# BIG DATA

# DS402

# FINAL PROJECT REPORT

## LINK SPAM DETECTION BASED ON SPAM MASS DETECTION IN PAGERANKING ALGORITHM

**COURSE IN-CHARGE - AMIT KUMAR SIR**

**GROUP MEMBERS-**

| | |
|---|---|
| KUMARI RENUKA | U101115FCS111 |
| RISHABH KUMAR KANDOI | U101115FCS283 |
| SHAILESH MOHTA | U101115FCS305 |
| UMESH KUMAR CHIDARI | U101115FCS169 |

# ACKNOWLEDGEMENT

*We'd first and foremost like to thank each of our fellow teammates for their equal and invaluable contribution towards the making of this Big Data Project successful, which is given as a part of the course – Big Data. Without the whole team working dedicatedly for the complete semester, we would not have been able to complete a project at this scale on time.*

*Next, we would also like to extend our special thanks of gratitude to our primary mentors Mr. Amit Kumar who helped us in making this project successful.* *Through this project we learned about many new technologies which we were unaware off such as working with the cluster. He also helped us in doing a lot of research and we came to know about so many new things, methods, procedures, and standards and we are really thankful to him for this. We were able to complete this project with all the requirements implemented, only with constant support and guidance from him over time. We would also like to thank him for his suggestions and feedback which played an important role in shaping this project to its output.*

*And lastly, we would like to thank NIIT University and the CSE Department at NIIT University, including the Head of Department (Professor Prosenjit Gupta), for bringing this course in our academic curriculum. We learned a lot through the project we undertook during this semester, concepts that surely cannot be grasped in the classroom, concepts that'll definitely help us frame our career in the future*

# TABLE OF CONTENTS

Introduction to the problem statement .......... 4

Reason for being a Big Data Problem .......... 6

Tools which can be used and what we have choosen .......... 8

Source, Dataset and Algorithm Design .......... 11

Output Screenshot .......... 14

Effectiveness of the solution .......... 19

Performance Analysis .......... 20

Graphical Representation .......... 22

# INTRODUCTION TO THE PROBLEM STATEMENT

## INTRODUCTION

*In the era of google where everything can be found in google search bar, the PageRank relevance has got more weightage than anything else. Most of the businesses are going online to avail the online market. Everything is just a click away. Customers find their stores online to making payment and registering taxes. More businesses want their websites to be on top of every possible search relevant to their type of business. But what about those cases where even after humble tries business agents fail to upgrade their website's PageRank because of web spams popping up every now and then. Tracing these spams is definitely a cumbersome task and managing them is the main concern. PageRank is an algorithm used for calculation of position of a web pages amongst multiple web pages across the web. It's basically correspondence regarding the importance of page and how relevant the page is when compared with other pages and how many links are related to it directly or indirectly. PageRank finds its main usage in our problem of link spam detection*.

## PROBLEM STATEMENT

*Since the PageRank importance has increased drastically over the web so, people are propelled to get listed in first initial pages as much as possible, so if they don't have enough integrity they try to fool google and have their website's PageRank boosted. This leads lot of irrelevant pages finding its listing among first initial pages of dominant search engine like Google. This which in turn leads to inconvenience for actual websites and find hard to get listed with proper ranking. This is basically done by spammers who try to alter ranks in order to bring lot of traffic to their targeted web pages.*

## SOLUTION APPROACH TO THE PROBLEM STATEMENT

*The main goal is the web spam detection, specifically a link spam detection. Spamming is any measured action to boost web page's position in search engine. Spam are collection of those boosted web pages which are result of spamming. Currently, we can find approximately 10-15% of web pages are spam. This are basically done by the humans or web crawlers using boosting and hiding techniques. They mainly focus on fooling dominant search engines like Wikipedia, government, educational websites etc. Hence, this can be achieved by building spam farms in order to increase the page rank. Spam farm consist of several nodes which tends* *to point to a single node in order to increase the PageRank of the pointed node. There are three types of pages from spammers point of view which are Inaccessible, accessible and target pages. Accessible pages are like blog comment pages or any page where spammer can post links to his/her pages. Owned pages are completely controlled by spammers and may spam multiple domain names. We focus on link spamming that targets the PageRank algorithm. PageRank is fairly robust to spamming: a significant increase in score requires a large number of links from low-PageRank nodes and/or some hard to-obtain links from popular nodes, such as The New York Times site www.nytimes.com. Spammers usually try to blend these two strategies, though the former is more prevalent. Hence, initially we find the page rank of each page in the graph. We estimate the spam mass of all web pages by computing and combining two PageRank scores: the regular PageRank of each page and a biased one, in which a large group of known reputable pages receives more weight. Mass estimates can then be used to identify pages that are significant beneficiaries of link spamming with high probability.*

## REASON FOR BEING A BIG DATA PROBLEM

**A) LARGE DATA SET**:

*The problem we are solving involves dealing with a large data set of multiple GBs which is dealt very inefficiently with traditional database management systems. Hence management techniques of big data are needed.* ***Spamming****: Spam consists of varieties of contents like text, image, embedded HTML, MIME attachments.* ***To handle this high volume, high velocity and large varieties of spam****, traditional database management systems are inefficient to use and hence this fall under the category of "big data".*

**B) COMPLEX PROCESSING**:

*Traditionally in cases where data set is smaller or relatable in terms of size to the computation system that are loaded in a single system, to get results out of the data using those computation system, data set is moved to and loaded into the system. While in our case, since the data is much greater in size, if we follow the traditional way of moving data to computation system, the processing of such huge amount of data of different varieties and complexities, by a single system will turn out to be more complex and time consuming and more inefficiency. Thus, we use the data management techniques of big data i.e. of bringing computation processes to data.*

**C) ANALYSIS OF SEARCH ENGINES IS REQUIRED:**

*As we are dealing with a large dataset of webpages and hosts, we have to make use of a search engine for many reasons. In order to perform spam detection and filtering on the dataset, we expect the search engine to have some reliable white-list or black-list of webpages that would assist in filtering of webpages and performing analytical functions in calculation of page rank. Search engine can perform simple analytical functions such as counts, sum, averages, and so forth over hundreds of millions (billions!) of records in a second or two, which is 1000's of times faster than any alternative. S****earch engines can search over structured content (e.g. tables of data) way better than relational databases.*** *Search engines are so much faster and more flexible than other searching techniques about 1000x better and about 1,000,000x faster.* ***Search engines are more scalable than just about any other type of access systems.***

## D) BIG DEPENDENCY GRAPHS MAY EXIST:

*Since we are performing spam detection and filtering in a large dataset of webpages and hosts, big spam farms and spam alliances have to be dealt with. The spam farms and alliances generally involve lots of direct and indirect links between boosting and other spam nodes and the target nodes with the motive to increase the PageRank score of the target nodes. Managing such dependencies with traditional techniques will be inefficient due to the complexity involved. Hence, we resort to Big Data management techniques to deal with such scenarios.*

# TOOLS USED TO SOLVE OUR BIG DATA PROBLEM AND WHAT WE TEND TO CHOOSE

## 1. APACHE HADOOP

*Apache Hadoop is an open source software framework used for the distributed storage and processing of large data sets using the MapReduce programming model. It consists of computer clusters built using commodity hardware. All the different modules in Hadoop are actually designed with the assumption that different hardware failures are commonly observed occurrences and they should be automatically handled by the framework.*

*Features:*

- *The Hadoop framework is mostly written in Java, with some of its native code in C. Its command line utilities are written as shell scripts.*
- *Apache Hadoop consists of a large storage part, known as the Hadoop Distributed File System.*
- *It uses the* ***MapReduce*** *programming model to process large data sets.*
- *Hadoop splits different files into large blocks and then distributes them across various nodes in a cluster.*
- *It transfers packaged code into different nodes to process the data in parallel.*
- *Apache Hadoop makes use of the data locality approach, where nodes manipulate all the data they have access to. This allows the large dataset to be processed faster and even more efficiently.*
- *The base Apache Hadoop framework is composed of different modules:* ***Hadoop Common, HDFS, Hadoop YARN*** *and* ***Hadoop MapReduce****.*

*Taking the above features into account, Apache Hadoop is suitable for implementing PageRank algorithm on a large data set of web pages and performing spam detection and filtering on it.*

## 1. APACHE PIG

*Apache Pig is a platform for analysing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets. Pig is more of a data flow language i.e. you can decide how your data should be computed for your data analysis. Pig scripts are internally converted into Map-Reduce tasks using a component called Pig Engine. We can perform all data manipulation operations in pig.*

***Below are few advantages of pig-***

***Easy to learn** - Pig will be helpful for people like me who don't know java but want to analyse my dataset. Yes, a 100 lines of java code required to analyse your data set can be done by simple 10 lines of Pig Latin*

***Extensibility** - You can create your own user defined functions in java and you can use it in pig programming*

***Handles all kinds of data**- Pig can handle all kinds of data-structured and unstructured data. Another advantage is that you don't even need to know the schema of the data set.*

*Apart from the primitive map-reduce advantages, it is capable enough to handle all kinds of data (structured and unstructured). Hence it is a suitable choice for spam detection in a large data set of webpages as spam consists of varieties of contents like text, image, embedded HTML, MIME attachments.*

## 2. APACHE SPARK

*Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.*

*Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these work load in a respective system, it reduces the management burden of maintaining separate tools.*

*Apache Spark has following features:*

***Speed** − Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.*

***Supports multiple languages** − Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.*

***Advanced Analytics** − Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.*

*Apart from the primitive map-reduce advantages, Spark is noted to be much more efficient than MapReduce because it decreases I/O cost and allows iterative computing much faster. Hence it is an apt choice for implementation of PageRank algorithm on a large dataset of web pages and performing spam detection and filtering on it.*

## TOOL SELECTED FOR SOLVING OUR PROBLEM:

### *Hadoop MapReduce and HDFS*

*Our primary reason for choosing the above-mentioned tool is our comfort in using the tool. We feel that tweaking a set of code is relatively easy in MapReduce framework. Working on other platforms like Apache Pig involves an extra step of converting the Pig Latin code to MapReduce tasks and hence time consumed is more. In Hadoop MapReduce framework, we can write our code in multiple programming languages such as C, Python, Java and so forth. Testability is easier in MapReduce Framework. At the initially stage, for large amount of data, finding optimum memory in Spark is difficult and thus the memory consumption in Spark is more than that of MapReduce. In MapReduce framework, once the job is done, the memory is freed while Spark is really greedy when it comes to cluster utilization.*

# DATASET, SOURCE AND ALGORITHM DESIGN

## ALGORITHM DESIGN

---

### *ALGORITHM 1: Normal PageRank Calculation*

```
1.  Input = Search engine data in the form of graph with each node and it's out degree list.
2.  Output = PageRank of each node.
3.Initialization:
      temp=0, sum=0, c=0.85 (Damping Factor), n=Total number of nodes in the graph.
      O [ ]: In-degree list of each node.
      Out degree of each node in O: Out [ ]
4. for all x node do
5.     PageRank $p_x = 1/n$
6. end for
7.        for all x do
8                   for all i do
9.                           temp = p (O[i]) / Out [O[i]]
10.                          sum = sum + temp
11.                 end for
12.         $p_x = c * sum + (1 - c)/n$
13. end for
```

---

### *ALGORITHM 2: Core PageRank Calculation*

```
1.  Input = Search engine data in the form of graph with each node and it's out degree list.
2.  Output = Core PageRank of each node.
3.Initialization:
      temp=0, sum=0, c=0.85 (Damping Factor), n=Total number of nodes in the graph.
      O [ ]: In-degree list of each node.
      Out degree of each node in O: Out [ ]
```

```
 Obtain $V^+$ (estimated good core nodes) & $V^-$ (estimated spam nodes)
 Obtain x where x = number of ($V^+$)
4. for all x node do
5.    PageRank $p_x = 1/n$
6. end for
7.       for all x do
8              for all i do
9.                   temp = p (O[i]) / Out [O[i]]
10.                  sum = sum + temp
11.            end for
12.      $p_x = c * sum + (1 - c)/n$
13. end for
```

----------------------------------------------------------------------------------------------------------------------------------

***ALGORITHM 3:* Mass Based Spam Nodes Detection**

```
1.  Input = PageRank (output of Algorithm 1) p of each node. Good core $V^+$, Relative Mass
             Threshold T, PageRank threshold ρ (output of Algorithm 2), PageRank p' of $V^+$.
2.  Output = Mass based spam detection of nodes.
3.Initialization:
     Consider set of spam nodes S, S [ ] = null
4. for all x node in $V^-$ do
5.    Calculate m
6.     m = (p – p')/p
7.        for all x: $p_x \geq \rho$ do
8.              if $m_x \geq$ T then
9.                    S = S U {x}
10.             end if
11.       end for
12.  end for
```

## DATASET

*We have considered the web spam dataset from this provided source. This is a large collection of annotated spam/nonspam hosts labelled by a group of volunteers. The base data is a set of 105,896,555 pages in 114,529 hosts in the .UK domain. The data was downloaded in May 2006 by the Laboratory of Web Algorithmic, Università degli Studi di Milano, with the support of the DELIS EU - FET research project.*

*Initially we have implemented the project using a smaller dataset which consist of 5 nodes labelled as spam/ non-spam in order to check the working of the code written. The code is written in Hadoop, using MapReduce in java. The code works fine with the smaller dataset that we have generated.*

## DATASET SOURCE

*http://chato.cl/webspam/datasets/uk2006/*

## REFERENCES

*[1] Gyongyi, Zoltan, Pavel Berkhin, Hector Garcia-Molina, and Jan Pedersen. "Link spam detection based on mass estimation." In Proceedings of the 32nd international conference on Very large data bases, pp. 439-450. VLDB Endowment, 2006.*

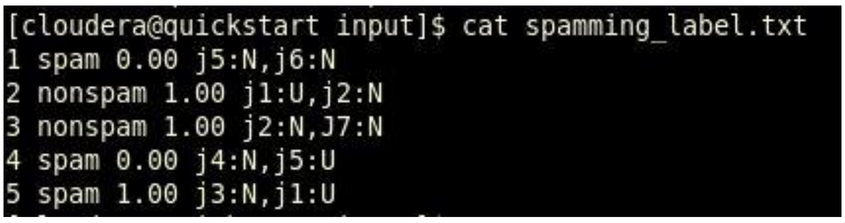
## OUTPUT SCREENSHOT

*The code is written in Hadoop, using MapReduce in java. Initially we have implemented the project using a smaller dataset which consist of 5 nodes labelled as spam/ non-spam in order to check the working of the code written. The code works fine with the smaller dataset that we have generated.*

*Figure 1, 2, 3 represents the inputs of the MapReduce program. The Figure 1 gives the information about the nodes with labelled as spam or non-spam. The Figure 2 gives the information about the nodes with its hostname and Figure 3 represents the graph in which the information regarding the nodes with its adjacency list is given.*

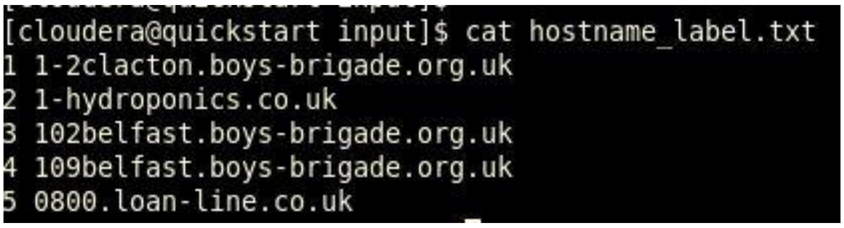
.

**FIG 1: NODES WITH LABEL SPAM/NONSPAM**



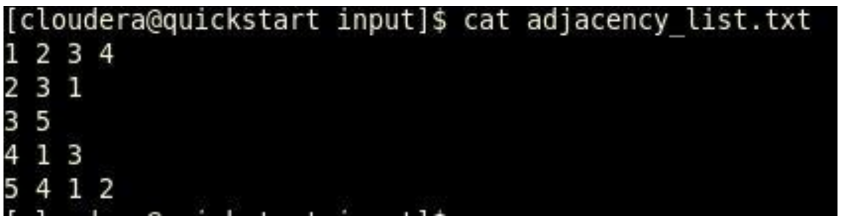
```
[cloudera@quickstart input]$ cat spamming_label.txt
1 spam 0.00 j5:N,j6:N
2 nonspam 1.00 j1:U,j2:N
3 nonspam 1.00 j2:N,J7:N
4 spam 0.00 j4:N,j5:U
5 spam 1.00 j3:N,j1:U
```

**FIG 2: NODES WITH HOSTNAME LABEL**



```
[cloudera@quickstart input]$ cat hostname_label.txt
1 1-2clacton.boys-brigade.org.uk
2 1-hydroponics.co.uk
3 102belfast.boys-brigade.org.uk
4 109belfast.boys-brigade.org.uk
5 0800.loan-line.co.uk
```

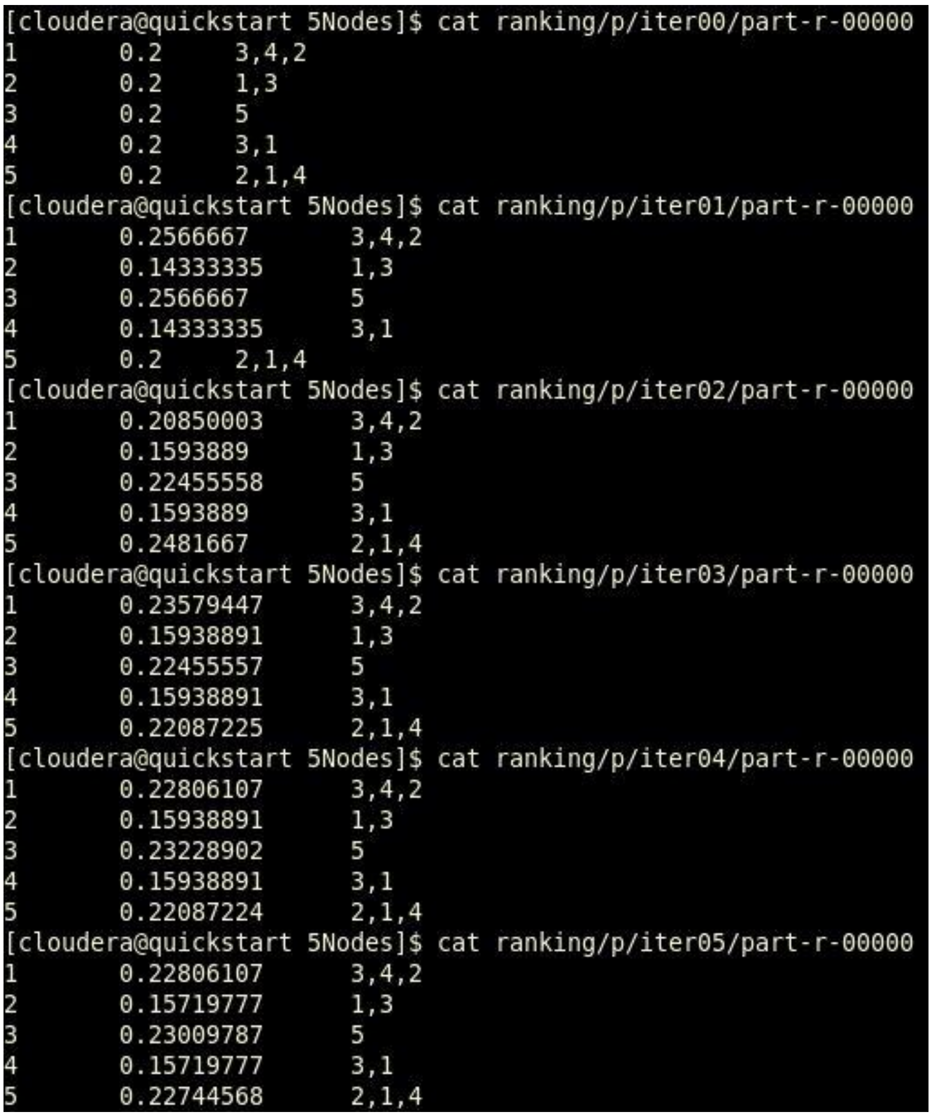
**FIG 3: NODES WITH ITS ADJACENCY LIST**



```
[cloudera@quickstart input]$ cat adjacency_list.txt
1 2 3 4
2 3 1
3 5
4 1 3
5 4 1 2
```

*Initially we have calculated the normal PageRank of each node by considering 1/n as the initial PageRank of each node. Five iterations are considered to calculate the final page rank of each nodes. Figure 4 represents the output of each iteration used to calculate the normal PageRank.*

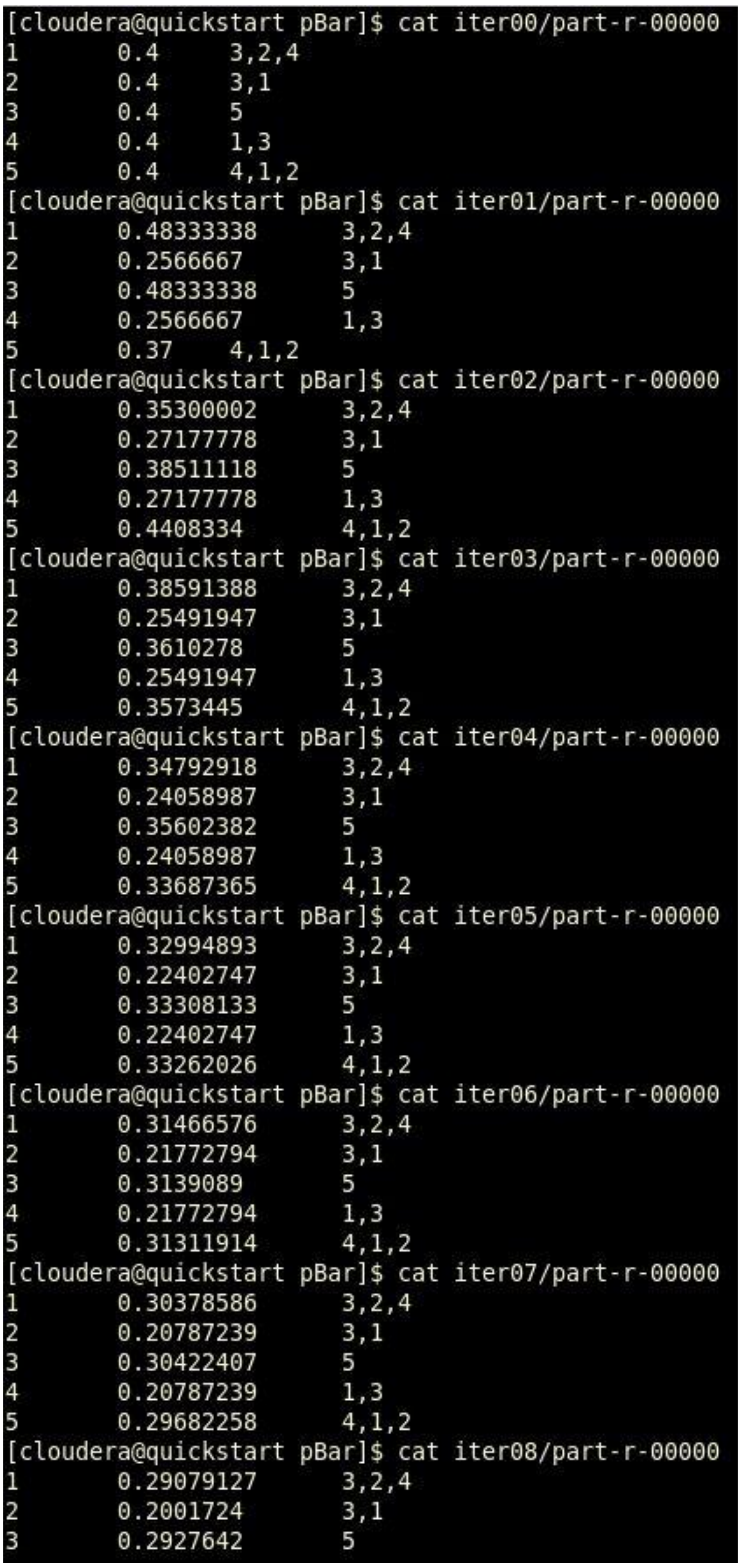
**FIG 4: ITERATION OUTPUT FOR CALCULATING NORMAL PAGERANK**



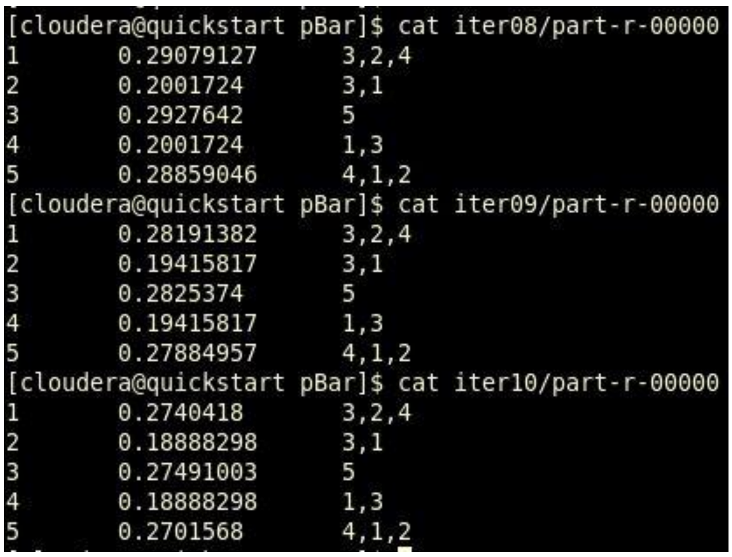
```
[cloudera@quickstart 5Nodes]$ cat ranking/p/iter00/part-r-00000
1       0.2     3,4,2
2       0.2     1,3
3       0.2     5
4       0.2     3,1
5       0.2     2,1,4
[cloudera@quickstart 5Nodes]$ cat ranking/p/iter01/part-r-00000
1       0.2566667       3,4,2
2       0.14333335      1,3
3       0.2566667       5
4       0.14333335      3,1
5       0.2     2,1,4
[cloudera@quickstart 5Nodes]$ cat ranking/p/iter02/part-r-00000
1       0.20850003      3,4,2
2       0.1593889       1,3
3       0.22455558      5
4       0.1593889       3,1
5       0.2481667       2,1,4
[cloudera@quickstart 5Nodes]$ cat ranking/p/iter03/part-r-00000
1       0.23579447      3,4,2
2       0.15938891      1,3
3       0.22455557      5
4       0.15938891      3,1
5       0.22087225      2,1,4
[cloudera@quickstart 5Nodes]$ cat ranking/p/iter04/part-r-00000
1       0.22806107      3,4,2
2       0.15938891      1,3
3       0.23228902      5
4       0.15938891      3,1
5       0.22087224      2,1,4
[cloudera@quickstart 5Nodes]$ cat ranking/p/iter05/part-r-00000
1       0.22806107      3,4,2
2       0.15719777      1,3
3       0.23009787      5
4       0.15719777      3,1
5       0.22744568      2,1,4
```

*Then we have calculated the core PageRank of each node by considering x/n as the initial PageRank of each node where x represents the number of good nodes(V+). Ten iterations are considered to calculate the final core page rank of each nodes. Figure 5 represents the output of each iteration used to calculate the core PageRank.*

**FIG 5: ITERATION OUTPUT FOR CALCULATING CORE PAGERANK**



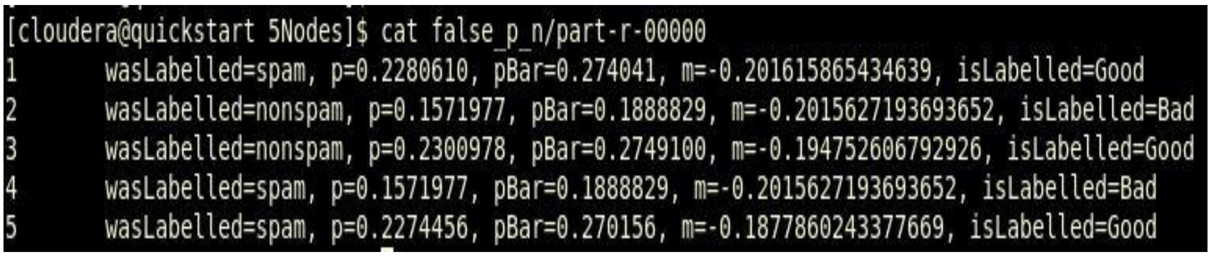
```
[cloudera@quickstart pBar]$ cat iter00/part-r-00000
1       0.4     3,2,4
2       0.4     3,1
3       0.4     5
4       0.4     1,3
5       0.4     4,1,2
[cloudera@quickstart pBar]$ cat iter01/part-r-00000
1       0.48333338      3,2,4
2       0.2566667       3,1
3       0.48333338      5
4       0.2566667       1,3
5       0.37    4,1,2
[cloudera@quickstart pBar]$ cat iter02/part-r-00000
1       0.35300002      3,2,4
2       0.27177778      3,1
3       0.38511118      5
4       0.27177778      1,3
5       0.4408334       4,1,2
[cloudera@quickstart pBar]$ cat iter03/part-r-00000
1       0.38591388      3,2,4
2       0.25491947      3,1
3       0.3610278       5
4       0.25491947      1,3
5       0.3573445       4,1,2
[cloudera@quickstart pBar]$ cat iter04/part-r-00000
1       0.34792918      3,2,4
2       0.24058987      3,1
3       0.35602382      5
4       0.24058987      1,3
5       0.33687365      4,1,2
[cloudera@quickstart pBar]$ cat iter05/part-r-00000
1       0.32994893      3,2,4
2       0.22402747      3,1
3       0.33308133      5
4       0.22402747      1,3
5       0.33262026      4,1,2
[cloudera@quickstart pBar]$ cat iter06/part-r-00000
1       0.31466576      3,2,4
2       0.21772794      3,1
3       0.3139089       5
4       0.21772794      1,3
5       0.31311914      4,1,2
[cloudera@quickstart pBar]$ cat iter07/part-r-00000
1       0.30378586      3,2,4
2       0.20787239      3,1
3       0.30422407      5
4       0.20787239      1,3
5       0.29682258      4,1,2
[cloudera@quickstart pBar]$ cat iter08/part-r-00000
1       0.29079127      3,2,4
2       0.2001724       3,1
3       0.2927642       5
```



```
[cloudera@quickstart pBar]$ cat iter08/part-r-00000
1       0.29079127      3,2,4
2       0.2001724       3,1
3       0.2927642       5
4       0.2001724       1,3
5       0.28859046      4,1,2
[cloudera@quickstart pBar]$ cat iter09/part-r-00000
1       0.28191382      3,2,4
2       0.19415817      3,1
3       0.2825374       5
4       0.19415817      1,3
5       0.27884957      4,1,2
[cloudera@quickstart pBar]$ cat iter10/part-r-00000
1       0.2740418       3,2,4
2       0.18888298      3,1
3       0.27491003      5
4       0.18888298      1,3
5       0.2701568       4,1,2
```

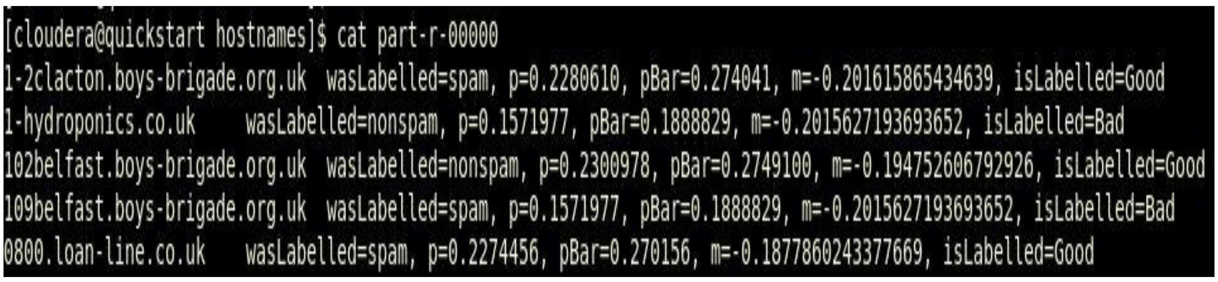
*Figure 6,7 represents the final output after the implementation of the spam detection algorithm which gives the information regarding the nodes which are actually spam and it also gives the information regarding false positive and false negative nodes.*

**FIG 6: FINAL OUTPUT AFTER IMPLEMENTATION OF SPAM DETECTION ALGORITHM WITH NODE NUMBER**



```
[cloudera@quickstart 5Nodes]$ cat false_p_n/part-r-00000
1       wasLabelled=spam, p=0.2280610, pBar=0.274041, m=-0.201615865434639, isLabelled=Good
2       wasLabelled=nonspam, p=0.1571977, pBar=0.1888829, m=-0.2015627193693652, isLabelled=Bad
3       wasLabelled=nonspam, p=0.2300978, pBar=0.2749100, m=-0.194752606792926, isLabelled=Good
4       wasLabelled=spam, p=0.1571977, pBar=0.1888829, m=-0.2015627193693652, isLabelled=Bad
5       wasLabelled=spam, p=0.2274456, pBar=0.270156, m=-0.1877860243377669, isLabelled=Good
```

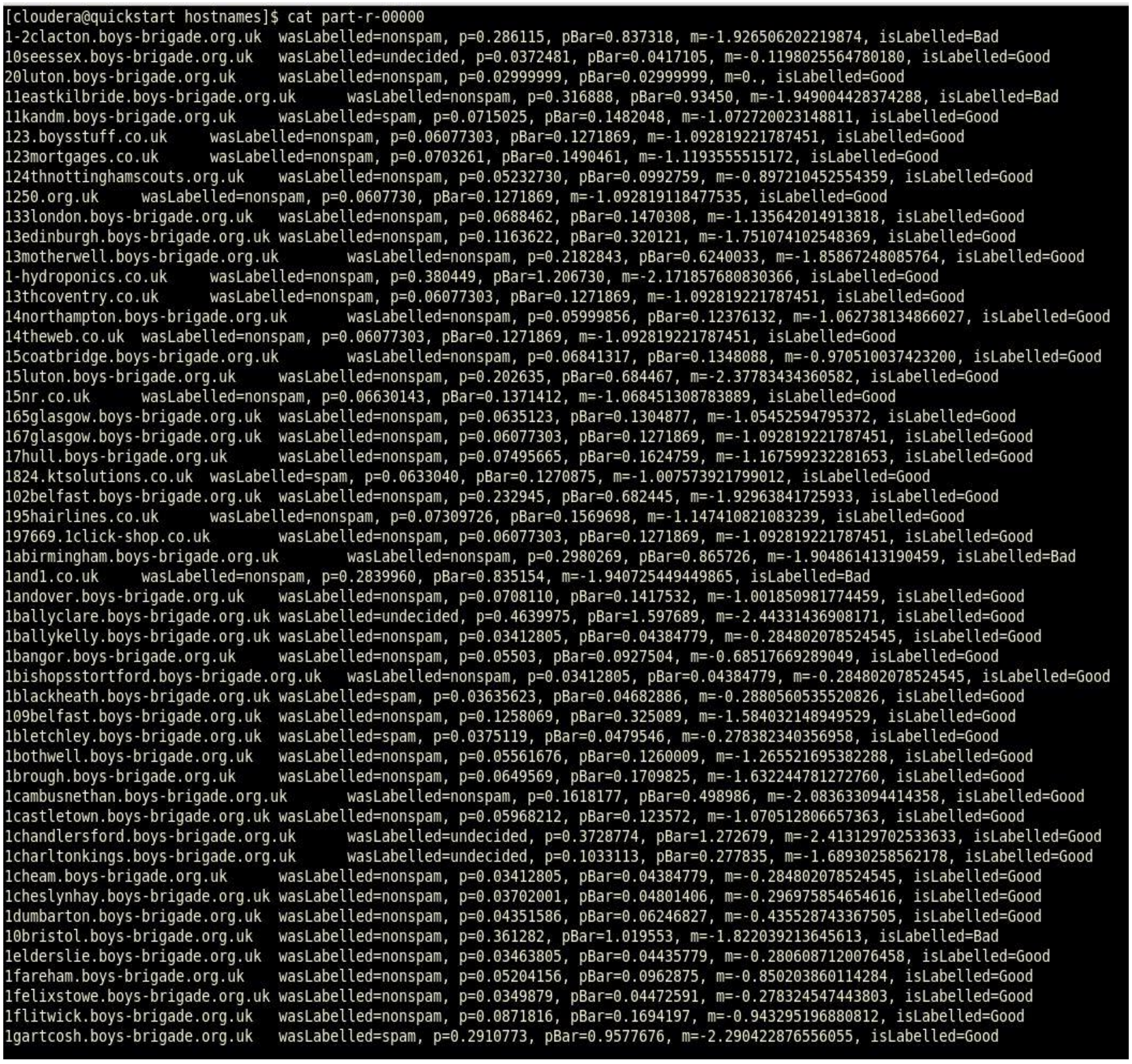
**FIG 7: FINAL OUTPUT AFTER IMPLEMENTATION OF SPAM DETECTION ALGORITHM WITH NODE HOSTNAME LABEL**



```
[cloudera@quickstart hostnames]$ cat part-r-00000
1-2clacton.boys-brigade.org.uk  wasLabelled=spam, p=0.2280610, pBar=0.274041, m=-0.201615865434639, isLabelled=Good
1-hydroponics.co.uk     wasLabelled=nonspam, p=0.1571977, pBar=0.1888829, m=-0.2015627193693652, isLabelled=Bad
102belfast.boys-brigade.org.uk  wasLabelled=nonspam, p=0.2300978, pBar=0.2749100, m=-0.194752606792926, isLabelled=Good
109belfast.boys-brigade.org.uk  wasLabelled=spam, p=0.1571977, pBar=0.1888829, m=-0.2015627193693652, isLabelled=Bad
0800.loan-line.co.uk    wasLabelled=spam, p=0.2274456, pBar=0.270156, m=-0.1877860243377669, isLabelled=Good
```

*After testing the code with a smaller dataset, we used a larger dataset with 1000 nodes labelled as spam/ non-spam from the dataset source provided in above section. We have taken 3 inputs, 1 gives the information about the nodes with labelled as spam or non-spam, 2 gives the information about the nodes with its hostname and 3 represents the graph in which the information regarding the nodes with its adjacency list is given. Initially we have calculated the normal PageRank of each node by considering 1/n as the initial PageRank of each node. Then we have calculated the core PageRank of each node by considering x/n as the initial PageRank of each node where x represents the number of good nodes(V+). Figure 8 represents the final output after the implementation of the spam detection algorithm.*

**FIG 8: FINAL OUTPUT AFTER IMPLEMENTATION OF SPAM DETECTION ALGORITHM WITH NODE HOSTNAME LABEL FOR 1000 NODES**



```
[cloudera@quickstart hostnames]$ cat part-r-00000
1-2clacton.boys-brigade.org.uk  wasLabelled=nonspam, p=0.286115, pBar=0.837318, m=-1.926506202219874, isLabelled=Bad
10seessex.boys-brigade.org.uk   wasLabelled=undecided, p=0.0372481, pBar=0.0417105, m=-0.1198025564780180, isLabelled=Good
20luton.boys-brigade.org.uk     wasLabelled=nonspam, p=0.02999999, pBar=0.02999999, m=0., isLabelled=Good
11eastkilbride.boys-brigade.org.uk      wasLabelled=nonspam, p=0.316888, pBar=0.93450, m=-1.949004428374288, isLabelled=Bad
11kandm.boys-brigade.org.uk     wasLabelled=spam, p=0.0715025, pBar=0.1482048, m=-1.072720023148811, isLabelled=Good
123.boysstuff.co.uk     wasLabelled=nonspam, p=0.06077303, pBar=0.1271869, m=-1.092819221787451, isLabelled=Good
123mortgages.co.uk      wasLabelled=nonspam, p=0.0703261, pBar=0.1490461, m=-1.1193555515172, isLabelled=Good
124thnottinghamscouts.org.uk    wasLabelled=nonspam, p=0.05232730, pBar=0.0992759, m=-0.897210452554359, isLabelled=Good
1250.org.uk     wasLabelled=nonspam, p=0.0607730, pBar=0.1271869, m=-1.092819118477535, isLabelled=Good
133london.boys-brigade.org.uk   wasLabelled=nonspam, p=0.0688462, pBar=0.1470308, m=-1.135642014913818, isLabelled=Good
13edinburgh.boys-brigade.org.uk wasLabelled=nonspam, p=0.1163622, pBar=0.320121, m=-1.751074102548369, isLabelled=Good
13motherwell.boys-brigade.org.uk        wasLabelled=nonspam, p=0.2182843, pBar=0.6240033, m=-1.85867248085764, isLabelled=Good
1-hydroponics.co.uk     wasLabelled=nonspam, p=0.380449, pBar=1.206730, m=-2.171857680830366, isLabelled=Good
13thcoventry.co.uk      wasLabelled=nonspam, p=0.06077303, pBar=0.1271869, m=-1.092819221787451, isLabelled=Good
14northampton.boys-brigade.org.uk       wasLabelled=nonspam, p=0.05999856, pBar=0.12376132, m=-1.062738134866027, isLabelled=Good
14theweb.co.uk  wasLabelled=nonspam, p=0.06077303, pBar=0.1271869, m=-1.092819221787451, isLabelled=Good
15coatbridge.boys-brigade.org.uk        wasLabelled=nonspam, p=0.06841317, pBar=0.1348088, m=-0.970510037423200, isLabelled=Good
15luton.boys-brigade.org.uk     wasLabelled=nonspam, p=0.202635, pBar=0.684467, m=-2.37783434360582, isLabelled=Good
15nr.co.uk      wasLabelled=nonspam, p=0.06630143, pBar=0.1371412, m=-1.068451308783889, isLabelled=Good
165glasgow.boys-brigade.org.uk  wasLabelled=nonspam, p=0.0635123, pBar=0.1304877, m=-1.05452594795372, isLabelled=Good
167glasgow.boys-brigade.org.uk  wasLabelled=nonspam, p=0.06077303, pBar=0.1271869, m=-1.092819221787451, isLabelled=Good
17hull.boys-brigade.org.uk      wasLabelled=nonspam, p=0.07495665, pBar=0.1624759, m=-1.167599232281653, isLabelled=Good
1824.ktsolutions.co.uk  wasLabelled=spam, p=0.0633040, pBar=0.1270875, m=-1.007573921799012, isLabelled=Good
102belfast.boys-brigade.org.uk  wasLabelled=nonspam, p=0.232945, pBar=0.682445, m=-1.92963841725933, isLabelled=Good
195hairlines.co.uk      wasLabelled=nonspam, p=0.07309726, pBar=0.1569698, m=-1.147410821083239, isLabelled=Good
197669.1click-shop.co.uk        wasLabelled=nonspam, p=0.06077303, pBar=0.1271869, m=-1.092819221787451, isLabelled=Good
1abirmingham.boys-brigade.org.uk        wasLabelled=nonspam, p=0.2980269, pBar=0.865726, m=-1.904861413190459, isLabelled=Bad
1and1.co.uk     wasLabelled=nonspam, p=0.2839960, pBar=0.835154, m=-1.940725449449865, isLabelled=Bad
1andover.boys-brigade.org.uk    wasLabelled=nonspam, p=0.0708110, pBar=0.1417532, m=-1.001850981774459, isLabelled=Good
1ballyclare.boys-brigade.org.uk wasLabelled=undecided, p=0.4639975, pBar=1.597689, m=-2.44331436908171, isLabelled=Good
1ballykelly.boys-brigade.org.uk wasLabelled=nonspam, p=0.03412805, pBar=0.04384779, m=-0.284802078524545, isLabelled=Good
1bangor.boys-brigade.org.uk     wasLabelled=nonspam, p=0.05503, pBar=0.0927504, m=-0.68517669289049, isLabelled=Good
1bishopsstortford.boys-brigade.org.uk   wasLabelled=nonspam, p=0.03412805, pBar=0.04384779, m=-0.284802078524545, isLabelled=Good
1blackheath.boys-brigade.org.uk wasLabelled=spam, p=0.03635623, pBar=0.04682886, m=-0.2880560535520826, isLabelled=Good
109belfast.boys-brigade.org.uk  wasLabelled=nonspam, p=0.1258069, pBar=0.325089, m=-1.584032148949529, isLabelled=Good
1bletchley.boys-brigade.org.uk  wasLabelled=spam, p=0.0375119, pBar=0.0479546, m=-0.278382340356958, isLabelled=Good
1bothwell.boys-brigade.org.uk   wasLabelled=nonspam, p=0.05561676, pBar=0.1260009, m=-1.265521695382288, isLabelled=Good
1brough.boys-brigade.org.uk     wasLabelled=nonspam, p=0.0649569, pBar=0.1709825, m=-1.632244781272760, isLabelled=Good
1cambusnethan.boys-brigade.org.uk       wasLabelled=nonspam, p=0.1618177, pBar=0.498986, m=-2.083633094414358, isLabelled=Good
1castletown.boys-brigade.org.uk wasLabelled=nonspam, p=0.05968212, pBar=0.123572, m=-1.070512806657363, isLabelled=Good
1chandlersford.boys-brigade.org.uk      wasLabelled=undecided, p=0.3728774, pBar=1.272679, m=-2.413129702533633, isLabelled=Good
1charltonkings.boys-brigade.org.uk      wasLabelled=undecided, p=0.1033113, pBar=0.277835, m=-1.68930258562178, isLabelled=Good
1cheam.boys-brigade.org.uk      wasLabelled=nonspam, p=0.03412805, pBar=0.04384779, m=-0.284802078524545, isLabelled=Good
1cheslynhay.boys-brigade.org.uk wasLabelled=nonspam, p=0.03702001, pBar=0.04801406, m=-0.296975854654616, isLabelled=Good
1dumbarton.boys-brigade.org.uk  wasLabelled=nonspam, p=0.04351586, pBar=0.06246827, m=-0.435528743367505, isLabelled=Good
10bristol.boys-brigade.org.uk   wasLabelled=nonspam, p=0.361282, pBar=1.019553, m=-1.822039213645613, isLabelled=Bad
1elderslie.boys-brigade.org.uk  wasLabelled=nonspam, p=0.03463805, pBar=0.04435779, m=-0.2806087120076458, isLabelled=Good
1fareham.boys-brigade.org.uk    wasLabelled=nonspam, p=0.05204156, pBar=0.0962875, m=-0.850203860114284, isLabelled=Good
1felixstowe.boys-brigade.org.uk wasLabelled=nonspam, p=0.0349879, pBar=0.04472591, m=-0.278324547443803, isLabelled=Good
1flitwick.boys-brigade.org.uk   wasLabelled=nonspam, p=0.0871816, pBar=0.1694197, m=-0.943295196880812, isLabelled=Good
1gartcosh.boys-brigade.org.uk   wasLabelled=spam, p=0.2910773, pBar=0.9577676, m=-2.290422876556055, isLabelled=Good
```

# EFFECTIVENESS OF THE SOLUTION

## NUMBER OF ITERATION

*Initially we considered five number of iteration in order to calculate both the normal PageRank and core PageRank of each node with the smaller dataset. Hence, we observed that the value that was obtained for the core PageRank for each node was not precise so we increased the number of iteration for the calculation of core PageRank. But five iterations were not sufficient for the larger dataset with 1000 number of nodes, hence we increased the number of iteration for both the normal PageRank Calculation and core PageRank Calculation in order to get the appropriate and correct results. Hence with increase in number of iterations the effectiveness of the solution also increases and we obtain the precise value of both the normal PageRank and core PageRank for each node.*

## THRESHOLD VALUE FOR PAGERANK AND MASS ESTIMATE

*Initially we considered the average value for the PageRank of the nodes as the PageRank threshold. But with the several results obtained after several output the threshold was revised and changed again to obtain the correct and precise output. Similarly, for the threshold value of mass estimate initially we calculated the mass value for each node then took the average in order to set the value of mass threshold. Hence by observing the several outputs obtained we selected an appropriate value for the mass threshold. Both the values were updated according to the observation in order to obtain precise output, this led to the effectiveness of the solution.*

# PERFORMANCE ANALYSIS

*In order to evaluate the performance of the project we divided the actual dataset into several chucks. We considered the parameters such as the execution time, false positive and false negative in order to show performance of the implemented code. We calculated all these parameters value by changing the number of nodes. Initially we considered 100 number of nodes, then the number of nodes were increased to 200, 300, 400, 500, 600, 700, 800, 900 and at last 1000.*

*Number of Iterations for normal PageRank-10*

*Number of Iterations for core PageRank- 20*

*PageRank Threshold- 0.25*

*Mass Estimate Threshold- (-2.0)*

**TABLE 1: PERFORMANCE EVALUTION 1**

| ***S NO.*** | ***NUMBER OF NODES*** | ***EXECUTION TIME (min)*** | ***FALSE POSITVE*** | ***FALSE NEGATIVE*** |
|---|---|---|---|---|
| *1* | 100 | 11.30 | 10 | 6 |
| *2* | 200 | 13 | 13 | 8 |
| *3* | 300 | 14 | 14 | 14 |
| *4* | 400 | 16.20 | 16 | 20 |
| *5* | 500 | 16.40 | 19 | 30 |
| *6* | 600 | 17 | 21 | 35 |
| *7* | 700 | 19 | 21 | 41 |
| *8* | 800 | 21.30 | 21 | 48 |
| *9* | 900 | 21.50 | 25 | 60 |
| *10* | 1000 | 22.30 | 27 | 72 |

*Number of Iterations for normal PageRank- 10*

*Number of Iterations for core PageRank- 20*

*PageRank Threshold- 0.22*

*Mass Estimate Threshold- (-2.1)*

**TABLE 2: PERFORMANCE EVALUTION 2**

| *S NO.* | *NUMBER OF NODES* | *EXECUTION TIME (min)* | *FALSE POSITVE* | *FALSE NEGATIVE* |
|---|---|---|---|---|
| *1* | 100 | 11.20 | 9 | 8 |
| *2* | 200 | 12.50 | 10 | 9 |
| *3* | 300 | 14.10 | 13 | 12 |
| *4* | 400 | 15.30 | 14 | 18 |
| *5* | 500 | 16.20 | 17 | 27 |
| *6* | 600 | 17.10 | 20 | 37 |
| *7* | 700 | 18 | 21 | 45 |
| *8* | 800 | 19.20 | 22 | 58 |
| *9* | 900 | 21 | 22 | 66 |
| *10* | 1000 | 22.10 | 24 | 80 |

## GRAPHICAL REPRESENTATION OF PERFORMANCE

FIG 9: NUMBER OF NODES VS EXECUTION TIME FOR TABLE 1

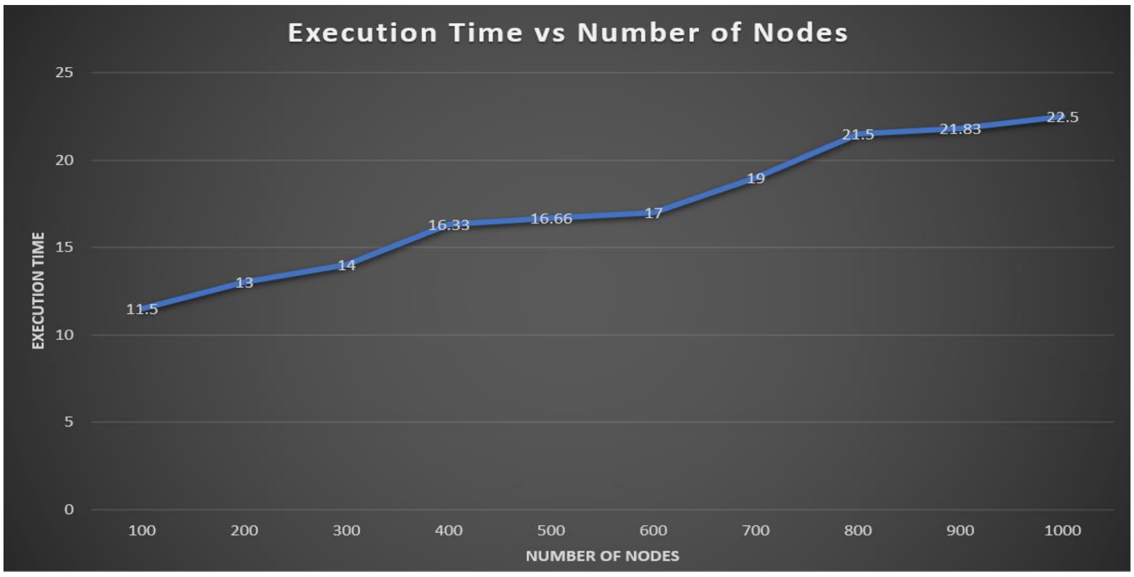


FIG 10: NUMBER OF NODES VS FALSE POSITIVE FOR TABLE 1

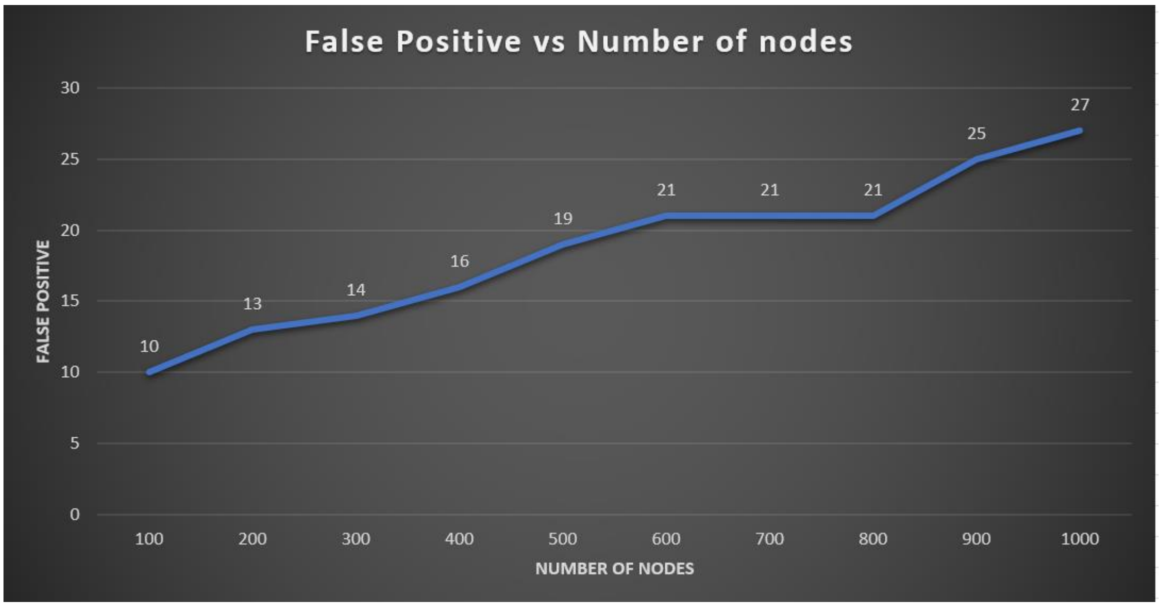


**FIG 11: NUMBER OF NODES VS FALSE NEGATIVE FOR TABLE 1**

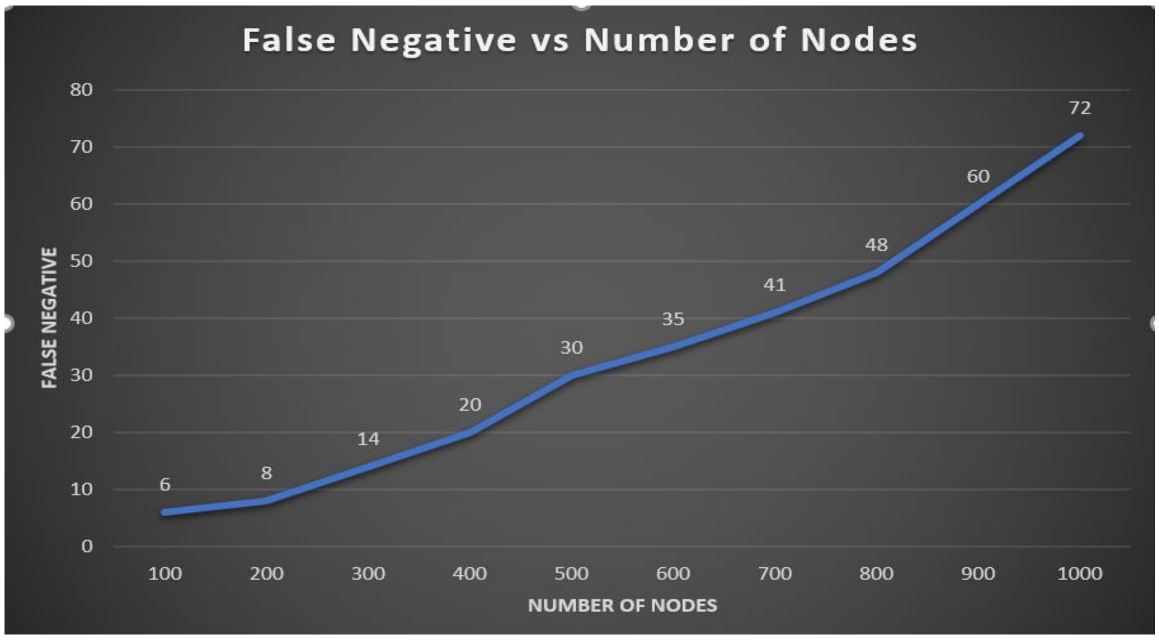


**FIG 12: NUMBER OF NODES VS EXECUTION TIME FOR TABLE 2**

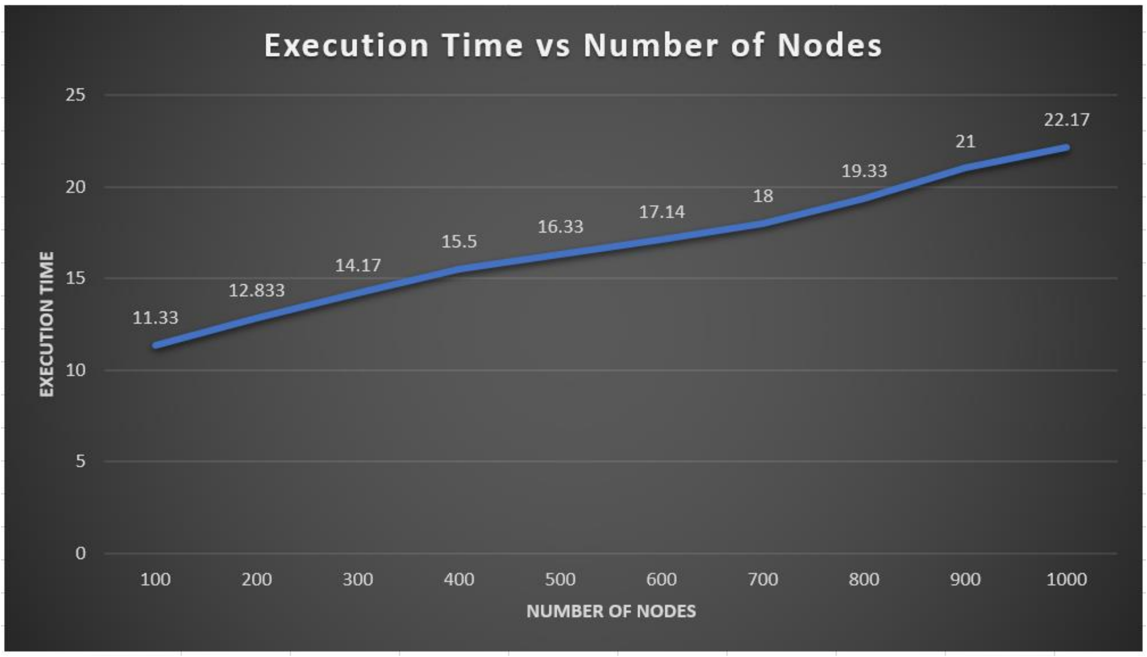


**FIG 13: NUMBER OF NODES VS FALSE POSITIVE FOR TABLE 2**

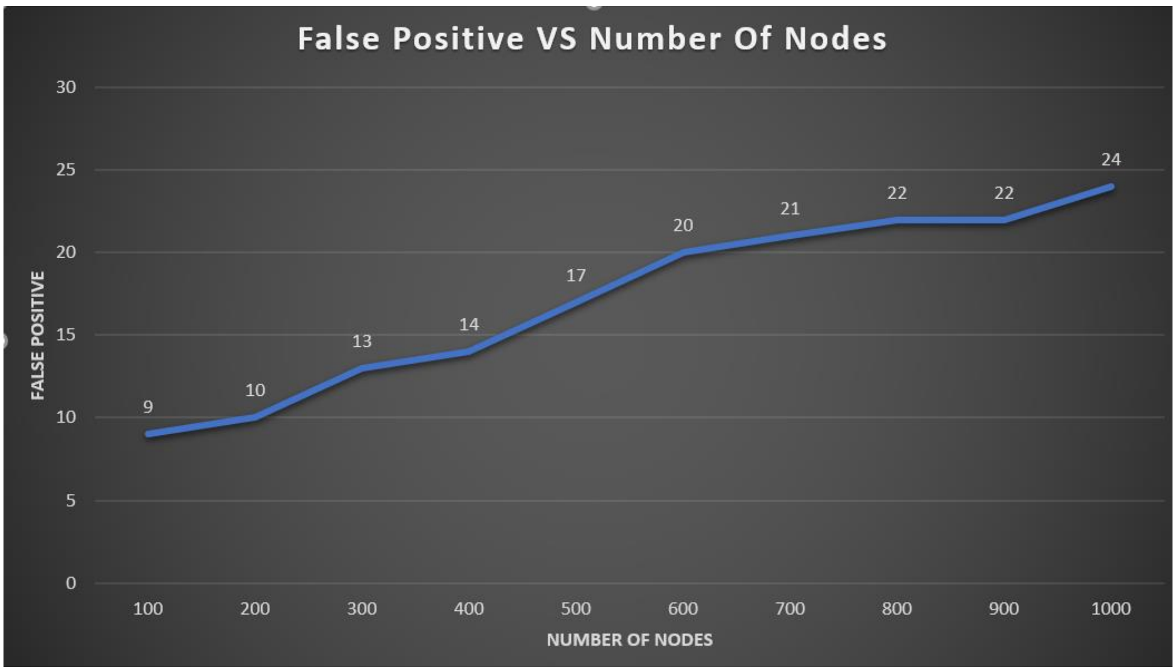
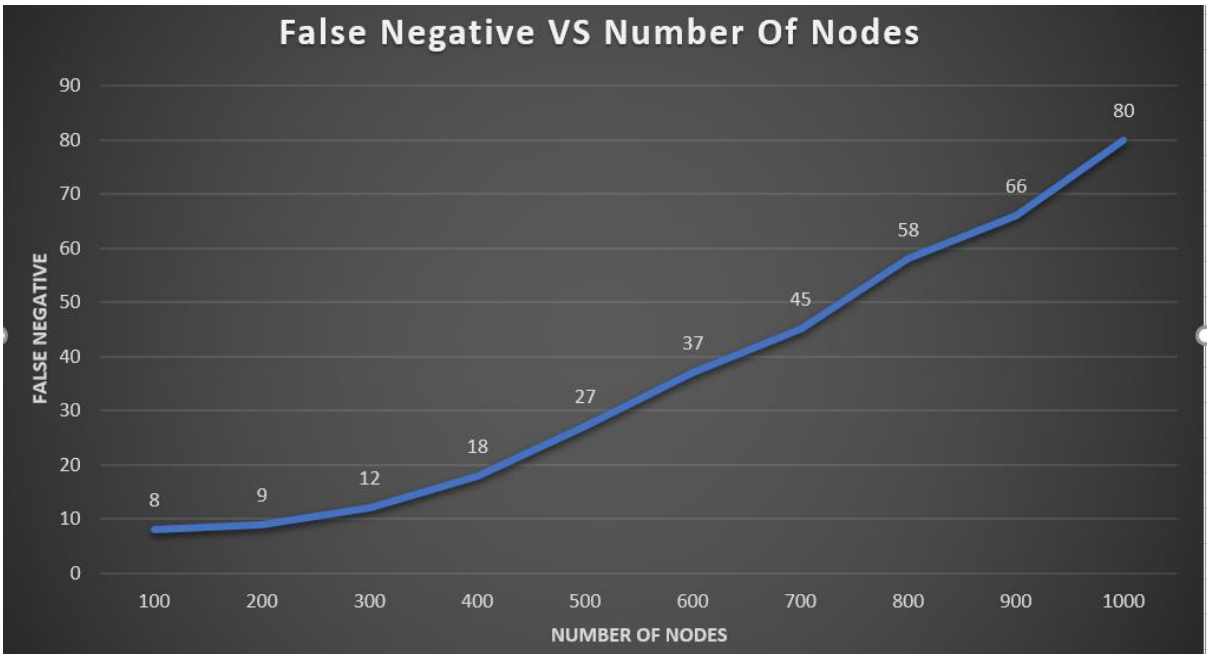


**FIG 14: NUMBER OF NODES VS FALSE NEGATIVE FOR TABLE 2**


False Negative VS Number Of Nodes
FALSE NEGATIVE
NUMBER OF NODES