A Meta-Classifier Link Prediction Model for False Profile Identification in Facebook

S. Saranya, M. Rajalakshmi, S. Devi, and R. M. Suruthi

Abstract Nowadays, social networks have become part of our everyday routine to connect and communicate to diverse communities. The detection of abnormal vertices in these networks has become more vital in exposing network intruders or malicious users. In this paper, we have included a novel meta-classifier which will notice anomalous vertices in complicated interactive network topology by extracting useful patterns. We identify that a vertex with several link connections will incorporate a higher probability of being abnormal. Henceforth, we choose to apply the link prediction model on the Facebook dataset with high interconnectivity. In each step, we determine abnormal vertices are detected with minimal false positive estimates and better accuracy when compared to alternative prevailing methods. Hence, the method incorporated reveals the outliers in the social networks. We proposed an approach that can identify the false profiles with 93% accuracy and lower false positive rates with sustainable accuracy.

Keywords Social networks · Anomaly detection · Link prediction classifier · Random forest · *K*-nearest neighbor · Facebook

S. Saranya (✉) · R. M. Suruthi
Faculty of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India
e-mail: saranya.s@cit.edu.in

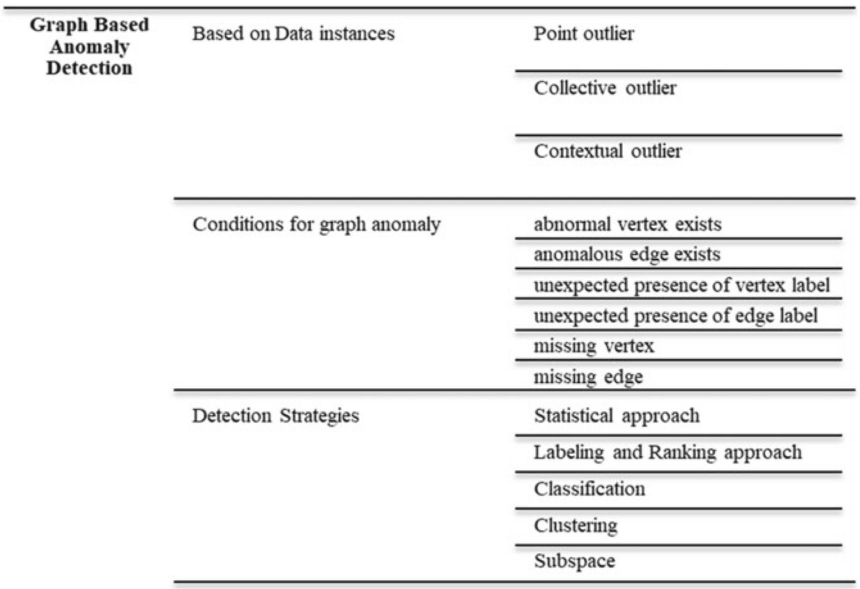
R. M. Suruthi
e-mail: suruthi.rm@cit.edu.in

M. Rajalakshmi · S. Devi
Faculty of Information Technology, Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India
e-mail: rajalakshmi@cit.edu.in

S. Devi
e-mail: devi.s@cit.edu.in

1 Introduction

In recent years, every person across the globe uses social networks as a platform to publicly share their knowledge and views, as an embodiment of their life. Fake profile identification is a salient problem within the field of social network analysis (SNA) and knowledge discovery (KDD). To maintain the network security and privacy in an interactive environment like social media, it is required to detect structural abnormalities that violate typical behavior of social networks. An outlier is a data point or a collection which deviates so much from the other observations as to arouse suspicions [1, 2]. Generally, normal data follow a salient pattern behavior differentiating it from anomaly. A general overview of graph-based anomaly detection methods is shown in Fig. 1. The anomaly in the graph can prevail as individual data outlier, in groups and disguises its nature on contextual analysis. Detection strategies calibers in using approaches based on statistics, classification, clustering, subspace, community detection, labeling/ranking, and so on.



Graph Based Anomaly Detection	Based on Data instances	Point outlier
		Collective outlier
		Contextual outlier
	Conditions for graph anomaly	abnormal vertex exists
		anomalous edge exists
		unexpected presence of vertex label
		unexpected presence of edge label
		missing vertex
		missing edge
	Detection Strategies	Statistical approach
		Labeling and Ranking approach
		Classification
		Clustering
		Subspace

Fig. 1 General overview of anomaly detection in social network

Studies show that the vertices which deviate from the typical behavior of social networks offer necessary insights into a network. There is a vast amount of bogus id existing in social networks than legitimate users present in different communities. In order to identify the fake profiles which are a serious threat to the secure use of social media, it is highly mandated to detect the anomaly behavior. Graph data reveals inter-dependencies that should be accounted for during the outlier detection process. When compared to traditional data analysis, graph data processing is highly beneficial. They maintain significant interconnections between data in the long run and are flexible to change.

In this paper, anomalous vertices in graph topology are identified by two phases. The primary phase predicts the edge probability in the network by applying a link prediction classifier. Next phase produces the new features set as output. The contributions of the work:

- Deploy sampling of vertices and edges using positive and negative sample strategy.
- Generate a link prediction classifier to find anomalous vertices in a graph
- We adopt a Meta classifier model in the training and testing phase.

2 Related Work

Graphical data analysis has grown over the past years and hence has increased the need for research in social networks. In a graph, data structure substructure reoccur [3]. Hence, it was proposed that the anomalies are substructures that occur infrequently. To understand the problems associated with fake profile creation, several detection methods have been discussed [4]. Most research contributions in machine learning and deep learning prevail but still fake accounts thrive in social networks [5]. In network sampling, there is a comparison of two alternatives for sampling networks that have a predefined number of edges [6]. A random selection of edges with uniform distribution and the edges of the 1.5 ego-networks finds network hubs that are the nodes with high traffic. The hubs are used as they are showing new information due to their high degree. However, the use of hubs may result in major bottlenecks in the networks. Infiltration of fake users into social networks using random friend requests [7]. Feature extraction of the networks are done based on the degree of user, connections that exist midst the friends of the user, amount of communities the user is coupled to. However, this may result in high anonymity level and may fail in detecting the spammer profiles. Feature extraction uses principal component analysis [8]. Main components in the networks are evaluated by reducing the total number of variables and also reduces the dimensions of all observations based on the classification of alike observations. Communication structure among the variables is also found by PCA. However, another dilemma in the PCA method is based on the selection of core components. Relational dependency networks accomplish collective classification of attributes of the designated variables and are assessed based on a joint probability distribution model. The technique should be trained with more unstructured multimedia data, databases, graph-based analytics, and conception for improved results.

Detection of strange nodes in bipartite graphs involved calculation of normality scores [9] created on the neighborhood relevance score where nodes with a minor normality score had a higher probability of being irregular. In machine learning, using simple heuristics or more refined algorithms is created that is based on the strength of the connection between the users [10]. The first evaluation method involves splitting of datasets hooked on training sets and testing sets. In the second evaluation method, the goal was to measure the classifiers recommendations average users precision. The connection-strength heuristic does not propose a general method of identifying which of the users are to be avoided.

Relational dependency networks (RDN) [11] can achieve collective classification when several attributes midst the selected variables are assessed along on the basis of a joint probability model. Multiple Gibbs sampling iterations were used to fairly accurate the joint distribution. The RDN results are nearly as good as the upper bound data-rich condition. The RDN is capable as it becomes possible to predict leadership parts with some degree of accuracy for circumstances in which very little specific data is known about the individual actors. The suggested method should be qualified with more unstructured data like multimedia data, graph data which will give imagining for improved results. In profile similarity technique [12] similar user profiles are grouped on the basis of profile attributes. User profiles are validated remotely. Using this strategy will drastically improve the search speed of a profile and gradually reduce the memory consumption.

Randomwalk [5, 13] extracts random path sequence from graph. Randomwalk is often used as a part of algorithms that create node embeddings. Deepwalk [14] builds upon a random walk approach which aids to learn latent meaningful graph representations. After the online clustering, anomaly detection is done. If the vertex or edges are far from all current clustering centroid points, then it will be claimed as abnormal. Deepwalk [15] is scalable [4]. Results show that when compared to spectral methods, this approach shows significantly better results for large graphs. This method is designed to operate for sparse graphs too. Netwalk [16], predicts anomalous nodes on dynamic traversal of nodes. As the graph network evolves, the network feature representation is updated dynamically. SDNE [16] and dLSTM [17] are deep network embedding models that facilitate learning vertex representations. It studies the network behavior using autoencoder and locality-preserving constraints.

A generic link prediction classifier [18, 19] aims to detect anomalous vertices by estimating the probability of edge absence in the graph. Based on features collected from graph topology, for each vertex the average probability of missing edges is calculated. Nodes with the highest average probability are predicted as anomalous nodes. This method was not applied on bipartite and weighted graphs [20, 21]. Label propagation algorithms (LPA) [22] is the fastest algorithm that involves labeling nodes in graph structure. The core idea is to handle highly influential nodes through label propagation. On edge traversal, we find the nodes with the more link connectivity are inferred as leader nodes or influential nodes. LPA is applicable to static plain graphs and it suffers from inconsistency and unstable behavior. Random forest algorithm [23– 25] is a classification method to find link patterns in a given graph structure. It is handled on static plain graphs. This method performs probabilistic ranking to find irregular vertices with good accuracy prediction results. Node similarity communication matching algorithm [26] is proposed to identify the cloned profile in online social network (OSN). It monitors the behavior of profile and finds for similarity matching with recent activity with 93% detection accuracy. It outperforms well over K-nearest neighbor (KNN) and support vector machine (SVM) methods.

In this section, many researchers have contributed their machine learning strategies to solve the problem domain. Most of the strategies either use feature reduction or feature extraction to identify the characteristic behavior of the graph data. The analysis is focused to find patterns from node interconnections as tracing malicious behavior is problematic. Though KNN and random forest classification provides better prediction independently, the performance aspect on integrating the classifier to anomaly detection is targeted.

3 Proposed Methodology

The input of the proposed system is a graph dataset that represents the social networks in the form of nodes and edges. The work flow of the proposed methodology is shown in Fig. 2. The working model is categorized into three stages involving anomalous node identification processes from the original graph.

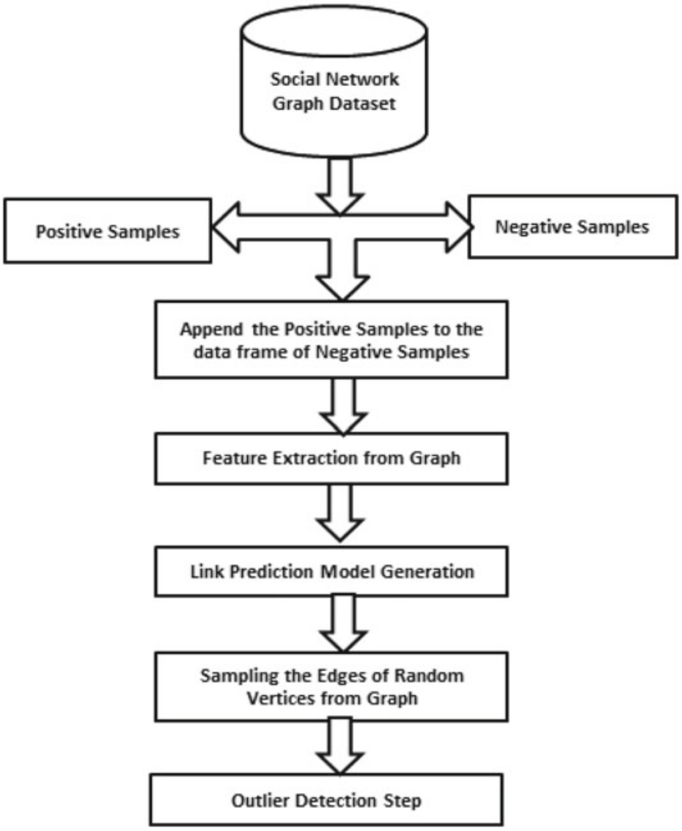

Social Network Graph Dataset
Positive Samples
Negative Samples
Append the Positive Samples to the data frame of Negative Samples
Feature Extraction from Graph
Link Prediction Model Generation
Sampling the Edges of Random Vertices from Graph
Outlier Detection Step

Fig. 2 Work model of the proposed methodology

3.1 Phase 1: Building a Classifier Model for Edge Connectivity

For every different graph manifest, a link prediction classifier is built which predicts a link between two unconnected nodes. For this, a training dataset is prepared from the graph that consists of negative and positive and samples. Negative samples are the major part of the dataset as they consist of the unconnected node pairs. Both Algorithm 1 and 2 shows steps in generating the positive and negative samples from graph input data, respectively.

Algorithm 1

Negative samples generation

```
Input: Graph G with node pairs
Output: Negative sample pairs
# Get unconnected node-pairs
all_unconnected_pairs ← new list()
#Traverse adjacency matrix
Initialize offset ← 0
for each i ← tqdm(range(G[0]) do
    for each j ← range(offset, G[6]) do
        if ( (i ≠ j) and (shortest_path_length(G,i, j) ≤ 2)) and (adjacency(i,j) = 0) then
          all_unconnected_pairs ← append( [node_list[i], node_list[j]])
         end if;
      offset = offset + 1
    end for;
end for;
```

Algorithm 2

Positive samples generation

```
Input: Graph G with node pairs
Output: Positive sample pairs
Initialize
         initial_node_count ← len(G nodes)
         #copy of nodes pairs in facebook graph dataframe
         fb_df_temp ← fb_ df
#empty list to store removable links
omissible_links_index ← new list()
for i ← tqdm(fb_df.index.values) do
   # remove a node pair and build a new graph
     G_temp ← fb_df_temp.drop(index = i)
        # check if there is no splitting of graph and number of nodes is same
             if (number_connected_components(G_temp) = = 1) and (len(G_temp.nodes) = =
initial_node_count) then
        omissible_links_index ← append(i)
        fb_df_temp ← fb_df_temp.drop(index = i)
    end if;
end for;
```

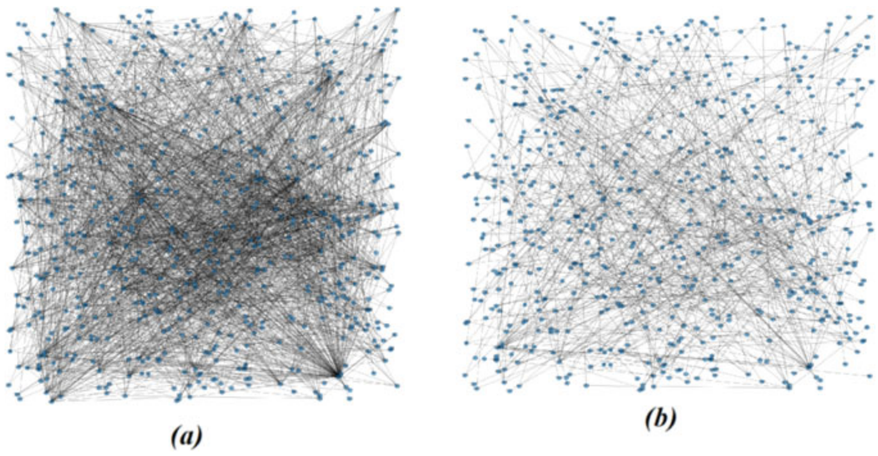


Fig. 3 **a** Original graph, **b** graph from link prediction classifier

Edges are randomly dropped from the graph in a condition that even in the process of dropping edges the nodes of the graph remain connected. These removable edges are appended to the data frame of unconnected node pairs. Feature extraction is performed on the graph after the dropping of the links. In the stage to obtain a trained dataset, jaccard coefficients/jaccard index is used to integrate the overlapping features. Let *x*,*y* be some graph vertex, the jaccard index is performed by comparing its neighbor nodes which is represented in Eq. 1.

$$\text{jaccard Index}(x, y) = \frac{|\text{neighborhood } (x) \cap \text{neighborhood}(y)|}{|\text{neighborhood } (x) \cup \text{neighborhood } (y)|} \tag{1}$$

After feature extraction to validate the model, the dataset is split for the training and testing of the model's performance. Random forest classifier is employed to generate the link prediction model based on the obtained features. Figure 3 represents the resultant graph after building the link prediction classifier.

3.2 Phase 2: Sampling Vertices from Graphs

In this phase, the samples are generated from a graph using a specific criterion. The threshold is set to be min_friends to satisfy the feature extraction process. The calculation of min_friends uses the knowledge of degree distribution to set the scale. Algorithm 3 shows the steps to perform the sampling of vertices and edges for a given graph *G*.

Algorithm 3

```
Graph vertex sampling
Input: Graph G with node pairs, min_ friends
Output: Selected edge list
   Initialize
      selected_edges ← new set()
      list_nodes = list(G.nodes())
   for each val ∈ list_nodes do
      if len(list(G.neighbors(val))) ≥ min_friends then
         Temp_edges ← new set()
         for node in list(G.neighbors(val)) do
            if len(list(G.neighbors(node))) ≥  min_friends then
               Temp_edges ← add(val)
               Temp_edges ← add(node)
       end if;
       if len(Temp_edges) ≥  min_friends then
          selected_edges ← union(selected_edges, Temp_edges)
         end if;
      end for
  end if;
end for;
```

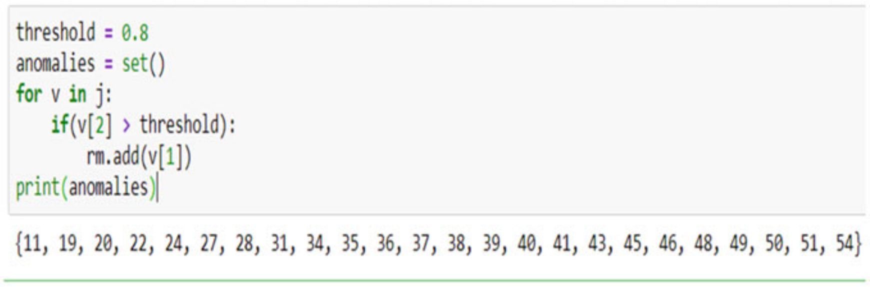
3.3 *Phase 3: Detection of Outlier from Resultant Graph*

In this phase, the link prediction classifier will generate interesting features that enable it to perform outlier detection. This method is centered on the hypothesis that a vertex with many low variance link connections has a higher likelihood of being anomalous. Test set is obtained by performing random edge samples from the graph. The algorithm steps involved in selecting node links that are completely traversed and chooses neighbors with more than minimum friend neighbors. We configure the min friend to be set to three. We omit the nodes with very low neighbors connected as it provides poor network characteristics. These sets of vertices are considered for further evaluation of anomaly detection and the outliers are detected which is represented in Fig. 4.

4 Experimental Evaluation

4.1 *Data Preparation*

The experiments are conducted on a system operating with a 64 bit windows platform with Intel quad core processors supporting GPU specification. For experimental



```
threshold = 0.8
anomalies = set()
for v in j:
    if(v[2] > threshold):
        rm.add(v[1])
print(anomalies)
```

```
{11, 19, 20, 22, 24, 27, 28, 31, 34, 35, 36, 37, 38, 39, 40, 41, 43, 45, 46, 48, 49, 50, 51, 54}
```

Fig. 4 Result of graph vertex outlier

evaluation, we use Python-3.6 utilizing machine learning libraries comprising scikit-learn version 0.19.1 and NumPy version 1.14.0 to perform mathematical operations.

We evaluated our algorithm on the Facebook dataset from network repository which has 620 nodes and 2102 edges. The fb_pages_food is an undirected graph with nodes representing the Facebook pages and edges having mutual likes among them. In addition to that we have also added 20 randomly generated edges or anomalies using Erdos_Renyi algorithm. To incorporate anomalous nodes, reindexing of the node pairs can be done in the initial dataset.

4.2 Generation of Anomalous Edges

To generate random anomalous edges in the existing graph, Erdos–Renyi Model is used. In this model, each edge distribution has a fixed probability of being available or missing, regardless of the number of connecting links in the network. The probability of edge distribution existing for our model has been set as 0.5 since it gives an equal distribution of the edges. Figure 5 gives the degree of distribution of the edges across the existing network.

4.3 Result and Experimental Setup

Figure 6 shows the anomalous nodes that are obtained from the proposed algorithm. The node information helps to prevent malicious activity. Nodes are crawled such that outlier nodes are extracted on configuring the threshold value set to 0.8. A node exceeding the limit is outliers and monitoring the network activity of the node is encouraged.

To evaluate the performance metric of the detection model, a confusion matrix is employed. The confusion matrix is used to solve this binary classification as either genuine/ normal node or outlier node. The confusion matrix gives values such as

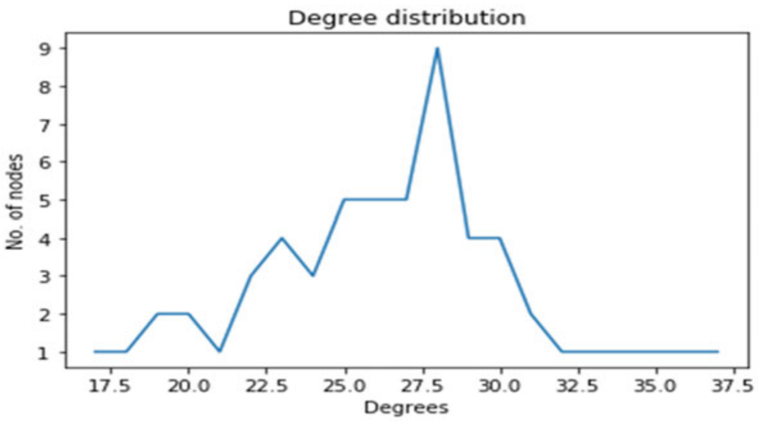
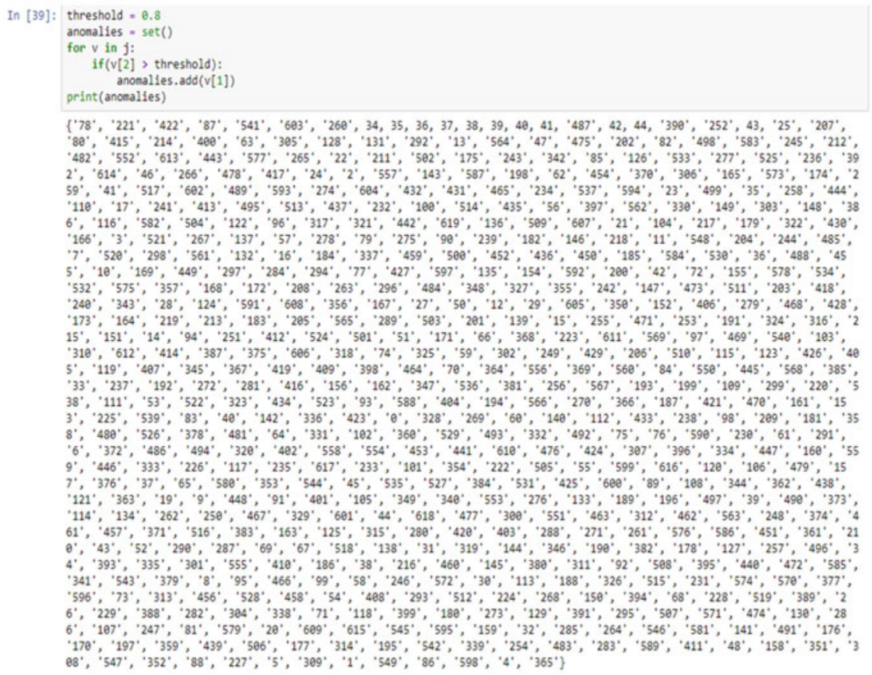


Fig. 5 Degree distribution of Facebook network



```
In [39]: threshold = 0.8
         anomalies = set()
         for v in j:
             if(v[2] > threshold):
                 anomalies.add(v[1])
         print(anomalies)
```

```
{'78', '221', '422', '87', '541', '603', '260', 34, 35, 36, 37, 38, 39, 40, 41, '487', 42, 44, '390', '252', 43, '25', '207',
'80', '415', '214', '400', '63', '305', '128', '131', '292', '13', '564', '47', '475', '202', '82', '498', '583', '245', '212',
'482', '552', '613', '443', '577', '265', '22', '211', '502', '175', '243', '342', '85', '126', '533', '277', '525', '236', '39
2', '614', '46', '266', '478', '417', '24', '2', '557', '143', '587', '198', '62', '454', '370', '306', '165', '573', '174', '2
59', '41', '517', '602', '489', '593', '274', '604', '432', '431', '465', '234', '537', '594', '23', '499', '35', '258', '444',
'110', '17', '241', '413', '495', '513', '437', '232', '100', '514', '435', '56', '397', '562', '330', '149', '303', '148', '38
6', '116', '582', '504', '122', '96', '317', '321', '442', '619', '136', '509', '607', '21', '104', '217', '179', '322', '430',
'166', '3', '521', '267', '137', '57', '278', '79', '275', '90', '239', '182', '146', '218', '11', '548', '204', '244', '485',
'7', '520', '298', '561', '132', '16', '184', '337', '459', '500', '452', '436', '450', '185', '584', '530', '36', '488', '45
5', '10', '169', '449', '297', '284', '294', '77', '427', '597', '135', '154', '592', '200', '42', '72', '155', '578', '534',
'532', '575', '357', '168', '172', '208', '263', '296', '484', '348', '327', '355', '242', '147', '473', '511', '203', '418',
'240', '343', '28', '124', '591', '608', '356', '167', '27', '50', '12', '29', '605', '350', '152', '406', '279', '468', '428',
'173', '164', '219', '213', '183', '205', '565', '289', '503', '201', '139', '15', '255', '471', '253', '191', '324', '316', '2
15', '151', '14', '94', '251', '412', '524', '501', '51', '171', '66', '368', '223', '611', '569', '97', '469', '540', '103',
'310', '612', '414', '387', '375', '606', '318', '74', '325', '59', '302', '249', '429', '206', '510', '115', '123', '426', '40
5', '119', '407', '345', '367', '419', '409', '398', '464', '70', '364', '556', '369', '560', '84', '550', '445', '568', '385',
'33', '237', '192', '272', '281', '416', '156', '162', '347', '536', '381', '256', '567', '193', '199', '109', '299', '220', '5
38', '111', '53', '522', '323', '434', '523', '93', '588', '404', '194', '566', '270', '366', '187', '421', '470', '161', '15
3', '225', '539', '83', '40', '142', '336', '423', '0', '328', '269', '60', '140', '112', '433', '238', '98', '209', '181', '35
8', '480', '526', '378', '481', '64', '331', '102', '360', '529', '493', '332', '492', '75', '76', '590', '230', '61', '291',
'6', '372', '486', '494', '320', '402', '558', '554', '453', '441', '610', '476', '424', '307', '396', '334', '447', '160', '55
9', '446', '333', '226', '117', '235', '617', '233', '101', '354', '222', '505', '55', '599', '616', '120', '106', '479', '15
7', '376', '37', '65', '580', '353', '544', '45', '535', '527', '384', '531', '425', '600', '89', '108', '344', '362', '438',
'121', '363', '19', '9', '448', '91', '401', '105', '349', '340', '553', '276', '133', '189', '196', '497', '39', '490', '373',
'114', '134', '262', '250', '467', '329', '601', '44', '618', '477', '300', '551', '463', '312', '462', '563', '248', '374', '4
61', '457', '371', '516', '383', '163', '125', '315', '280', '420', '403', '288', '271', '261', '576', '586', '451', '361', '21
0', '43', '52', '290', '287', '69', '67', '518', '138', '31', '319', '144', '346', '190', '382', '178', '127', '257', '496', '3
4', '393', '335', '301', '555', '410', '186', '38', '216', '460', '145', '380', '311', '92', '508', '395', '440', '472', '585',
'341', '543', '379', '8', '95', '466', '99', '58', '246', '572', '30', '113', '188', '326', '515', '231', '574', '570', '377',
'596', '73', '313', '456', '528', '458', '54', '408', '293', '512', '224', '268', '150', '394', '68', '228', '519', '389', '2
6', '229', '388', '282', '304', '338', '71', '118', '399', '180', '273', '129', '391', '295', '507', '571', '474', '130', '28
6', '107', '247', '81', '579', '20', '609', '615', '545', '595', '159', '32', '285', '264', '546', '581', '141', '491', '176',
'170', '197', '359', '439', '506', '177', '314', '195', '542', '339', '254', '483', '283', '589', '411', '48', '158', '351', '3
08', '547', '352', '88', '227', '5', '309', '1', '549', '86', '598', '4', '365'}
```

Fig. 6 Outlier nodes detected

True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The mathematical formulation of the performance parameters are represented in Eqns. 2–5, where TP is nodes correctly classified as outlier, TN is node correctly classified as normal, FP is nodes misclassified as outlier from normal, and FN is nodes misclassified as normal from outlier.

$$\text{Accuracy} = (\text{TP} + \text{TN})/\text{Total Nodes} \tag{2}$$

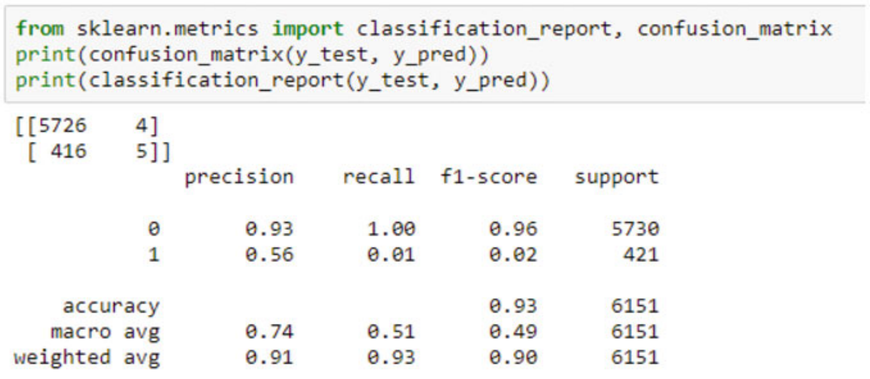
$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}) \tag{3}$$

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN}) \tag{4}$$

$$f1 - \text{score} = (2 \times \text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall}) \tag{5}$$

Figure 7 tabulates the $f1$-score, support, precision, and recall for the trained model. The KNN uses the Minkowski metric for this step. We have also measured the error rate for different k values (k lies between 1 and 40).

Figure 8 depicts that the highest error rate is 11% for $k = 0$ and the lowest error rate of 6% is for $k = 6$. The error rate remains constant (i.e.) 6% for the value of k greater than 6. The algorithm has been trained using KNN model, and its accuracy is found to be 93% with the maximum error rate of 0.14. Table 1 summarizes the comparison of the proposed model outperforming other methods [23, 26].



```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[5726    4]
 [ 416    5]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96      5730
           1       0.56      0.01      0.02       421

    accuracy                           0.93      6151
   macro avg       0.74      0.51      0.49      6151
weighted avg       0.91      0.93      0.90      6151
```

Fig. 7 Classification report

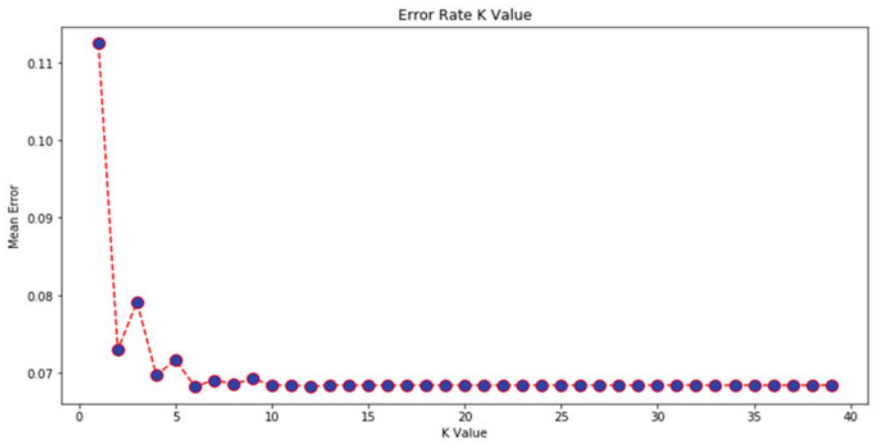


Fig. 8 Error rate calculation

Table 1 Comparison table

S. No	Model Used	Accuracy(%)	Time (ms)
1	K-nearest neighbor (KNN)	89	0.89
2	Support vector machine (SVM)	84	0.78
3	Node similarity communication matching	93.14	0.45
4	Naive bayes	83	0.64
5	Proposed model (Random forest + KNN)	93.495	0.40

5 Conclusion

To understand the behavior patterns of large networks, detection of anomalies are very important. We propose a novel strategy that discovers anomalous nodes on the basis of characteristic traits of the underlying network structure. The method adopts current techniques in machine learning to generate useful communities. We experimented this method in the Facebook social network dataset, but there is considerable scope to be tested with a diverse application. The proposed method is trained using KNN model and its accuracy is found to be 93% with the maximum error rate of 0.14. As for future work, the algorithm can be prolonged to be applied on undirected and dynamic real-world graph datasets. We plan to extend the work to incorporate a neural network learning model so as to increase the performance of the proposed model.

References

1. Hodge VJ, Austin J (2004) A survey of outlier detection methodologies. Artif Intell Rev 22(2):85–126
2. Saranya S, Rajalakshmi M (2022) Certain St rategic study on machine learning-based graph anomaly detection. In: Shakya S, Bestak R, Palanisamy R, Kamel KA (eds) Mobile computing and sustainable informatics. Lecture notes on data engineering and communications technologies, vol 68. Springer, Singapore. https://doi.org/10.1007/978-981-16-1866-6_5
3. Noble CC, Cook DJ (2003) Graph-based anomaly detection", ACM SIGKDD '03, August 24–27, 2003
4. Roy PK, Chahar S (Dec.2020) Fake profile detection on social networking websites: a comprehensive review. IEEE Trans Artif Intell 1(3):271–285. https://doi.org/10.1109/TAI.2021.3064901
5. Moonesinghe HDK, Tan PN (2008) OutRank: a graph-based outlier detection framework using random walk. Int J Artif Intell Tools 17(1)
6. Altshuler Y et al (2013) Detecting Anomalous behaviors using structural properties of social networks. In: Greenberg AM, Kennedy WG, Bos ND (eds) Social computing, behavioral-cultural modeling and prediction, SBP 2013. Lecture Notes in Computer Science, vol 7812. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37210-0_47
7. Fire M, Katz G, Elovici Y (2012) Strangers Intrusion detection—detecting spammers and fake profiles in social networks based on topology anomalies. In: ASE Human J
8. Mohammadrezaei M, Shiri ME, Rahmani AM (2018) Identifying fake accounts on social networks based on graph analysis and classification algorithms. Secur Commun Netw 1–8. https://doi.org/10.1155/2018/5923156
9. Sun J, Qu H, Chakrabarti D, Faloutsos C (200) Neighborhood formation and anomaly detection in bipartite graphs. In: Fifth IEEE international conference on data mining (ICDM'05), p 8. https://doi.org/10.1109/ICDM.2005.103
10. Fire M, Kagan D, Elyashar A et al (2014) Friend or foe? Fake profile identification in online social networks. Soc Netw Anal Min 4:194. https://doi.org/10.1007/s13278-014-0194-4
11. Campbell WM, Dagli CK, Weinstein CJ (2013) Social network analysis with content and graphs
12. Nandhini DM, Das BB (2016) Profile similarity technique for detection of duplicate profiles in online social network
13. Vempala S (2005) Geometric random walks: a survey. Comb Comput Geom MSRI Publ 52:573–612
14. Perozzi B, Al-Rfou R, Skiena S (2019) DeepWalk: online learning of social representations. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining. https://doi.org/10.1145/2623330.2623732
15. Tan Q, Liu N, Hu X (2014) Deep representation learning for social network analysis. Front Big Data 2:2
16. Yu W, Cheng W, Aggarwal CC, Zhang K, Chen H, Wang W (2018) NetWalk: a flexible deep embedding approach for anomaly detection in dynamic networks. pp 2672–2681. https://doi.org/10.1145/3219819.3220024.
17. Maya S, Ueno K, Nishikawa T (2019) dLSTM: a new approach for anomaly detection using deep learning with delayed prediction. Int J Data Sci Anal
18. Kagan D, Elovichi Y, Fire M (2018) Generic anomalous vertices detection utilizing a link prediction algorithm. Soc Netw Anal Mining
19. Fadaee SA, Haeri MA (2019) Classification using link prediction. Neurocomputing 359:395–407. https://doi.org/10.1016/j.neucom.2019.06.026
20. Fire M, Tenenboim L, Lesser O, Puzis R, Rokach L, Elovici Y (2011) Link prediction in social networks using computationally efficient topological features. In: IEEE Third Int'l conference on privacy, security, risk and trust and IEEE third Int'l conference on social computing. https://doi.org/10.1109/passat/socialcom.2011.20

21. Lichtenwalter RN, Lussier JT, Chawla NV (2010) New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining—KDD '10,2010. https://doi.org/10.1145/1835804.1835837
22. Bhatia V, Saneja B, Rani (2017) INGC: graph clustering & outlier detection algorithm using label propagation. In: International conference on machine learning and data science 2017
23. Homsi A, Al Nemri J, Naimat N, Kareem HA, Al-Fayoumi M, Snober MA (2021) Detecting twitter fake accounts using machine learning and data reduction techniques. DATA
24. Primartha R, Tama BA (2017) Anomaly detection using random forest: a performance revisited. https://doi.org/10.1109/ICODSE.2017.8285847
25. Rahman O, Quraishi MA (2019) Experimental analysis of random forest, K-nearest neighbor and support vector machine anomaly detection. https://doi.org/10.13140/RG.2.2.19998.18245
26. Revathi S, Suriakala M (2018) Profile similarity communication matching approaches for detection of duplicate profiles in online social network. In: 2018 3rd International conference on computational systems and information technology for sustainable solutions, pp 174–182