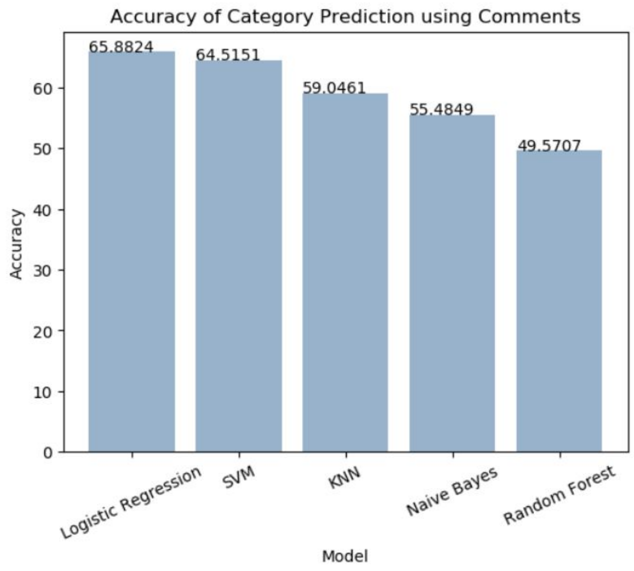
TITLE_AND_TAGS MODELS:

- MultinomialNB - Best: 0.709563 using {'alpha': 0.12, 'fit_prior': False}
- LogisticRegression - Best: 0.703365 using {'C': 50, 'penalty': 'l2', 'random_state': 42}
- SVC - Best: 0.693329 using {'C': 11.1, 'gamma': 0.09, 'kernel': 'rbf', 'random_state': 42}
- RandomForestClassifier - Best: 0.688312 using {'n_estimators': 87, 'random_state': 42}
- KNeighboursClassifier- Best:0.649941 using {'algorithm': 'auto', 'n_neighbors': 6, 'weights': 'distance'}
- DecisionTreeClassifier - Best: 0.603601 using {'random_state': 42}



COMMENT MODELS:

- LogisticRegression - Best: 0.658824 using {'C': 35, 'penalty': 'l2', 'random_state': 42}
- SVC - Best: 0.645151 using {'C': 21, 'gamma': 0.1, 'kernel': 'rbf', 'random_state': 42}
- KNeighboursClassifier - Best: 0.590461 using {'algorithm': 'auto', 'n_neighbors': 20, 'weights': 'distance'}
- MultinomialNB - Best: 0.554849 using {'alpha': 0.0, 'fit_prior': False}
- RandomForestClassifier - Best: 0.495707 using {'n_estimators': 88, 'random_state': 42}



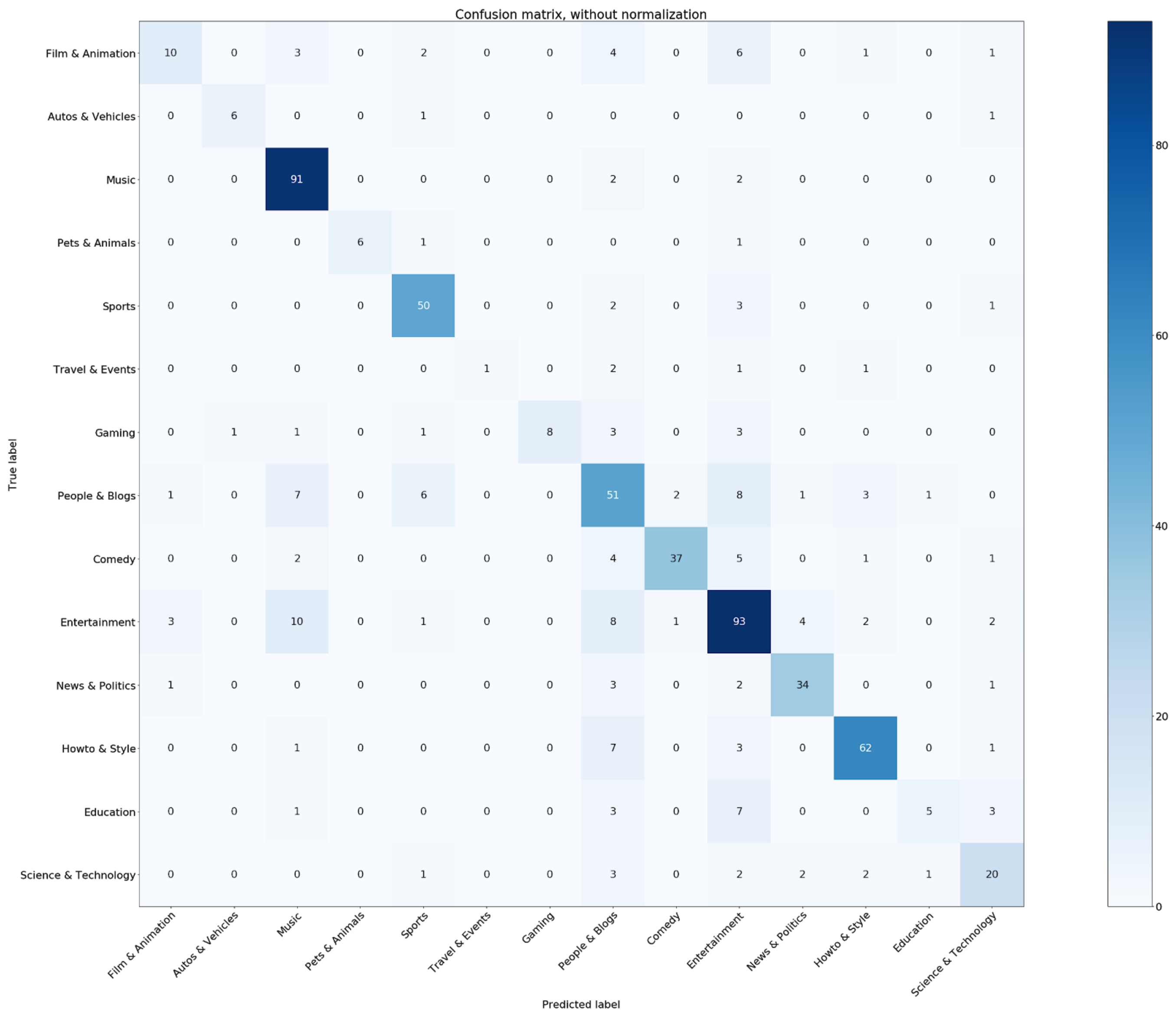
ENSEMBLE MODELS:

- Ensemble model's 5 fold stratified CV score is: 0.7379404129504286

Shown above are the tuned accuracies for the comment models and the title and tags models. As mentioned earlier, Multinomial Naive Bayes model turned out to be the best at predicting the category of the video from the tags and titles at 71% accuracy, while the Logistic Regression model is the best at predicting the category of the video from the comments at 66% accuracy. We further ensembled the model which gives us a 74% accuracy.

6.1.4 Analysis on Prediction Result

The comparison above is done with one single measure - accuracy of prediction. However, for classification problems, this measure might be biased or misleading sometimes, especially when there exists imbalanced classes. This refers to situations where there is a disproportionate ratio of observations in each class. In our case, all the 14 categories do not have similar number of videos. As such, it would be a better choice for us to take a closer look at the confusion matrix.



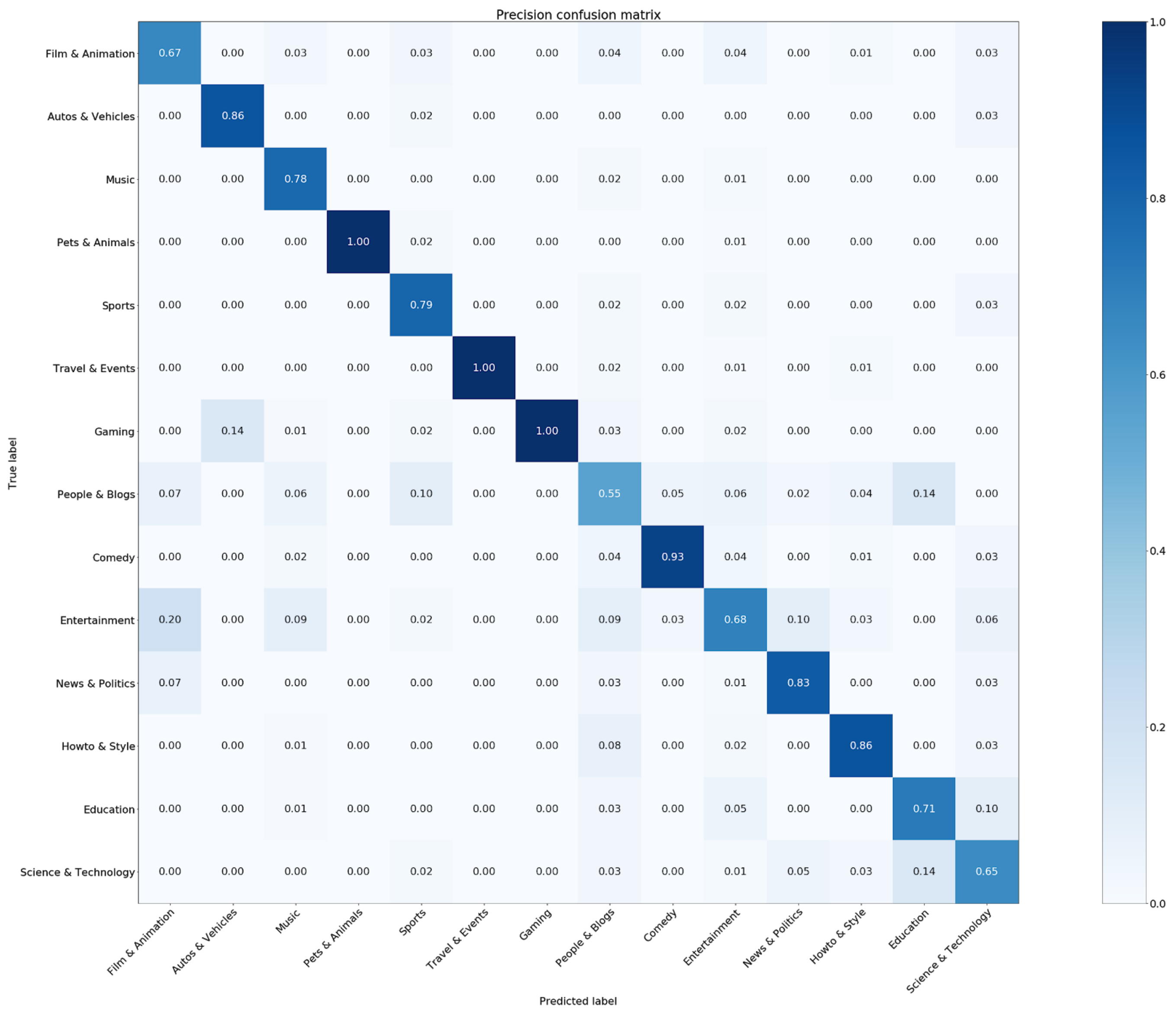
True label \ Predicted label	Film & Animation	Autos & Vehicles	Music	Pets & Animals	Sports	Travel & Events	Gaming	People & Blogs	Comedy	Entertainment	News & Politics	Howto & Style	Education	Science & Technology
Film & Animation	10	0	3	0	2	0	0	4	0	6	0	1	0	1
Autos & Vehicles	0	6	0	0	1	0	0	0	0	0	0	0	0	1
Music	0	0	91	0	0	0	0	2	0	2	0	0	0	0
Pets & Animals	0	0	0	6	1	0	0	0	0	1	0	0	0	0
Sports	0	0	0	0	50	0	0	2	0	3	0	0	0	1
Travel & Events	0	0	0	0	0	1	0	2	0	1	0	1	0	0
Gaming	0	1	1	0	1	0	8	3	0	3	0	0	0	0
People & Blogs	1	0	7	0	6	0	0	51	2	8	1	3	1	0
Comedy	0	0	2	0	0	0	0	4	37	5	0	1	0	1
Entertainment	3	0	10	0	1	0	0	8	1	93	4	2	0	2
News & Politics	1	0	0	0	0	0	0	3	0	2	34	0	0	1
Howto & Style	0	0	1	0	0	0	0	7	0	3	0	62	0	1
Education	0	0	1	0	0	0	0	3	0	7	0	0	5	3
Science & Technology	0	0	0	0	1	0	0	3	0	2	2	2	1	20

To illustrate, the above is the confusion matrix for our final ensembled model. Each row represents the true label of the videos, while each column represents the predicted label given by the model. The different cell values show the distribution of the prediction versus the true labels.

It would be best if we have all the non-diagonal entries (entries that are not on the diagonal line) as 0, because it would mean 100% accuracy for the model. In our case, we can see that despite having relatively large numbers at the diagonal entries, we still have quite a number of non-zero cells elsewhere. This explains why the accuracy is only 73.8%.

To visualise the result better, we also plot the normalised confusion matrices. The steps are as below:

- Sum up all the counts across the columns
- Divide values at each entry by its column sum



With the steps mentioned above, we can see that the diagonal entries now represent the precision of the model in predicting each category. For example, out of all test samples that our model predicted to be in category *Science & Technology*, 65% of them are correct predictions (True Positive).

From the plot above, we can derive more insights into the performance of the model, instead of relying solely on the overall accuracy. For instance, we can see that our model performs well in predicting *Pets & Animals*, *Travel & Events*, *Gaming* and *Comedy*, with above 90% precision for each of these categories. On the other hand, the performance is not very decent in predicting *Film & Animation*, *People & Blogs*, *Entertainment* and *Science & Technology*, with below 70% precision for each of these categories.

The implication of this is that when we apply the model to solve the problem identified, we need to take into account the category predicted, instead of just blindly apply the model because of the over 70% overall accuracy. To be more specific, in suggesting categories to the YouTube video uploader, if our model predicts that the label should be *Film & Animation*, we may consider suggesting *Film & Animation*, *People & Blogs*, *Entertainment* and *News & Politics* as 4 potential categories. This is because the precision for *Film & Animation* is not very high, and in fact the true label might be one of the 4 potential categories based on the above plot.

We have also plotted the normalised confusion matrix, summing up all the counts across the rows and dividing values at each entry by its row sum. This means that the diagonal entries now represent the recall of the model for each category. The plotted matrices can be found in *4 - Category prediction with Ensemble Modelling.ipynb*.

Additionally, other than the final ensemble model, the aforementioned plotting of 3 confusion matrices was also done on the other two models - the best Video Titles and Tags model, and the best Comments model. These matrices can be found in *3 - Category Prediction using Video Titles and Tags - Part 1.ipynb* and *2 - Category Prediction using Comments.ipynb* respectively.

6.2 Mining Association Rule for Tags

In order to find rules that can predict the occurrence of tag based on other tags, we use the FP-growth algorithm [11]. We first create an FP-tree and store the records (processed by L-order) into the tree. Next we construct the conditional pattern base and conditional FP-tree for each frequent 1-itemset. From there, we can mine all the frequent itemsets. In our own implementation (**no external library**), we used a small minimum support and focus on maximal frequent itemsets due to lack of RAM.

6.2.1 Implementation

In this section, we will discuss the main ideas behind the association rule mining for tags. For the code implementation, please refer to *“6 - Mining Association Rules for Tags.ipynb”.*

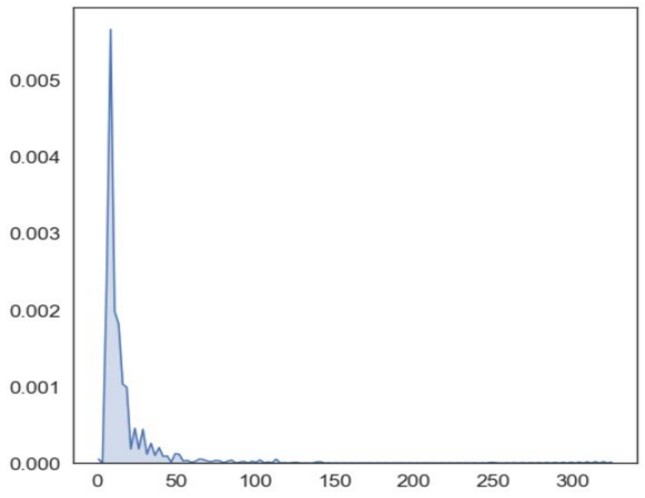
After reading in the *“videos.csv”* file as a pandas dataframe, we drop data samples with the same video id. We then extract the tags from each data sample and perform some text cleaning (removing non-alphanumeric characters and setting to lowercase). From now on, I will refer to each data sample as a record and each tag as an item.

First, we do some preliminary analysis on the items. We found that there are 28,281 unique items. Here is a table of containing the top 10 most frequent 1-itemsets:

Item	**Support Count**
funny	325
comedy	250
2017	140
vlog	140
makeup	125
interview	114
music	113
how to	112
humor	112
news	107

From this list, we can already tell some common topics include comedy, vlogging, makeup videos, music etc.

Also, we calculated some statistics for 1 itemsets (kde plot):



Notice that the distribution of the 1-itemsets is extremely skewed. A large number of 1-itemsets have very few occurrences. Hence, we use minimum support of 0.5%, which translates to a minimum support count of 16 to be considered frequent. This turns out to be 346 frequent 1-itemsets out of 3082 records

Next, we change the records to L-order and add them to a FP-Tree. From here, we can mine the frequent itemsets. Because of the large number of frequent itemsets. We elect to focus on obtaining the maximal frequent itemsets of at least size 2. We found 98 maximal frequent itemsets.

Next, we generate the rules from the maximal frequent itemsets. We noticed that for a maximal frequent itemset of size n, if we segment it into n different binary partitions of sizes {1, n-1}, and used them as rules, the worst confidence of any rule is 99.1%. This implies that all possible rules have at least 99.1% confidence.

Explanation - any other rules generated are guaranteed to have a LHS that is a superset of the LHS of some rule with 1 item in the LHS. A superset will have a lower support count than its subset. Since Confidence = Count(LHS ∪ RHS) / Count(LHS) and Count(LHS) is lower for these rules, their confidence are at least 99.1%.

Consider any maximal frequent itemset {A, B, C, D}. The rules with 1-item on LHS are A -> B C D, B -> A C D, C -> A B D and D -> A B C and they all have confidence greater than 99.1%. Hence, we can say that for any item in a maximal frequent itemset, it can predict the occurence of the rest of the items with at least 99.1% confidence. As a result, it is more meaningful to simply analyze the maximal frequent itemsets.

6.2.2 Analysis on the Maximal Frequent Itemsets generated

We will now proceed to analyse 4 interesting maximal frequent itemsets and one bad itemset generated from the association rules mining model. Please refer to *“6 - Mining Association Rules for Tags.ipynb”* to see all the maximal frequent itemsets that we generated.

The first maximal frequent itemset is *['donald trump', 'politics']* that has a support count of 15. It is very reasonable that ‘donald trump’ is associated with ‘politics’ because he is the current president of the United States and probably the most famous politician in the 21st century due to trade wars.

The second maximal frequent itemset is *[‘nba’, ‘basketball’]* that has a support count of 28. This pair of tags are frequently used together because NBA (National Basketball Association) is a men's professional basketball league in North America.

The third maximal frequent itemset is *['makeup', 'how to', 'tutorial', 'beauty', 'makeup tutorial']* which has a support count of 19. Makeup videos on youtube are usually tutorials and tutorial is synonym of ‘how to’. Besides, makeup is to beautify appearance. Hence, the probability of these tags being used together is very high.

The forth maximal frequent itemset is *['trailer', 'film', 'movie']* which has a support count of 21. The reason that these 3 tags are tightly associated with each other is because trailer videos give advance publicity to a film or movie by releasing extracts or selected details. Besides, there are only trailer videos of movies available on youtube due to profit and copyright issue.

One example of unreasonable maximal frequent itemset that is generated is *['vlog', 'daily', 'british', 'puppy', 'zoella', 'zoe', 'pointlessblog', 'nala']* which has a support count of 18. For our dataset, vlog can predict ‘zoe’ and ‘zoella’ with high confidence but this is probably because the dataset is biased for this period and does not hold in general.

To conclude, most of the maximal frequent itemsets generated by the association rule mining model are very reasonable. A potential way to use this association rule model is to suggest tags when the youtuber is tagging the video as the occurrence of other tags can be predicted by the rules in the model once the first tag is selected. Beside saving time needed for tagging videos, the model may also suggest related tags that the youtuber never think of.

7 Conclusion

After experimenting with 11 different classification methods, we derived a final ensemble model using the top 9 classification model which has the best accuracy of 73.8% in predicting the video category. Using FP-Growth algorithm, we obtained a list of maximal frequent itemsets. The confidences of rules generated from these maximal frequent itemsets are at least 99.1%.

Further improvement on the video category prediction can be expected by using a larger dataset and deep learning models. Particularly, the use of bi-directional LSTMs with Bert embeddings is likely to yield better results on text classification, albeit requiring more computational time. Similarly, tag associations that are more general can be achieved with larger dataset.

8 References

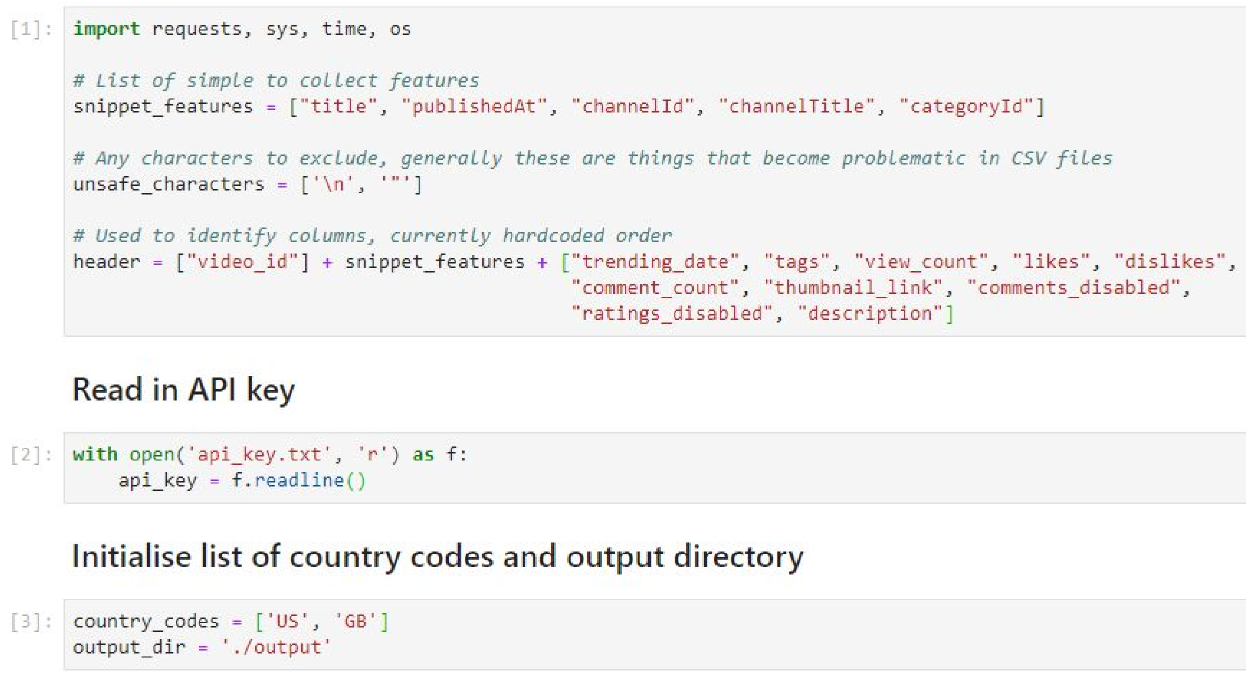
[1] Anonymous. YouTube - Wikipedia.https://en.wikipedia.org/wiki/YouTube.
[2]Maryam Mohsin. 10 Youtube Stats Every Marketer Should Know in 2019 - Oberlo. (2019,Jun). https://www.oberlo.com/blog/youtube-statistics.
[3][5] DataSnaek. “Trending YouTube Videos Scraper”. Retrieved from https://github.com/DataSnaek/Trending-YouTube-Scraper

[4] YouTube. YouTube Data API. Retrieved from https://developers.google.com/youtube/registering_an_application
[6] Kaggle. “Kaggle API”. Retrieved from https://github.com/Kaggle/kaggle-api#api-credentials
[7] scikit-learn developers. sklearn.feature_extraction.text.CountVectorizer. http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.
[8] Why Is Removing Stop Words Not Always a Good Idea. Lima Vallantin - https://medium.com/@limavallantin/why-is-removing-stop-words-not-always-a-good-idea-c8d35bd77214
[9] Porter Stemmer. Retrieved from http://people.scs.carleton.ca/~armyunis/projects/KAPI/porter.pdf
[10]Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning (Vol. 242, pp. 133-142).
[11] Han, J., Pei, J., & Yin, Y. (2000, May). Mining frequent patterns without candidate generation. In ACM sigmod record (Vol. 29, No. 2, pp. 1-12). ACM.

9 Appendix

9.1 Data Scraper

Some of the crucial portions of the scraper are as follows:



```
[1]: import requests, sys, time, os

     # List of simple to collect features
     snippet_features = ["title", "publishedAt", "channelId", "channelTitle", "categoryId"]

     # Any characters to exclude, generally these are things that become problematic in CSV files
     unsafe_characters = ['\n', '"']

     # Used to identify columns, currently hardcoded order
     header = ["video_id"] + snippet_features + ["trending_date", "tags", "view_count", "likes", "dislikes",
                                                  "comment_count", "thumbnail_link", "comments_disabled",
                                                  "ratings_disabled", "description"]
```

Read in API key

```
[2]: with open('api_key.txt', 'r') as f:
         api_key = f.readline()
```

Initialise list of country codes and output directory

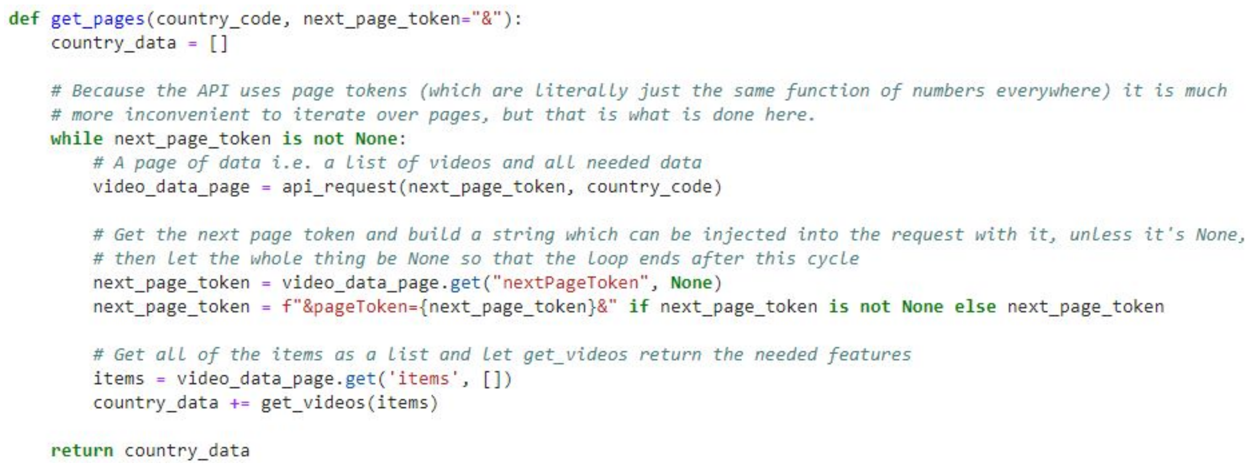
```
[3]: country_codes = ['US', 'GB']
     output_dir = './output'
```

Figure 9.1.1: Initialisation of the required variables

NOTES: *requests* is a non-standard library, and must be installed via *pip* or *conda*.

API key is stored in a .txt file

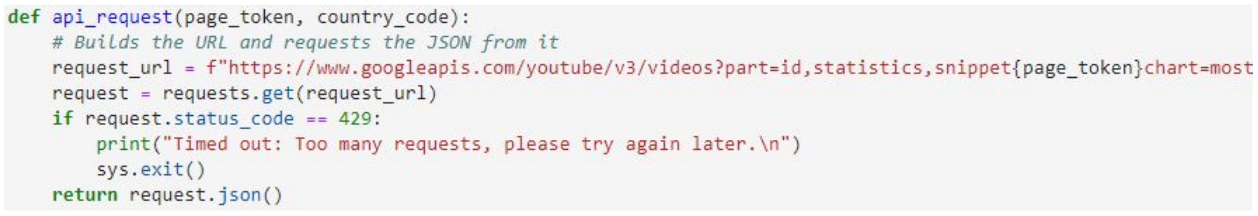
The noteworthy variables initialised are: *header* which columns to determine the output .csv format, *api_key* which contains the API key required for obtaining data from the YouTube Data API, and *country_codes* containing the input list of countries to scrape top trending daily videos.



```python
def get_pages(country_code, next_page_token="&"):
    country_data = []

    # Because the API uses page tokens (which are literally just the same function of numbers everywhere) it is much
    # more inconvenient to iterate over pages, but that is what is done here.
    while next_page_token is not None:
        # A page of data i.e. a list of videos and all needed data
        video_data_page = api_request(next_page_token, country_code)

        # Get the next page token and build a string which can be injected into the request with it, unless it's None,
        # then let the whole thing be None so that the loop ends after this cycle
        next_page_token = video_data_page.get("nextPageToken", None)
        next_page_token = f"&pageToken={next_page_token}&" if next_page_token is not None else next_page_token

        # Get all of the items as a list and let get_videos return the needed features
        items = video_data_page.get('items', [])
        country_data += get_videos(items)

    return country_data
```

Figure 9.1.2: Iterating through the page tokens & building *country_data*

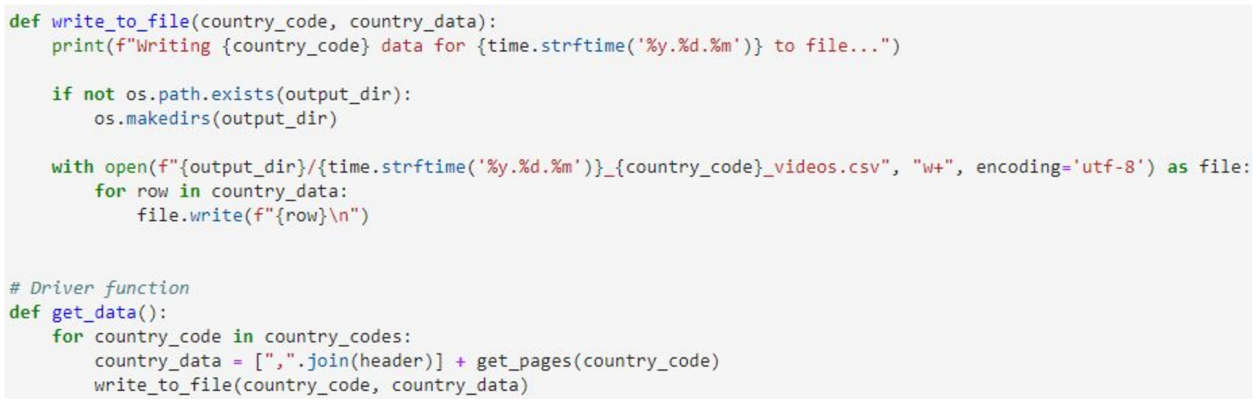
The code iterates through each page token and performs an api_request for each page, passing the next page token to be embedded in the request URL until the last page is reached. The data for the country is returned.



```python
def api_request(page_token, country_code):
    # Builds the URL and requests the JSON from it
    request_url = f"https://www.googleapis.com/youtube/v3/videos?part=id,statistics,snippet{page_token}chart=most
    request = requests.get(request_url)
    if request.status_code == 429:
        print("Timed out: Too many requests, please try again later.\n")
        sys.exit()
    return request.json()
```

Figure 9.1.3: Code block of the YouTube Data API via requests

The code block builds the request URL, embedding the page_token, api_key and the country_code of the country of interest. Requests are timed out if too many are made. A JSON object containing the required information is returned.



```python
def write_to_file(country_code, country_data):
    print(f"Writing {country_code} data for {time.strftime('%y.%d.%m')} to file...")

    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    with open(f"{output_dir}/{time.strftime('%y.%d.%m')}_{country_code}_videos.csv", "w+", encoding='utf-8') as file:
        for row in country_data:
            file.write(f"{row}\n")


# Driver function
def get_data():
    for country_code in country_codes:
        country_data = [",".join(header)] + get_pages(country_code)
        write_to_file(country_code, country_data)
```

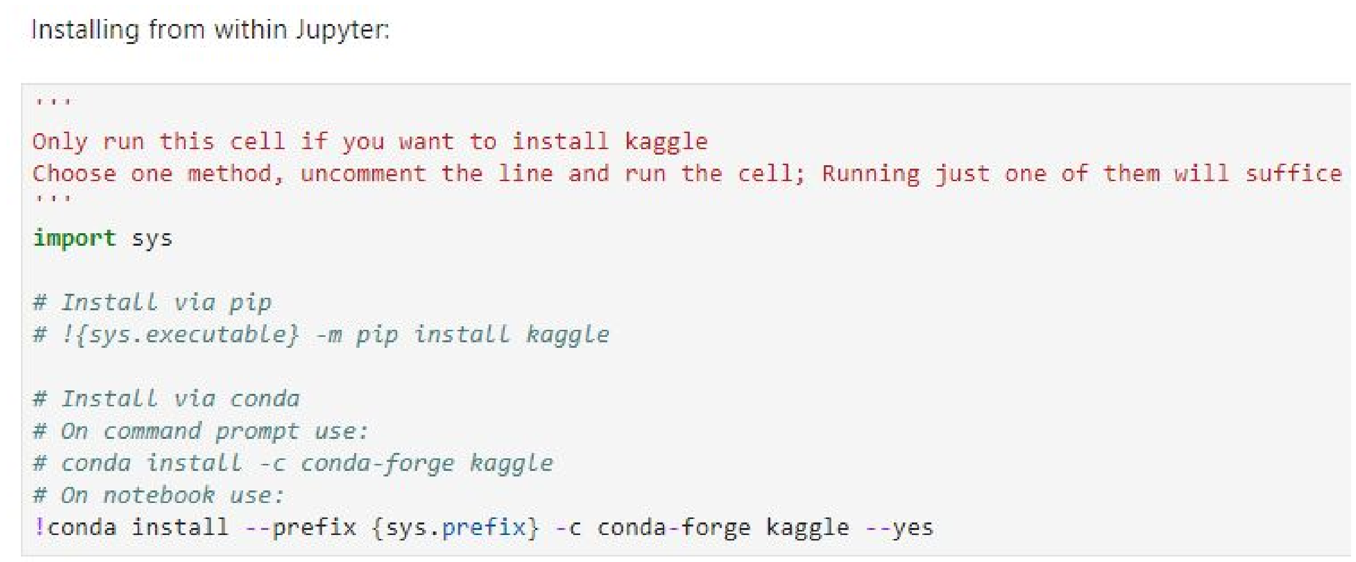
Figure 9.1.4: Writing the data into .csv files, organised by the headers defined

9.2 Downloading the Dataset via kaggle API

9.2.1 Installing kaggle

Installing from Command Prompt (recommended):

- conda install -c conda-forge kaggle **OR** pip install kaggle



Installing from within Jupyter:

```
'''
Only run this cell if you want to install kaggle
Choose one method, uncomment the line and run the cell; Running just one of them will suffice
'''
import sys

# Install via pip
# !{sys.executable} -m pip install kaggle

# Install via conda
# On command prompt use:
# conda install -c conda-forge kaggle
# On notebook use:
!conda install --prefix {sys.prefix} -c conda-forge kaggle --yes
```

Figure 9.2.1: Installing *kaggle* from within Jupyter, via *pip* or *conda*

9.2.2 Preparing API Credentials

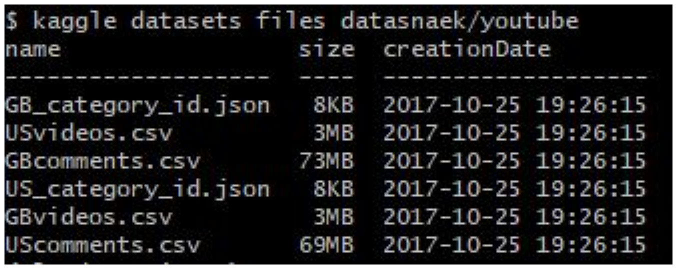
Important:

- To use the Kaggle API, sign up for a Kaggle account[1].
- Then go to the 'Account' tab of your user profile and select 'Create API Token'. This will trigger the download of *kaggle.json*, a file containing your API credentials.
- Place this file in the location ~/.kaggle/kaggle.json (on Windows in the location C:\Users\{your-username}\.kaggle\kaggle.json)

9.2.3 Downloading Datasets via a Command Line Interface

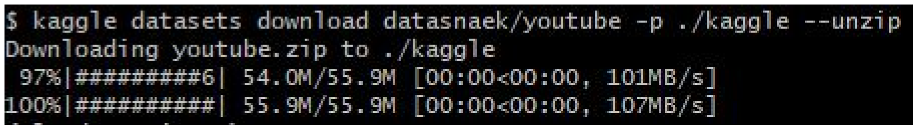
Enter 'kaggle datasets files datasnaek/youtube', we verify that the files available match those on the official kaggle dataset.[2]

[1] https://www.kaggle.com/
[2] https://www.kaggle.com/datasnaek/youtube



```
$ kaggle datasets files datasnaek/youtube
name                    size  creationDate
--------------------    ----  -------------------
GB_category_id.json      8KB  2017-10-25 19:26:15
USvideos.csv             3MB  2017-10-25 19:26:15
GBcomments.csv          73MB  2017-10-25 19:26:15
US_category_id.json      8KB  2017-10-25 19:26:15
GBvideos.csv             3MB  2017-10-25 19:26:15
UScomments.csv          69MB  2017-10-25 19:26:15
```

Enter ‘kaggle datasets download datasnaek/youtube -p <path to extract to> --unzip

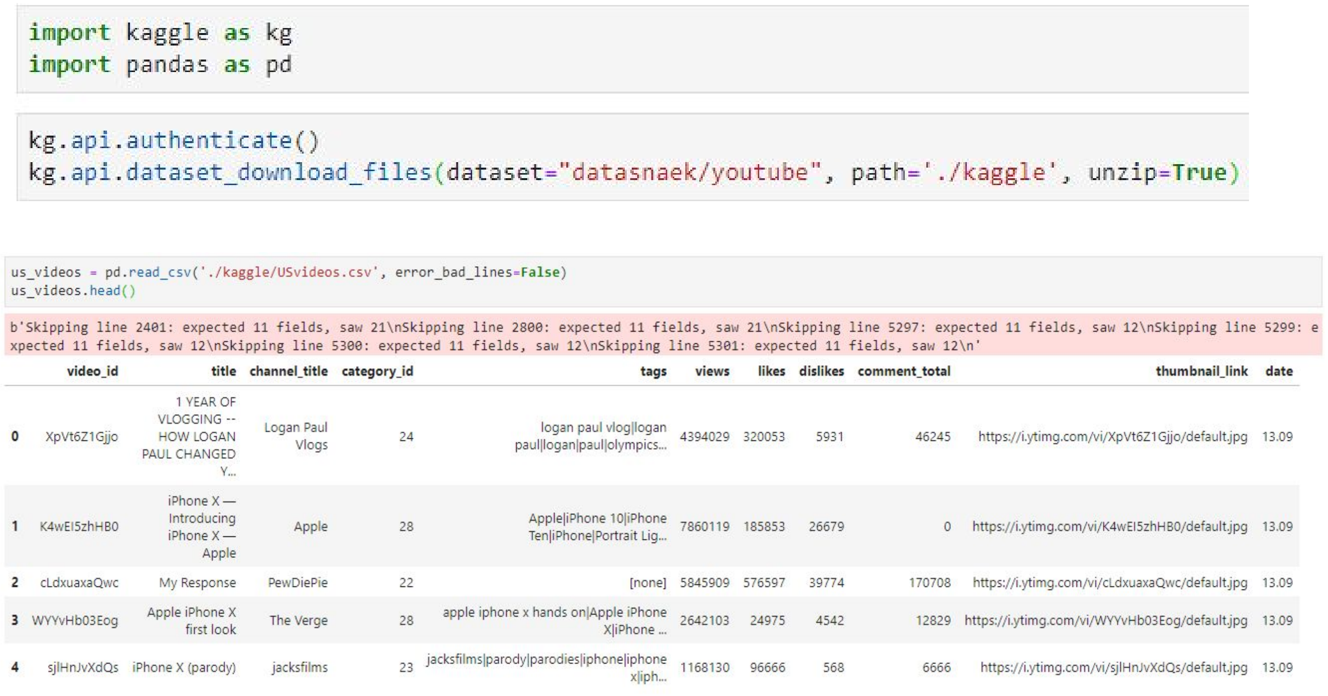


```
$ kaggle datasets download datasnaek/youtube -p ./kaggle --unzip
Downloading youtube.zip to ./kaggle
 97%|#########6| 54.0M/55.9M [00:00<00:00, 101MB/s]
100%|##########| 55.9M/55.9M [00:00<00:00, 107MB/s]
```

-p flag sets download path; *--unzip* to unzip all files

9.2.4 Downloading Datasets via a Jupyter Notebook

Refer to the ‘kaggle.ipynb’ in the ‘data_scraper’ folder for a notebook prepared, for the purpose of downloading the dataset.



```python
import kaggle as kg
import pandas as pd
```

```python
kg.api.authenticate()
kg.api.dataset_download_files(dataset="datasnaek/youtube", path='./kaggle', unzip=True)
```

```python
us_videos = pd.read_csv('./kaggle/USvideos.csv', error_bad_lines=False)
us_videos.head()
```

```
b'Skipping line 2401: expected 11 fields, saw 21\nSkipping line 2800: expected 11 fields, saw 21\nSkipping line 5297: expected 11 fields, saw 12\nSkipping line 5299: e
xpected 11 fields, saw 12\nSkipping line 5300: expected 11 fields, saw 12\nSkipping line 5301: expected 11 fields, saw 12\n'
```

	video_id	title	channel_title	category_id	tags	views	likes	dislikes	comment_total	thumbnail_link	date
0	XpVt6Z1Gjjo	1 YEAR OF VLOGGING -- HOW LOGAN PAUL CHANGED Y...	Logan Paul Vlogs	24	logan paul vlog|logan paul|logan|paul|olympics...	4394029	320053	5931	46245	https://i.ytimg.com/vi/XpVt6Z1Gjjo/default.jpg	13.09
1	K4wEI5zhHB0	iPhone X — Introducing iPhone X — Apple	Apple	28	Apple|iPhone 10|iPhone Ten|iPhone|Portrait Lig...	7860119	185853	26679	0	https://i.ytimg.com/vi/K4wEI5zhHB0/default.jpg	13.09
2	cLdxuaxaQwc	My Response	PewDiePie	22	[none]	5845909	576597	39774	170708	https://i.ytimg.com/vi/cLdxuaxaQwc/default.jpg	13.09
3	WYYvHb03Eog	Apple iPhone X first look	The Verge	28	apple iphone x hands on|Apple iPhone X|iPhone ...	2642103	24975	4542	12829	https://i.ytimg.com/vi/WYYvHb03Eog/default.jpg	13.09
4	sjlHnJvXdQs	iPhone X (parody)	jacksfilms	23	jacksfilms|parody|parodies|iphone|iphone x|iph...	1168130	96666	568	6666	https://i.ytimg.com/vi/sjlHnJvXdQs/default.jpg	13.09

Figure 9.2.4: Importing *kaggle*, authenticating and downloading datasets into a folder

NOTE: *kaggle.json* must be placed in its proper location prior