**Advances In Data Science**

**Youtube Network Analysis**

**Final Project**

**INFO 7250**

A project by Team1

Vishal Satam & Manasi Dalvi

Under the guidance of Prof. Srikanth Krishnamurthy

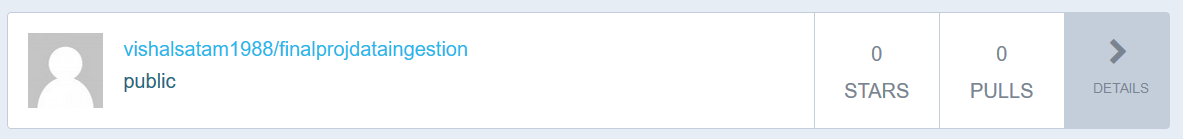
**Summary**

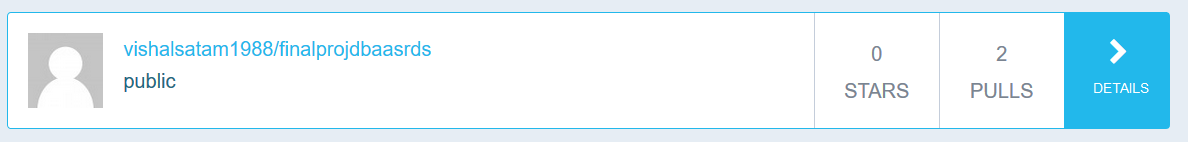
The University of Simon Fraser has performed a crawl of the youtube videos data from the public API available by Youtube. Since there is a rate limit on the API, the researchers had to perform the crawl over 2 months. For this Network Analytics project, we have examined the network of videos created which can be derived from the dataset. We perform exploratory data analysis on this network to understand the features of this dataset and to provide a framework for performing predictive modelling. We have also analyzed features of the dataset such as page rank and the in degree centrality measures to understand how our network is setup. Hadoop along with Python and Spark were used to perform compute intensive tasks. These created flat files which in turn were used to populate databases on AWS cloud which could be used to serve the final web application. Neo4J and AWS – RDS were used as data stores to serve as relational as well as graph oriented data stores. Predictive Modelling was performed on the videos dataset to estimate the views which a video can generate. We used the videos dataset to derive a set of summary features for the uploaders which in turn could enable us to form clusters of users based on some interesting features which we derived from the network of vides. A web application was designed for the user to easily access and view his summary features as well as an interface to predict the views which his videos can generate. We also inform the user of what cluster he belongs to, i.e whether he is a regular user or an inactive user.

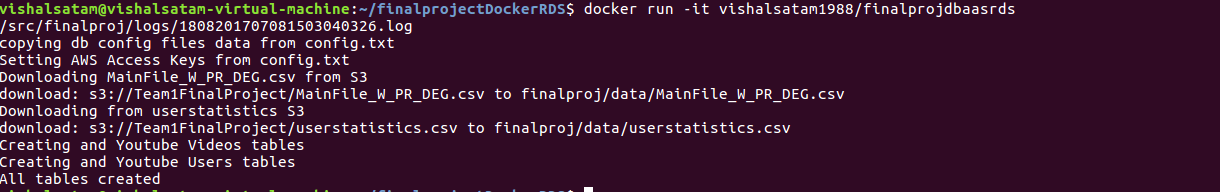
**DOCKER**

2 Docker images have been created for this project. One for the purpose of Data Ingestion and Wrangling, the other for the purpose of creating and deploying databases as a service.

The Docker related instructions and execution details have been provided on the github repository.







**Dataset**

The dataset consists of 2 types of flat files which are available for download. The website consists of downloadable links for these files. The main table is the Videos table.

The Videos table consists of details about each video from about 40 different web crawls. The video features are as follows

|  |  |
| --- | --- |
| video ID | an 11-digit string, which is unique |
| uploader | a string of the video uploader's username |
| age | an integer number of days between the date when the video was uploaded and Feb.15, 2007 (YouTube's establishment) |
| category | a string of the video category chosen by the uploader |
| length | an integer number of the video length |
| views | an integer number of the views |
| rate | a float number of the video rate |
| ratings | an integer number of the ratings |
| comments | an integer number of the comments |
| related IDs | up to 20 strings of the related video IDs |

The user’s tabe has the following features

|  |  |
| --- | --- |
| userID | an 11-digit string, which is unique |
| uploads | Number of uploads by this user |
| friends | Number of friends (subscriber’s) this user has |

The related links (max 20) contained in the dataset form a network of videos. Every video has one associated video.

We create a directed graph with links coming from a video to it’s related videos.

**Data Ingestion And Wrangling**

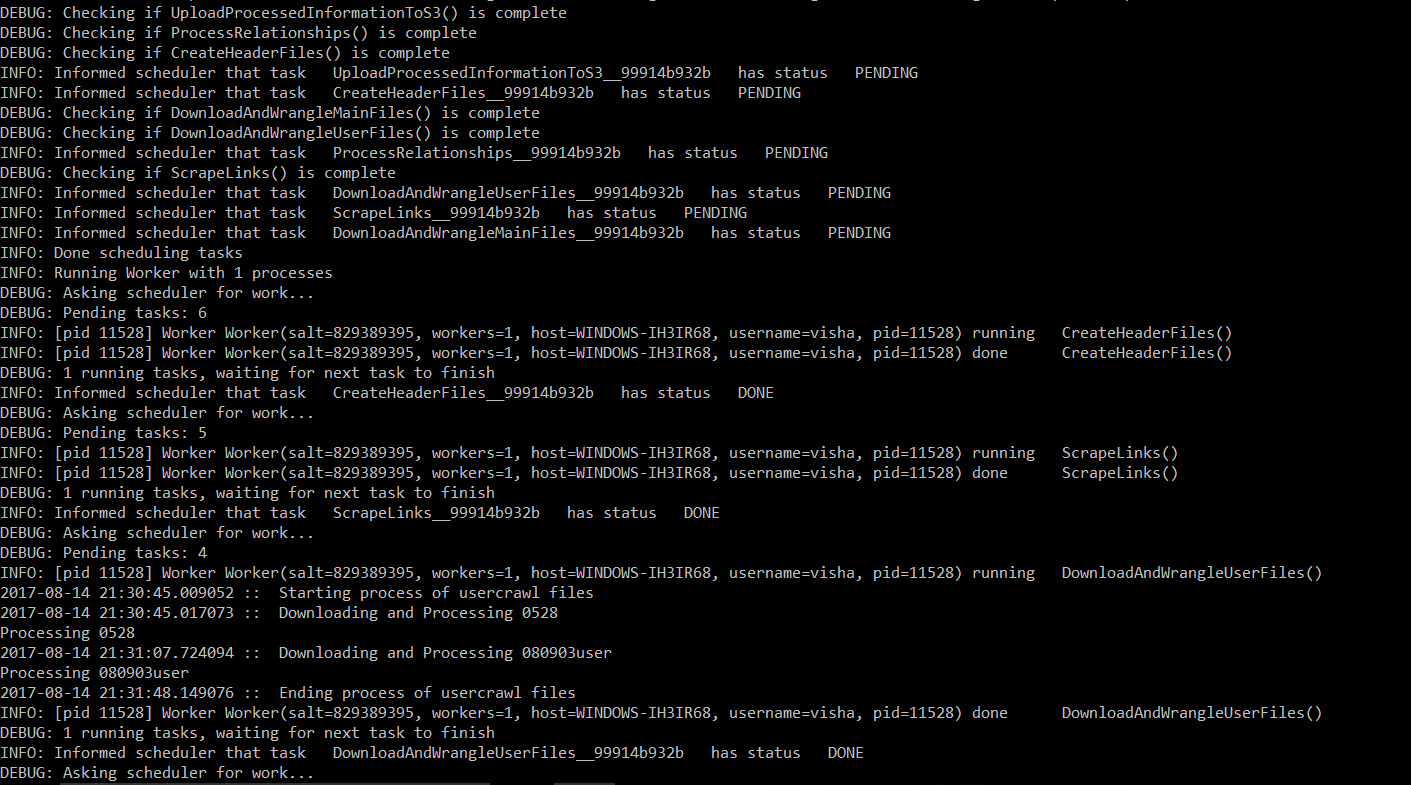
The data ingestion phase happens in a luigi pipeline as illustrated in the below image

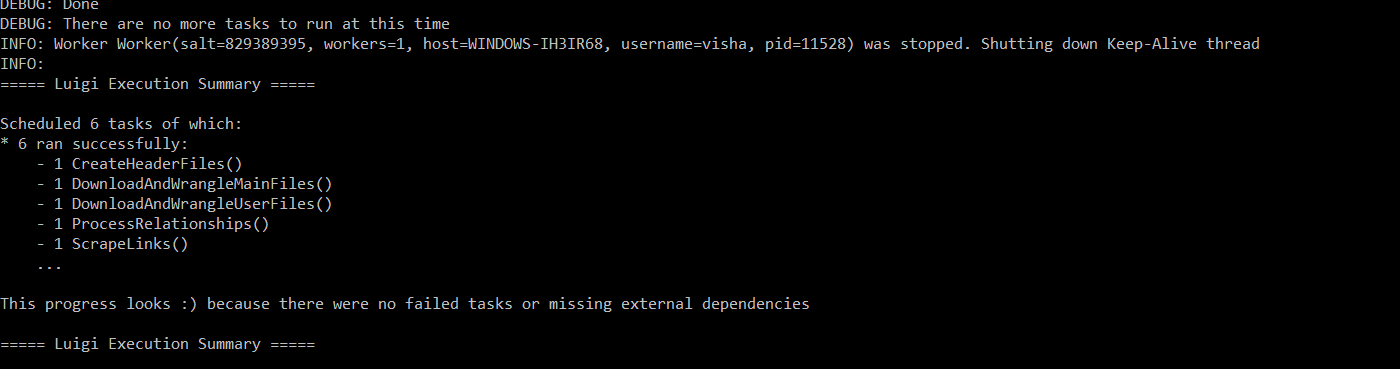
We first scrape links from the web page and download the files using the Beautiful Soup library of Python. After scraping the links, we download the corresponding files from these URL’s. We then perform cleaning and transformation using the pipeline. After the data cleaning and wrangling phase, the cleaned and transformed data is uploaded to Amazon S3 for storage.

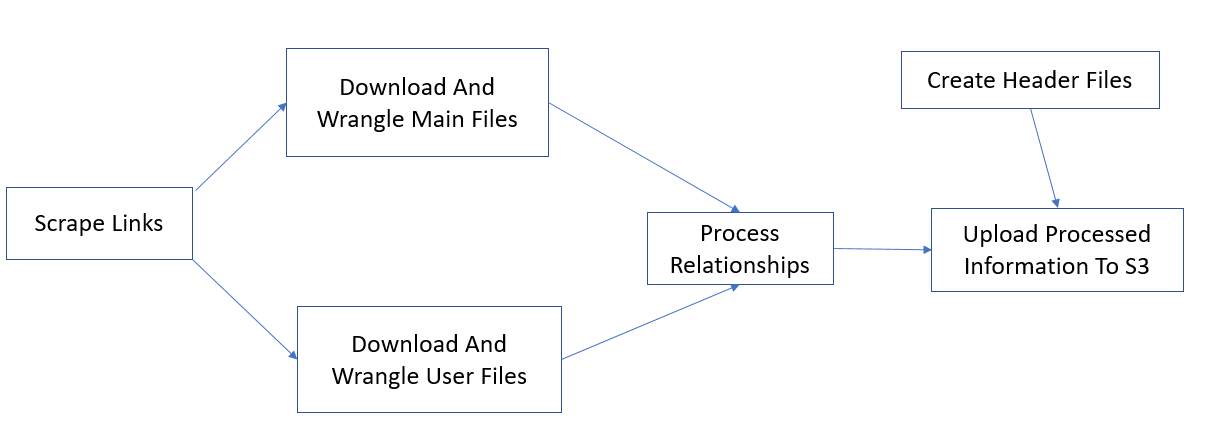
We have also written a Map Reduce job to perform aggregations on the videos file to compute summaries for the user.

**Pipeline**

Luigi was used as the pipelining tool for building the data ingestion, wrangling and transformation pipeline. The process relationships phase converts the videos file into a file which maintains relationships of nodes as from and to nodes.



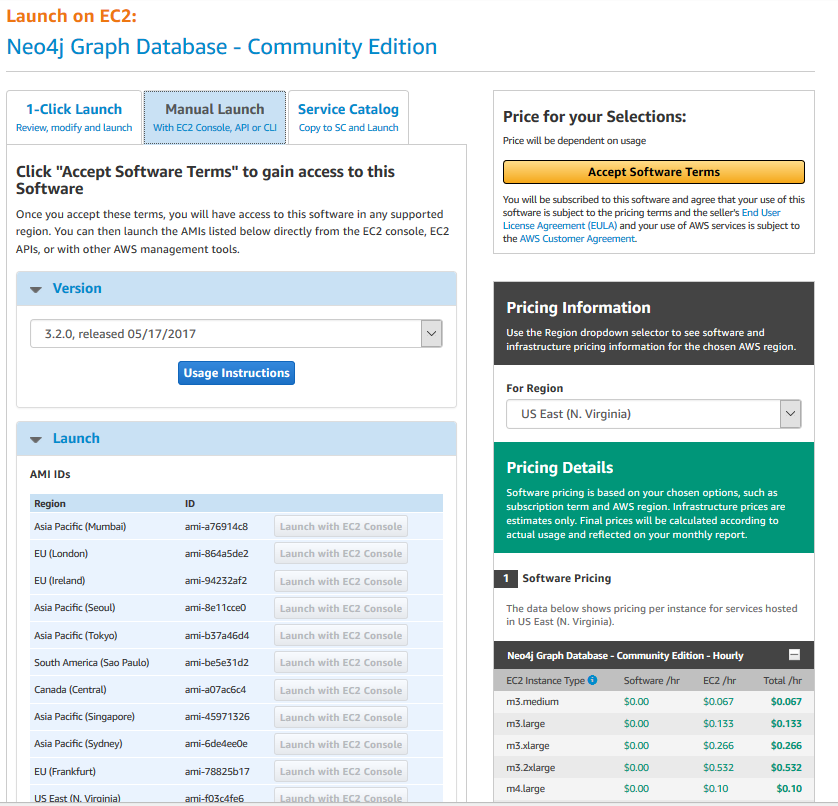




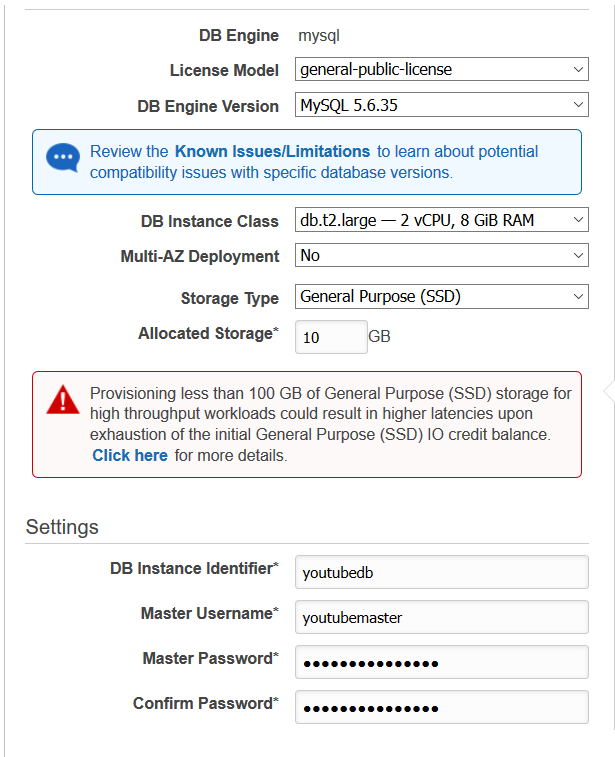
**Database**

We used Neo4J and Amazon RDS for persistence and stored data on these databases to be served to the application. RDS was used because we had derived summary files which were relational in nature.

For creating the Neo4J database on the cloud environment, you will have to select the Neo4J’s AMI and then create this on EC2.



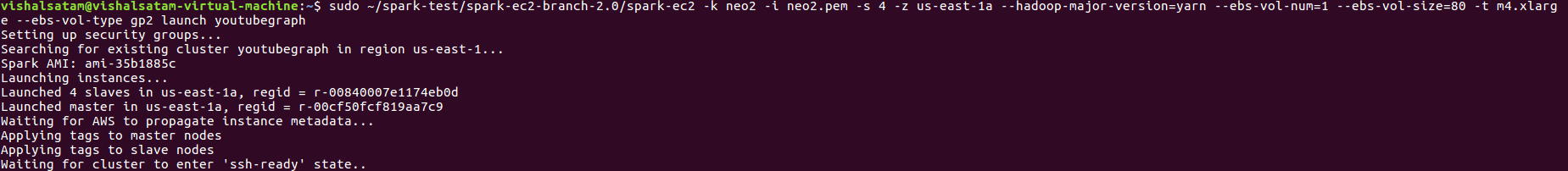
The Amazon RDS database was created to store the user statistics and videos data.

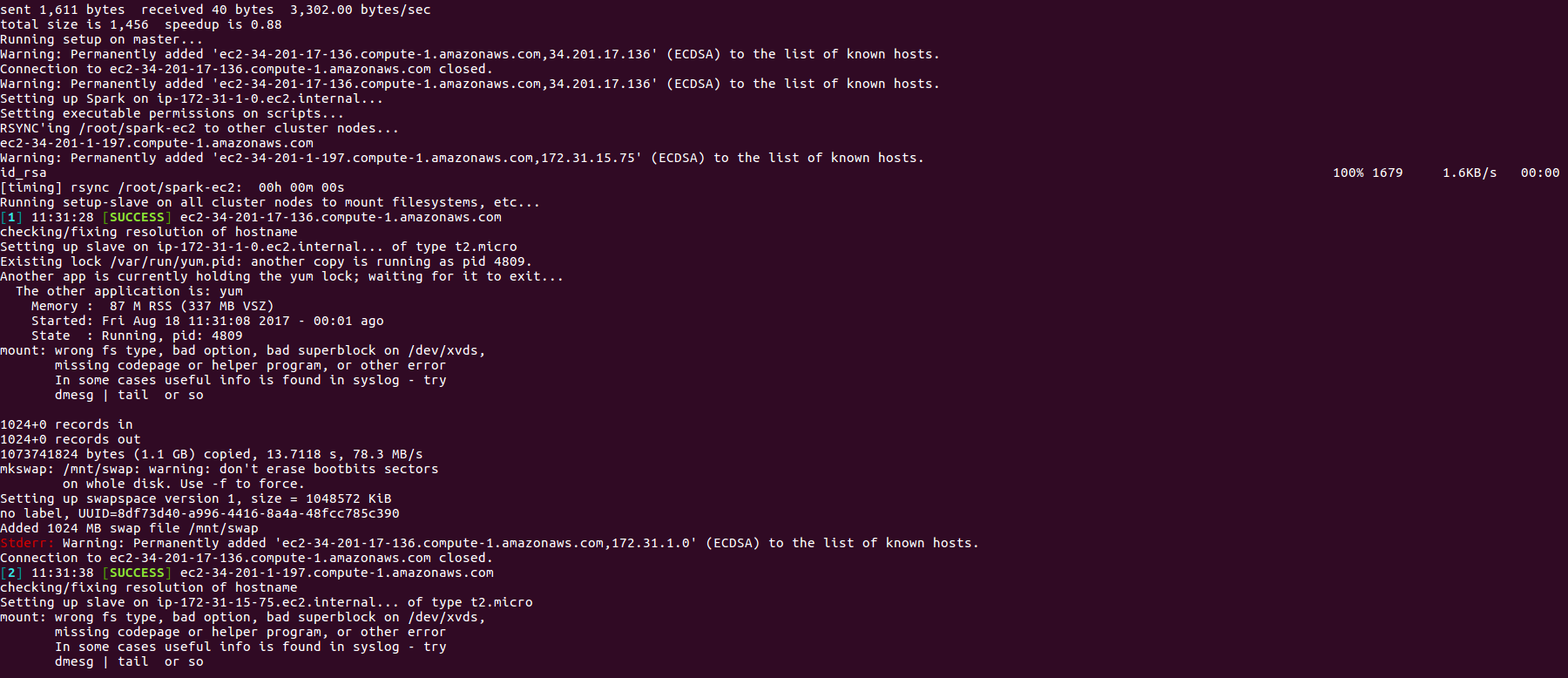


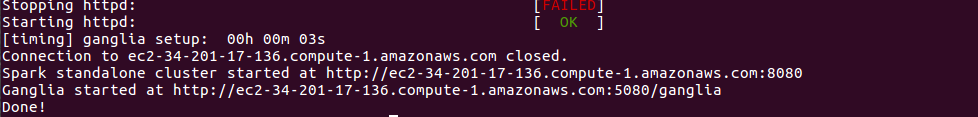
**SPARK**

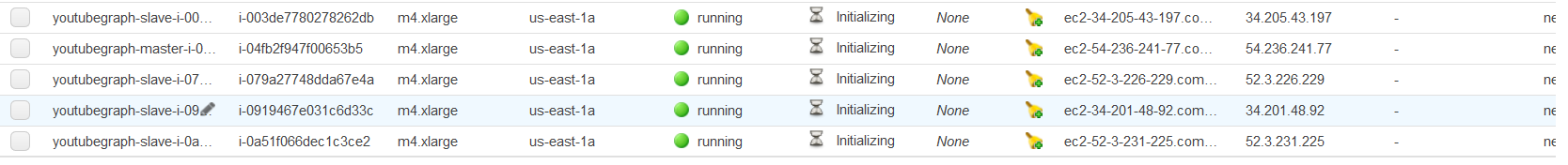
The most important measure in a network type of dataset is the Pagerank which is a measure of it’s Centraity. Pagerank algorithm calculates the score for each page based on the links that come into it. The weights provided by every node are aggregated and the final score is calculated till it converges.

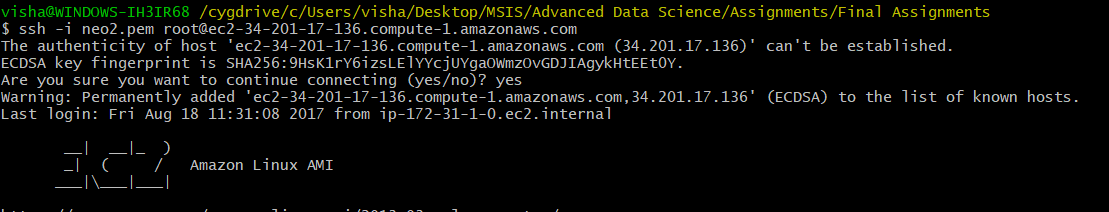
We created and used a 4 node cluster of m4.large EC2 machines using the python’s ec2 package to create a distributed spark environment to compute the pagerank for all the nodes. This was then appended as a column to the main videos file and uploaded to S3.

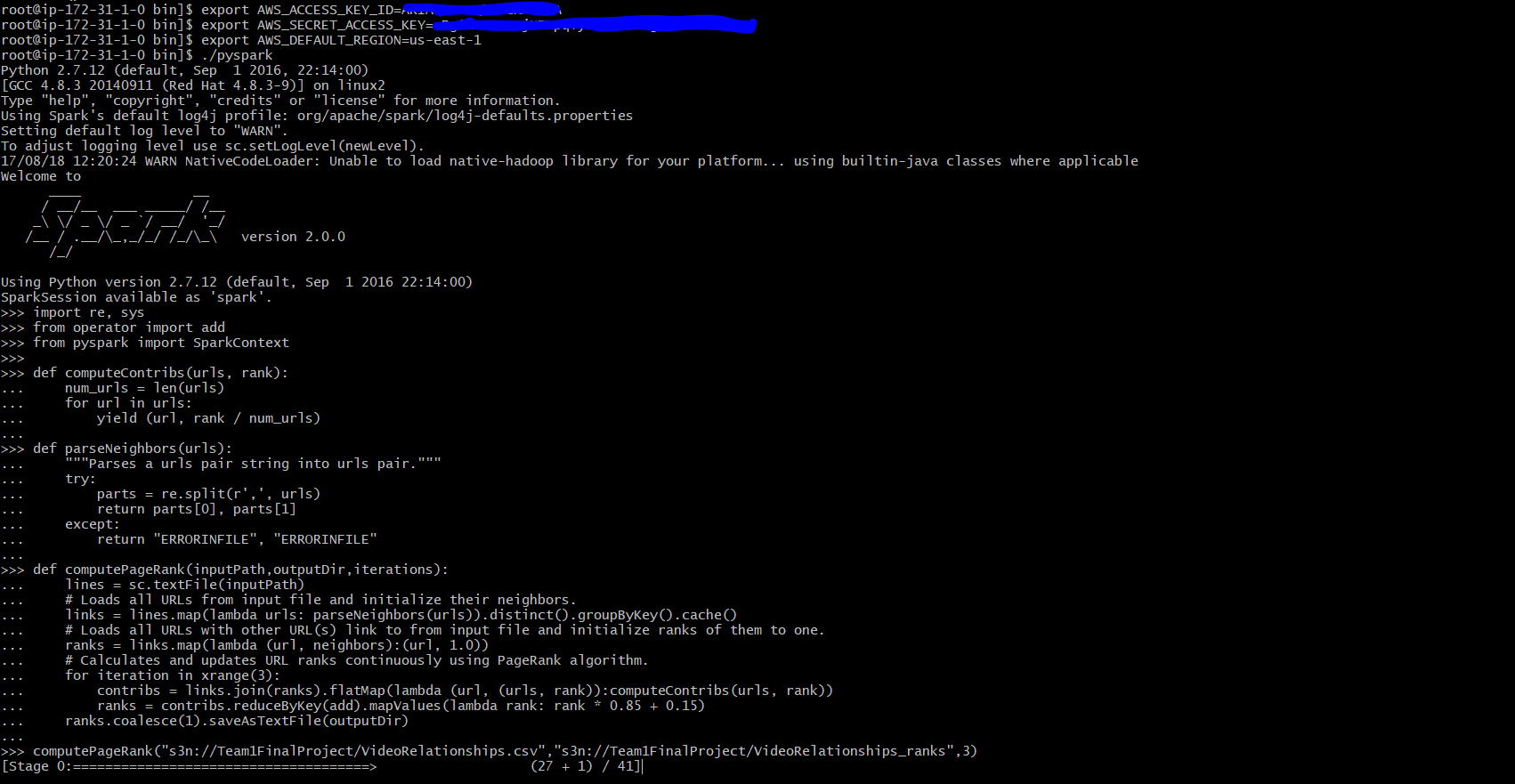






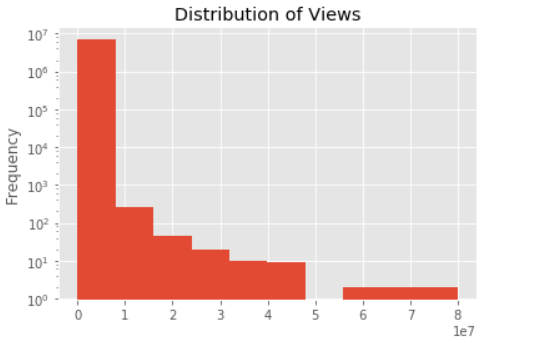




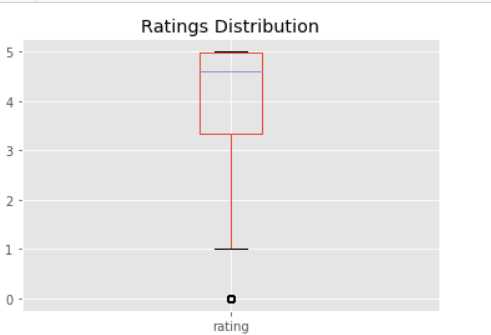


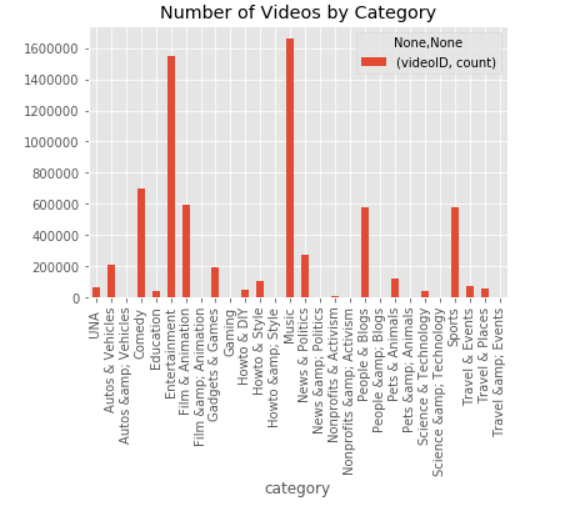
We also computed the in degree centrality measure of the graph for all the nodes using neo4j’s library.

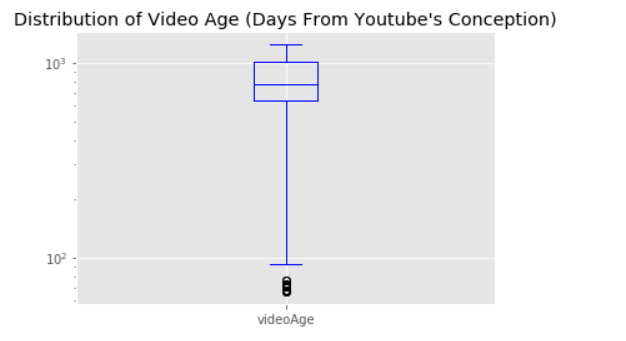
**Exploratory Data Analysis**

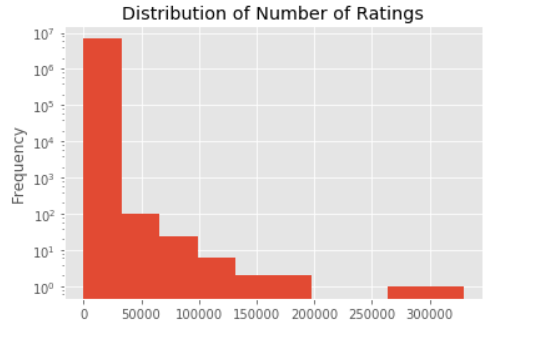


Most of the Features in this dataset follow a power law distribution





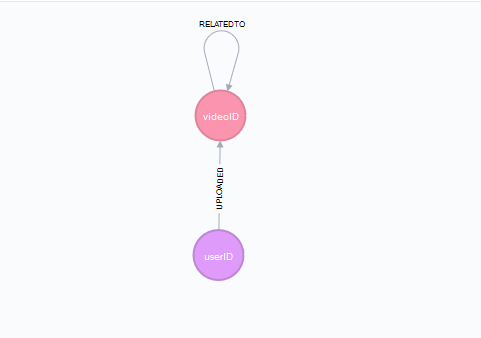




**Network Data Model**

We are using Neo4 J which has been hosted on a large EC2 instance. Using the following query we can retrive the structure of out nodes and relationships. In out dataset, the videos are the nodes of the network. They have their links of 20 related videos. Every video has an uploader or user. VIDEO is RELATEDTO other VIDEO UPLOADED by USER

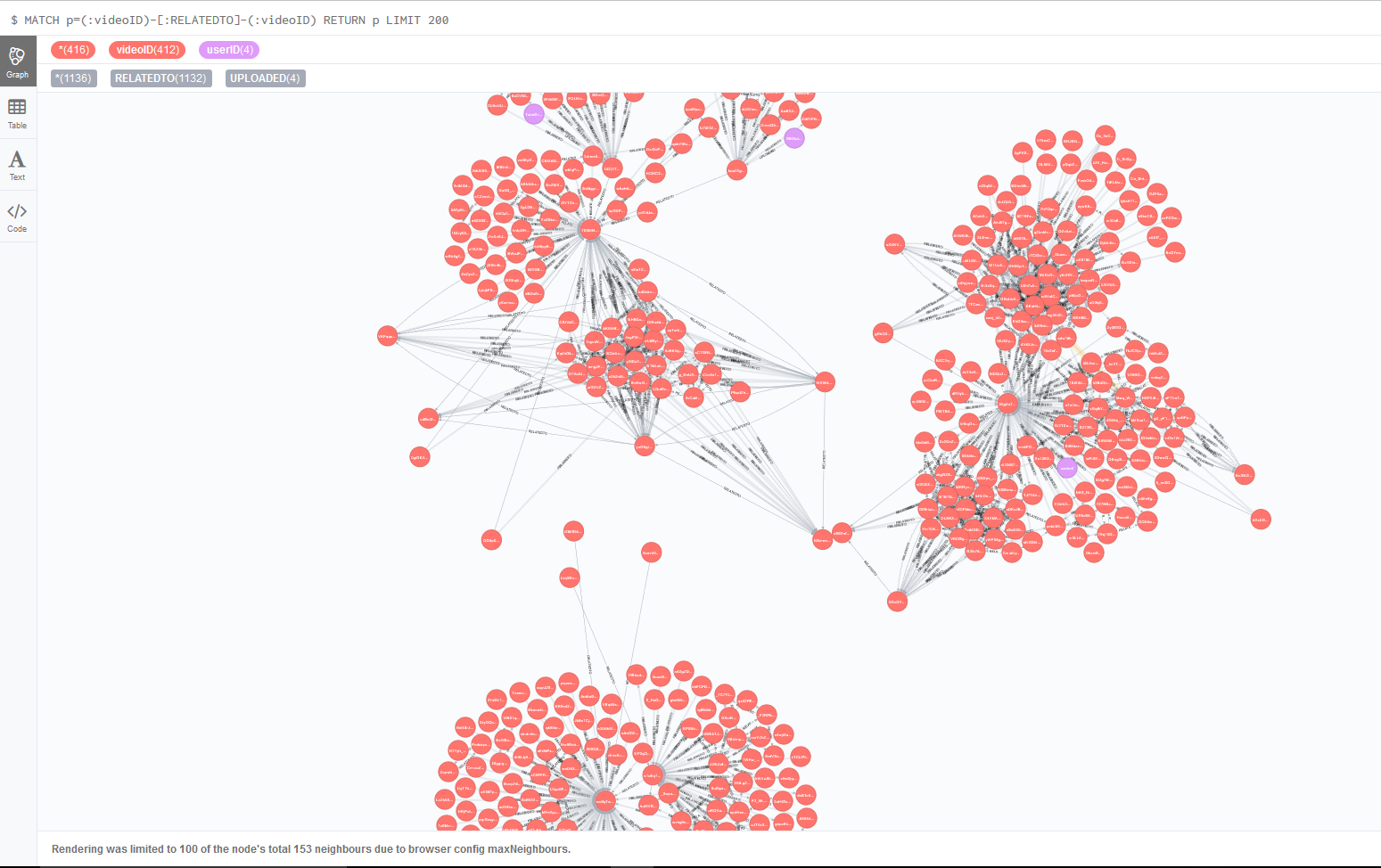




We can visualize a network of videos using the following query, limit the result to a list of 200 related videos with the following query.

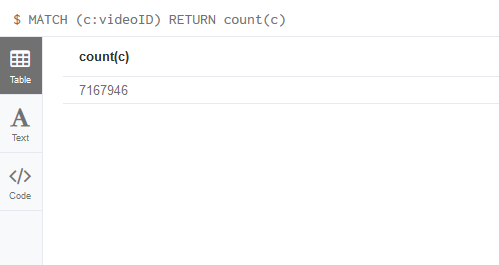


Displaying 416 nodes, 1150 relationships. Rendering is reserved to maximum 100 related nodes by Neo4j configuration.

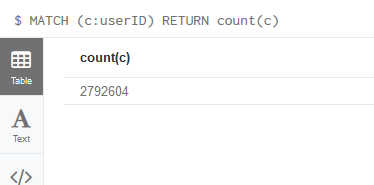


**Analyzing the network:**

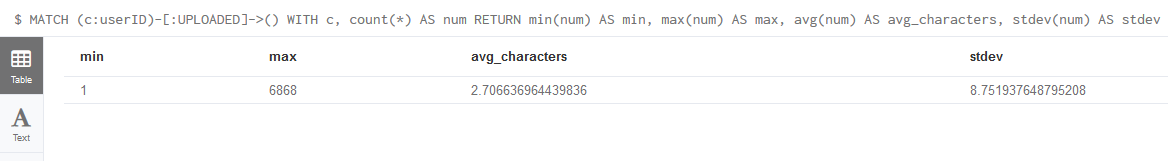
Total number of video nodes in the graph:



**Total number of users/uploaders in the network:**

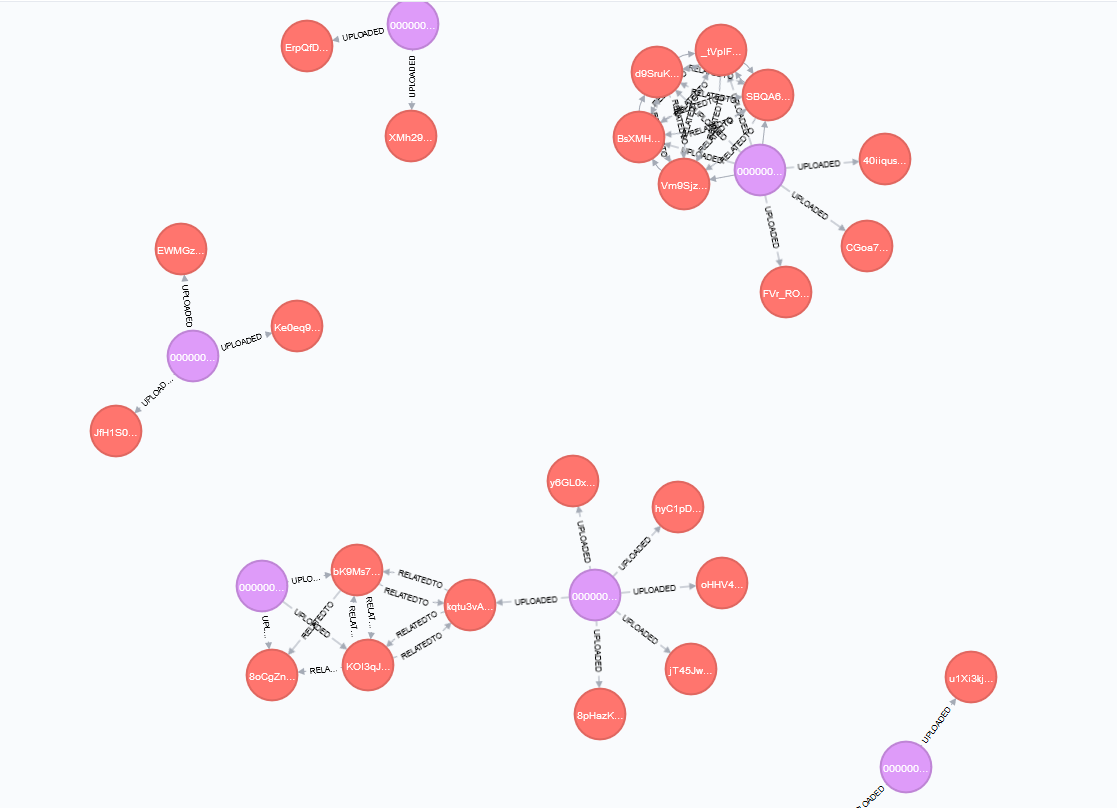


**Summary Statistics for number of other videos each video is related to**



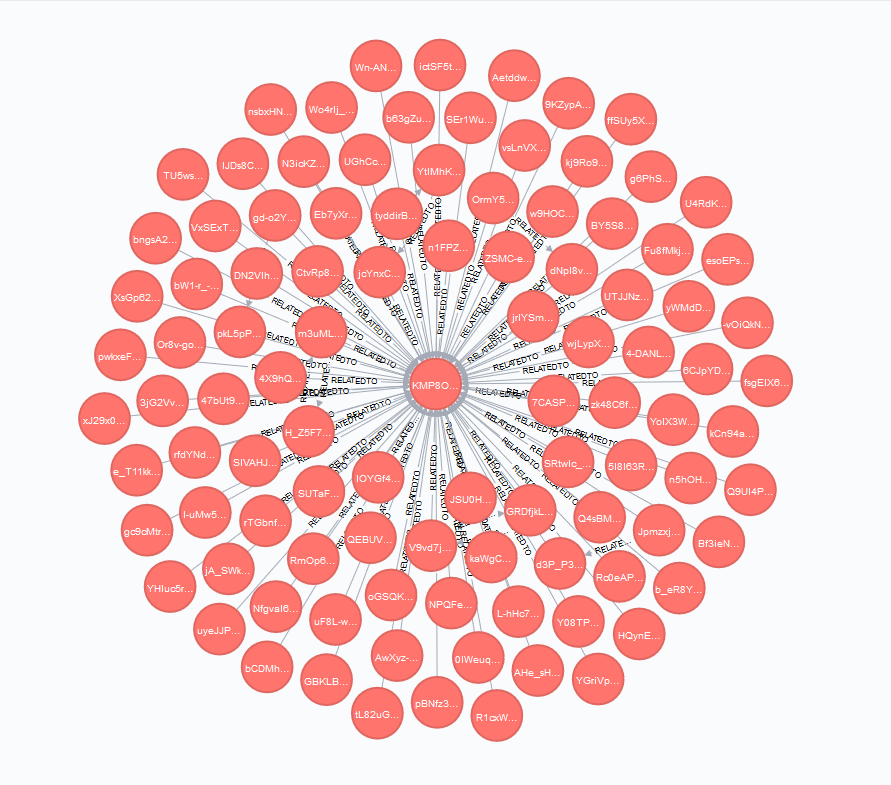
**User Network:**

WE can visualize the user and his uploaded videos and the videos related to it. (purple nodes are users)



Retrieve the network of a single video node, we can run a match query on the videoid property passing the videoid. It returns all the nodes coming in or going out of the nodes.



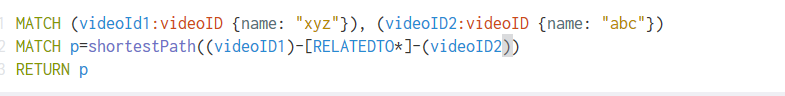


Neo4j has Graph algorithms and Apoc Graph algorithms libraries for running Graph specific algorithms. Graph algorithms are used for traversing the graphs, detecting communities, calculating pagerank, finding shortest paths between nodes and forming clusters.

**Graph Algorithms**

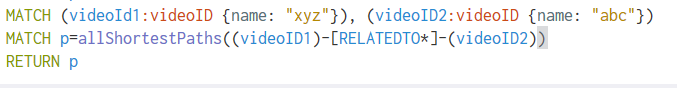
**Shortest Path Algorithm:**

The following query will find the shortest path between two nodes videoID1, videoID2.



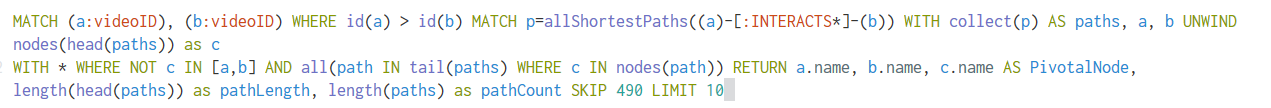
**All Shortest Path Algortihm:**

Returns all the short paths computed between videoID1 and videoID2



**Pivotal Nodes:**

A node is said to be pivotal if it lies on all shortest paths between two other nodes in the network. We can find all pivotal nodes in the network:



**Centrality measures**

Centrality measures give us relative measures of importance in the network. There are many different centrality measures and each measures a different type of “importance”.

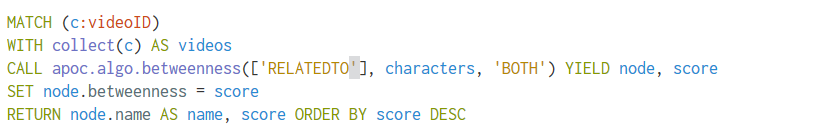
**Degree Centrality**

Degree centrality is simply the number of connections that a node has in the the network. In the context of the graph of thrones, the degree centrality of a character is the number of other characters that character interacted with. We can calculate degree centrality using Cypher using the following query



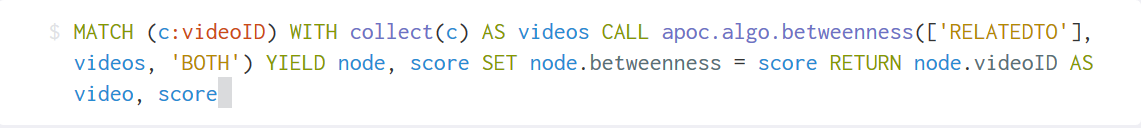
**Betweenness Centrality**

The betweenness centrality of a node in a network is the number of shortest paths between two other members in the network on which a given node appears. Betweenness centrality is an important metric because it can be used to identify “brokers of information” in the network or nodes that connect disparate clusters. The following query is used for the same



**PageRank**

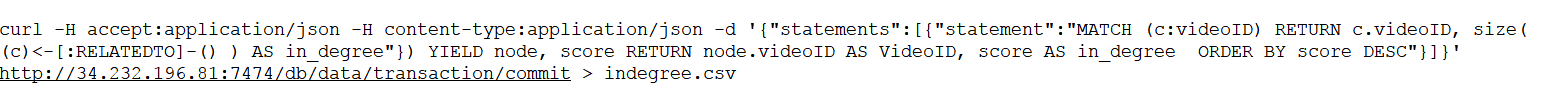
PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set.



These were few of many algorithms implemented in Neo4j Graph library. For our analysis we are computing PageRank to assign weights to the relationships and Degree Centrality to compute how well connected a node is within the network.

Using the browser app to compute PageRank and Degree Centrality was memory intensive and had to be done offline or in a parallel processing manner. As PageRank is an iterative process we opted for SPARK distributed environment deployed on AWS cloud.

For computing Degree Centrality, we used CURL command, to run an offline query our Neo4j database, which runs the following query on the database and writes the outputs a json which is then parsed to csv format.



**USER PROFILE FILE FROM MAPREDUCE JOBS**

Based on the mapReduce output files we are returning as user profile summary statistics. IT contains

UserID : Unique Id of the user

TotalVideos: total number of videos uploaded

TotalVIews: total number of views user has received for all his videos combined

MaxViewedVideo: title of the max viewed video

MaxViews: maximum views

MinViewedVideo: title of the minimum viewed video

MinViews: minimum views

AvgViews: average of the views

TotalLengthInMinutes: the length of all the videos combined

AvgRating : the average total rating received

TotalRatings: total ratings

TotalComments: total comments received

MaxAgeInDays: maximum age of video in days

MinAgeInDays: minimum age of video in days

PopularityScore: total score based on pagerank

TotalInDegree: total of all the indegrees of the uploaded video

Friends: number of friends the user has

This file forms the basis for Clustering Users based on their profile summary statistics.

**K-Means Clustering** :

***k*-means clustering** is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. *k*-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. IT is a unsupervised  learning  algorithm. he goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*. The algorithm works iteratively to assign each data point to one of *K* groups based on the features that are provided. Data points are clustered based on feature similarity.

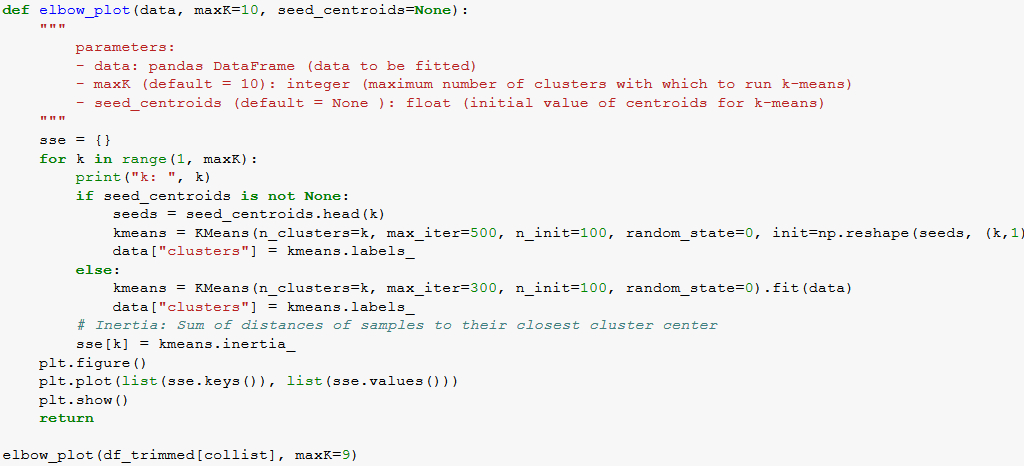
Choosing *K*

The algorithm described above finds the clusters and data set labels for a particular pre-chosen *K*. To find the number of clusters in the data, the user needs to run the *K*-means clustering algorithm for a range of *K* values and compare the results.

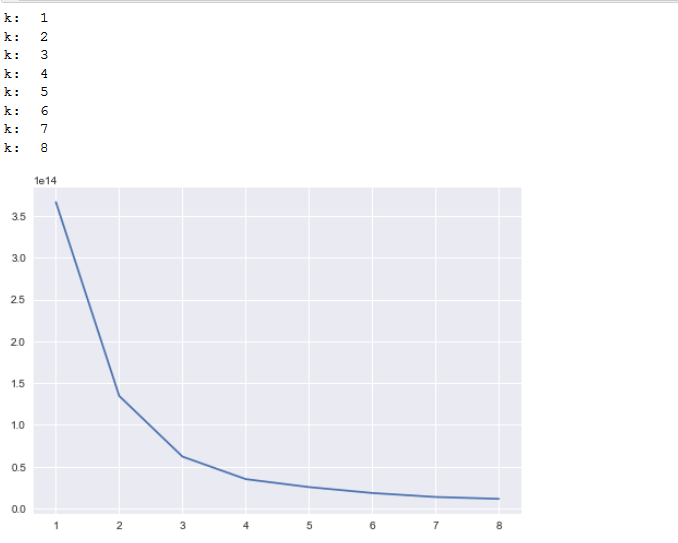
**Elbow Point Curve:**

The **Elbow method** is a method of interpretation and validation of consistency within cluster analysis designed to help finding the appropriate number of clusters in a dataset.

The following function returns the elbow curve,

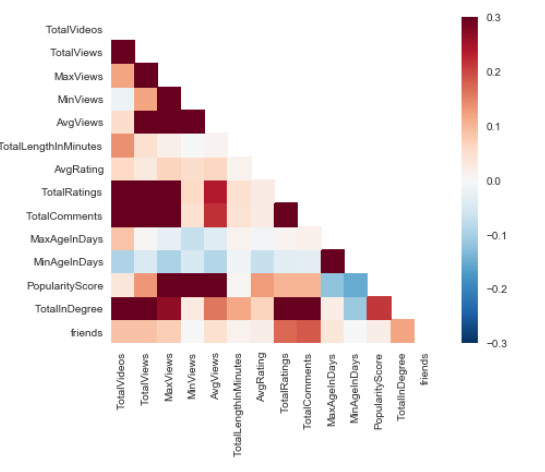


**Elbow curve with trimmed features. Sample of 2000 rows**

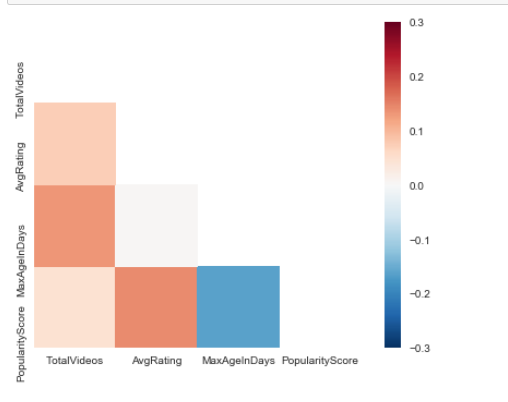


To check the impact of correlation of features on the curve, we plot a correlation map using seaborn.

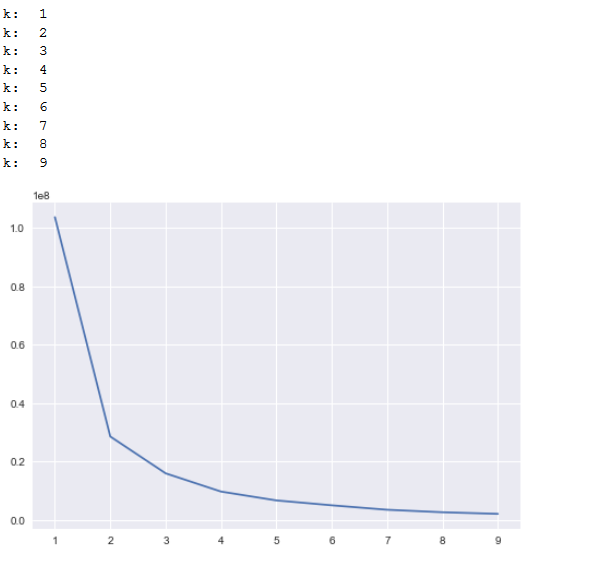
**Map for all the features.**



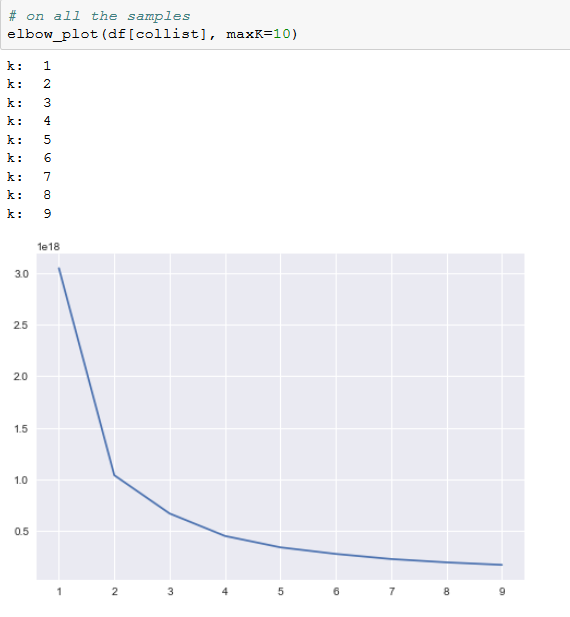
**After removing the correlated features and selecting the remaining**



**Elbow curve with feature selection determined by the above maps**

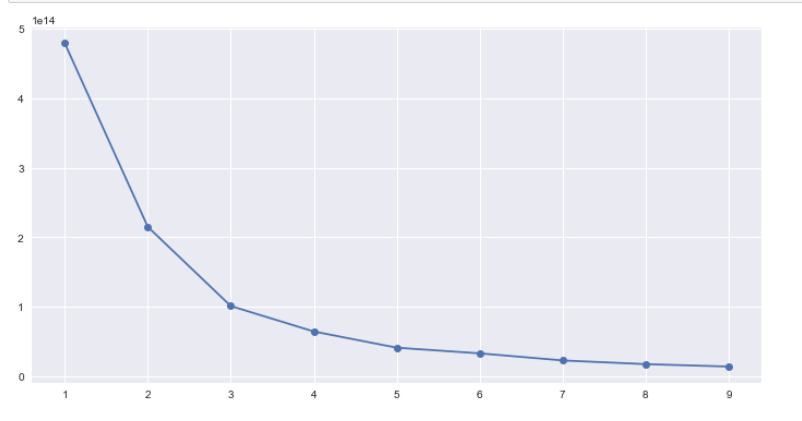


**Elbow Curve for all the entire dataset 5000000 rows**



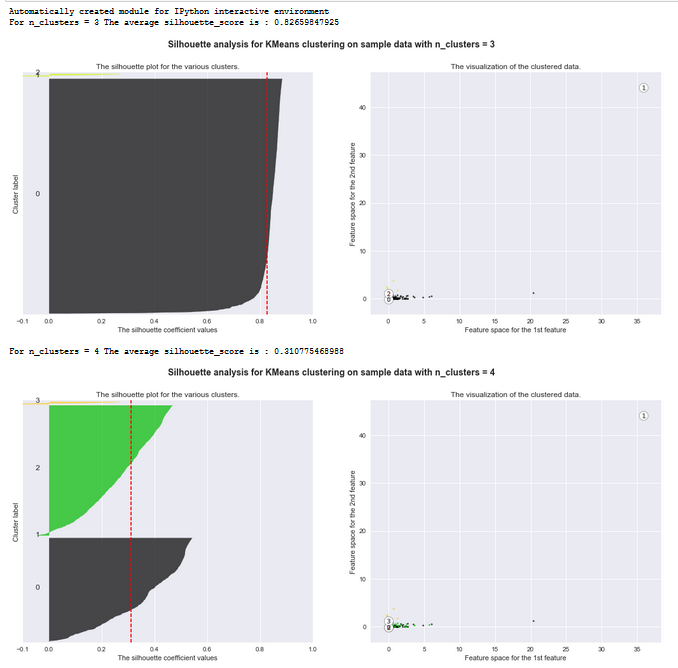
**At k=3 the graph begins to flatten significantly. This point where the graph starts to smooth out is the prophesied “elbow” for which we have been looking.**

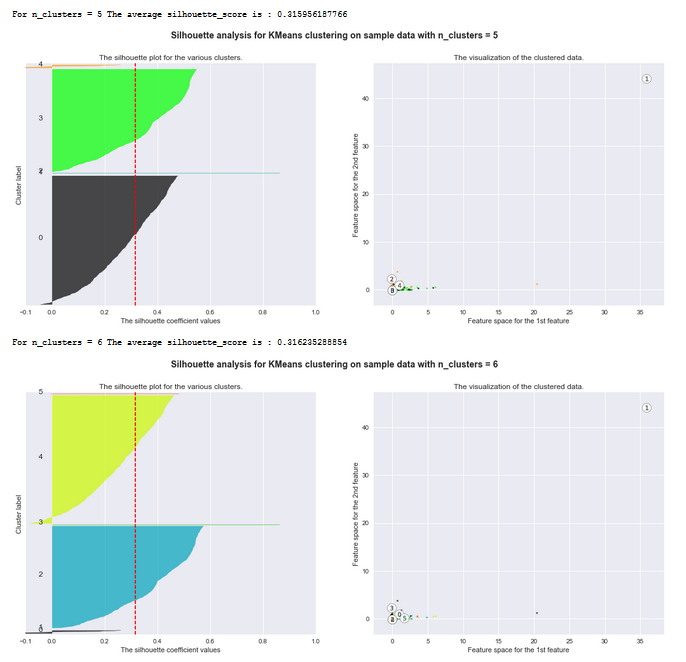
**Same elbow curve with data points.**



**Silhouette Graph**

Silhouette refers to a method of interpretation and validation of consistency within clusters of data. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters. The silhouette can be calculated with any distance metric, such as the Euclidean distance or the Manhattan distance.

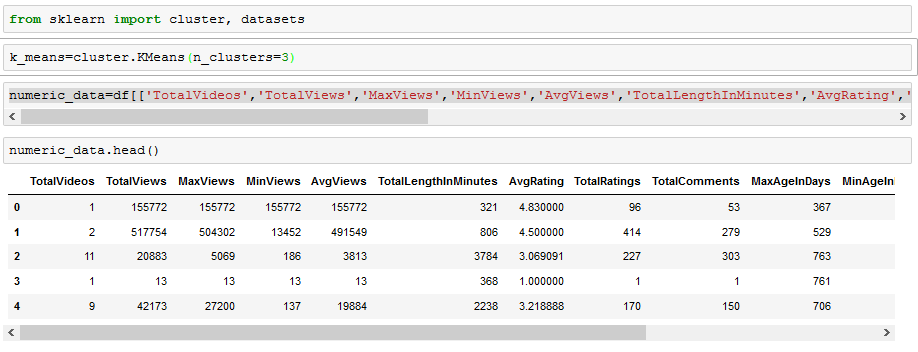


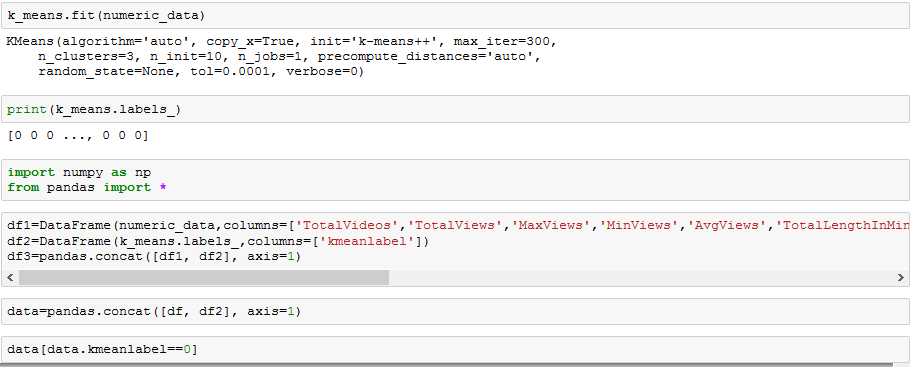


Silhouette coefficients (as these values are referred to as) near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster. Plot 4 to 6 are bad pick as the score is way below the mean. Plot with n = 3 has a 0.8 score and can be considered.

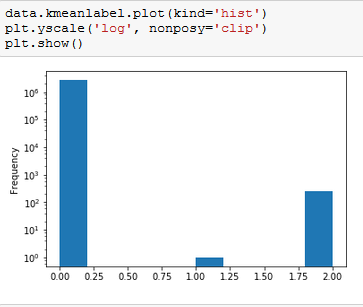
Based on these techniques for selecting clusters, we decided with k = 3

**K-Means model**





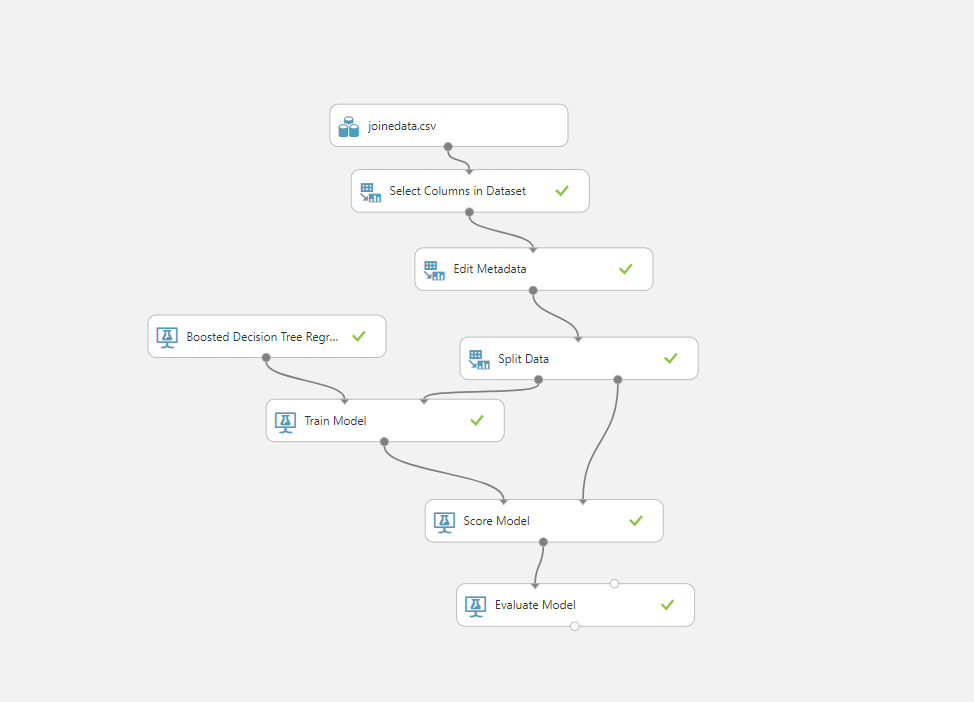
We clustered the users in three groups. A look at the formed clusters tells us that the users grouped in cluster 0 have low popularity score, less almost no values in friends, low view statistics. Cluster 2 has higher statistics in all fields. Cluster 1 has one value.

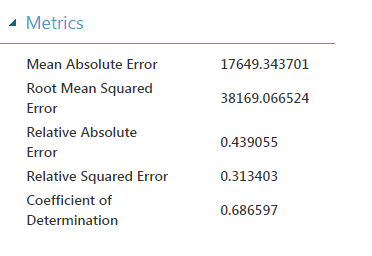


**Predictive Modelling:**

Using the video statistics files we aim to predict the number of views given a set of features. This use case is to provide the user a tool to manipulate the features to check if his video can reach for higher views. We have used Azure to deploy the model, and Boosted Decision Tree Regression as the training algorithm.

The data is used is power distributed, hence leading to a high skew.





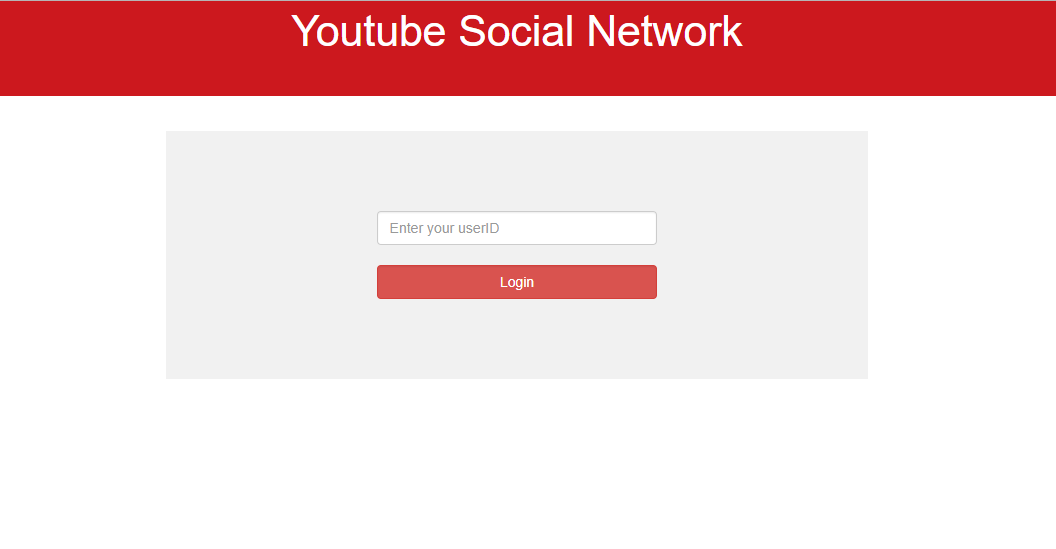
The model gives a high error after normalizing the data, hence we decided to go with the base model.

Web Application

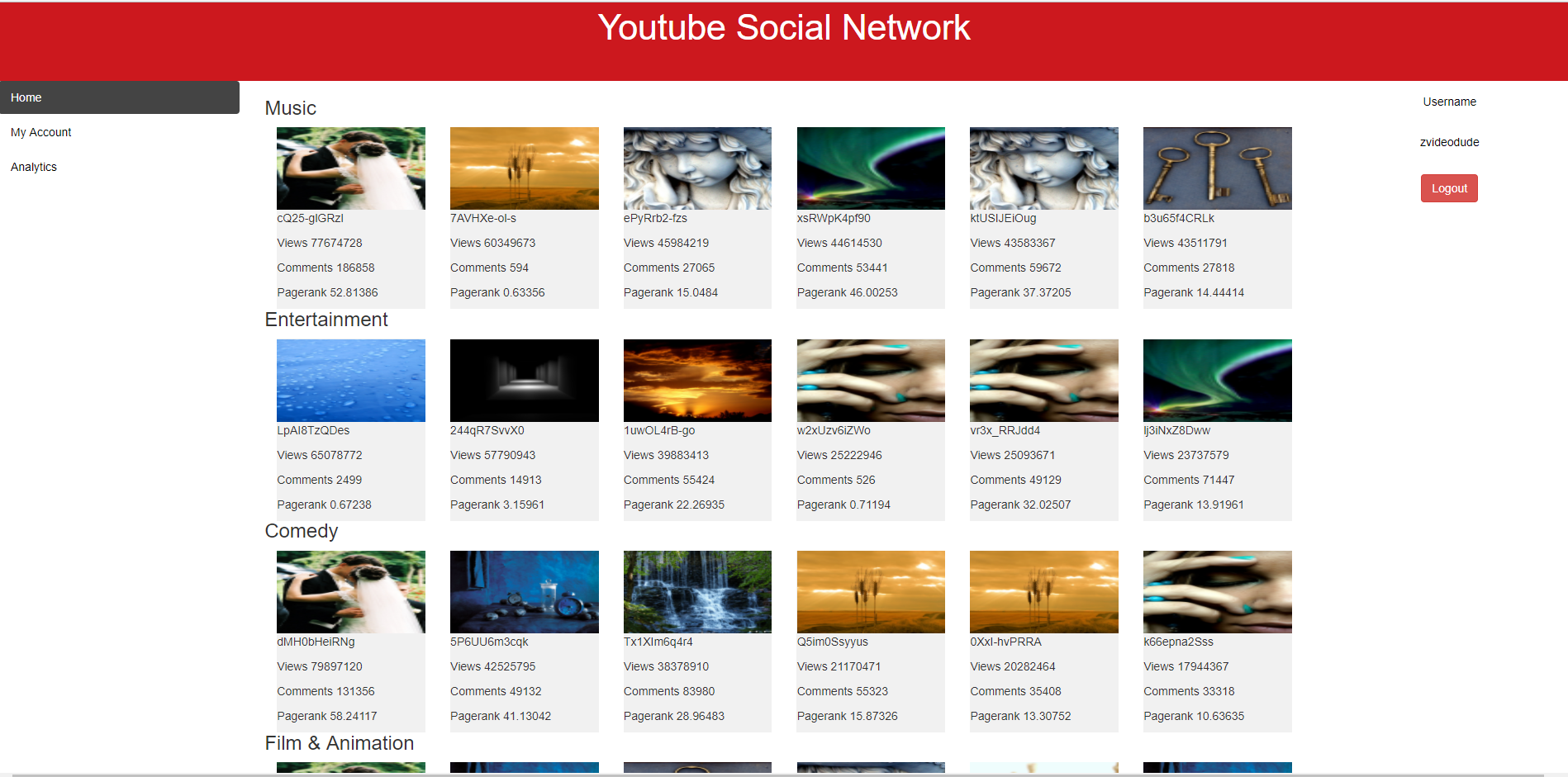
**WebApp is deployed on IBM BlueMix**

http://finalproject-biparietal-camshaft.mybluemix.net/

**Home Page provides a login . Any user existing in the database can login.**



**User homepage featuring top videos for each category**



**MY ACCOUNT: PROVIDES THE USERS STATISTICS FOR THAT USER**

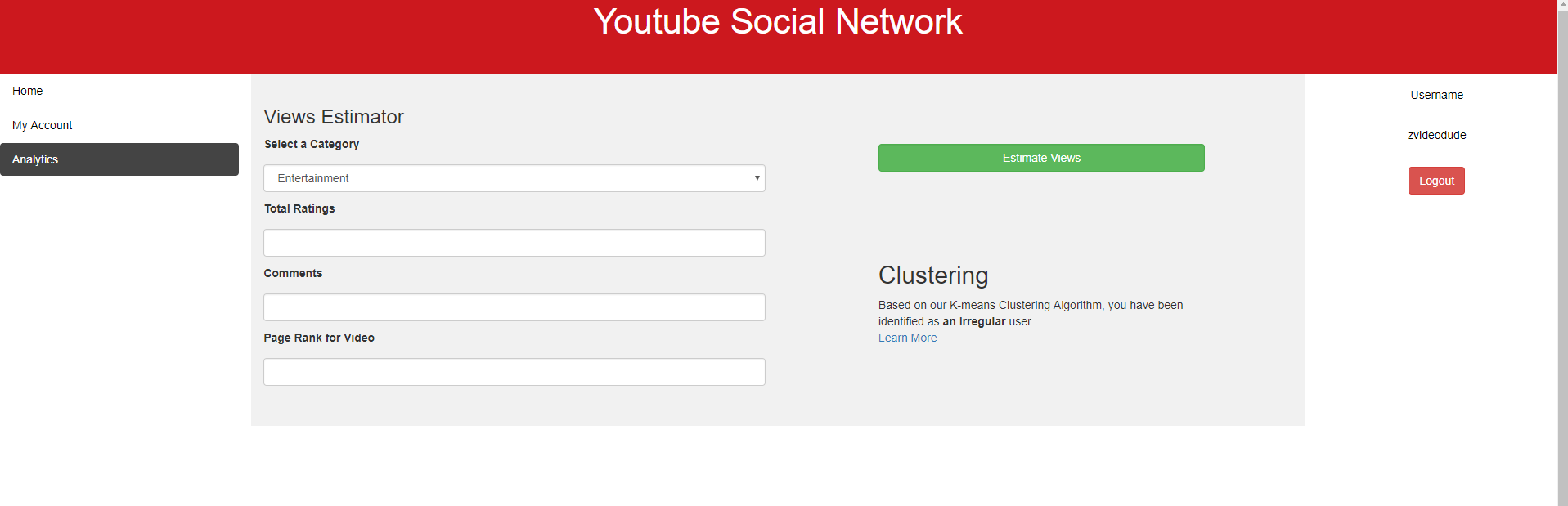


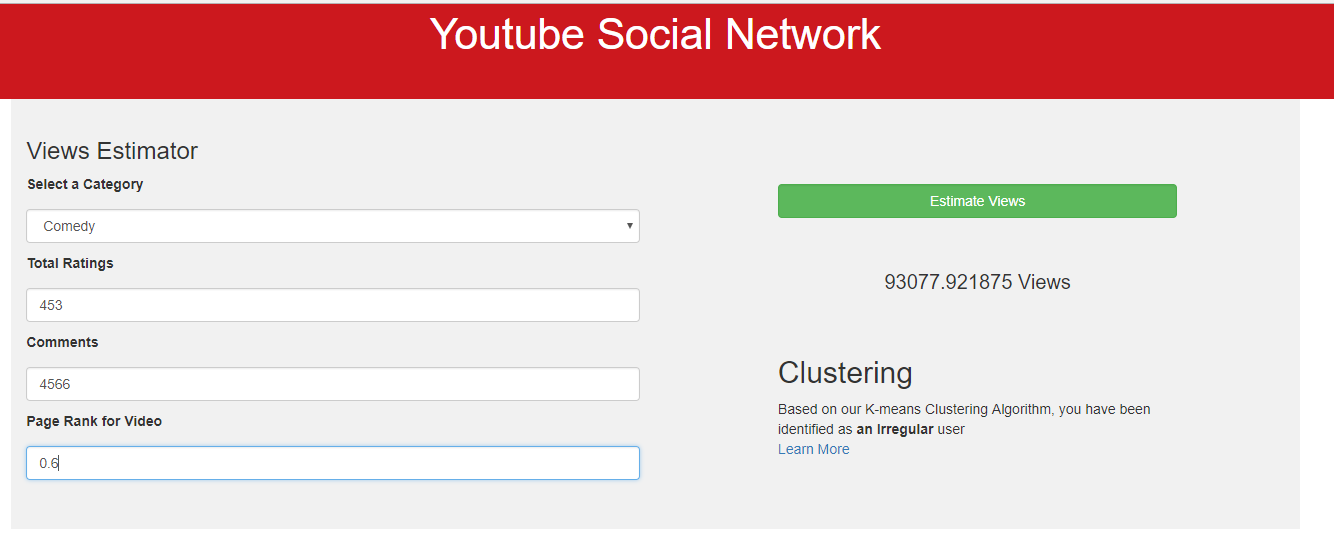
**ANALYTICS:**

Views Estimator form, and the cluster information.

User can input the features to get an estimated vies for his video.

Also based on K-Means the user is given the information about his resident cluster.





**Summary**

In this project, we analyzed youtube networks dataset which consisted of 7 million nodes. We transformed the raw dataset into a useful relational and graph model which we could showcase and discover the network related properties of this data (eg: pagerank and in degree measure). From our exploratory analysis, we could see that majority of the features follow a power law distribution. We also implemented a prediction and clustering algorithm which were deployed on the cloud using Azure and have been made available to the web application created using Flask. This application provides the user with a view of his profile and statistics. It also provides a feature of estimating views based on certain features as well as give him an idea of which cluster he belongs to.