Webology (ISSN: 1735-188X)
Volume 18, Number 6, 2021

Rumor Recognition On Summarized Graph

Sakshi Srivastava[1] , Anil Kumar Singh[2]

[1]Research Scholar at MNNIT Allahabad.

[2]Professor at MNNIT Allahabad.

Abstract — As the attractiveness of social networking sites continues to grow, rumors are also on rising respectively. Over the last few years, social networking and information-sharing microblogging websites such as Twitter and Sina Weibo have gained popularity. Unsolicited content, such as social spam, has also been exploited by spammers to overwhelm most users unfairly. In contrast to existing work, this paper uses a novel graph-based approach for spam detection. The problem of graph summarization has practical applications involving visualization and graph compression. As graph-structured databases become popular and prominent, summarizing and compressing graph-structured databases can become more and more valuable. Our experimental results demonstrate the usefulness and efficiency of our proposed strategy. The accuracy of the graph is considered before and after Graph Summarization using Multi Nominal NB and then compared with other machine learning algorithms. Various algorithms are considered, and it is found that Multi Nominal NB gives the lowest training time and the highest accuracy. The training time of Multi Nominal NB is found to be 0.55 sec before graph summarization. After graph summarization, the training time is optimized to be 0.02 seconds, and the accuracy value is 96.64%.

Keywords: Spam detection; Graph summarization; Conceptual summarization; Neo4j; Similarity

I. INTRODUCTION

When it comes to representing data and their interactions, graphs are employed in a wide range of applications. Chemical compounds, biological networks, online graphs, communication networks, social networks, and other types of data models are examples of data models represented by graphs. Science has taken an interest in graph theory and its applications [1], and in particular, research on graph matching is summarized here which describes variances across issues, general and specialized solution approaches, evaluation procedures, and future research possibilities [2]. When it comes to strategies that apply to broad graphs with semantic properties, the survey places a strong emphasis on those techniques.

Numerous applications today produce gigantic networks including millions of nodes and edges on a vast scale, and much study on the Tera-scale graphs in terms of their engineering and theory. [3]. To be more specific, graphs with enormous amounts of data and a quick rate of expansion are in front of us; this is not a good thing. According to the news site Yahoo, Facebook has 1.11 billion members in March 2013, when it had just approximately 1 million members at the end of 2004. The process of responding queries on these enormous graphs takes a long time. As a possible solution to this problem, graph summarization has been offered. In recent years, a number of graph summarizing algorithms have been presented, can shrink to a smaller one, a large graph by removing its aspects while retaining the graph's overall characteristics [4, [5], [6], and [7]. Following that, the smaller graph can be used for query response and other purposes. The inaccuracy in the graph is tolerable since it results in a shorter response time, which is essential in many applications.

Graphs have been more essential in various applications and areas, particularly in the management of large amounts of data, over the previous few decades. When it comes to graph databases, big data analysis is defined as the study of exponentially rising huge interconnected data that is related to time. The analysis of large amounts of related data in social networks, as well as the detection of synthetic identities, is difficult [11].

A native graph database that is open-source and NoSQL, Neo4j is a transactional backend for your applications [13],[14] that provides an ACID-compliant transactional backend for your applications. A Community Edition and an Enterprise Edition are available for Neo4j. It offers all of the features of the Community Edition as well as additional enterprise-level features like as backups, clustering, and failover functionality. As a native graph database, Neo4j is characterised by the fact that it executes the PGM (property graph model) as efficiently as possible, all the way down to the level of storage [1, 2]. As a result, the data is stored precisely as it appears on your whiteboard, and the database makes use of pointers to explore and traverse the graph[15]. For example, in contrast to graph processing or in-memory libraries, Neo4j includes complete database characteristics[16], such as ACID transaction conformance, cluster support, and runtime failover, making it ideal for theusage of graphs for data in production applications.

Developing appropriate processing and analytical strategies for dealing with the constant and rapid increase of highly interconnected datasets that are both large and complicated is necessitated by the development of appropriate processing and analytical procedures. For such datasets, graph summarization is a useful technique for compressing and simplifying them. Data reduction, speeding up query evaluation, and making it easier to see and analyze a graph are the three main goals of this project. Summary graphs are created by applying a sequence of application-specific methods to turn graphs into more compact representations while maintaining structural patterns, query answers, and specific property distributions. In light of the fact that this problem is similar to numerous areas of research into network topologies, a variety of approaches, including clustering, compression, sampling, and influence detection, have been presented, most of which are based on statistical and optimization methods. Although significant progress has been made in recent years, the topic of graph summarization continues to present open research challenges, particularly when dealing with more complex graph models, such as property-based models, when defining appropriate quality metrics, and when dealing with updates.

Users choose who to follow on a micro blogging Web site based on their own expertise and experience. However, despite the fact that spammers can imitate typical link patterns between their bogus accounts, they have little ability to influence the decisions of actual users. Such links are regarded as a reliable source of information for spammer detection by our team. In this article, we will discuss how to identify spammers based on the links they use. Spammers follow legitimate userssince it is both logical and important for them to gain the attention of legitimate users and distribute spam. On the other hand, there have been conflicting reports on whether spammers would establish connections with other spammers. Researchers Zhu et al. [12] discovered that spammers and legitimate users are segregated on Renren, a social networking site that is similar to Facebook. When

it comes to Twitter, however, Yang et al. [12] discovered the polar opposite: spammers tend to be networked, maybe in an attempt to mimic natural connection patterns. The two networks weretreated differently as a result of this. As a result, distinct spammer detection techniques weredeveloped for the two networks.

Spammers are difficult to detect, especially given the dynamic nature of social networks, where members' activity and contacts change frequently. Furthermore, the situation is exacerbated by the massive volume of data supplied by users. A large number of systems for automatic spam detection employing machine learning techniques depending on binary classification have been developed by the research community. The behavioral differences between legitimate users and spammers are used to inform the design of spam detection methods in general. The essential principle, as demonstrated by previous research, is that spammer activity appears abnormal when compared to typical user behavior.

In the present method, spammers are detected by Summaries with Super nodes, Super edges, and Corrections(SSSC). This method uses the Neo4j database which is a Property Graph Model (PGM). A property graph is a directed labeled multigraph with the special trait that each node or edge could maintain a collection (which could be empty) of property-value pairs. Nodes represent entities, edges represent relationships between entities, and properties represent specific characteristics of either the entity or the relationship from the perspective of data modeling.

Machine learning algorithms, MultiNominal NB is applied before and after summarization to find the prediction accuracy. Training and testing of the dataset is done on KNN, Random Forest, Logistic Regression, Decision Tree, Gradient Boosting and Linear SVC algorithms.

The paper is divided into five sections. The second section discusses the related works to graph summarization that are in the literature. The third section discusses the graph summarization method used in this research. The evaluation and experimental results are discussed in fourth section and the fifth section concludes the findings.

II. RELATED WORKS

Some of the most recently suggested summarization algorithms are briefly explained to provide an overview of the extent of the topic in question. When Navlakha and colleagues [5] proposed a summary technique in 2008, they said that graph compression is accomplished by grouping together into super-nodes, a collection of closely related nodes and constructing amongst all the pair of super nodes, a super-edge. Based on the MDL1 concept, it attempts to design a compression graph with thelowest possible representation cost.

This was accomplished through the development of two iterative algorithms, GREEDY & RANDOMIZED. When performing each stage, the GREEDY algorithm determines which pair of nodes is the best candidate for merging depending on the given cost decrease. Inevitably, the algorithm's execution time is exceedingly high. Authors have presented a RANDOM-IZED algorithm in order to minimise the running duration of the programme (Navlakha et al.). In contrast to the GREEDY algorithm, 2 merging nodes are chosen at random in this algorithm.

For the purpose of grouping nodes and building summary, Tian et al. [6] presented a summarization approach that included two procedures of summarization known as k-SNAP and SNAP2, both of which were introduced in 2008. In the case of attributed graphs, this summarization approach has been presented. According to Tian et al., relation compatible grouping and attribute compatible

grouping are two types of attribute compatible grouping. Additionally, they enhanced the SNAP procedure by providing k-SNAP, in which of the resulting summary k is the proper size, which is determined first by user.
Zhang et al. [7], in 2009, made two significant improvements to the k-SNAP method. In reality, the k-SNAP approach has two significant drawbacks. Because users must categorize the attribute values,and because there is no criterion for evaluating the quality of the final summary, there are two problems. In order to address these shortcomings, the CANAL approach was proposed by Zhang, which automatically categorizes attribute values, as well as a criterion for determining the overall quality of the summary, as well as a criterion for estimating the quality of the summary.
OLAP is an open-source framework developed by Chen et al. [11] that allows users to perform OLAP-like operations on graphs. In order to allow cubes to be formed from graphs depending on measurements and dimensions, a new framework known as the OLAP framework has been established. The OLAP framework's inherent property is that it automatically generates a summary based on the qualities that have been selected and the information that has been provided.
The data is modelled as a graph database using Neo4J for the purpose of analysing the Panama Papers, which revealed a fraud ring operating over offshore firms and was exposed [12].
Chen et al. [13] introduced another summarising approach in 2009 for mining frequent patterns, which was afterwards adopted by other researchers.
In order for this strategy to operate, randomised summary graphs must be generated. Because of random access time, Chen et al. confirmed that typical pattern mining methods are extremely time-consuming and inefficient when applied to huge graphs, as demonstrated by their findings. Instead of mining the full disc, they created a summarising method which first makes summaries and afterwards mines them, as opposed to mining the actual disc -graphs of residents - and then mining the summaries.
Using graph summarization, a method has been presented in [14] that assures that the reliability of the summary is maintained at an acceptable level. As far as the reconstruction error is concerned, this method gives a summary that is as accurate as it is possible. When the summary and original graphs are compared, the error is calculated by subtracting their adjacency matrices and multiplying them together to get the difference. There is a relationship between graphs, as demonstrated by the authors.
From the literature it can be seen that there is no graph summarization method for spam detection. The graph summarization method, SSSC is used for spam detection which is becoming increasingly prevalent in micro blogging sites.

III. GRAPH SUMMERIZATION USING SSSC

The database community has shown a great deal of interest in the topic of graph summarization.

i. Summerization

Large graph datasets are common in distinct fields, such as social networking and biological research. These types of domains require the usage of graph summarising techniques since they can aid in the discovery about the patterns of useful insights concealed within the data that is underlying. Significant type of graph summarising is the generation of compact and useful summaries depending on user-selected associations and node attributes, as well as the ability for users to interactively roll- up or drill-down to traverse among summaries having varying levels of detail.

The use of graph summarising techniques is essential for understanding the underlying properties of big graphs [3, 4]. However, the majority of the available graph summarising algorithms are statistical in nature (they investigate statistics likehop-plots, degree distributions, and clustering[27] (coeficients, among others). These statistical tools are extremely valuable, but it is difficult to maintain control over the resolutions of the summaries. A simple statistical approach is used to describe the characteristics of graphs in most existing summarization methods; for example, When investigating the property of graphs which is scale-free, researchers plot degree distributions; when investigating the small world effect, researchers use hop-plots; and when measuring the"clumpiness" of huge graphs, researchers employ clustering coeficients. It is possible to make use of the summaries provided by the given methods; but, they provide minimal data and can be tough to comprehend and manipulate. It is also possible to gain an understanding of the properties of huge graphs by using methods that mine graphs for recurring patterns.

The given algorithms, on the other hand, typically return a high number of results, which might be overwhelming to the user. Community structures (dense subgraphs) in big networks have been discovered through the use of graph partitioning algorithms, which has allowed researchers to detect community patterns in vast networks. In contrast, the community recognition algorithm is basedsolely on the connectivities of nodes, with no consideration given to their qualities. Visualizing huge graphs can be difficult due to the fact that they are difficult to visualise using graph sketching techniques. It is necessary to provide consumers with a more regulated and intuitive technique of summarising graphics. User-friendly properties and relationships should be available for selection in the summary technique [28, 2], which should subsequently utilize the given aspects to provide concise as well as useful summaries. Aside from that, users must be able to choose how thegenerated summaries are shown and "drill-down" or "roll-up" the information, in a manner similar to how OLAP-style aggregation techniques are used in usual database systems to aggregate information. It is possible to construct considerably more compact and informative graphs by using Graph Summarization. These graphs will summarise the high-level structure properties of the original graph as well as the prominent interactions between groups of nodes. For each node in the summary graph, there are links between related subsets of nodes in the original network, and for every edge in the summary graph, there are connections between related subsets of nodes in the original network.

iii. Graph Summarization

Summaries with Super nodes, Super edges, and Corrections (SSSC) is used for graph summarization. In the summarising approach, which is defined as the compression of data into a meaningful representation, the notion of summary graph serves as the conceptual underpinning. Due to the fact that it is suitable to an interactive querying strategy, this method is quite unique in that it allows users to modify the summary depending on node qualities and associations that they choose. Users can also adjust the resolutions of the resultant summaries [4], which can be used together along with an intuitive "drill-down" or "roll-up" paradigm to travel between summaries of varying resolutions [5– 7]. This final element of roll-up or drill-down capabilities is influenced by the OLAP-style aggregation processes that are commonly employed in regular database systems to gather information.

a. Graph and summary representation and definitions

A graph, G = (V,E), is a set of vertices, V, and a set of edges, E. Edges are represented as (u,v), where u,v∈ V. If we were to represent Property Graph Model, data as a graph, subjects and objects would be nodes, and there would be a directed edge from subject to object for each triple. The edge would be annotated with the predicate value. Notably, vertices could also have multiple edges between each other. For our purposes, however, we only consider undirected, unannotated graphs, and so we simplify the PGM data to fit this kind of graph.

A graph summary of graph G = (V,E), is denoted by GS = (S, C). The summary graph, S = (VS , ES,) is a collection of supernodes, VS, which contain original nodes in V, a collection of superedges,ES, and a collection of corrections, C. A superedge in a summary connects two supernodes. Asuperedge represents that an edge between every original node contained by the two connected supernodes is also connected in the original graph, G. However, a summary may have superedges that represent some edges between original nodes that do not actually exist in the original data. If thisis the case, we must record that such edges do not actually exist in the original data by adding appropriate corrections to C. For instance, suppose s,t∈ V, S,T ∈ VS, s ∈ S, t ∈ T, (S,T) ∈ ES, and (s,t) ¬∈ E. Thus, according to the summary, s and t are connected originally, but this is actually not true. We would fix this by adding the correction (-, s, t) to C. Corrections are negative when representing the subtraction of an edge represented by the summary which is not actually in the original data. Corrections may also be positive when an edge is not represented by the summary that exists in the original data.

In Figure 1, there is an example of a SSSC representation of a small graph. The original nodes b and c are merged into super node w. Notice how w has a self-loop because b and c are connected in the original graph. Similarly, original nodes h and g are merged into super node y, and original nodes d, e, and f are merged into super node z. Original node a is simply assigned super node x, which only contains a. Notice how super edges can represent the majority of the original edges, but the edge (a, e) must be added as a correction because no super edge represents it. Also, the edge (g, d) is represented by the super edge (y, z), but (g, d) is not present in the original graph, so a negative correction must be added to the correction set.

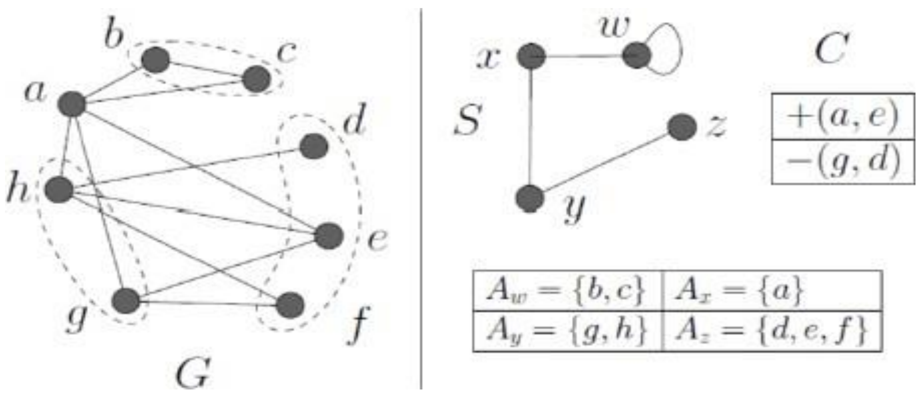


Figure 1 [1] A summary produced by merging nodes with the same neighbors, allowing for corrections.

b. Summary cost, compression ratio, and MDL

The cost of a summary is defined in [1], and uses MDL (Minimum Description Length) principle for its justification. If the theory and data are encoded using MDL, the optimum theory to deduce from a given dataset is one that minimises both their sizes [1]. In our graph summarization, we can conceptualize summary graph S to be our theory, and the corrections C to be our data encoded in terms of the theory. Therefore, [1] declares the best summary S to be the one which minimizes the storage cost of S and C. This cost includes the cost of storing super edges, corrections, and mappings from original nodes to super nodes. Note that the mappings from original nodes from V to the super nodes in V_S, should cost roughly the same regardless of the summary, so we can ignore the costof the mappings in the summary cost we try to minimize. Assuming the cost of storing a super edge and the cost of a correction is roughly the same, we are left with a relative cost of a summary to be the sum of the number of super edges and the number of corrections.

$\mathrm{Cost}(G_S) = |E_S| + |C|$

[1] also states that we can ignore the cost of the mappings because they will generally be small compared to the storage costs E_S and C. However, for our use of summarizing data, theoriginal vertices are typically URIs stored as strings, which do not have negligible storage costs, especially when super nodes can be represented integers, which are much cheaper than long URI strings.Therefore, in this paper, we make a distinction between the theoretical cost presented in [1], or the storage cost without the cost of mappings, and the implemented cost, or the storage cost in our implementation which includes mappings.

Finally, notice that the original graph can be represented as a summary where every supernode just contains one original node, and there are no corrections. Then, the theoretical cost of a graph G is just the number of edges |E |. Therefore, we can calculate a theoretical compression ratio of a summary to be Compression Ratio(G_S,G) = Cost(G_S) / |E |

Since the goal is to minimize Cost(G_S), it is important to note that, given a VS, there is exactly one E_S and exactly one C which minimizes Cost(G_S). This is because, if we look at pairs of super nodes one at a time, we can easily determine whether or not a super edge should exist between those two nodes, using the cost of a super edge as our decision criteria. If $u,v \in V_S$, let Π_{uv} be the set of all pairs (a,b) where $a \in u$, and $b \in v$ (Note that $|\Pi_{uv}| = |u| * |v|$), and let $A_{uv} \subseteq \Pi_{uv}$ be the set of edges which actually exist in the original graph. Then, the cost of having a super edge between u and v would be $|\Pi_{uv}| - |A_{uv}| + 1$ because storing the super edge adds a cost of 1, and we would need a negative correction for all $|\Pi_{uv}| - |A_{uv}|$ edges that do not exist in the original graph. On the other hand, the cost of not having a super edge between u and v would be $|A_{uv}|$ because we would need a positive correction for each edge in the original graph. Finally, we can decide whether or not (u,v) shouldexist in ES based on which option has the lower cost.

c. Edge representation Cost, Supernode Cost, and Reduced Cost

As discussed previously, the cost of representing the original edges between two super nodes, u and v, can be calculated deterministically based on $|\Pi_{uv}|$, the number of possible edges represented, and $|A_{uv}|$, the number of edges between nodes in u and v which are present in the original graph. Since we will be choosing whether or not to include a super edge between u and v based on which option has a lower cost, we can denote the cost of the representing the original edges between u and v as $\mathrm{Cost}(u,v) = c_{uv} = \min(|\Pi_{uv}| - |A_{uv}| + 1, |A_{uv}|)$

Furthermore, we can denote the cost of a super node, u, to be the sum of the cost of all its related edges

$$\text{Cost}(u) = c_u = \sum c_{ui} i \in VS$$

Note that the calculation of this cost in implementation need not actually iterate over all super nodes if we already know the set of super nodes for which there is at least one original connection to u; all nodes not in this set would simply add zero to the cost of u.

Finally, with the cost of a node defined, we can calculate the reduced cost of merging nodes. If we wanted to merge two super nodes, u and v, into one super node, w, we can easily calculate how much this merge would contribute to the lowering the overall summary cost: it would be the cost of the original super nodes minus the cost of the new super node, $cu + cv - cw$. As long as this sum is positive, merging u and v would reduce the overall summary cost. In our definition of reduced cost, we also normalize this value. The purpose behind this is to try to avoid suboptimal local minima in summary costs, particularly if a greedy approach is being used. For instance, if we are trying to find the "best" merge for super node u, we may greedily pick the node which has the highest absolute reduced cost. However, this approach favors merges which may be inefficient despite a high absolute cost, when a more efficient (higher normalized reduced cost) merge may be present.

Therefore, it is important to use the normalized reduced cost, which we refer to as reduced cost from now on, when making decisions greedily. So, we define a reduced cost value of

$s(u,v) = cu + cv - cwcu + cv$

Note that this metric has a maximum value of 0.5, which would be the case if u and v had identical sets of neighbors and no corrections. This metric does not have a general lower bound, but any pair of nodes which have a non-positive reduced cost will not benefit the overall summary cost by being merged.

The algorithm for the graph summarization process is given. The algorithm represents the process that is taking place in graph summarization.

Algorithm:

Input: - T[i]=Node Details Tabular Data and Its G[i]=Neo4j Graph

Output: - ST[i]=Summery Graph Labeled Node Links in Tabular Format

SSSC (T[i],G[i])

1 Appling Node_ID based on Data Subgraph with its (V(vertex),E(edges)) and Start Random WalkQueue based on Node Links

2 Traversing Nodes based on S,T ∈V, S,T ∈VS, s ∈S, t ∈T, (S,T) ∈ES, and (S,T) ¬∈E

3 Determine Super edges based on Cost of Traversing b/w nodes through `Cost(GS) = |ES| + |S| +
|C(correlation)|` and Minimal cost `A[Gs]=Compression Ratio (GS, Gv) = Cost(GS) / |E | ` with thehelp of FFCM()

4 Determine Super nodes „u‟ based on Cost representing the original edges between two nodes, „Cost(u,v) = cuv= min(|Πuv | - |Auv| + 1, |Auv|)‟ and selecting the super node
„$(u,v)=(cu+cv-cw)/cu+cv$‟, and Correlation |C| through RWFB().

5 Summarize(G)

a. GS ← initialize_summary(G)

```
b. unvisited ← initialize_unvisited(GS )
c. while unvisited is not empty
d. seed ← pick_seed_node(unvisited)
e. merge_candidates ← get_merge_candidates(seed)
f. to_merge_with_seed ← get_best_merge(merge_candidates, seed )
g. new_supernode ←merge_supernodes(seed, to_merge_with_seed )
h. update_unvisited(unvisited, seed, to_merge_with_seed, new_supernode )
i. put_edges_in_summary (G, GS )
j. return GS
6 Based on Summaries Graph Label the Super edge/Super node and Links on Tabular Data with
  thehelp of node_id through SB().
7 RETURN Summery Graph Labeled Node Links in Tabular Format
```

IV. METHODOLOGY

The steps used in the proposed research can be represented as a flowchart and algorithm. The flowchart is represented in Fig. 2.

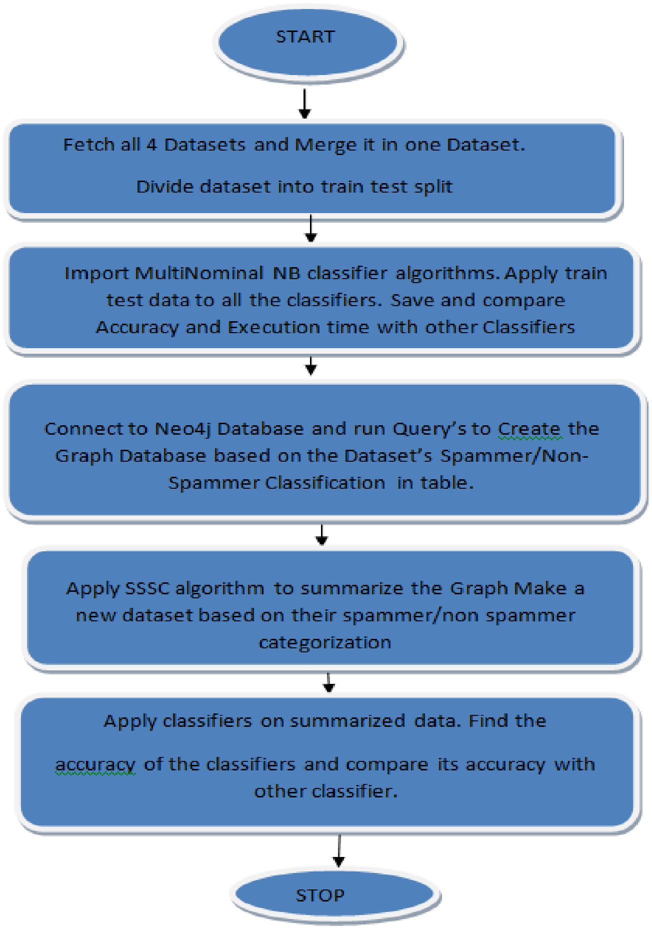

START
Fetch all 4 Datasets and Merge it in one Dataset.
Divide dataset into train test split
Import MultiNominal NB classifier algorithms. Apply train test data to all the classifiers. Save and compare Accuracy and Execution time with other Classifiers
Connect to Neo4j Database and run Query's to Create the Graph Database based on the Dataset's Spammer/Non-Spammer Classification in table.
Apply SSSC algorithm to summarize the Graph Make a new dataset based on their spammer/non spammer categorization
Apply classifiers on summarized data. Find the accuracy of the classifiers and compare its accuracy with other classifier.
STOP

Fig. 2(a) Flowchart representing the steps involved in the proposed research

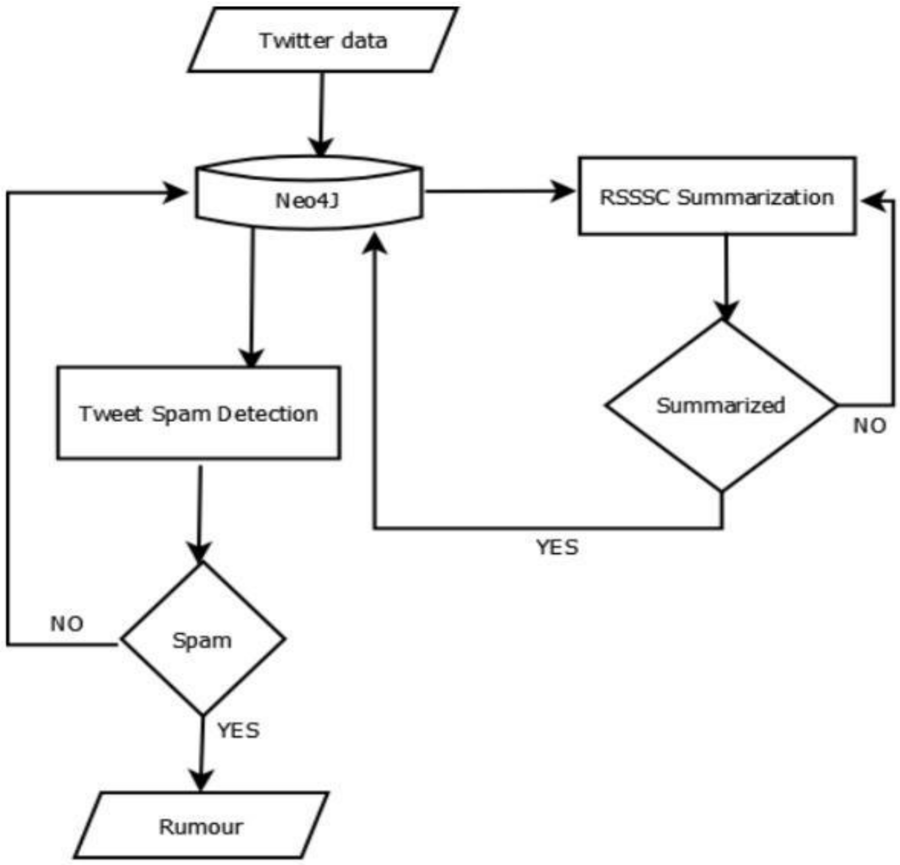


Figure 2(b): Flowchart for the representation of the process in the proposed research

Step 1: The datasets are combined and merged as one dataset. Divide dataset into train test split. Import KNN, Logistic Regression, Decision Tree, Random Forrest, Gradient Boosting, Linear SVC classifier algorithms. Apply train test data to all the classifiers. Save Accuracy and Execution time of Each Classifier
Step 2: Connect Neo4j Database. Run Quarry"s to create the Graph Database based on the Data set"s Spammer/Non-Spammer Classification in table. Create all Graphs in Neo4j and also save it in Tabular form
Step 3: Apply SSSC algorithm to created Graph Database. Label all super node, super edges and corelations on the graph"s respective table as spammer/nons pammer. Categories Spammer/ Non-Spammer Data based on the Labeled Noes/Edges. Make a new dataset based on their spammer/non spammer categorization
Step 4: Divide Summarize Dataset into train test split. Import KNN, Logistic Regression, Decision Tree, Random Forrest, Gradient Boosting, Linear SVC classifier algorithm. Apply train test data(after Summarization) to all the classifiers. Save and compare Accuracy and Execution time of Each Classifier. Use best classifier from comparison and find Mean Square error and mean test score. Use Multinominal NB Classifier to classify the mean test score and find the accuracy through MNB.

V. EVALUATION AND EXPERIMENTAL RESULTS

i. Data set used:

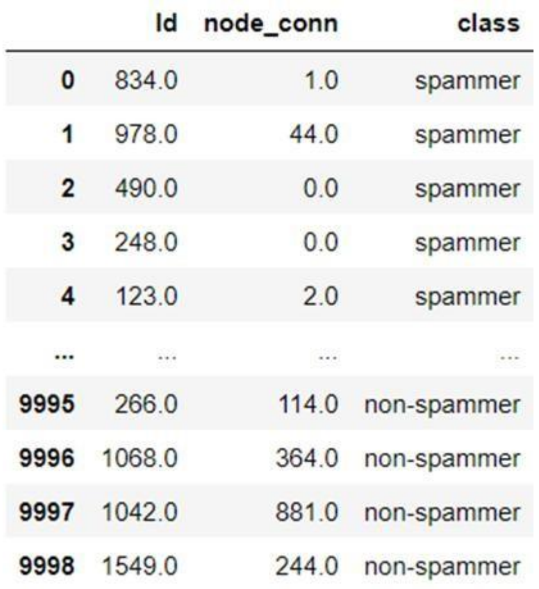
The data set used is from twitter. The data in the twitter dataset consists of both spam data and non-spam data. Both spam and non-spam data are considered. 220000 entries were considered in which the attributes are given in Table. 1.This dataset from twitter is used and Neo4j graph database is applied to the dataset. Graph summarization is applied by using super nodes, super edges and corrections. Machine learning algorithms are used before and after summarization is done.

Table 1: All Attributes of Datasets

account_age	0	no_follower	0
no_following	0	no_user favourites	0
no_lists	0	no_tweets	0
no_retweets	0	no_hashtag	0
no_user mention	0	no_urls	0
no_char	0	no_digits	0
class	0		

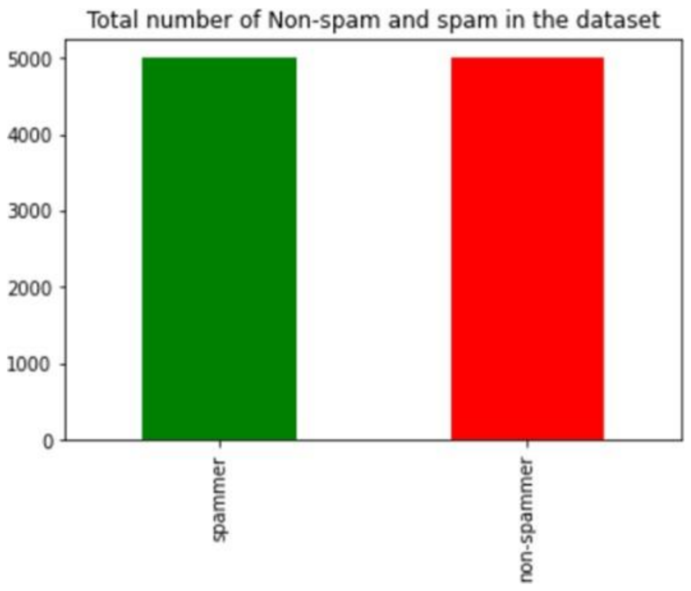
ii. Identification of spam:

Identify and prevent spam as it occurs during the execution process: It is necessary to do real-time link analysis on an interconnected dataset and find out the amount of spam data that are presentthere. The Node ID compilation for spam detection in the summaries data is given in Table. 2 and theratio of spam and non-spam data is given in Figure. 3. The spam detection has an accuracy score of 94.427%.



	Id	node_conn	class
0	834.0	1.0	spammer
1	978.0	44.0	spammer
2	490.0	0.0	spammer
3	248.0	0.0	spammer
4	123.0	2.0	spammer
...	...	...	...
9995	266.0	114.0	non-spammer
9996	1068.0	364.0	non-spammer
9997	1042.0	881.0	non-spammer
9998	1549.0	244.0	non-spammer

Table 2: Node ID Compilation for Spam Detection in the Summaries data

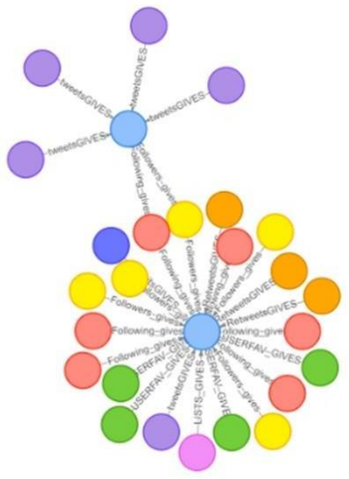


(a)

Figure 3: Spam/Non Spam Ratio

iii. Neo4j Graph for all queries

The graph database that was considered for this research is the Neo4j database. Neo4j is the world's most popular open source Graph Database, and it was created utilising Java technology to achieve this position. It has a great degree of scalability and is schema-free (NoSQL).In its role as native graph database, Neo4j distinguishes itself by executing the property graph model in an efficient manner way to the bottom to the storage level [1]. Consequently, the data is saved exactly as it appears on the whiteboard, as well as the database makes extensive use of pointers to explore and navigate the graph. [15].



(b)

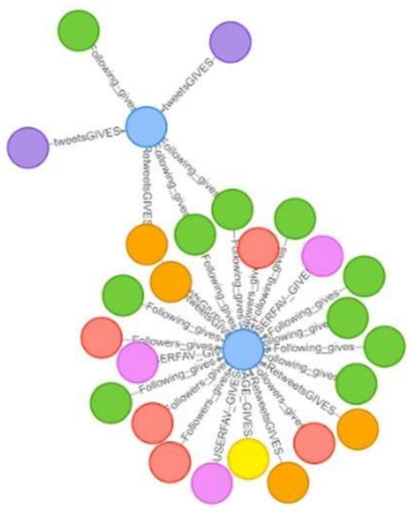
Figure 4: (a) Non-spammer representation of data attributes and (b) spammer representation for one of the data attributes of class relations (Account age)

iv. Summaries with Super nodes, Super edges and Categories (SSSC)

SSSC algorithms were initially studied by the authors of [19]. They proposed a greedy and randomised method for SSSC. According to the Minimum Description Length principle, which statesthat the best theory to infer from a set of data is the theory that has the least amount of data when encoded using the theory of the SSSC summary model which will discussed, and then supported this proposal with evidence from previous research. We can reduce the model's cost by defining the theory's size as a graph summary and the data it contains as a collection of adjustments; this allowsus to discover the "best" model's summary or compression. The detailed representation of the SSSC model is given in Figure. 3.
The data is first fetched after feature selection. It is first tested if the node is available and then graph summarization using SSSC takes place.

v. Machine learning algorithm for finding the accuracy of graph summarization

It is the Multinomial Nave Bayes Classifier (MNB) that has been used in this implementation, which is a modified form of the Nave Bayes Classifier. MNB is also a probabilistic strategy that is comparable to Nave Bayes. MNB is a text document analysis tool that is specifically developed to compute the frequency of occurrence of each word.
In general, the Naive Bayes (NB) algorithm works on the conditional probability (it takes into account the conditional independence of the features), whereas the Multinomial Naive Bayes algorithm works on the multinomial distribution (it takes into account the conditional independence of the features). The multinomial Nave Bayes classifier takes into account the fact that each phrase appears several times. The MNB method is combined with the K-Means algorithm to obtain better accuracy.
The K-Means algorithm, which is based on dividing [4] [5], is a type of cluster algorithm developed by J.B.Mac Queen. Unsupervised algorithms, such as this one, are commonly employed in data mining and pattern recognition applications. The square-error and error criterion are the foundations of this technique, which is aimed at reducing the cluster performance index to the bare minimum. For



the purpose of achieving the best possible result, this algorithm attempts to discover K divisions that satisfy a certain requirement. To begin with, select some dots to represent the initial cluster focal points (typically, we choose the first K sample dots of income to represent the initial cluster focal point); second, gather the remaining sample dots to their focal points in accordance with the criterion of minimum distance; third, obtain the initial classification; and if the classification is unreasonable, modify it (calculate each cluster focal point again); and finally, iterate repeatedly until we obtain a satisfactory classification. The K-Means algorithm, which is based on division, is a type of cluster algorithm that has the advantages of being quick, efficient, and timely. This method, on the other hand, is very dependent on the initial dots and the differences in initial sample selection, which always results in varied outputs. Furthermore, the gradient approach is always used to obtain the extremum in this algorithm based on the goal function. When using the gradient method, thedirection of search is always along the direction in which energy decreases. This results in the fact that if the initial cluster focal point is not appropriate, the entire algorithm will easily sink into the local minimum point, resulting in the fact that the initial cluster focal point is not appropriate.
When measuring the performance of a classification model, a confusion matrix, NxN is employed, where N is the number of target classes. With this matrix, you may compare real target values with the values predicted by the machine learning model. Fig. 5 depicts a confusion matrix for the numberN = 2, with the following entries having the following meanings:
• a is the number of correct negative predictions;
• b is the number of incorrect positive predictions;
• c is the number of incorrect negative predictions; and
• d is the number of correct positive predictions. This matrix can be used to calculate the classification error and the prediction accuracy, which are as follows:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d}$$

$$\text{Error} = \frac{b+c}{a+b+c+d}$$

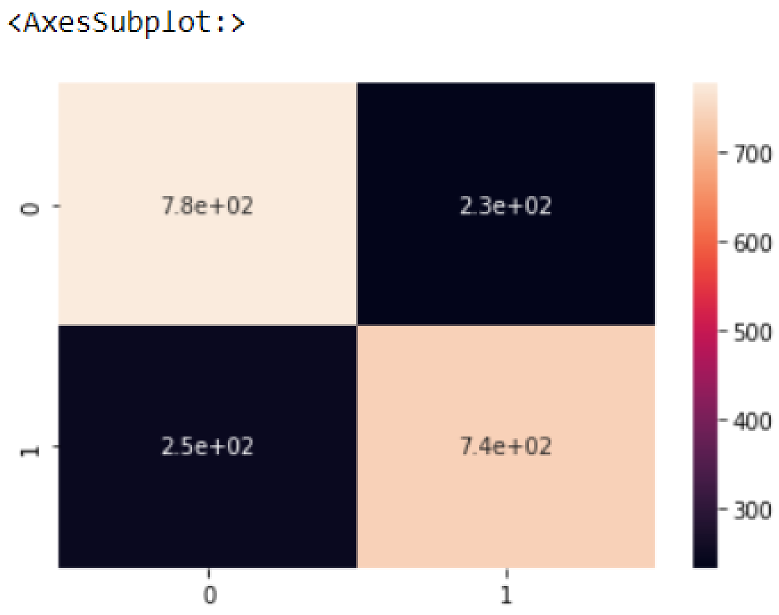


Fig. 5: Confusion Matrix of Multi Nominal NB Classifier to represents its accuracy

vi. Comparison of graphs using machine learning

The different machine learning algorithms such as Logistic Regression, K-NN Classifier, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier and Linear SVC are used for training and testing the graphs. The accuracy of each algorithms is also calculated. The time taken for training the data, testing the data and the accuracy score is given in Table. 3. The time taken is represented as a histogram in Figure 4 and the accuracy is represented in Figure 5.

Accuracy is measured by the number of correctly classified examples. When you run tests, you can use a classifier to generate hypothesized class labels for each example. Every test example has a guess that is either correct or incorrect. The accuracy of your classifier may be calculated by simply counting the number of correct judgments made by your classifier and dividing the total number of test cases by the number of correct decisions.

The accuracy of the classifiers are calculated by using the formula:

$$\textbf{Accuracy} = \frac{1st\ index\ + 4th\ index}{(sum\ of\ all\ index\ )round\ \ of\ \ 2\ digits}$$

The various classifiers are discussed below:

Logistic Regression: Logistic regression is a widely used method for predicting a categorical answer in a variety of situations. Generalized Linear models (GLMs) are a subset of Generalized Linear models which predicts the likelihood of different outcomes.

K-NN Classifier: This technique is an instance-based learning method that classifies items based on the k training examples that are the closest to each other in the resource space, as shown in the figure below.

Decision Tree Classifier: Regression and classification techniques are used in decision tree classifier.

When solving a problem, a decision tree is used whereby every leaf node correlates to a class label while attributes is expressed on the internal node of the tree, the problem is solved using a tree representation.

Random Forest Classifier: The RF method is based on the building of many decision trees, which are then combined to produce a decision that is more accurate and consistent.

Gradient Boosting Classifier: Gradient-boosted trees (GBTs) are a common regressionand classification method that employs ensembles of decision trees to do classification and regression.

Linear SVC: Using a support vector machine, a hyperplane or series of hyperplanes is constructed in a high-dimensional space (or an infinite-dimensional space), which can then be utilised for regression, classification, and similar tasks.

Table 3: Comparison of various algorithms

Algorithm	Time taken for training data	Time taken for testing data	Accuracy Score
Logistic Regression	6.13	0.01	91.036
K-NN Classifier	2.99 m	18.82 s	91.036
Decision Tree Classifier	5.2 s	0.02 s	97.471
Random Forest Classifier	82.44 s	3.74 s	99.012
Gradient Boosting Classifier	90.71 s	0.33 s	98.068
Linear SVC	76.59 s	0.01 s	52.195
Multinominal NB	0.55 s	1.5 s	96.64

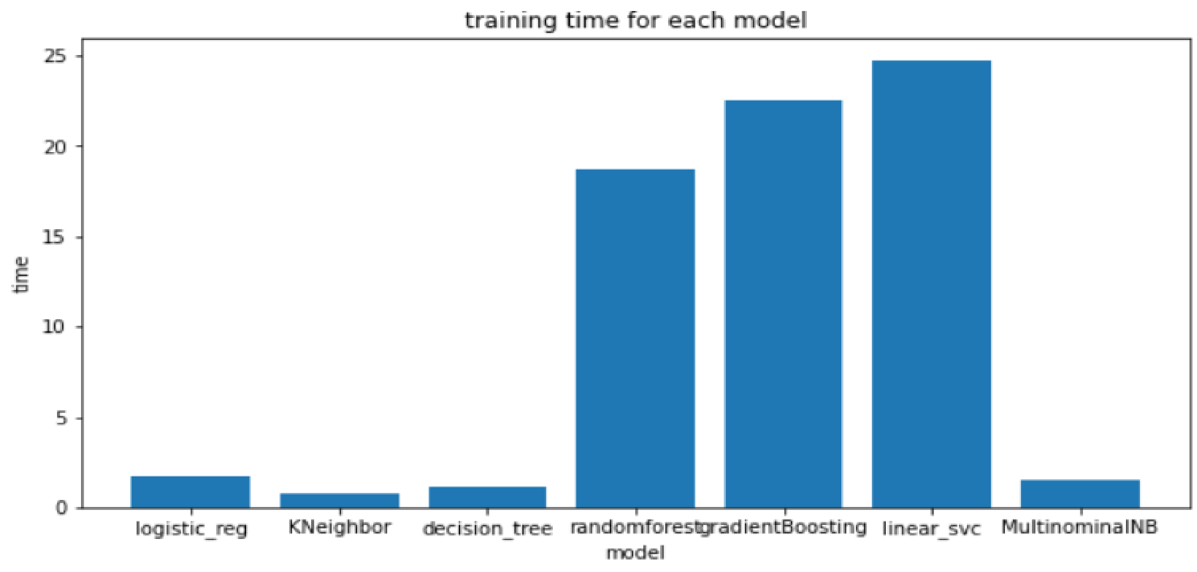


Figure 6: Graph Comparison (based on time) of all above classifier algorithm

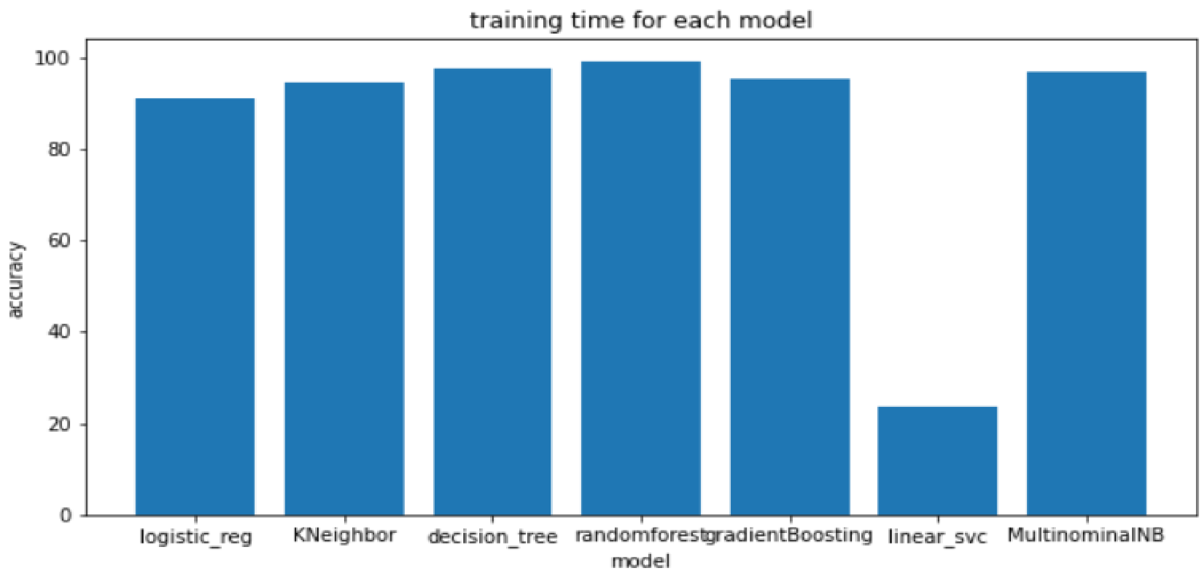
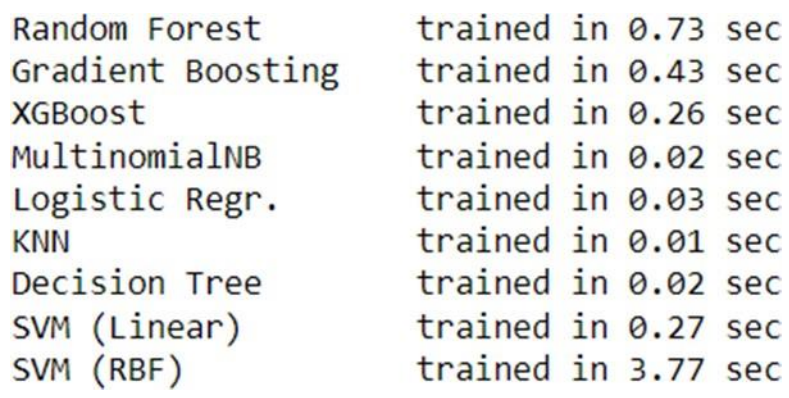


Figure 7: Graph Comparison (based on accuracy) of all above classifier algorithm

vii. Summarization

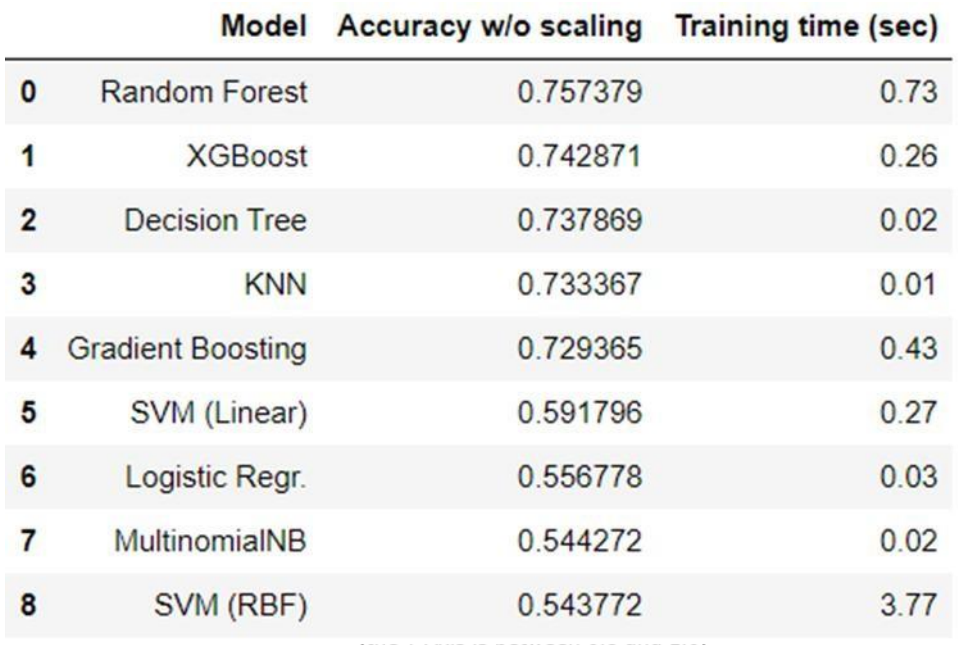
Summarization technique is applied to the graph. Graph summarization is done and the training score through various classifiers is calculated after Summerization. This is represented in Table. 4. The accuracy score of the graphs after summarization is given in Table.5. The training score is represented in Figure 6 and the accuracy score is given in Figure 7 of the various classifier after summarization. It can be seen that SVM (Linear) classifier gives the best accuracy of 59%.

Table. 4: Training Score Through Various Classifier after Summerization



```
Random Forest        trained in 0.73 sec
Gradient Boosting    trained in 0.43 sec
XGBoost              trained in 0.26 sec
MultinomialNB        trained in 0.02 sec
Logistic Regr.       trained in 0.03 sec
KNN                  trained in 0.01 sec
Decision Tree        trained in 0.02 sec
SVM (Linear)         trained in 0.27 sec
SVM (RBF)            trained in 3.77 sec
```

Table. 5: Accuracy score after summarization



	Model	Accuracy w/o scaling	Training time (sec)
0	Random Forest	0.757379	0.73
1	XGBoost	0.742871	0.26
2	Decision Tree	0.737869	0.02
3	KNN	0.733367	0.01
4	Gradient Boosting	0.729365	0.43
5	SVM (Linear)	0.591796	0.27
6	Logistic Regr.	0.556778	0.03
7	MultinomialNB	0.544272	0.02
8	SVM (RBF)	0.543772	3.77

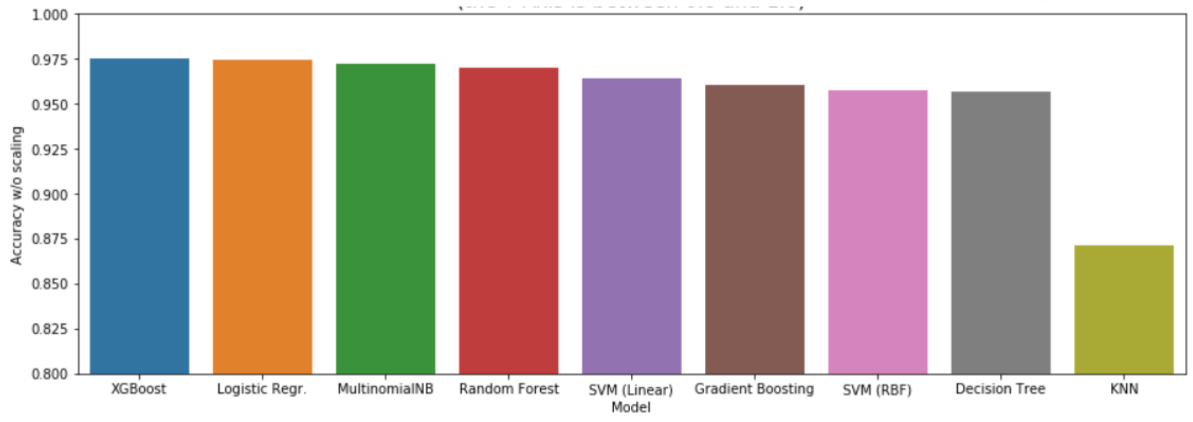


Figure 7: Accuracy Comp. Graph of all classifier after summarization

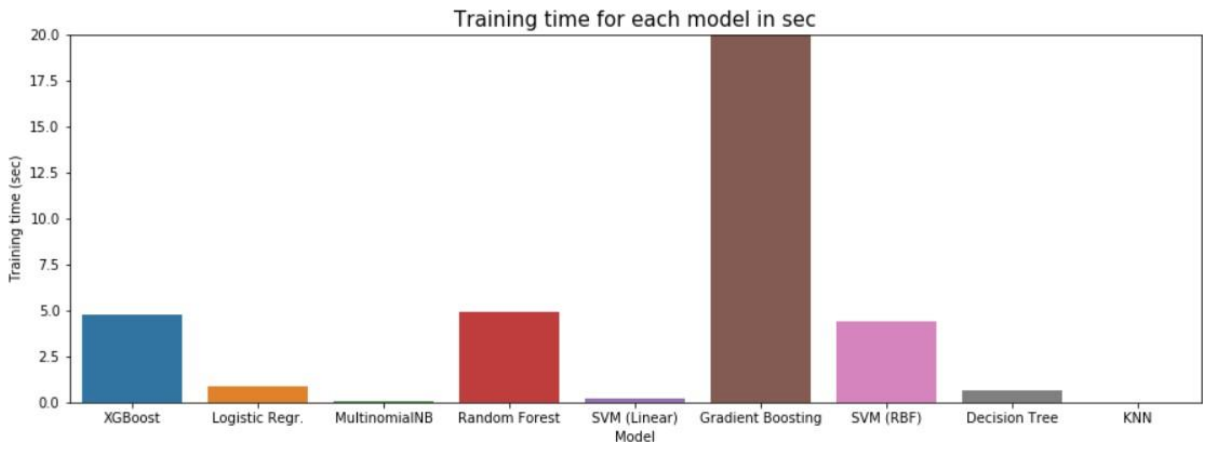


Figure 8: Time Count Comp. Graph of all classifier after summarization

V. Conclusion

In this era of digital communication the popularity of social networking sites increasing in every second and with this spam accounts are also on the rise. In contrast to existing work, in this paper wepresent a novel graph-based approach for spam detection. As the size of data is large, due to which analysis of spam became complex and to optimize this issue summarization of the graph has been performed using SSSC algorithm. The problem of graph summarization has practical applications involving visualization and graph compression. As graph structured databases become popular and large, summarizing and compressing graph structured databases can become more and more useful. The usefulness and efficiency of our proposed strategy were demonstrated by our experimental results. The accuracy of the graph is considered before and after Graph Summerization using Multi Nominal NB. The accuracy of the graph is compared with other machine learning algorithms. Various algorithms are considered and it is found that Multi Nominal NB gives the lowest training time and the highest accuracy.

REFERENCES

[1] S. Navlakha et al. 2008. Graph Summarization with Bounded Error. ACM SIGMOD Conference (2008).
[2] Aggarwal C C, Wang H (2010) Graph data management and mining: A survey of algorithms and applications. In: Aggarwal C (ed) Managing and Mining Graph Data. Springer US, pp 13-68.
[3] U. Kang, "Mining Tera-Scale Graphs: Theory, Engineering and Discoveries," PhD Thesis, no. May, 2012.
[4] K. LeFevre and E. Terzi, "Gra SS: Graph Structure Summarization," Proc. 2010 SIAM Int. Conf. Data Min., pp. 454–465, 2010.
[5] S. Navlakha, R. Rastogi, and N. Shrivastava, "Graph summarization with bounded error," Proc. 2008 ACM SIGMOD Int. Conf. Manag. data - SIGMOD "08, p. 419, 2008.
[6] Y. Tian, R. A. Hankins and J. M. Patel, "Efficient Aggregation for Graph Summarization Categories and Subject Descriptors," pp. 567–579.
[7] N. Zhang, Y. Tian, and J. M. Patel, "Discovery-driven graph summarization," 2010 IEEE 26th Int. Conf. Data Eng. (ICDE 2010), pp. 880–891, 2010.

[8] Y. Bei, Z. Lin, and D. Chen, "Summarizing scale-free networks based on virtual and real links," Phys. A Stat. Mech. its Appl., vol. 444, no. 2, pp. 360–372, 2016.

[9] H. Cheng, Y. Zhou, and J. X. Yu, "Clustering Large Attributed Graphs: A Balance between Structural and Attribute Similarities," ACM Trans. Knowl. Discov. Data, vol. 5, no. 2, p. 12:1-- 12:33, 2011.

[10] S. Navlakha, M. C. Schatz, and C. Kingsford, "Revealing biological modules via graph summarization.," J. Comput. Biol., vol. 16, no. 2, pp. 253–264, 2009.

[11] Srivastava, S., Singh, A.K. Fraud detection in the distributed graph database. Cluster Comput (2022). https://doi.org/10.1007/s10586-022-03540-3

[12] Srivastava, S., & Singh, A. K. (2018, December). Graph Based Analysis of Panama Papers. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (pp. 822-827). IEEE.

[13] M. I. Seltzer and P. Macko, "Provenance Map Orbiter: Interactive Exploration of Large Provenance Graphs," Proc. 3rd USENIX Work. Theory Pract. Proven., pp. 20–21, 2011.

[14] C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu, "Graph OLAP: Towards online analytical processing on graphs," Proc. Of IEEE Int. Conf. Data Mining, ICDM, pp. 103–112, 2008.

[15] C. Chen and C. Lin, "Mining graph patterns efficiently via randomized summaries," Proc. of the VLDB Endowment, vol. 2, no. 1, pp. 742–753, 2009.

[16] M. Riondato, D. Garcia-Soriano, and F. Bonchi, "Graph Summarization with Quality Guarantees," Proc. - IEEE Int. Conf. Data Mining, ICDM, vol. 2015–Janua, no. January, pp. 947–952, 2015.

[17] R. Pienta, A. Tamersoy, H. Tong, and D. H. Chau, "MAGE: Matching approximate patterns in richly-attributed graphs," Proc. - 2014 IEEE Int. Conf. Big Data, IEEE Big Data 2014, pp. 585–590, 2015.

[18] G. a. Bilodeau and R. Bergevin, "Matching Graphs with Fuzzy Attributes in Machine Vision," Int. J. Robot. Autom., vol. 20, pp. 1–22, 2005.

[19] N. Z. Gong et al., "Jointly predicting links and inferring attributes using a social-attribute network (san)," ar Xiv Prepr. arXiv1112.3265, p. 9, 2011.

[20] X. Liu, Y. Tian, Q. He, W.-C. Lee, and J. McPherson, "Distributed Graph Summarization," Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manag. - CIKM "14, pp. 799–808, 2014.

[21] S. Navlakha et al. 2008. Graph Summarization with Bounded Error. ACM SIGMOD Conference (2008).

[22] Y. Tian et al. 2008. Efficient Aggregation for Graph Summarization. ACM SIGMOD Conference (2008).

[23] N. Zhang et al. 2010. Discovery-Driven Graph Summarization. IEEE ICDE (2010).

[24] M. Habib and C. Paul. A survey of the algorithmic aspects of modular decomposition. Elsevier Inc, Amsterdam, Netherlands, 2010.

[25] M. Tedder et al. 2008. Simple, Linear-time Modular Decomposition (Extended Abstract)∗. arXiv archive, Paper 0710.3901v2.

[26] P. Serafino. 2013. Speeding up Graph Clustering via Modular Decomposition Based Compression. ACM SAC (2013).

[27] S. Alvarez et al. 2010. A Compact Representation of Graph Databases. MLG Proceedings of the

Eighth Workshop on Mining and Learning with Graphs (2010).

[28] C. Li et al. 2015. Modul Graph: modularity-based visualization of massive graphs. ACM SIGGRAPH (2015).

[29] Z. Liu et al. 2013. Frequent subgraph summarization with error control. WAIM (2013). [30] K. Khan. 2015. Set-based Approach for Lossless Graph Summarization using Locality Sensitive Hashing. IEEE ICDEW (2015).

[31] K. Khan et al. 2015. Lossless graph summarization using dense subgraphs discovery. ACM IMCOM (2015).

[32] K. Khan et al. 2014. An Efficient Algorithm for MDL Based Graph Summarization for Dense Graphs. HIKARI Ltd. 66

[33] S. Qiao, "Querying graph structured RDF data," Ph.D. dissertation, Dept. Electrical Engineering and Computer Science, Case Western Reserve Univ., Cleveland, OH, 2016.

[34] C. Bizer and A Schultz. 2009. The Berlin SPARQL Benchmark.

[35] D. J. Abadi et. al. 2007. Scalable semantic web data management using vertical partitioning, Proceedings of the 33rd international conference on Very large data bases, September 23-27, 2007, Vienna, Austria

[36] R Development Core Team (2008). R: A language and environment forstatistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

[37] B. Ripley, et. al. May 29, 2016. Package „rpart‟. Available: https://cran.rproject.org/web/packages/rpart

[38] G. J. Williams. Dec. 2, 2009. Rattle: a data mining GUI for R. The R Journal Vol. 1.