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

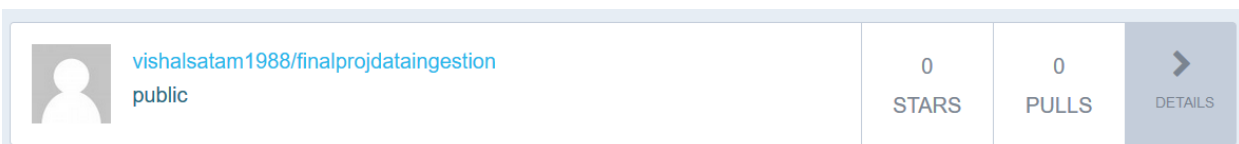
Summary

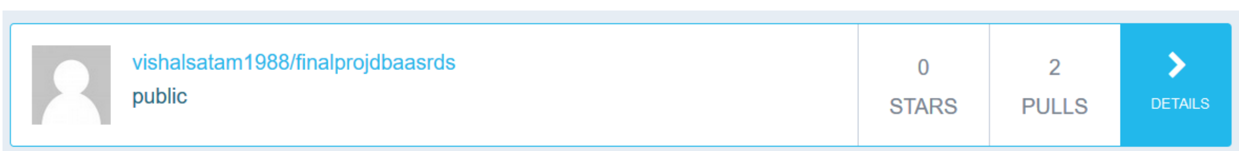
The University of Simon Fraser has performed a crawl of the youtube videos data from the public API available by Youtube. Since there is a rate limit on the API, the researchers had to perform the crawl over 2 months. For this Network Analytics project, we have examined the network of videos created 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.

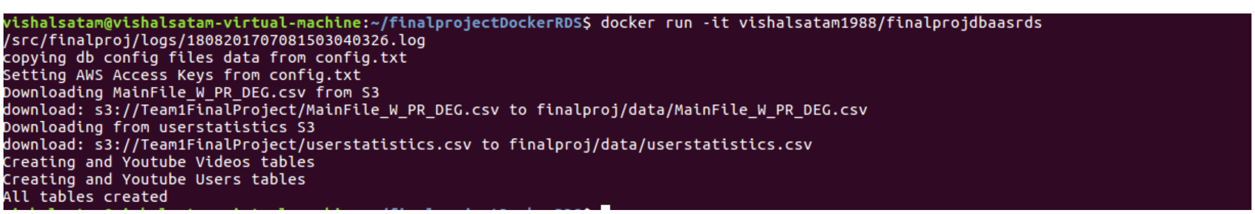
DOCKER

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.

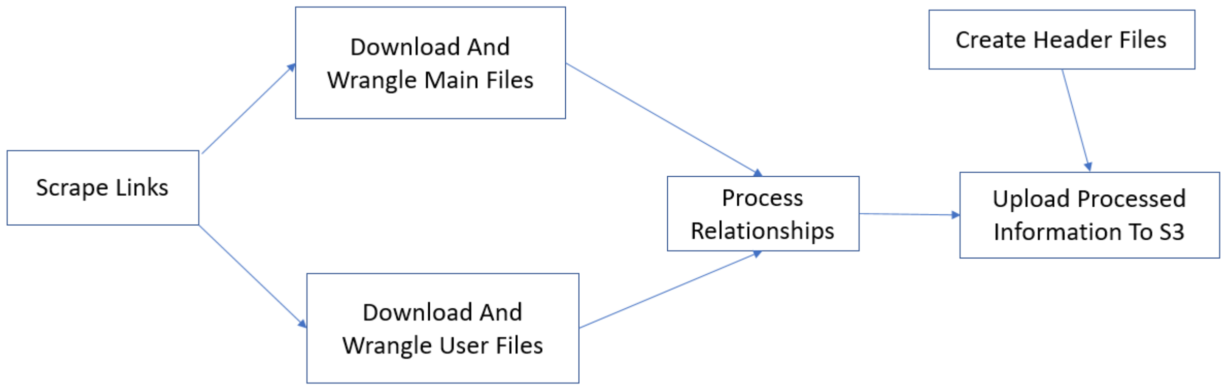
We have also written a Map Reduce job to perform aggregations on the videos file to compute summaries for the user.

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.

```
DEBUG: Checking if UploadProcessedInformationToS3() is complete
DEBUG: Checking if ProcessRelationships() is complete
DEBUG: Checking if CreateHeaderFiles() is complete
INFO: Informed scheduler that task   UploadProcessedInformationToS3__99914b932b   has status   PENDING
INFO: Informed scheduler that task   CreateHeaderFiles__99914b932b   has status   PENDING
DEBUG: Checking if DownloadAndWrangleMainFiles() is complete
DEBUG: Checking if DownloadAndWrangleUserFiles() is complete
INFO: Informed scheduler that task   ProcessRelationships__99914b932b   has status   PENDING
DEBUG: Checking if ScrapeLinks() is complete
INFO: Informed scheduler that task   DownloadAndWrangleUserFiles__99914b932b   has status   PENDING
INFO: Informed scheduler that task   ScrapeLinks__99914b932b   has status   PENDING
INFO: Informed scheduler that task   DownloadAndWrangleMainFiles__99914b932b   has status   PENDING
INFO: Done scheduling tasks
INFO: Running Worker with 1 processes
DEBUG: Asking scheduler for work...
DEBUG: Pending tasks: 6
INFO: [pid 11528] Worker Worker(salt=829389395, workers=1, host=WINDOWS-IH3IR68, username=visha, pid=11528) running   CreateHeaderFiles()
INFO: [pid 11528] Worker Worker(salt=829389395, workers=1, host=WINDOWS-IH3IR68, username=visha, pid=11528) done      CreateHeaderFiles()
DEBUG: 1 running tasks, waiting for next task to finish
INFO: Informed scheduler that task   CreateHeaderFiles__99914b932b   has status   DONE
DEBUG: Asking scheduler for work...
DEBUG: Pending tasks: 5
INFO: [pid 11528] Worker Worker(salt=829389395, workers=1, host=WINDOWS-IH3IR68, username=visha, pid=11528) running   ScrapeLinks()
INFO: [pid 11528] Worker Worker(salt=829389395, workers=1, host=WINDOWS-IH3IR68, username=visha, pid=11528) done      ScrapeLinks()
DEBUG: 1 running tasks, waiting for next task to finish
INFO: Informed scheduler that task   ScrapeLinks__99914b932b   has status   DONE
DEBUG: Asking scheduler for work...
DEBUG: Pending tasks: 4
INFO: [pid 11528] Worker Worker(salt=829389395, workers=1, host=WINDOWS-IH3IR68, username=visha, pid=11528) running   DownloadAndWrangleUserFiles()
2017-08-14 21:30:45.009052 ::  Starting process of usercrawl files
2017-08-14 21:30:45.017073 ::  Downloading and Processing 0528
Processing 0528
2017-08-14 21:31:07.724094 ::  Downloading and Processing 080903user
Processing 080903user
2017-08-14 21:31:48.149076 ::  Ending process of usercrawl files
INFO: [pid 11528] Worker Worker(salt=829389395, workers=1, host=WINDOWS-IH3IR68, username=visha, pid=11528) done      DownloadAndWrangleUserFiles()
DEBUG: 1 running tasks, waiting for next task to finish
INFO: Informed scheduler that task   DownloadAndWrangleUserFiles__99914b932b   has status   DONE
DEBUG: Asking scheduler for work...
```

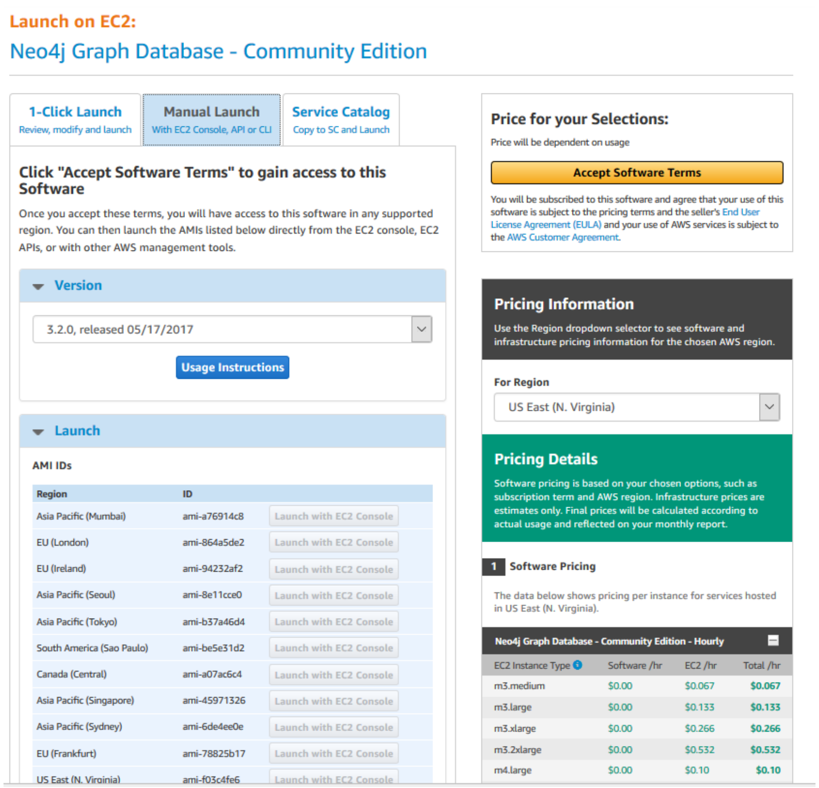
```
DEBUG: Done
DEBUG: There are no more tasks to run at this time
INFO: Worker Worker(salt=829389395, workers=1, host=WINDOWS-IH3IR68, username=visha, pid=11528) was stopped. Shutting down Keep-Alive thread
INFO:
===== Luigi Execution Summary =====

Scheduled 6 tasks of which:
* 6 ran successfully:
    - 1 CreateHeaderFiles()
    - 1 DownloadAndWrangleMainFiles()
    - 1 DownloadAndWrangleUserFiles()
    - 1 ProcessRelationships()
    - 1 ScrapeLinks()
    ...

This progress looks :) because there were no failed tasks or missing external dependencies

===== Luigi Execution Summary =====
```



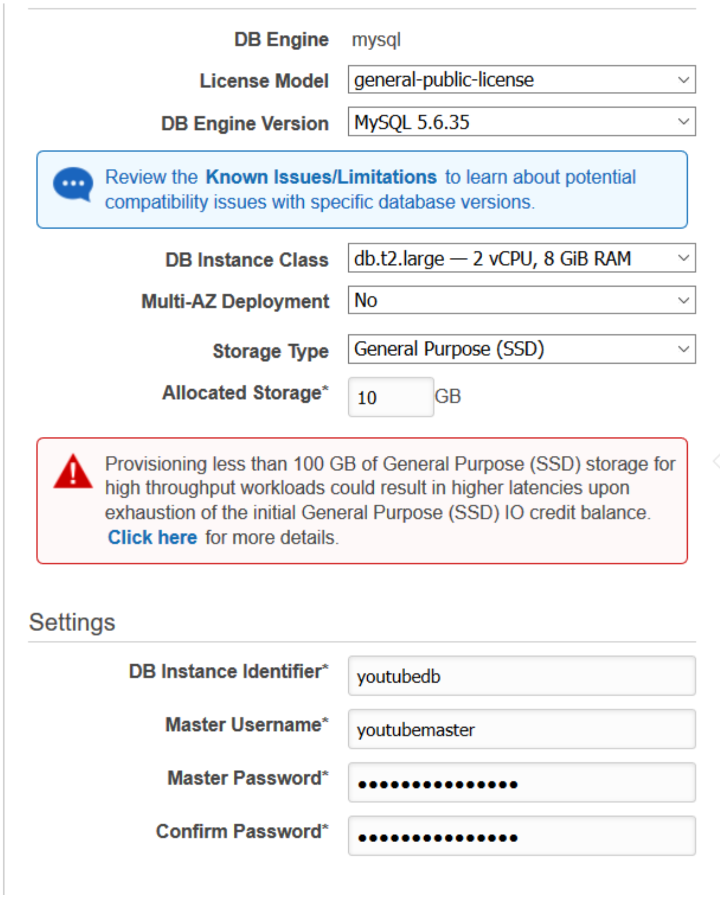
Database

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.



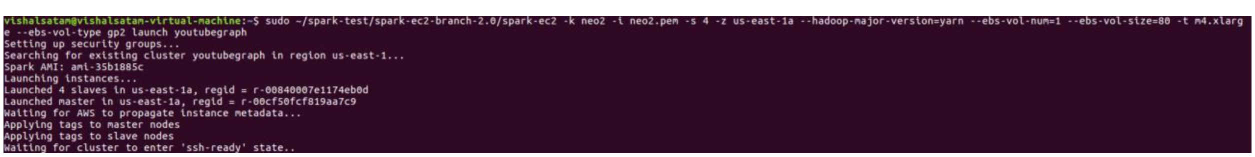
The Amazon RDS database was created to store the user statistics and videos data.



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.



```
sent 1,611 bytes  received 40 bytes  3,302.00 bytes/sec
total size is 1,456  speedup is 0.88
Running setup on master...
Warning: Permanently added 'ec2-34-201-17-136.compute-1.amazonaws.com,34.201.17.136' (ECDSA) to the list of known hosts.
Connection to ec2-34-201-17-136.compute-1.amazonaws.com closed.
Warning: Permanently added 'ec2-34-201-17-136.compute-1.amazonaws.com,34.201.17.136' (ECDSA) to the list of known hosts.
Setting up Spark on ip-172-31-1-0.ec2.internal...
Setting executable permissions on scripts...
RSYNC'ing /root/spark-ec2 to other cluster nodes...
ec2-34-201-1-197.compute-1.amazonaws.com
Warning: Permanently added 'ec2-34-201-1-197.compute-1.amazonaws.com,172.31.15.75' (ECDSA) to the list of known hosts.
id_rsa                                                                                          100% 1679     1.6KB/s   00:00
[timing] rsync /root/spark-ec2:  00h 00m 00s
Running setup-slave on all cluster nodes to mount filesystems, etc...
[1] 11:31:28 [SUCCESS] ec2-34-201-17-136.compute-1.amazonaws.com
checking/fixing resolution of hostname
Setting up slave on ip-172-31-1-0.ec2.internal... of type t2.micro
Existing lock /var/run/yum.pid: another copy is running as pid 4809.
Another app is currently holding the yum lock; waiting for it to exit...
  The other application is: yum
    Memory :  87 M RSS (337 MB VSZ)
    Started: Fri Aug 18 11:31:08 2017 - 00:01 ago
    State  : Running, pid: 4809
mount: wrong fs type, bad option, bad superblock on /dev/xvds,
       missing codepage or helper program, or other error
       In some cases useful info is found in syslog - try
       dmesg | tail  or so

1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 13.7118 s, 78.3 MB/s
mkswap: /mnt/swap: warning: don't erase bootbits sectors
        on whole disk. Use -f to force.
Setting up swapspace version 1, size = 1048572 KiB
no label, UUID=8df73d40-a996-4416-8a4a-48fcc785c390
Added 1024 MB swap file /mnt/swap
Stderr: Warning: Permanently added 'ec2-34-201-17-136.compute-1.amazonaws.com,172.31.1.0' (ECDSA) to the list of known hosts.
Connection to ec2-34-201-17-136.compute-1.amazonaws.com closed.
[2] 11:31:38 [SUCCESS] ec2-34-201-1-197.compute-1.amazonaws.com
checking/fixing resolution of hostname
Setting up slave on ip-172-31-15-75.ec2.internal... of type t2.micro
mount: wrong fs type, bad option, bad superblock on /dev/xvds,
       missing codepage or helper program, or other error
       In some cases useful info is found in syslog - try
       dmesg | tail  or so
```

```
Stopping httpd:                                            [FAILED]
Starting httpd:                                            [  OK  ]
[timing] ganglia setup:  00h 00m 03s
Connection to ec2-34-201-17-136.compute-1.amazonaws.com closed.
Spark standalone cluster started at http://ec2-34-201-17-136.compute-1.amazonaws.com:8080
Ganglia started at http://ec2-34-201-17-136.compute-1.amazonaws.com:5080/ganglia
Done!
```

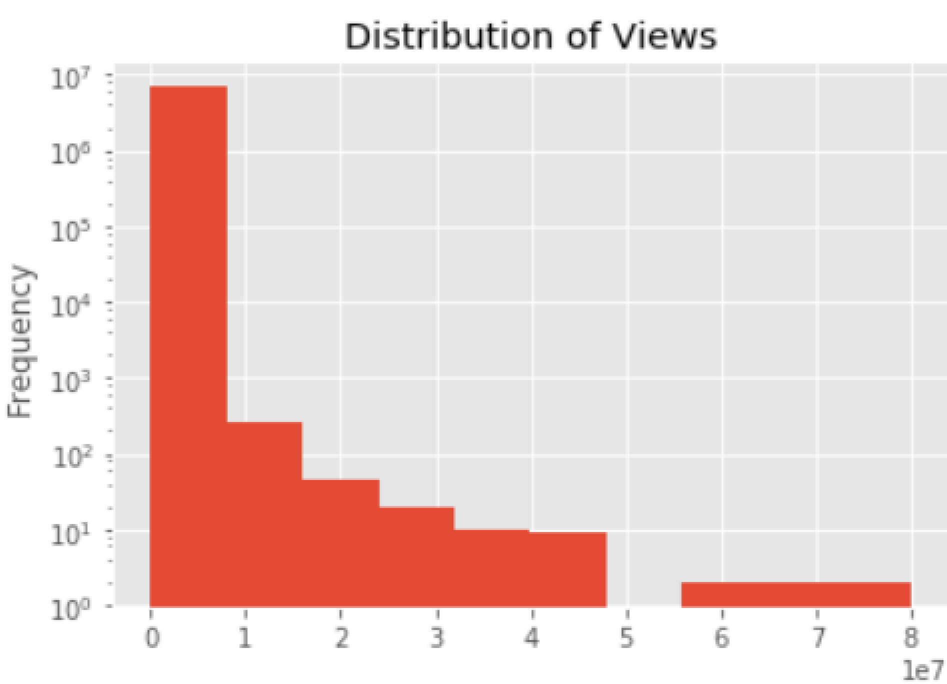
youtubegraph-slave-i-00...	i-003de7780278262db	m4.xlarge	us-east-1a	running	Initializing	None	ec2-34-205-43-197.co...	34.205.43.197	-	ne
youtubegraph-master-i-0...	i-04fb2f947f00653b5	m4.xlarge	us-east-1a	running	Initializing	None	ec2-54-236-241-77.co...	54.236.241.77	-	ne
youtubegraph-slave-i-07...	i-079a27748dda67e4a	m4.xlarge	us-east-1a	running	Initializing	None	ec2-52-3-226-229.com...	52.3.226.229	-	ne
youtubegraph-slave-i-09...	i-0919467e031c6d33c	m4.xlarge	us-east-1a	running	Initializing	None	ec2-34-201-48-92.com...	34.201.48.92	-	ne
youtubegraph-slave-i-0a...	i-0a51f066dec1c3ce2	m4.xlarge	us-east-1a	running	Initializing	None	ec2-52-3-231-225.com...	52.3.231.225	-	ne

```
visha@WINDOWS-IH3IR68 /cygdrive/c/Users/visha/Desktop/MSIS/Advanced Data Science/Assignments/Final Assignments
$ ssh -i neo2.pem root@ec2-34-201-17-136.compute-1.amazonaws.com
The authenticity of host 'ec2-34-201-17-136.compute-1.amazonaws.com (34.201.17.136)' can't be established.
ECDSA key fingerprint is SHA256:9HsK1rY6izsLElYYcjUYgaOWmzOvGDJIAgykHtEEtOY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-34-201-17-136.compute-1.amazonaws.com,34.201.17.136' (ECDSA) to the list of known hosts.
Last login: Fri Aug 18 11:31:08 2017 from ip-172-31-1-0.ec2.internal

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|
```

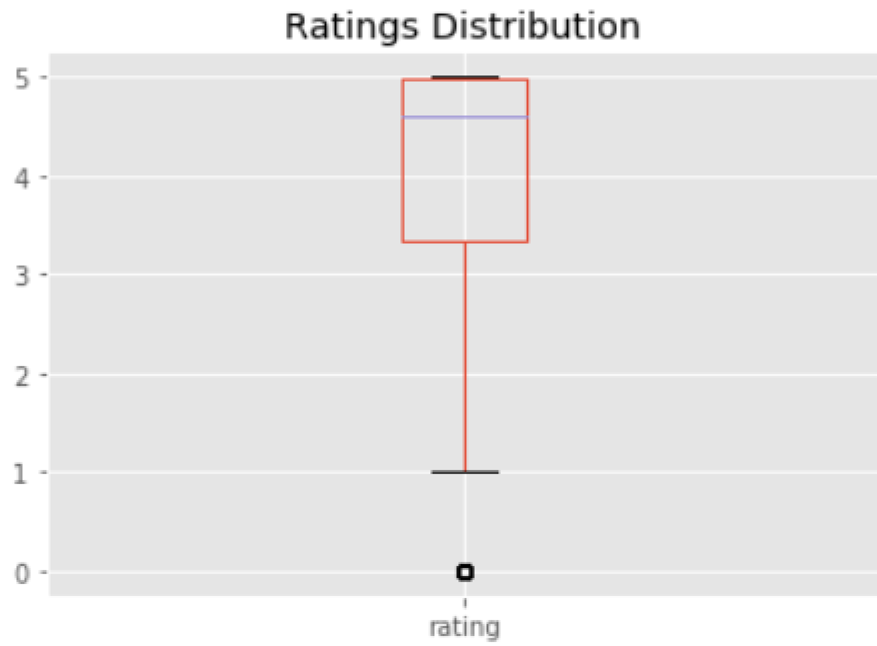
```
root@ip-172-31-1-0 bin]$ export AWS_ACCESS_KEY_ID=[redacted]
root@ip-172-31-1-0 bin]$ export AWS_SECRET_ACCESS_KEY=[redacted]
root@ip-172-31-1-0 bin]$ export AWS_DEFAULT_REGION=us-east-1
root@ip-172-31-1-0 bin]$ ./pyspark
Python 2.7.12 (default, Sep  1 2016, 22:14:00)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/08/18 12:20:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Python version 2.7.12 (default, Sep  1 2016 22:14:00)
SparkSession available as 'spark'.
>>> import re, sys
>>> from operator import add
>>> from pyspark import SparkContext
>>>
>>> def computeContribs(urls, rank):
...     num_urls = len(urls)
...     for url in urls:
...         yield (url, rank / num_urls)
...
>>> def parseNeighbors(urls):
...     """Parses a urls pair string into urls pair."""
...     try:
...         parts = re.split(r',', urls)
...         return parts[0], parts[1]
...     except:
...         return "ERRORINFILE", "ERRORINFILE"
...
>>> def computePageRank(inputPath,outputDir,iterations):
...     lines = sc.textFile(inputPath)
...     # Loads all URLs from input file and initialize their neighbors.
...     links = lines.map(lambda urls: parseNeighbors(urls)).distinct().groupByKey().cache()
...     # Loads all URLs with other URL(s) link to from input file and initialize ranks of them to one.
...     ranks = links.map(lambda (url, neighbors):(url, 1.0))
...     # Calculates and updates URL ranks continuously using PageRank algorithm.
...     for iteration in xrange(3):
...         contribs = links.join(ranks).flatMap(lambda (url, (urls, rank)):computeContribs(urls, rank))
...         ranks = contribs.reduceByKey(add).mapValues(lambda rank: rank * 0.85 + 0.15)
...     ranks.coalesce(1).saveAsTextFile(outputDir)
...
>>> computePageRank("s3n://Team1FinalProject/VideoRelationships.csv","s3n://Team1FinalProject/VideoRelationships_ranks",3)
[Stage 0:====================================>                          (27 + 1) / 41]
```

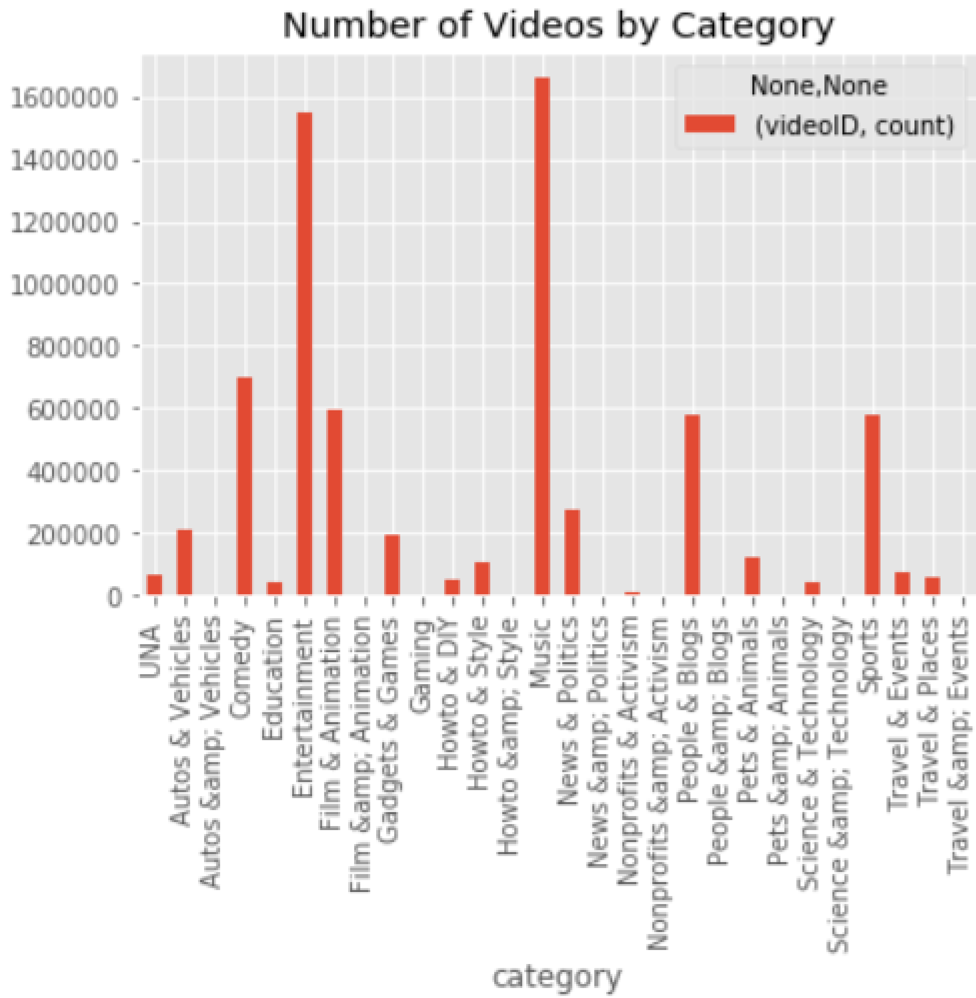
We also computed the in degree centrality measure of the graph for all the nodes using neo4j's library.

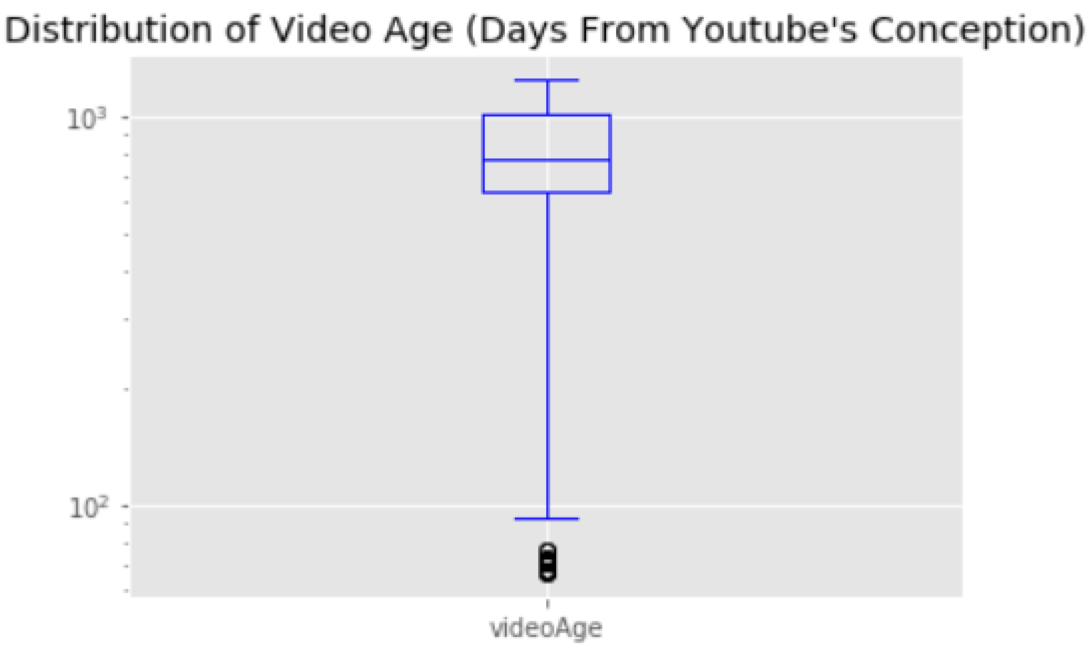
Exploratory Data Analysis

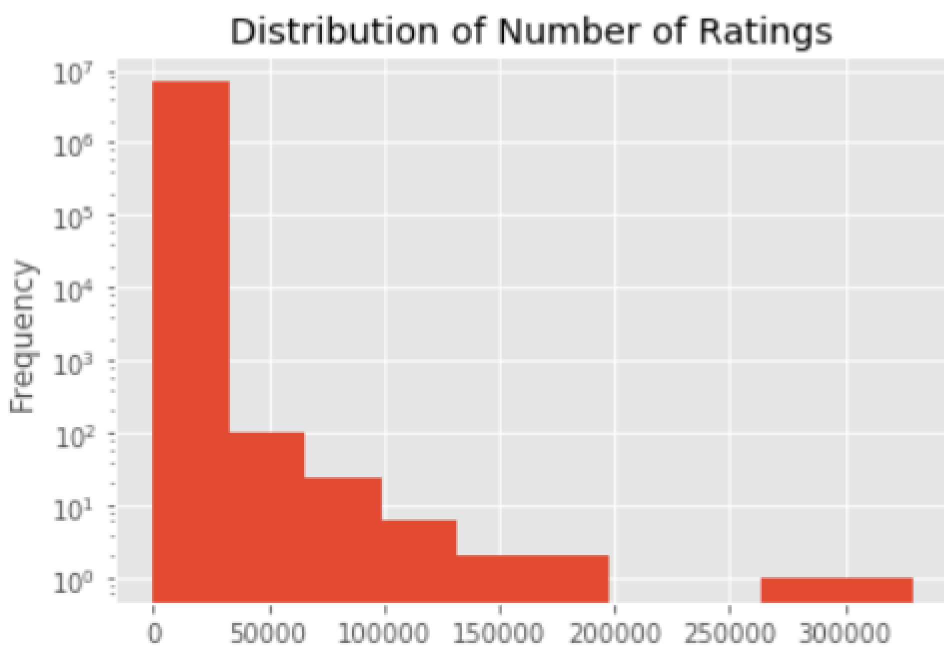


Most of the Features in this dataset follow a power law distribution




Number of Videos by Category
None,None
(videoID, count)
0
200000
400000
600000
800000
1000000
1200000
1400000
1600000
UNA
Autos & Vehicles
Autos & Vehicles
Comedy
Education
Entertainment
Film & Animation
Film & Animation
Gadgets & Games
Gaming
Howto & DIY
Howto & Style
Howto & Style
Music
News & Politics
News & Politics
Nonprofits & Activism
Nonprofits & Activism
People & Blogs
People & Blogs
Pets & Animals
Pets & Animals
Science & Technology
Science & Technology
Sports
Travel & Events
Travel & Places
Travel & Events
category

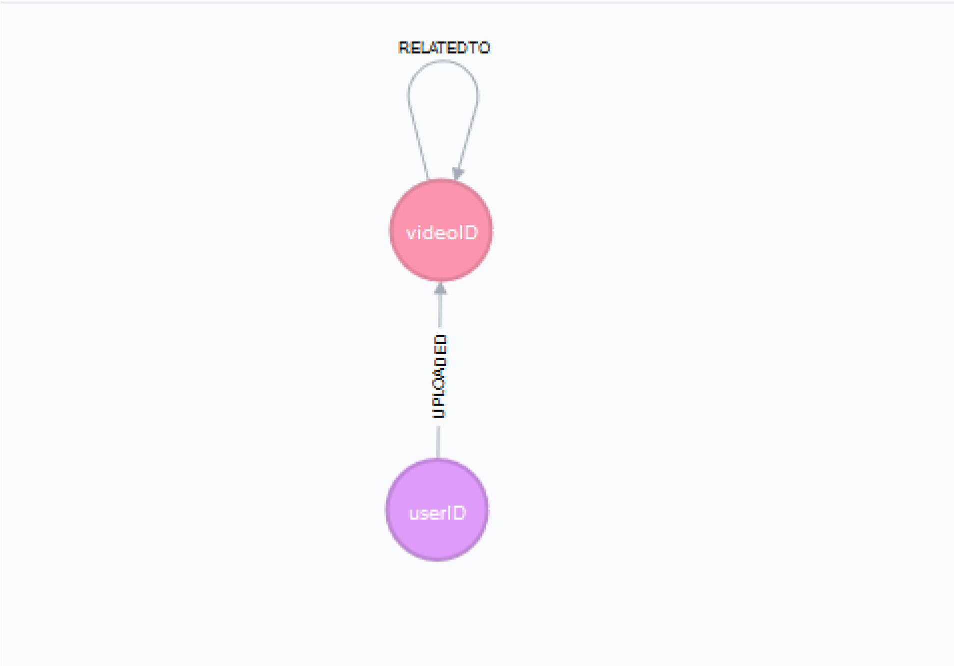




Network Data Model

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

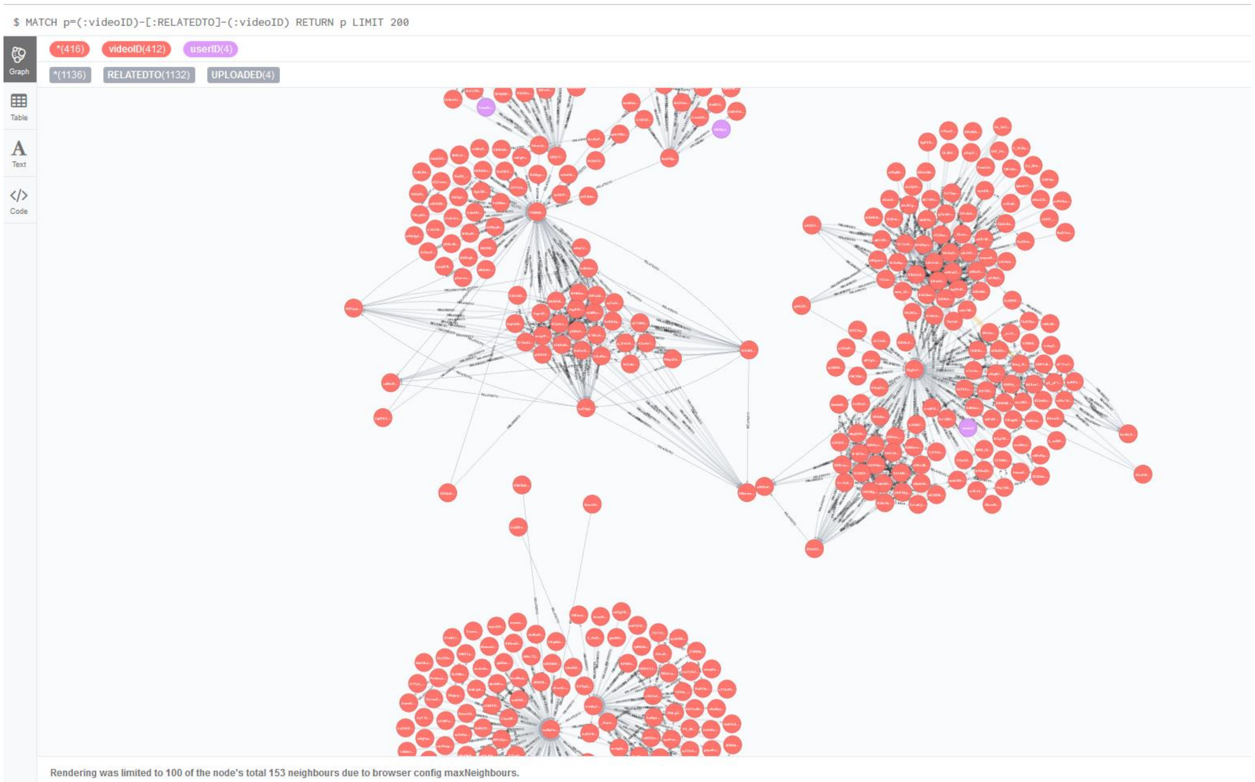
```
$ call apoc.meta.graph
```



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.

```
$ MATCH p=(:videoID)-[:RELATEDTO]-(:videoID) RETURN p LIMIT 200
```

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:

```
$ MATCH (c:videoID) RETURN count(c)
```

count(c)
7167946

Total number of users/uploaders in the network:

```
$ MATCH (c:userID) RETURN count(c)
```

count(c)
2792604

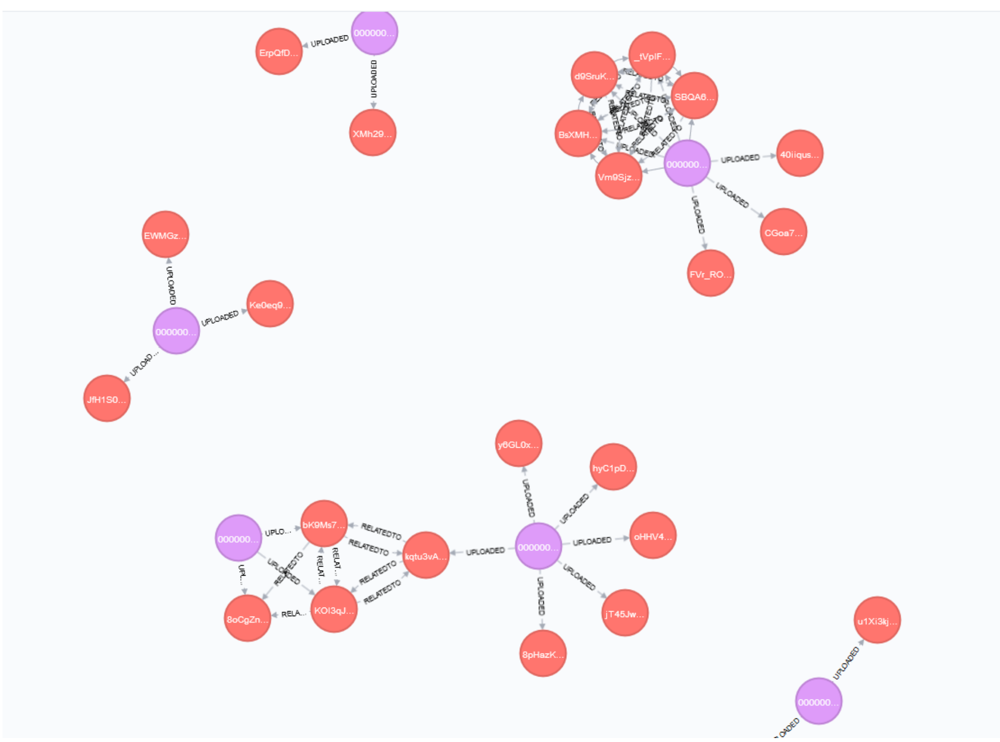
Summary Statistics for number of other videos each video is related to

```
$ MATCH (c:userID)-[:UPLOADED]->() WITH c, count(*) AS num RETURN min(num) AS min, max(num) AS max, avg(num) AS avg_characters, stdev(num) AS stdev
```

min	max	avg_characters	stdev
1	6868	2.706636964439836	8.751937648795208

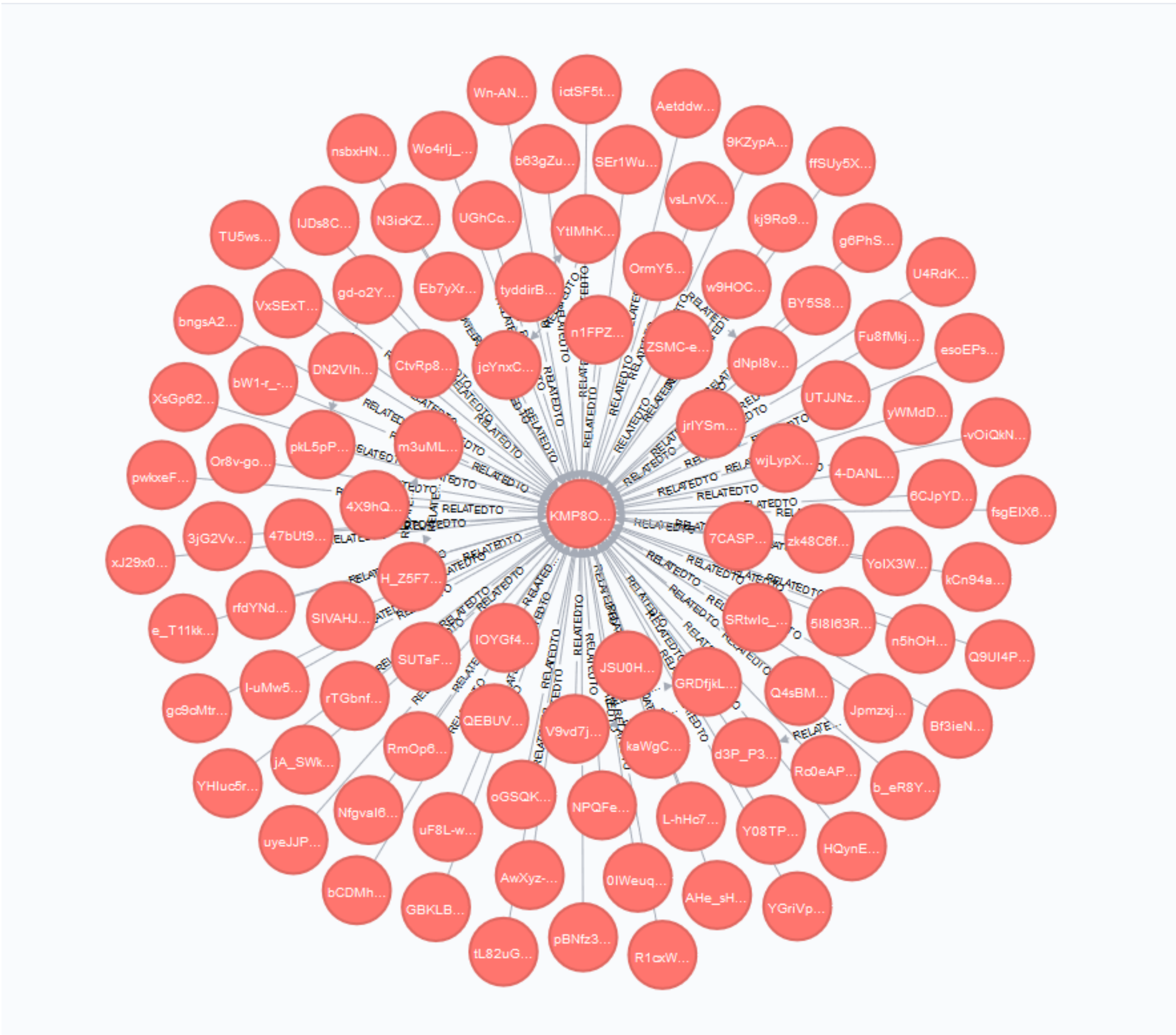
User Network:

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.

```
$ MATCH p=(n:videoID)-[r:RELATEDTO]->(c:videoID) where c.videoID="KMP8OSWGcss" RETURN p LIMIT 100
```



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.

```
MATCH (videoId1:videoID {name: "xyz"}), (videoID2:videoID {name: "abc"})
MATCH p=shortestPath((videoID1)-[RELATEDTO*]-(videoID2))
RETURN p
```

All Shortest Path Algortihm:

Returns all the short paths computed between videoID1 and videoID2

```
MATCH (videoId1:videoID {name: "xyz"}), (videoID2:videoID {name: "abc"})
MATCH p=allShortestPaths((videoID1)-[RELATEDTO*]-(videoID2))
RETURN p
```

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:

```
MATCH (a:videoID), (b:videoID) WHERE id(a) > id(b) MATCH p=allShortestPaths((a)-[:INTERACTS*]-(b)) WITH collect(p) AS paths, a, b UNWIND
nodes(head(paths)) as c
WITH * WHERE NOT c IN [a,b] AND all(path IN tail(paths) WHERE c IN nodes(path)) RETURN a.name, b.name, c.name AS PivotalNode,
length(head(paths)) as pathLength, length(paths) as pathCount SKIP 490 LIMIT 10
```

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

```
MATCH (c:videoID)
RETURN c.videoID AS character, size( (c)-[:RELATEDTO]-() ) AS degree ORDER BY degree DESC
```

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

```
MATCH (c:videoID)
WITH collect(c) AS videos
CALL apoc.algo.betweenness(['RELATEDTO'], characters, 'BOTH') YIELD node, score
SET node.betweenness = score
RETURN node.name AS name, score ORDER BY score DESC
```

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.

```
$ MATCH (c:videoID) WITH collect(c) AS videos CALL apoc.algo.betweenness(['RELATEDTO'],
  videos, 'BOTH') YIELD node, score SET node.betweenness = score RETURN node.videoID AS
  video, score
```

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.

```
curl -H accept:application/json -H content-type:application/json -d '{"statements":[{"statement":"MATCH (c:videoID) RETURN c.videoID, size((c)<-[:RELATEDTO]-() ) AS in_degree"}) YIELD node, score RETURN node.videoID AS VideoID, score AS in_degree  ORDER BY score DESC"}]}' http://34.232.196.81:7474/db/data/transaction/commit > indegree.csv
```

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:

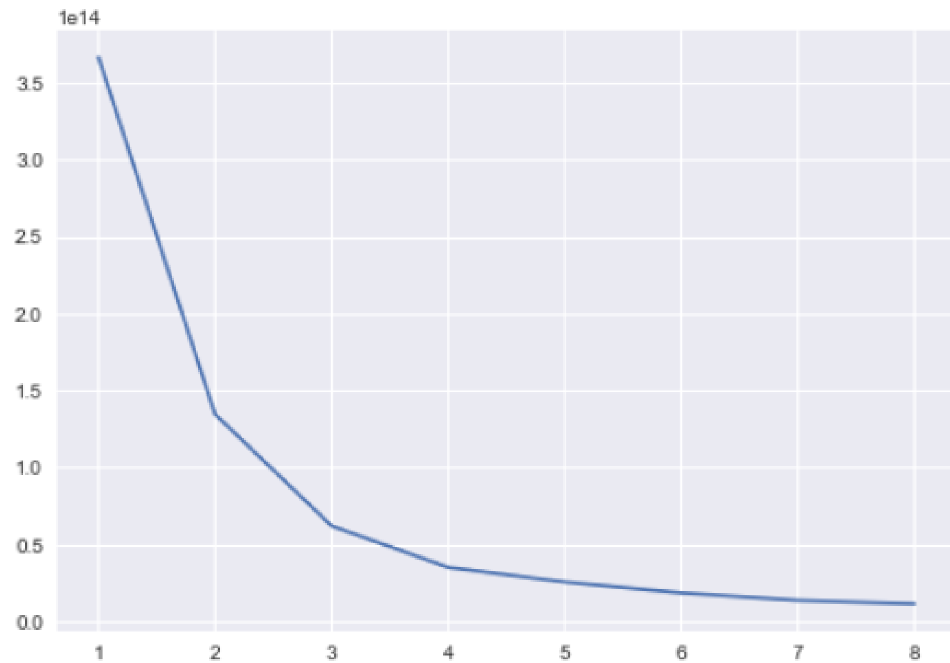
The **Elbow method** is a method of interpretation and validation of consistency within cluster analysis designed to help finding the appropriate number of clusters in a dataset.

The following function returns the elbow curve,

```
def elbow_plot(data, maxK=10, seed_centroids=None):
    """
        parameters:
        - data: pandas DataFrame (data to be fitted)
        - maxK (default = 10): integer (maximum number of clusters with which to run k-means)
        - seed_centroids (default = None ): float (initial value of centroids for k-means)
    """
    sse = {}
    for k in range(1, maxK):
        print("k: ", k)
        if seed_centroids is not None:
            seeds = seed_centroids.head(k)
            kmeans = KMeans(n_clusters=k, max_iter=500, n_init=100, random_state=0, init=np.reshape(seeds, (k,1)
            data["clusters"] = kmeans.labels_
        else:
            kmeans = KMeans(n_clusters=k, max_iter=300, n_init=100, random_state=0).fit(data)
            data["clusters"] = kmeans.labels_
        # Inertia: Sum of distances of samples to their closest cluster center
        sse[k] = kmeans.inertia_
    plt.figure()
    plt.plot(list(sse.keys()), list(sse.values()))
    plt.show()
    return

elbow_plot(df_trimmed[collist], maxK=9)
```

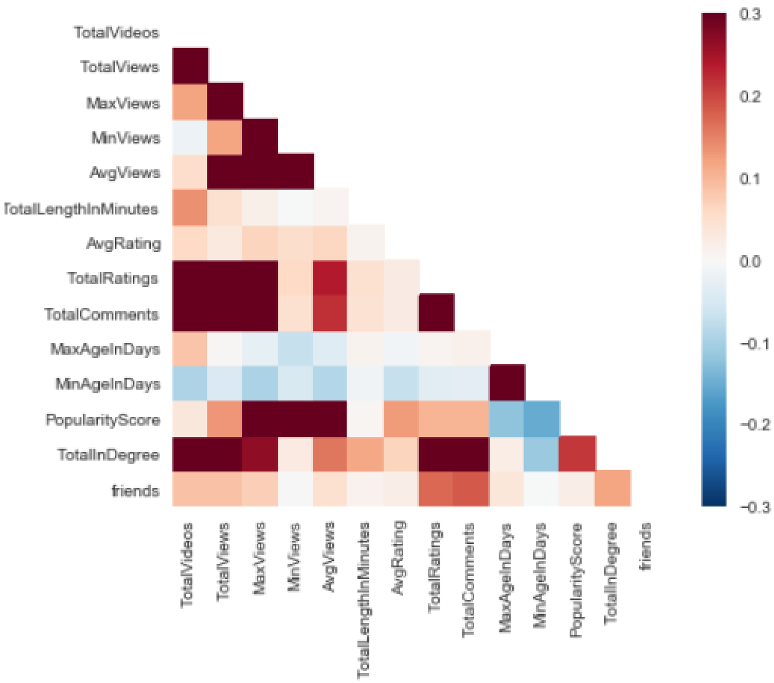
Elbow curve with trimmed features. Sample of 2000 rows

```
k:  1
k:  2
k:  3
k:  4
k:  5
k:  6
k:  7
k:  8
```

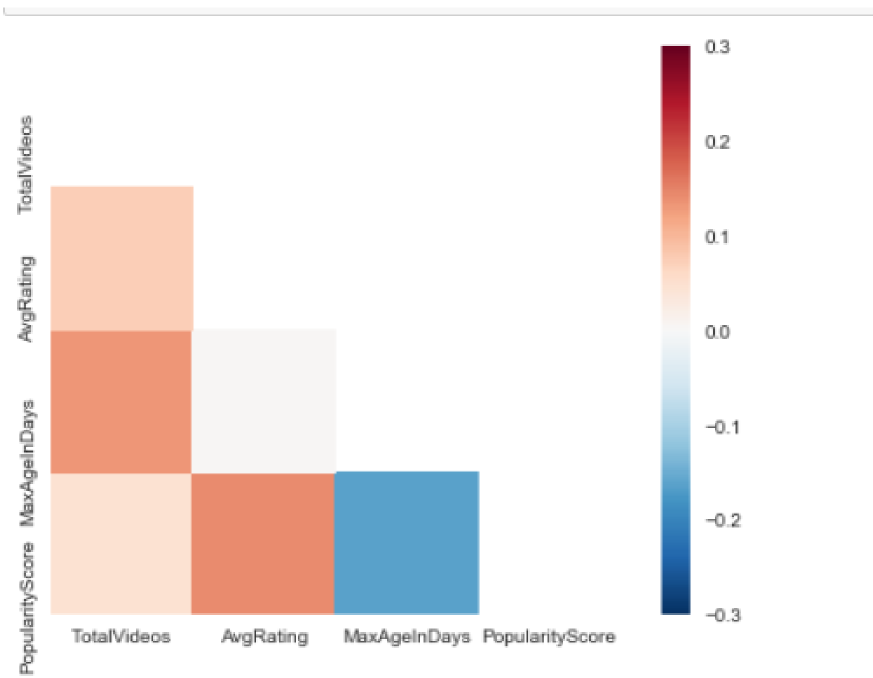


To check the impact of correlation of features on the curve, we plot a correlation map using seaborn.

Map for all the features.

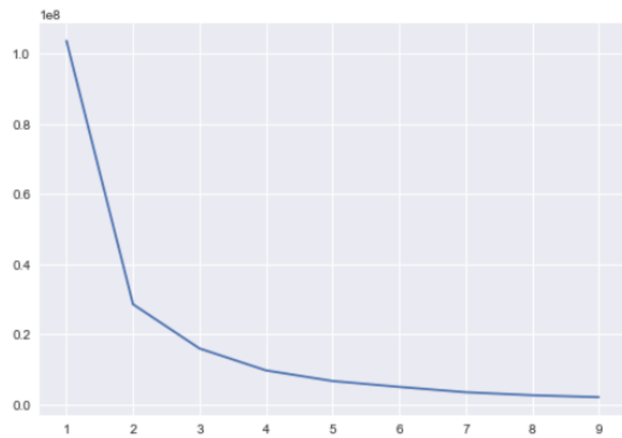


After removing the correlated features and selecting the remaining



Elbow curve with feature selection determined by the above maps

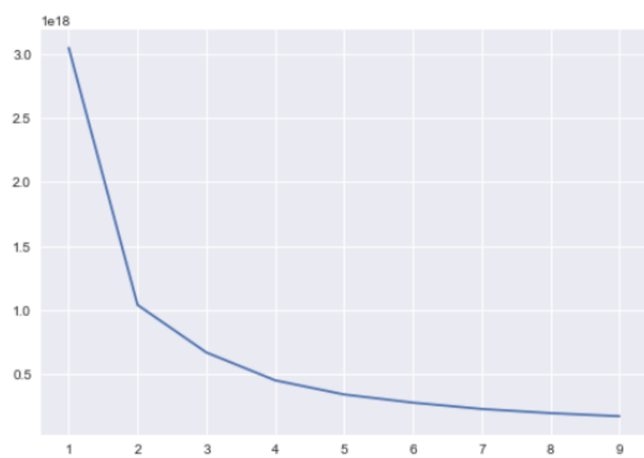
```
k:  1
k:  2
k:  3
k:  4
k:  5
k:  6
k:  7
k:  8
k:  9
```



Elbow Curve for all the entire dataset 5000000 rows

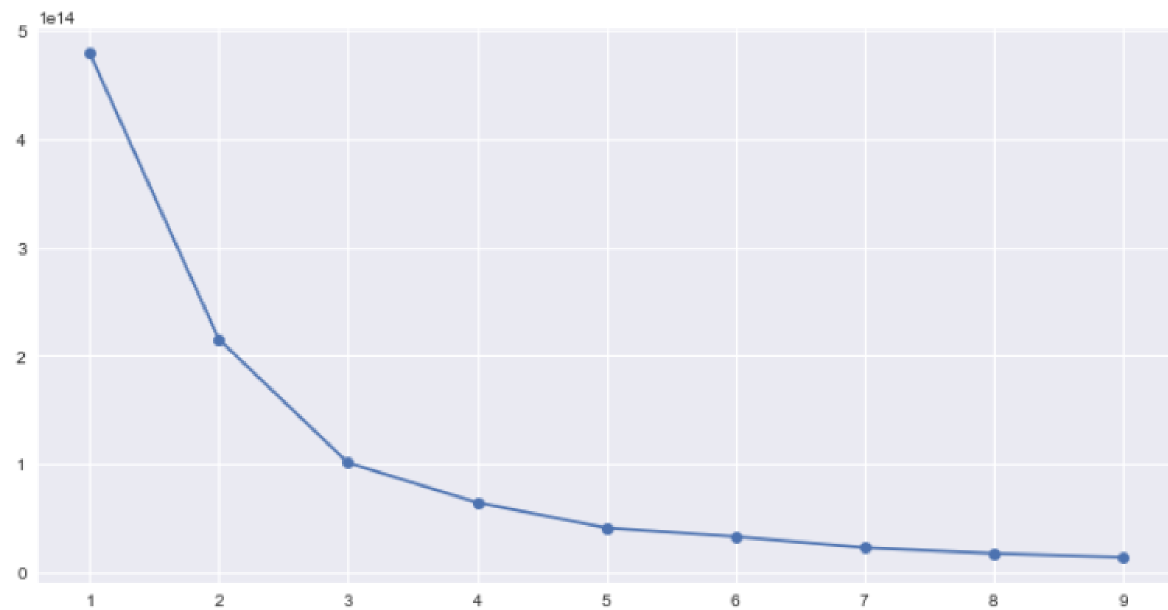
```
# on all the samples
elbow_plot(df[collist], maxK=10)
```

```
k:  1
k:  2
k:  3
k:  4
k:  5
k:  6
k:  7
k:  8
k:  9
```



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.

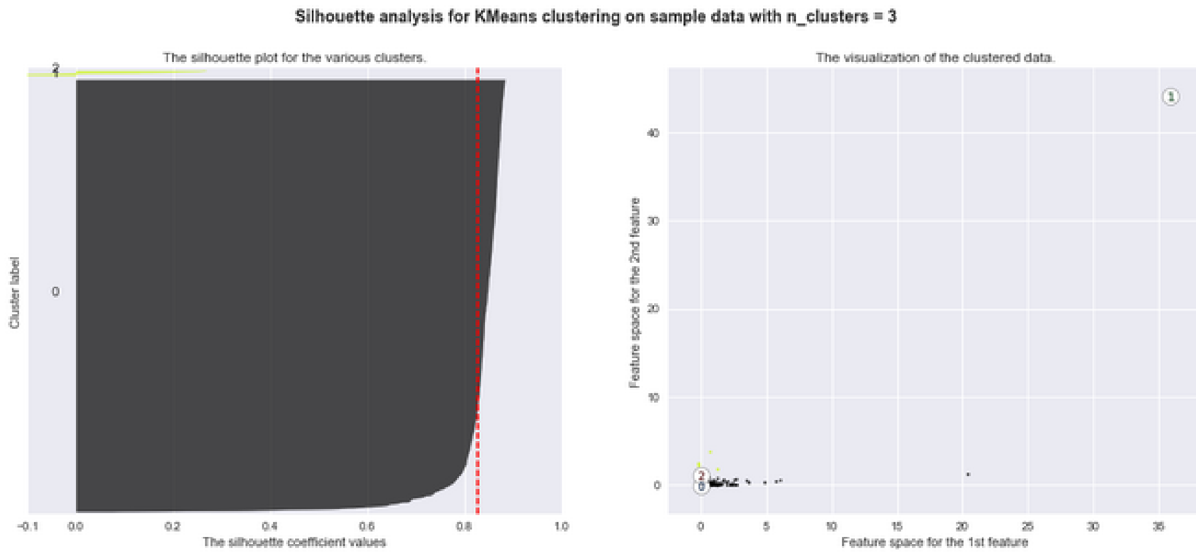
Same elbow curve with data points.



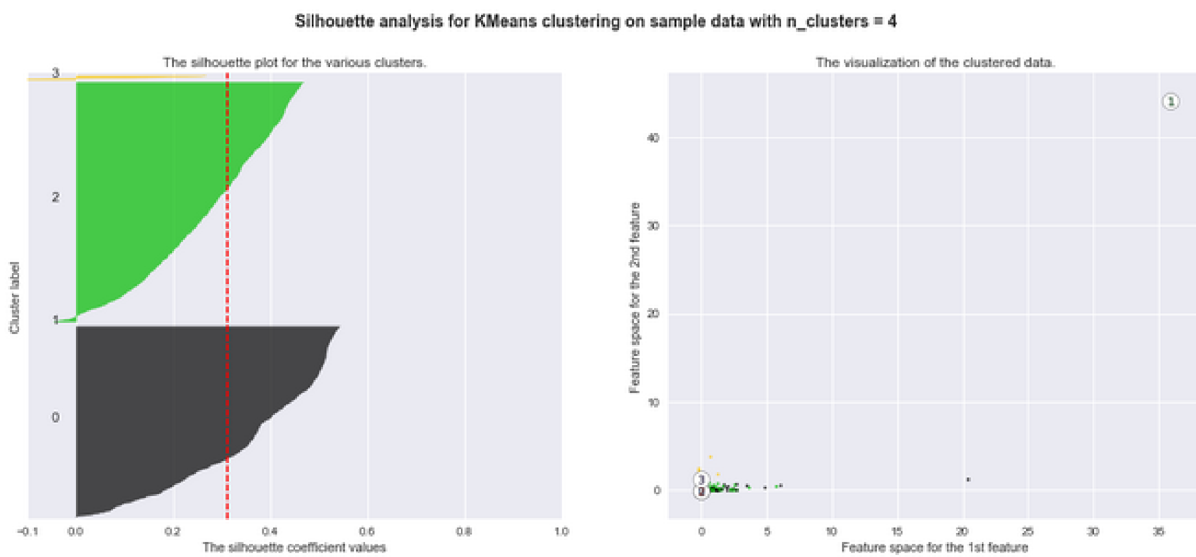
Silhouette Graph

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.

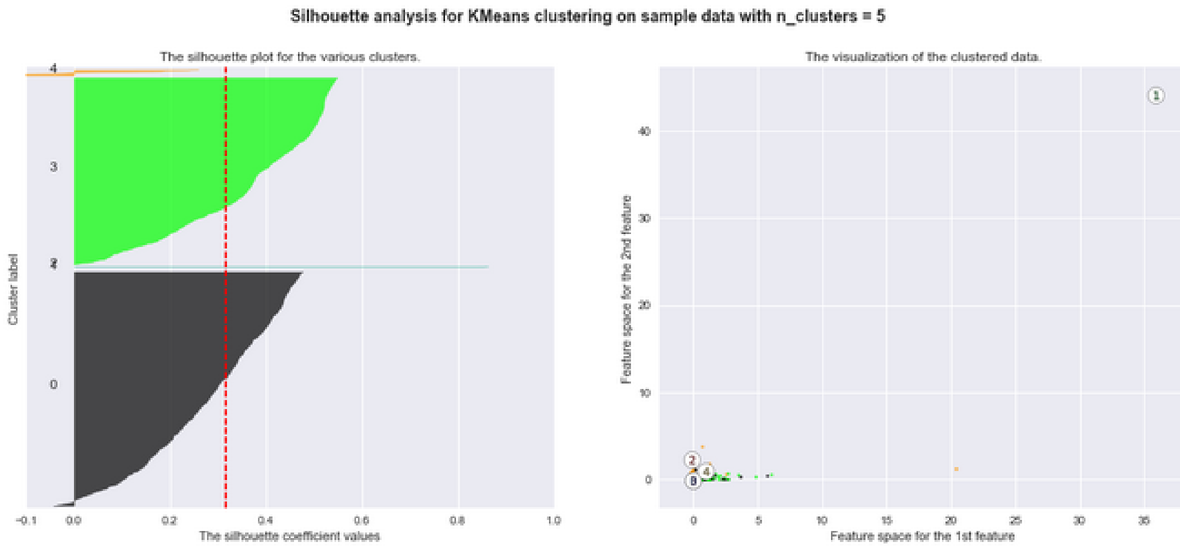
```
Automatically created module for IPython interactive environment
For n_clusters = 3 The average silhouette_score is : 0.82659847925
```



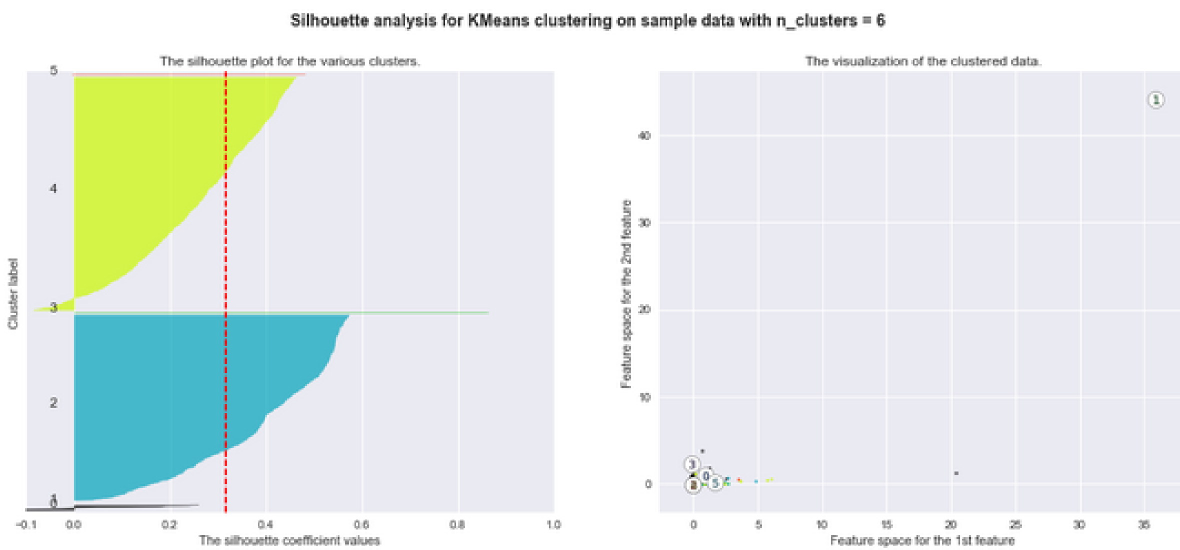

```
For n_clusters = 4 The average silhouette_score is : 0.310775468988
```



```
For n_clusters = 5 The average silhouette_score is : 0.315956187766
```




```
For n_clusters = 6 The average silhouette_score is : 0.316235288854
```



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

```
from sklearn import cluster, datasets
```

```
k_means=cluster.KMeans(n_clusters=3)
```

```
numeric_data=df[['TotalVideos','TotalViews','MaxViews','MinViews','AvgViews','TotalLengthInMinutes','AvgRating','
```

```
numeric_data.head()
```

	TotalVideos	TotalViews	MaxViews	MinViews	AvgViews	TotalLengthInMinutes	AvgRating	TotalRatings	TotalComments	MaxAgeInDays	MinAgeIn
0	1	155772	155772	155772	155772	321	4.830000	96	53	367	
1	2	517754	504302	13452	491549	806	4.500000	414	279	529	
2	11	20883	5069	186	3813	3784	3.069091	227	303	763	
3	1	13	13	13	13	368	1.000000	1	1	761	
4	9	42173	27200	137	19884	2238	3.218888	170	150	706	

```
k_means.fit(numeric_data)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

```
print(k_means.labels_)
```

```
[0 0 0 ..., 0 0 0]
```

```
import numpy as np
from pandas import *
```

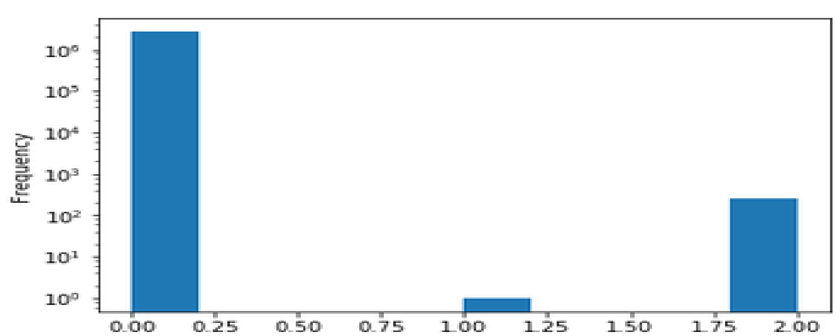
```
df1=DataFrame(numeric_data,columns=['TotalVideos','TotalViews','MaxViews','MinViews','AvgViews','TotalLengthInMin
df2=DataFrame(k_means.labels_,columns=['kmeanlabel'])
df3=pandas.concat([df1, df2], axis=1)
```

```
data=pandas.concat([df, df2], axis=1)
```

```
data[data.kmeanlabel==0]
```

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.

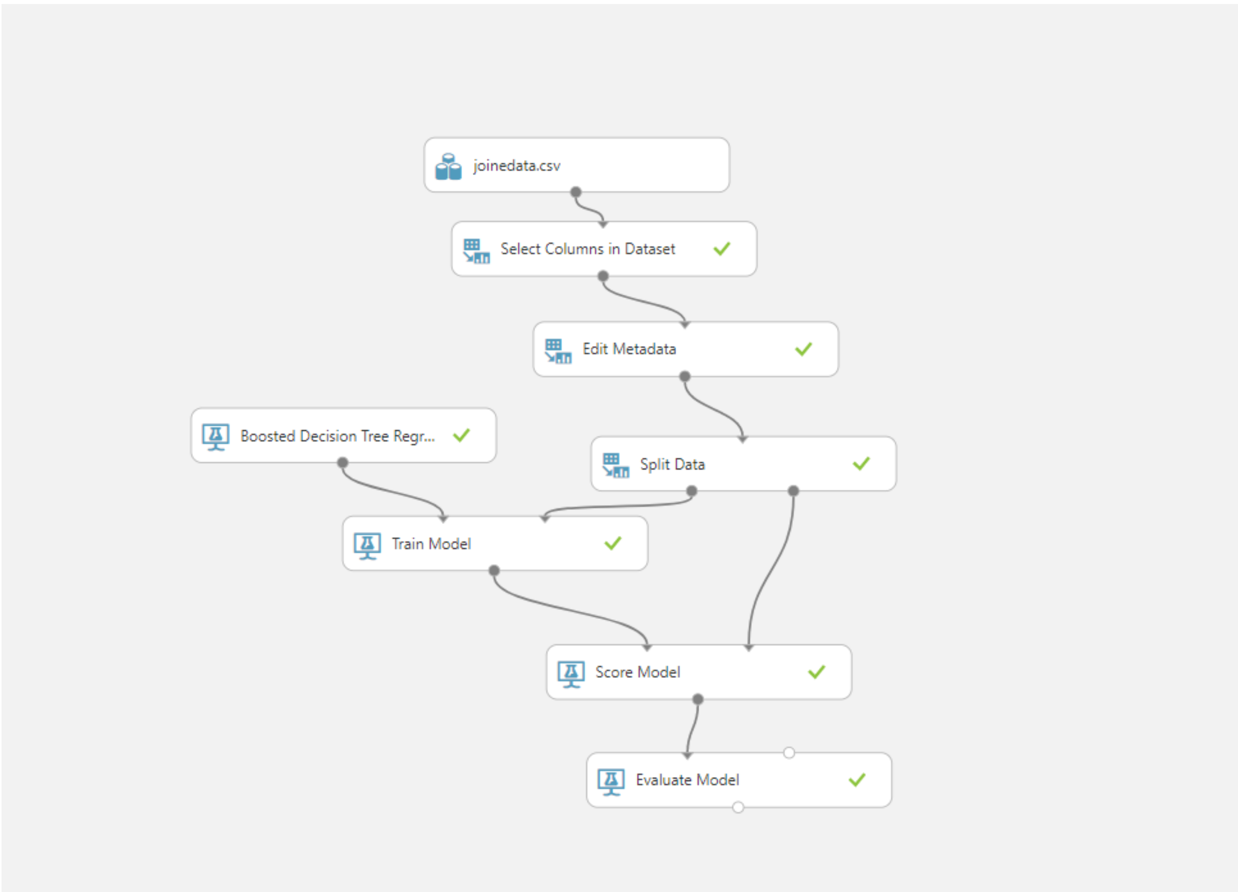
```
data.kmeanlabel.plot(kind='hist')
plt.yscale('log', nonposy='clip')
plt.show()
```



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.



Metrics

Mean Absolute Error	17649.343701
Root Mean Squared Error	38169.066524
Relative Absolute Error	0.439055
Relative Squared Error	0.313403
Coefficient of Determination	0.686597

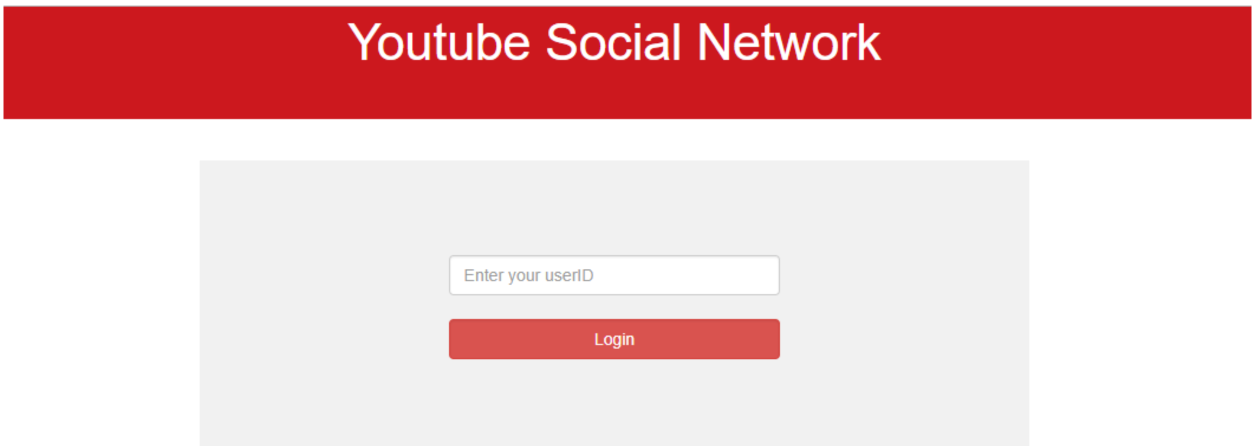
The model gives a high error after normalizing the data, hence we decided to go with the base model.

Web Application

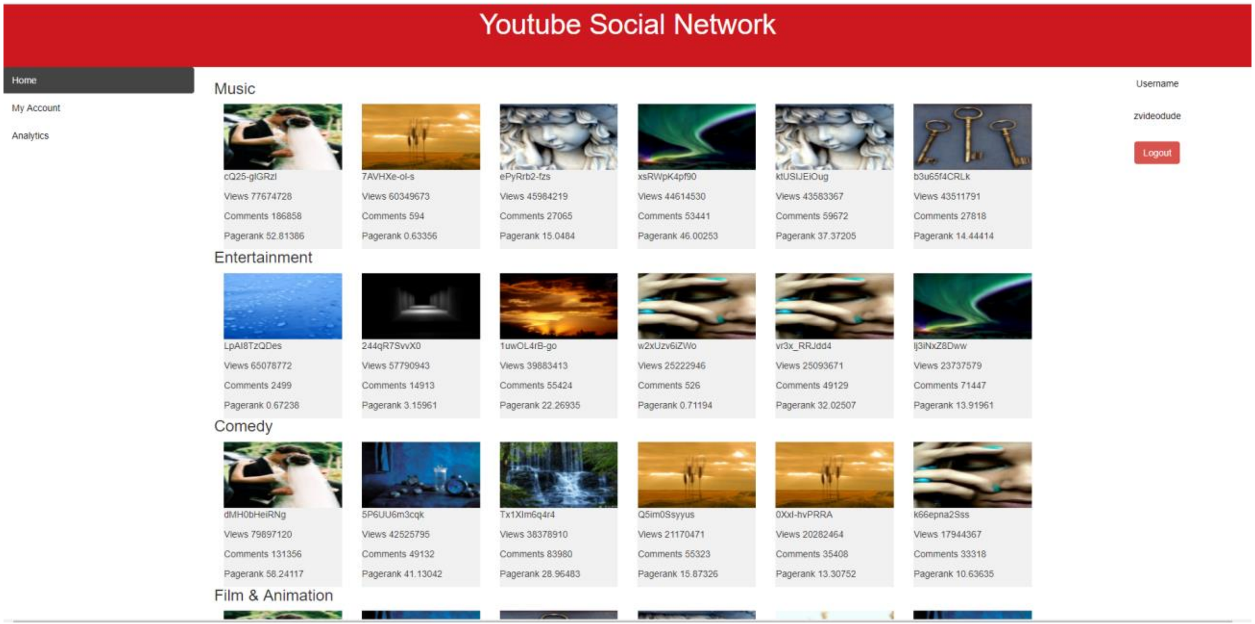
WebApp is deployed on IBM BlueMix

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

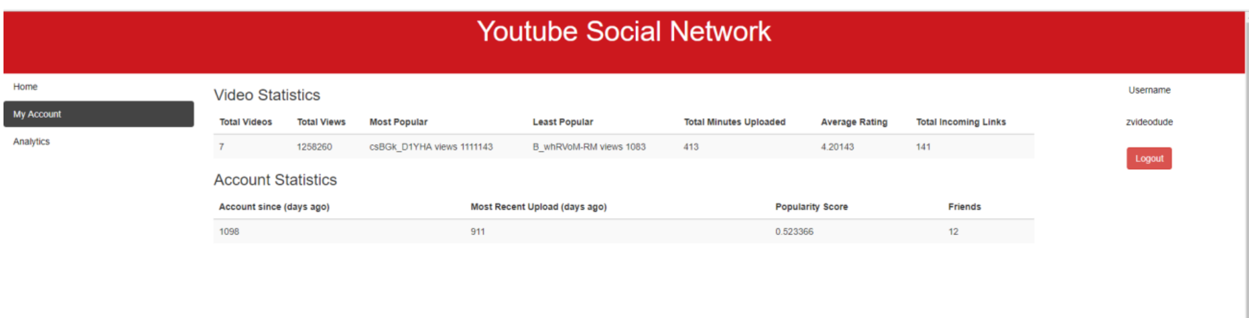
Home Page provides a login . Any user existing in the database can login.



User homepage featuring top videos for each category



MY ACCOUNT: PROVIDES THE USERS STATISTICS FOR THAT USER

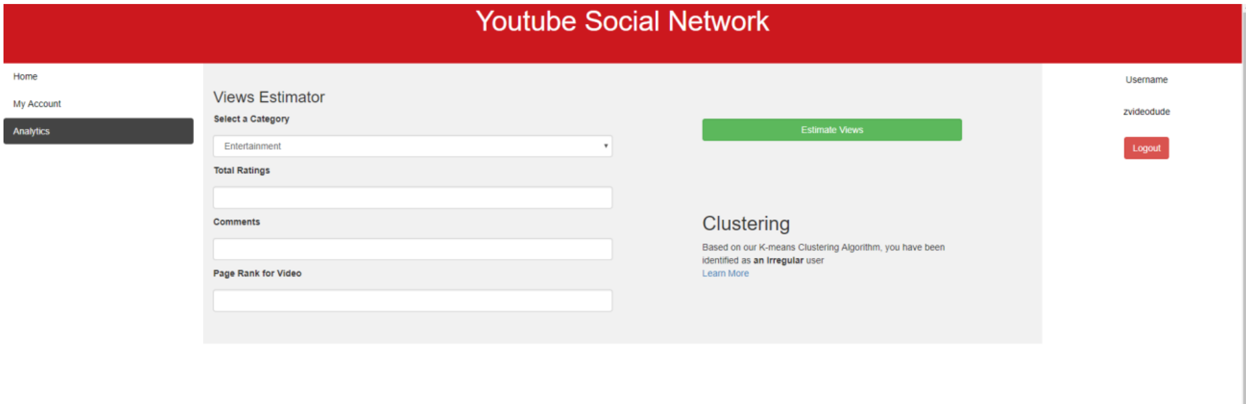


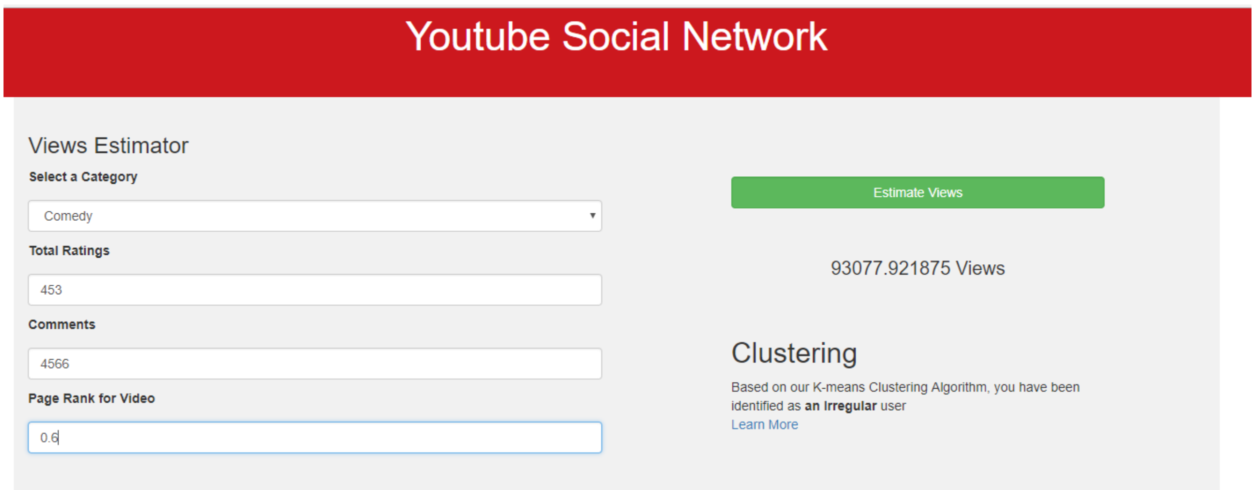
ANALYTICS:

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.