**Categorizing YouTube Videos**

**Capstone 1 Report**

**Introduction**

**Problem Statement:**

300 hours of videos are uploaded to YouTube every minute and 5 billion videos are watched every day. In this huge amount of content, it is very tough to know what makes the video stand out and be the trending or most watched video. This project is intended to provide some insights into the factors that influence in making the videos famous.

**Client:**

The client for this would be companies trying to advertise their products, individuals looking to maximize their monetization for videos.

**Data:**

The dataset used for this project is from Kaggle: <https://www.kaggle.com/datasnaek/youtube-new>

**Tools used:**

This project is done using python language using Jupyter notebook. Many of the machine learning, data wrangling libraries are used. The whole code can be found [here](https://github.com/nikhilsimha85/springboard-ds/tree/master/Capstone-Project1).

**Data Wrangling**

**Data retrieval:**

The CSV file available on Kaggle is downloaded and loaded into the notebook using pandas.

**Data Cleaning:**

Using the simple analysis on pandas dataframe, we can see that some of the columns don’t have the right data type. They have been transformed to the correct data types which will help in doing EDA.

Some of the rows have null values and since the null value rows are very less, they have been dropped leaving us with still a decent number of records in the dataset.

Pandas profiling library gives many important statistics and as per it, we see that there are some duplicate rows and the data is very skewed. The duplicate rows are dropped.

For the skewed data analysis, lets take one column views and do some analysis based on it. There are more than 3700 outliers (views more than 1.5 times the inner quartile from the 75th percentile) based on views. These outliers don’t look like they are bad data. Some more analysis on these outliers shows that they mostly belong to specific category id and channels. These analysis might be handy during the next steps in the project. We might need to deal with this outliers accordingly based on what kind of analysis we are performing. The skewed data is handled directly during the EDA and machine modeling.

A new column publish\_day is added which represents the day of the week the video was published.

**Final features:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Description** | **Data Type** | **Useful in prediction?** |
| trending\_date | The date video started trending | Datetime | No |
| title | Title of the video | Object | No |
| channel\_title | Title of the channel | Object | No |
| category\_id | Category the video | Category | Yes |
| publish\_time | Date and time when video was uploaded | Datetime | No |
| tags | Tags to search video | Object | No |
| views | Number of views | Int | Yes |
| likes | Number of likes | Int | Yes |
| dislikes | Number of dislikes | Int | Yes |
| comment\_count | Number of comments | Int | Yes |
| thumbnail\_link | Any links in description | Object | No |
| comments\_disabled | If comments are disabled or not | Bool | No |
| ratings\_disabled | If ratings are disabled | Bool | No |
| video\_error\_or\_removed | If video is erroring out or removed | Bool | No |
| description | Detailed description of the video | Object | No |
| publish\_day | Day of the week video was published | Object | No |

The cleaned data is stored using pickle so that it can be used for further analysis.

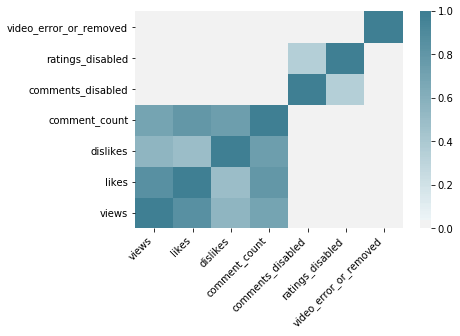
**Data Exploration**

EDA is done using visual analysis. The purpose of doing this is to better understand how category of the video, likes, dislikes, comments, the date etc influences how many views the video gets. The number of views is an important measure of video’s popularity and also determines how individual youtubers are paid.

**Visual analysis:**

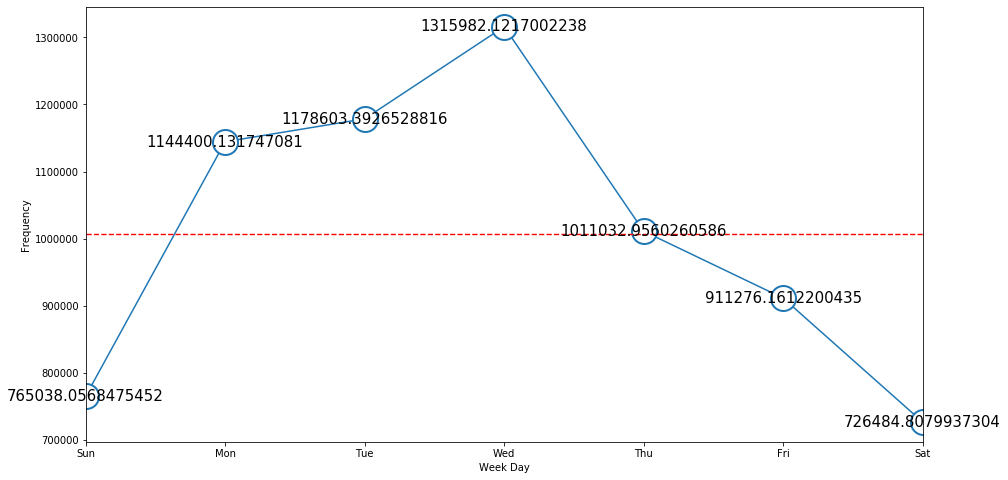
The data is very skewed with respect to likes, dislikes and views. The correlation between views and likes is 0.85 which is as expected since more likes will most likely lead to more views.

Lets look at a heatmap of correlation between the features to get a general idea of the dataset.



Here we can see that likes and views have a very strong correlation followed by likes and comment\_count.

We added a new column previously for the day of the week the video is published. Lets look at how many views videos get based on published day of the week.



We see that videos published during the week get more views than those published during the weekend.

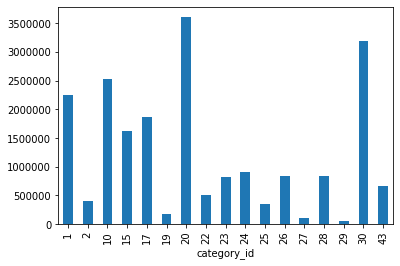
We can also see below that when comments are disabled, people generally don’t choose to like or dislike the video.(table shows the mean values)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Views | Likes | Dislikes |
| comments\_disabled |  |  |  |
| False | 1.027022e+06 | 26759.984815 | 1631.947042 |
| True | 4.283319e+05 | 2562.452952 | 575.595941 |

Similarly we see that when ratings are disabled then people don’t comment much on the video. (table shows the mean values)

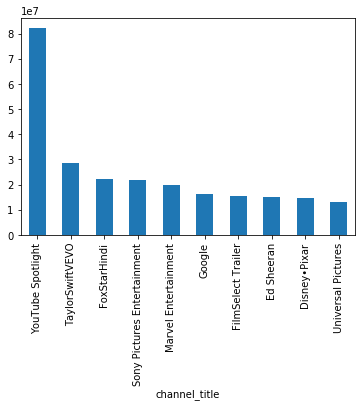
|  |  |  |
| --- | --- | --- |
|  | Views | Comment\_count |
| Ratings\_disabled |  |  |
| False | 1.022784e+06 | 2612.797383 |
| True | 2.811639e+05 | 196.657475 |

Lets now look at the category\_id and see which category of videos have maximum average number of the views



We can see the categories 10, 20 and 30 get more views in average.

Below is the histogram for the top 10 channel titles by average number of views.



**Data Modeling**

**Preprocessing:**

In data modeling stage, we first need to get the data in right format which can be run through machine learning models in sklearn library. We dropped all the datetime fields as well as text fields. Then, convert all the category features to integer using labelencoder. The final features list which is used for data modeling are below:

category\_id 32562 non-null int64

likes 32562 non-null int64

dislikes 32562 non-null int64

comment\_count 32562 non-null int64

comments\_disabled 32562 non-null bool

ratings\_disabled 32562 non-null bool

video\_error\_or\_removed 32562 non-null bool

publish\_day 32562 non-null int64

views\_category 32562 non-null int64

Also, since the data is very skewed, we create a new views\_category variable which is to divide the dataset based on number of views equally into 10 different buckets.

Since number of views is a numeric field which is very skewed and is not interpretable, we use the views\_category variable as independent feature which we try to predict using various models.

Finally before running the models, we convert the dataset into training data and testing data in ratio of 80:20.

**Cross Validation:**

For optimization of hyper parameters, we use cross validation for all below models using GridSearchCV. We also do K-fold cross validation by splitting the dataset randomly into training and testing data during cross validation. All these techniques improve hyper parameter tuning which will yield better result.

We use F1-score as a measure of accuracy of the models. It considers both the precision and the recall of the test.

Precision = tp

tp + fp

Recall = tp

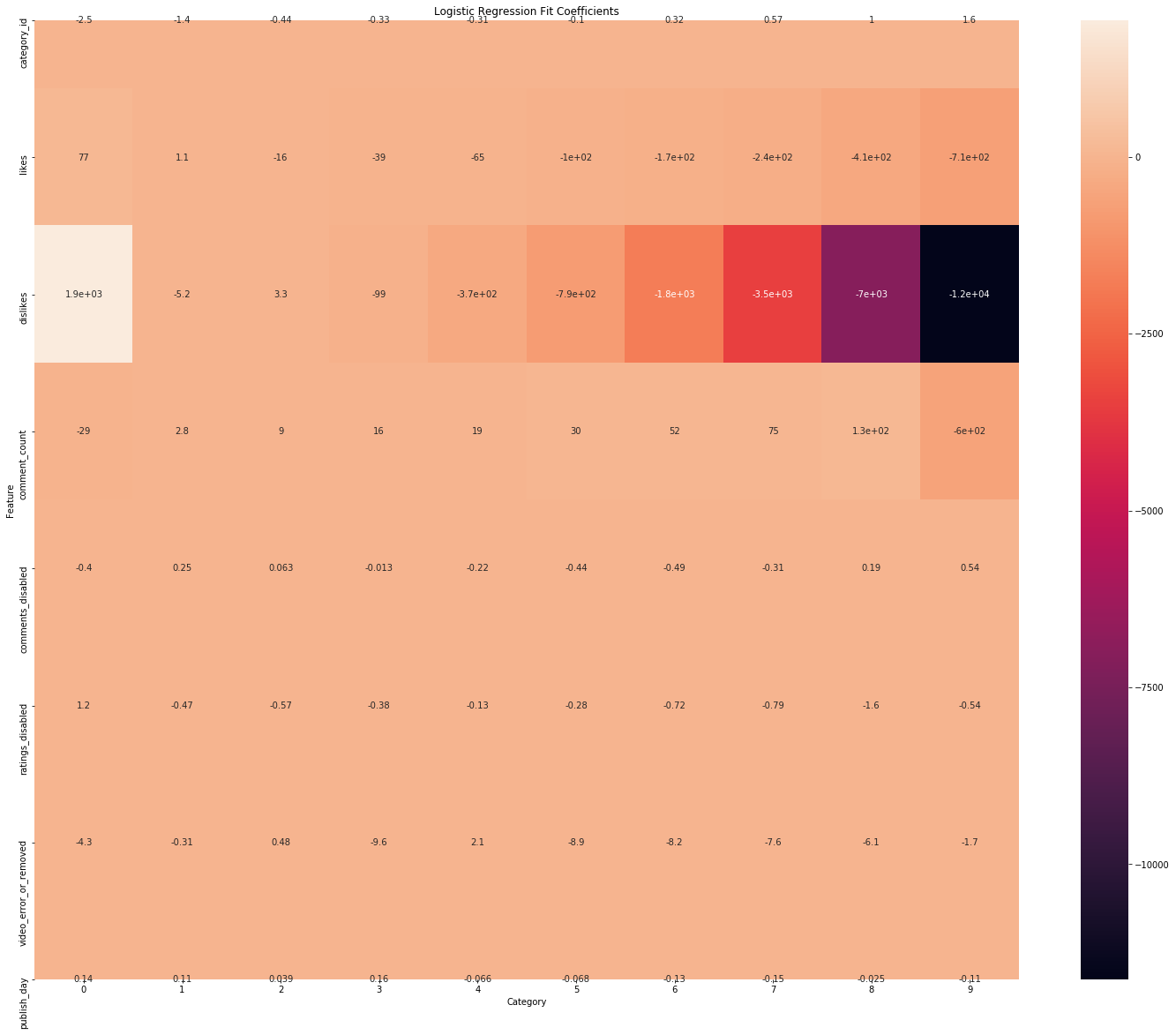
tp + fn

where tp = true positive, fp = false positive and fn = false negative.

**Logistic Regression:**

Since the dataset is skewed, we first need to scale the features. This is done using MaxAbsScaler which will scale each feature such that the maximum value of each feature is 1.0.

We also setup a parameterized grid search for the hyper parameter C which is inverse of regularization strength. Smaller value specify stronger regularization. We finally do a k-fold grid search as explained above. Using the best F1 score, we get a 25% accuracy.



From the heatmap, we can see that dislikes is the most significant feature. All other features seem irrelevant in this model.

**KNN:**

Next, we use KNN model. Here, we again use parameter grid search for the hyperparameters n\_neighbors which represent the number of neighbors to be used for determining the category and p which takes a value of 1(Manhattan distance) or 2(Euclidean distance).

We run the model through cross validation and using F1 score, we get accuracy of 47% which is a significant improvement over the logistic regression model. The best result is obtained with 5 neighbors using the Euclidean distance.

**SVC:**

Support vector classification is based on support vector machine concept for multi-class classification. The advantage of a SVC model is it is very memory efficient and effective in high dimensional spaces (we have many features).

We use the radial basis function kernel which avoids overfitting. The parameter grid search is carried out for gamma which is kernel coefficient for rbf.

We also carry out grid search for C, regularization parameter.

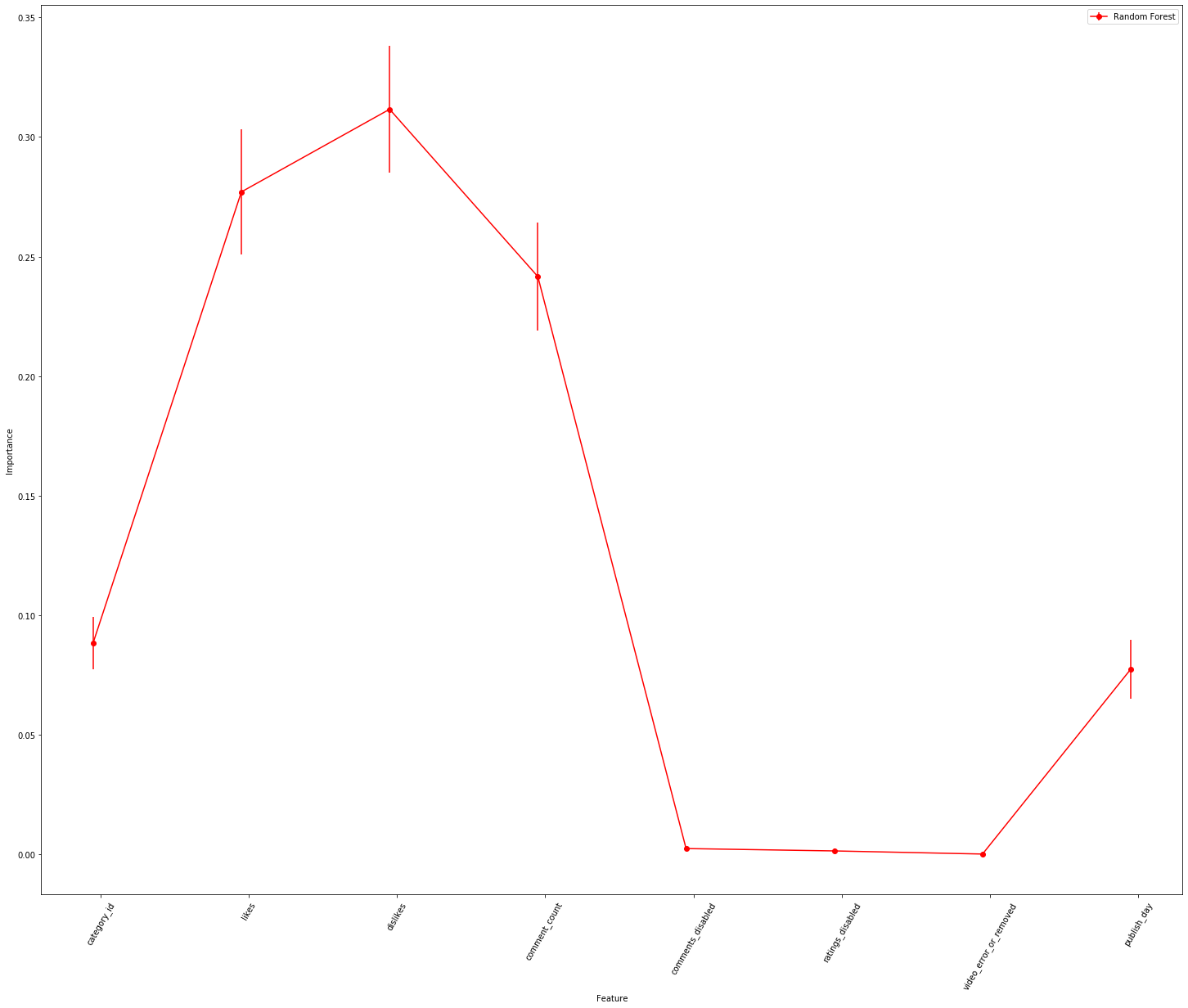
Using F1 score, we get a best accuracy of 31% which is lower than KNN model.

**Random Forest:**

Finally we use an ensemble method of random forest. It is an ensemble of many decision tree classifiers which improves accuracy while controlling over-fitting. Parameter tuning is done for n\_estimators (number of trees in the forest) and max\_features (number of features to be used for each split the trees).

Using the F1 score, we get the best accuracy of 48% which is better than all other models.

Using the below plot, we can see than likes, dislikes and comment\_count are the most significant features in this model.



**Conclusion**

With the multiple models that are built, lets summarize the final scores of each of these models.

|  |  |
| --- | --- |
| Model | F1 score |
| Logistic regression | 0.2515 |
| K-nearest neighbors | 0.4692 |
| Support Vector Classification | 0.3092 |
| Random Forest | 0.4817 |

The ensemble random forest model performs better than all other models. Lets look at confusion matrix to see how the model performs for each category.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P**  **R**  **E**  **D**  **E**  **C**  **T**  **I**  **O**  **N** | **ACTUAL** | | | | | | | | | | |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 3766 | 104 | 20 | 7 | 0 | 1 | 0 | 2 | 1 | 0 |
| 2 | 99 | 2987 | 143 | 14 | 18 | 4 | 6 | 2 | 2 | 0 |
| 3 | 20 | 124 | 3531 | 80 | 48 | 21 | 14 | 13 | 5 | 2 |
| 4 | 16 | 37 | 152 | 2630 | 107 | 57 | 33 | 11 | 3 | 1 |
| 5 | 2 | 23 | 78 | 128 | 2692 | 107 | 64 | 43 | 18 | 2 |
| 6 | 0 | 31 | 48 | 42 | 99 | 2624 | 108 | 88 | 20 | 3 |
| 7 | 0 | 8 | 19 | 33 | 58 | 109 | 2660 | 133 | 55 | 6 |
| 8 | 0 | 8 | 20 | 29 | 55 | 68 | 120 | 3260 | 144 | 12 |
| 9 | 0 | 1 | 8 | 10 | 17 | 29 | 48 | 180 | 3182 | 66 |
| 10 | 0 | 4 | 0 | 0 | 3 | 8 | 8 | 25 | 124 | 1745 |

We can see from the confusion matrix that most of the wrong predictions are to the previous or the next category than the actual. This might be because we randomly categorized the views to have equal frequency across each category but the dataset might be having natural breaks in the data which is being ignored.

Further work to improve the project could be to use K-means clustering to convert the views into categories considering inherent breaks in the data. Also, we can do PCA for dimensionality reduction. We can also add data from other sources or create a continuous stream of incoming data using the youtube API. This can provide continuous improvements to the model with real time data.

Finally, we can run many other machine learning models and artificial neural network models.