

# Identifying Politicians in the Panama Papers Network

CSCI 5352 Final Project

Irene Beckman, Santhanakrishnan Ramani, Ruhi Saraf

December 12, 2016

**Abstract**—Using features from vertex level exploration and ego networks, we sought to create a binary classification that would distinguish politician nodes from all other people officer nodes in a network drawn from the Panama Paper data leak. After trying several different sampling methods, classification algorithms, and feature combinations we achieved the best accuracy of 88% using AdaBoost and oversampling.

## I. INTRODUCTION

In April of 2016, Mossack Fonseca, a Panamanian law firm known for selling and coordinating the sales of anonymous offshore companies, notified its clients of a security breach. Over 11.5 million emails, images, invoices, and internal and client communications were released shedding light on the firm’s inner working and constituting the largest leak known to date. These documents became known as the Panama Papers. Originally released to the German newspaper Süddeutsche Zeitung (SZ), the International Consortium of Investigative Journalists (ICIJ) quickly partnered with the smaller organization to sift through these documents. Along with legitimate dealings, what they found was a group of well known and connected individuals who used this firm for tax evasion, fraud, bribery, arms deals and drug trafficking. Members of this group included world leaders, sports figures, and influential businessman, as well as their friends and family. Using network data provided by the ICIJ, we sought to build a classifier that could identify the politicians from all other people mentioned.[9] [11].<sup>1</sup>

## II. DATA

### A. Structure

While processing the leak, the ICIJ constructed a network that attempted to capture the relationships outlined in the documents. The final network they produced, and the one used for our classification, is a directed graph with 1,040,537 nodes and 4,429,023 edges. Each node is labeled as one of the following: [8]

- i. *Entity*: an offshore company, trust, or fund
- ii. *Officer*: individuals or companies involved with an entity
- iii. *Intermediate*: predominately law-firms or individuals who couple clients with offshore service providers
- iv. *Address*: a physical address

Out of the over 1 million nodes, only 20% are people and officers. Only 102 nodes are known politicians or political affiliates. A distribution of network labels can be found in

Fig. 1. All edges are directed and represent any direct affiliation between two nodes such as a shareholder relationship, an entity registered at a specific address, or a person having a similar name to another node.

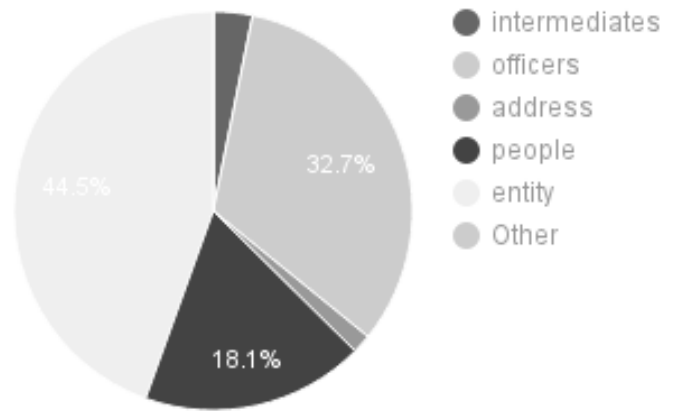


Fig. 1. Network Label Distribution

### B. Parsing and Filtering

We encountered several roadblocks when working with the data set that are worth mentioning. As can be seen in Fig. 1, the distribution of labels is uneven. Having a relatively large data set with distinct node types was making it difficult to pinpoint the politician subset within the officer group. To counteract this problem, we only looked at the set of around 200,000 officer nodes which are identified as people. This way we could narrow down which features were specifically distinguishing politicians instead of relying on good features for categorizing the officer. A second roadblock was determining which nodes were politicians for training and testing purposes. Although each node came with a name, they did not include any meta data on their politician affiliation. Flagging the appropriate subset of people and politicians nodes was a manual task involving other data provided by the ICIJ website. Finally, given the confidential and sometimes illegal nature of the business this network describes, the names associated with people nodes were often crafted to obfuscate the true identity of the person it represents. This took the form of multiple spelling variations and substituting in friends and family names. Ultimately, this resulted in duplicate nodes for the same person. The ideal solution to this problem is merging all replicate nodes.

<sup>1</sup><https://github.com/santro92/CSCI-5352>

Unfortunately, to our knowledge there exists no single truth of the necessary merges this solution requires. Due to the unknown extent of this issue and lack of data, we could not fix it globally and so left the data as it came

### III. FEATURES

Once the proper subset of nodes was selected and labeled, the first step in building our classification system was to gather different features for each person officer node. From the outset, we knew that no features should require knowledge of the entire graph. As such, the we calculated fell into two distinct categories: vertex level features and ego network features.

#### A. Vertex Level Features

We started by computing the following vertex level measures: In-degree, Out-degree, Average Neighbor Degree, Clustering Coefficient, Page Rank, Reciprocity, and Neighbor Label Distribution. All of these values only require knowledge of the vertex in question and its direct neighbors. Each property is detailed below and the means are displayed in figures following it.

- i. *In-Degree*: In-degree of vertex  $V_i$  in a directed graph  $G$  is the number of edges which are coming into  $V_i$  commonly denoted as  $deg^+(v)$ . We calculated this value on a directed version of the graph using a built in function from the networkx library. [2]

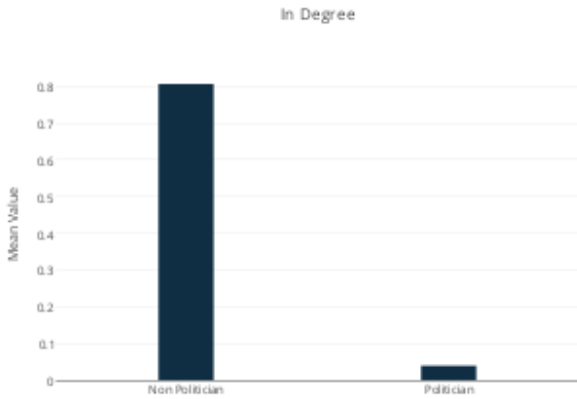


Fig. 2. In-Degree Feature Means

- ii. *Out-Degree*: Out-degree for vertex  $V_i$  in a directed graph  $G$  is the number of edges which originate at  $V_i$ , commonly denoted as  $deg^-(v)$ . We calculated this value on a directed version of the graph using a built in function from the networkx library. [2]

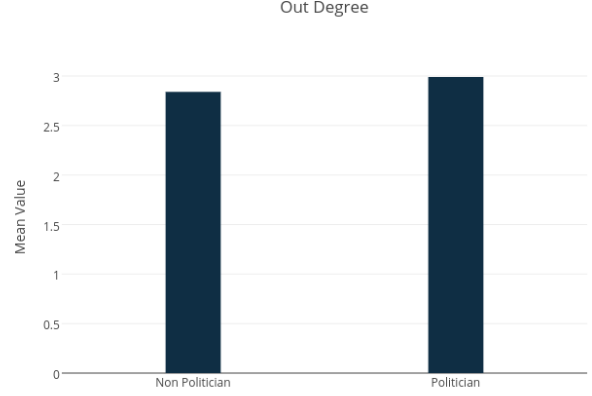


Fig. 3. Out-Degree Feature Means

- iii. *Average Neighbor Degree*: The average neighbor degree of a node  $v$  is calculated as follows:

$$k_{nn,v} = \frac{1}{|N(v)|} \sum_{j \in N(v)} k_j$$

where  $N(v)$  is the set of neighbors for node  $v$  and  $k_j$  is the degree of node  $j$  where  $j \in N(v)$ . We calculated this value on a undirected version of the graph using built functions from networkx. [2]

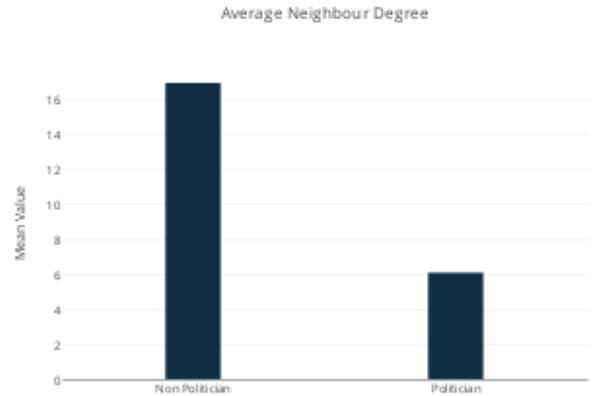


Fig. 4. Average Neighbor Feature Means

- iv. *Clustering Coefficient*: The clustering coefficient of a node  $v$  shows the fraction of existing triangles compared to the number that could possibly exist. Mathematically, it is shown below:

$$c = \frac{2T(v)}{k_v(k_v - 1)}$$

where  $T(v)$  is the number of triangles through  $v$  and  $k_v$  is the degree of  $v$ . The clustering coefficient was calculated on an undirected version of the graph using

networkx. [2]

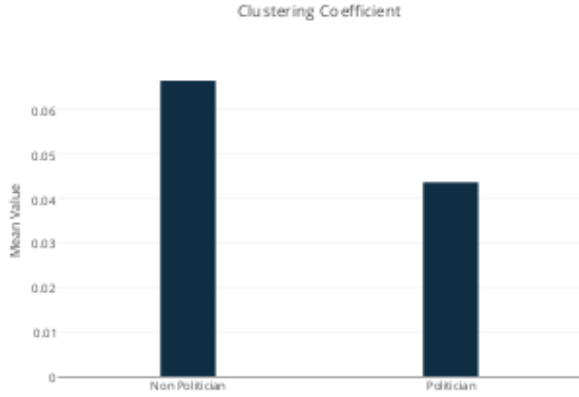


Fig. 5. Clustering Coefficient Feature Means

- v. *Page Rank*: The page rank computes a ranking of the nodes in the graph  $G$  based on the structure of the incoming links. Page Rank was calculated using networkx on a directed version of the graph. [2]

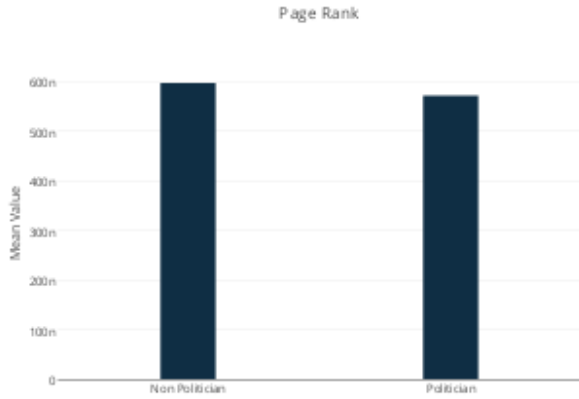


Fig. 6. Page Rank Feature Mean

- vi. *Reciprocity*: The reciprocity of node  $v$  is defined as the ratio of bidirectional edges to the total number of edges attached to node  $v$ .

$$r = \frac{|(v, u) \in G \cap (u, v) \in G|}{deg^+(v) + deg^-(v)}$$

Reciprocity was calculated on a directed version of the graph using networkx. [2] As all values for reciprocity were 0, we chose not to graph it or use it as a feature.

- vii. *Neighbor Label Distribution*: The neighbor label distribution calculations resulted in one value for of the four

possible labels for every node  $v$ . Each values represented the percentage of neighbors of  $v$  with a given label. This was calculated on an undirected version of the graph using build in networkx functions.[2] The equations for this follows:

$$p_{v,r} = \frac{\sum_{j \in N(v)} j | j \in L(r)}{|N(v)|}$$

where  $N(v)$  is the neighbor set of  $v$  and  $L(r)$  is the set of nodes with a given label  $r$ . In this implementation,  $r$  could be entity, address, intermediate, or officer.

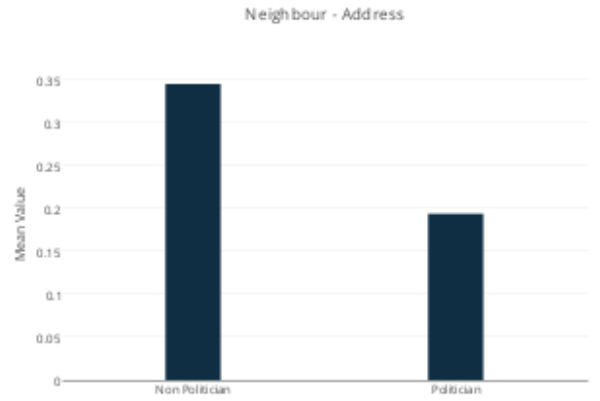


Fig. 7. Address Neighbor Label Feature Mean

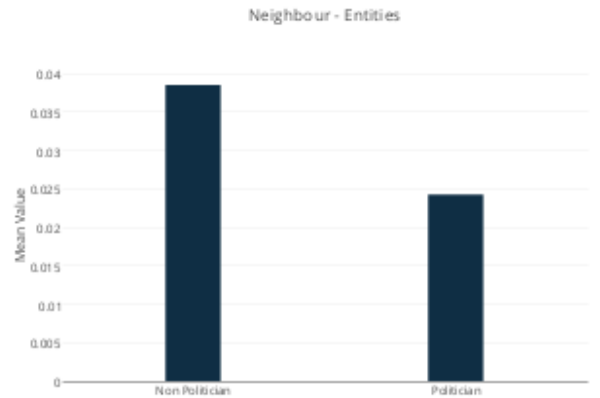


Fig. 8. Entity Neighbor Label Feature Mean

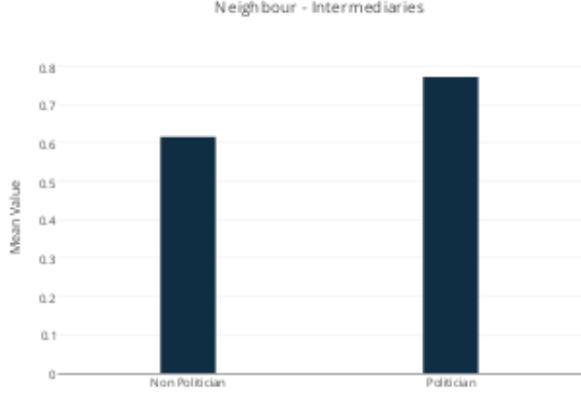


Fig. 9. Intermediate Neighbor Label Feature Mean

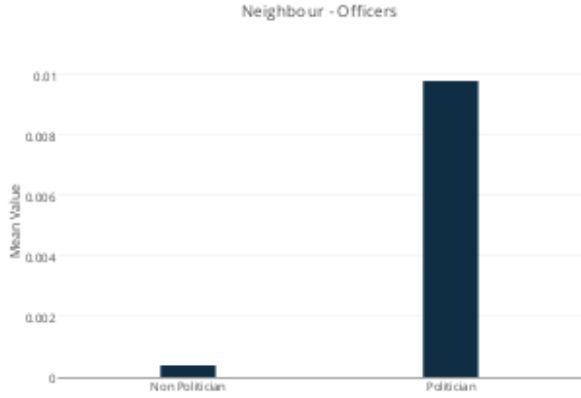


Fig. 10. Officer Neighbor Label Feature Mean

### B. Full Network and Ego-Network Features

While looking at metrics gathered based on vertex level knowledge provided some assistance in our classification, we wanted to expand our features to include those requiring global network knowledge. In an attempt to limited dependency on the entire network, we created ego networks for each person officer node. These networks consisted of a vertex, its neighbors and all of its neighbor's neighbors. In cases where it was possible, the original node was removed from the ego network. Some features still required the entire graph to calculate. The specifics of implementation graph type are detailed below and the means are displayed.

- i. *Rich-Club Coefficient*: The rich-club coefficient represents the tendency of highly connected nodes to link to one another. Given that the average degree,  $k$ , for a network  $G$  the rich club is defined by  $R(k) = \{v \in N(G) | k_v > k\}$ . Once  $R(k)$  is determined, the rich club coefficient is

$$c = \frac{1}{R(k)(R(k)-1)} \sum_{i,j \in R(k)} a_{ij}$$

where  $a_{i,j} = 1$  when an edge exists between any node  $i$  and rich club member  $j$ . [5] This was calculated using the Alg. 22 on the set of undirected ego networks that were missing the original vertex.

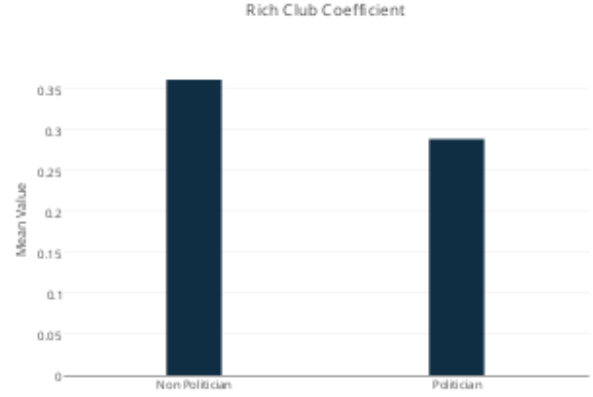


Fig. 11. Rich Club Feature Mean

- ii. *Local Cyclic Coefficient*: The local cyclic coefficient for node  $v$  is a value meant to provide a relative metric on how many cycles appear in a network. It averages the inverse of the shortest cycles between  $v$  and all of its neighbors. [5]. This was run a directed version of the original entire network using the Algorithm outline in Alg. 15.

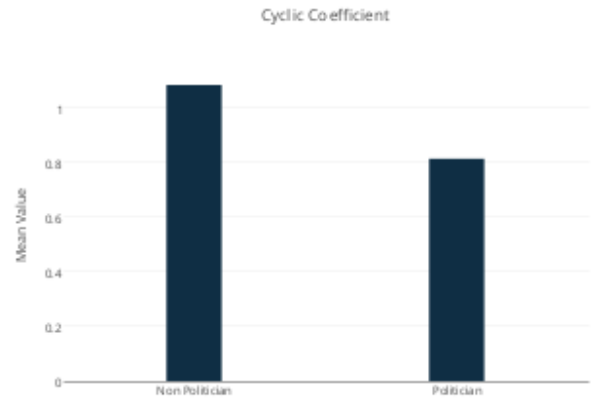


Fig. 12. Local Cyclic Coefficient Feature Means

- iii. *Modularity*: Modularity is a score that shows how well a graph can be segmented into communities. For our implementation of this we used a best partition function and modularity score calls from the Community Detection for Networkx library. [3] These were performed on undirected ego networks with the the original vertex removed and the results are outlined in Fig. 13.

---

**Algorithm 1** Local Cyclic Coefficient

---

```
1: procedure CALCULATECYCLICCOEFF( $v, E(v)$ )
2:   for all  $j \in \text{Neighbors}(v)$  do
3:      $c \leftarrow 0.0$ 
4:     for all  $k \in \text{Neighbors}(v)$  do
5:       if  $\text{pathExists}(j, k)$  then
6:          $c \leftarrow c + \frac{1}{\text{shortestPath}(j, k) + 2}$ 
7:       end if
8:     end for
9:   end for
10:   $\text{neighborSize} \leftarrow \text{size}(N(v))$ 
11:  if  $\text{neighborSize} > 1$  then
12:     $c \leftarrow \frac{2c}{\text{neighborSize} * (\text{neighborSize} - 1)}$ 
13:  return  $c$ 
14: end if
15: return 0
16: end procedure
```

---

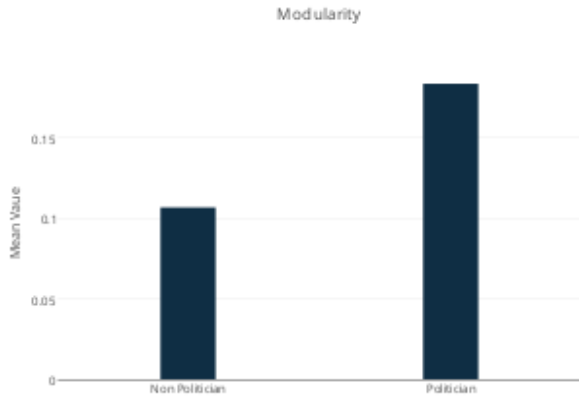


Fig. 13. Modularity Feature Means

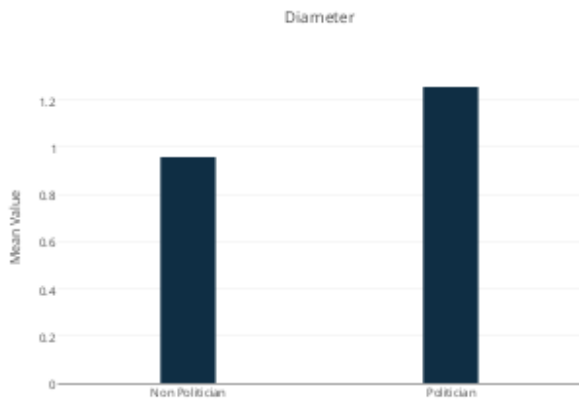


Fig. 14. Diameter Feature Means

- iv. *Diameter*: The diameter of a network is the most efficient path distance between the two farthest nodes. In other words, it is the largest shortest path between all possible combination of nodes in a given network. We calculated diameter using networkX on undirected ego networks missing their original vertex. [2] The means are displayed in Fig. 14.
- v. *Average Degree* : The average degree of a network,  $k$ , is the mean of all individual members edge count. This feature was determined using the networkX library on undirected ego graphs. [2]

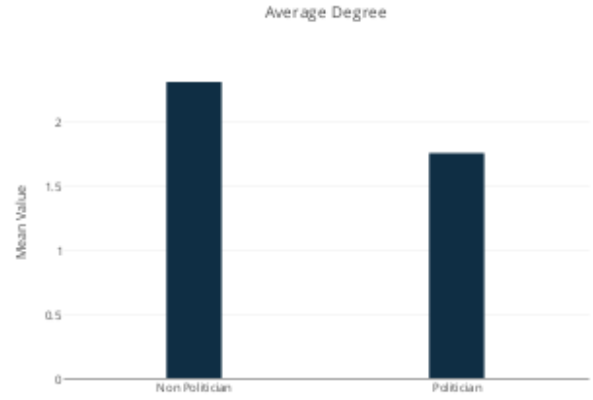


Fig. 15. Ego Network Average Degree Feature Means

---

**Algorithm 2** Rich-Club Coefficient

---

```
1: procedure CALCULATERICHCLUBCOEFF( $v, E(v)$ )
2:   $k \leftarrow \text{getAverageGraphDegree}(E(v))$ 
3:   $\text{nodesDegreesList} \leftarrow \text{getListOfNodesDegree}(E(v))$ 
4:   $\text{richClub} = []$ 
5:  for all  $\text{node} \in \text{nodesDegreesList}$  do
6:    if  $\text{node.degree}() > k$  then
7:       $\text{richClub.add}(\text{node})$ 
8:    end if
9:  end for
10:   $\text{coeff} \leftarrow \frac{1}{\text{richClub.size}() * (\text{richClub.size}() - 1)}$ 
11:   $\text{count} \leftarrow 0$ 
12:  for all  $\text{nodeA} \in \text{nodesDegreesList}$  do
13:    for all  $\text{nodeB} \in \text{nodesDegreesList}$  do
14:      if  $\text{nodeA} \neq \text{nodeB}$  then
15:        if  $\text{nodeB.isNeighbor}(\text{nodeA})$  then
16:           $\text{count}++$ 
17:        end if
18:      end if
19:    end for
20:  end for
21:   $\text{coeff} \leftarrow \text{coeff} * \text{count}$ 
22:  return  $\text{coeff}$ 
23: end procedure
```

---

- vi. *Radiality Centrality* : Radiality centrality is the measure of how close a given vertex is to every other vertex in relation to the diameter of the network. In other words, a node with high radial centrality will be closely connected to a large number of nodes in the graph.[10] The equation for this is seen below:

$$r_i = \frac{\sum_j D - d_{ij} + 1}{D(n-1)}$$

where  $d_{ij}$  is the distance between  $i$  and  $j$ ,  $n$  is the total number of nodes in the graph, and  $D$  is the diameter. This value was calculated using the Alg. 10 on the original vertex on the set of undirected ego networks that included the original vertex.

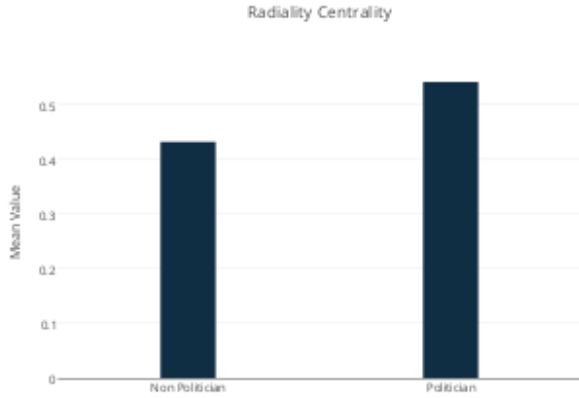


Fig. 16. Radiality Centrality Feature Means

---

**Algorithm 3** Radiality Centrality Coefficient

---

```

1: procedure CALCULATERADIALCOEFF( $v$ ,  $E(v)$ )
2:    $D \leftarrow E(v).diameter()$ 
3:    $n \leftarrow getNumberOfNodes(E(v))$ 
4:    $nodesList \leftarrow getListOfNodes(E(v))$ 
5:    $coeff \leftarrow 0.0$ 
6:   for all  $node \in nodesList$  do
7:      $coeff \leftarrow coeff + (D - shortestPath(v, node) + 1)$ 
8:   end for
9:    $coeff \leftarrow \frac{coeff}{(D * (n - 1))}$ 
10:  return  $coeff$ 
11: end procedure

```

---

### C. Feature Analysis

Looking at the feature averages for politicians versus non-politicians several trends come to light. While the ego networks of politicians tended to be bigger than that of non politicians, their average ego network degree is lower. This could be interpreted as a politician's way of creating longer chains of business transactions to obscure their role in the offshore account. Simultaneously,

this indicates they are trying to reduce the number of players involved. The in-degree and out-degree mean reaffirm this trend. Politicians have considerably lower in-degree's presumably making it harder to trace specific offshore dealings back to them. Their out-degree is roughly equivalent to that of non-politicians.

Another point of interest was the distribution of neighbor labels. Politicians are directly connected to a considerably higher number of officers and intermediates when compared to non-politicians. Intermediaries and officers can serve the purpose of adding layers in between a politician and an entity. On reading more about the investigations that have resulted from this leak, we learned that politicians commonly involve family and friends in their offshore activities to avoid public scrutiny. [6] If trying to distance themselves from a specific company or place, it follows that the average neighbor degree for entities and addresses is low.

In a similar vein, we were not surprised to find that the neighbor's average degree for a politician is higher than that of a non-politician. While politicians stray away from being directly linked to a large number of people and places, they still must be well connected. This is reinforced by the relatively high radiality centrality measures detected.

Looking at the modularity and rich club scores, it appears as though ego networks centered around a politician are much easier to divide into groups than non-politicians. However, the high degree vertices that exists are only slightly less likely to link to one another. Although we can only guess, this appears to be another consequence that stems from the contradicting needs for a politicians to be isolated and well-connected at the same time.



































Clustering coefficient and cyclic coefficient on the other hand, were not particularly helpful metrics. The purpose of money laundering is to create convoluted cycles of transactions. As such, it would be rare to find short cycles of only three parties or cycles that exist with direct neighbors in the network. This applies to both politicians and non-politicians. There seems to be little difference in the values produced by page rank and reciprocity for both our two people officer sets making these poor features for our classifier.

### D. Motifs

Recursive decomposition of networks is a widely used approach in network analysis to factorize the complex structure of real-world networks into small sub-graph patterns of size  $k$  nodes. These patterns are called graphlets or motifs. We implemented a fast and efficient algorithm in Python to compute the frequencies of these structures for nodes of size  $k = 3, 4$ . For each edge, we count a few graphlets, and with these counts along with the combinatorial arguments, we obtain the exact counts of others in constant time. We further divided this into the following macro and micro level features:

- Frequency of 6 motifs ( out of the 15) for each node in the network as computed by Algorithm 4 and Algorithm 5. The combinatorial arguments made in [1] are motif counts for full graphs - only 6 out of these are specific

TABLE I. SUMMARY OF GRAPHLET NOTATION

Summary of the notation and properties for the graphlets of size $k = \{2, 3, 4\}$ . Note that $\rho$ denotes density, $\Delta$ and $\bar{d}$ denote the max and mean degree, whereas assortativity is denoted by $r$ . Also, $ T $ denotes the total number of triangles, $K$ is the max $k$ -core number, $\chi$ denotes the Chromatic number, whereas $D$ denotes the diameter, $B$ denotes the max betweenness, and $ C $ denotes the number of components. Note that if $ C  > 1$ , then $r$ , $D$ , and $B$ are from the largest component.													
Graphlet	Description	Complement	$\rho$	$\Delta$	$\bar{d}$	$r$	$ T $	$K$	$\chi$	$D$	$B$	$ C $	
(k = 4) — GRAPHLETS													
CONNECTED		$g_{41}$	4-clique		1.00	3	3.0	1.00	4	3	4	1	0
		$g_{42}$	4-chordal cycle		0.83	3	2.5	-0.66	2	2	3	2	1
		$g_{43}$	4-tailed triangle		0.67	3	2.0	-0.71	1	2	3	2	2
		$g_{44}$	4-cycle		0.67	2	2.0	1.00	0	2	2	2	1
		$g_{45}$	3-star		0.50	3	1.5	-1.00	0	1	2	2	3
		$g_{46}$	4-path		0.50	2	1.5	-0.50	0	1	2	3	2
DISCONNECTED		$g_{47}$	4-node-1-triangle		0.50	2	1.5	1.00	1	2	3	1	0
		$g_{48}$	4-node-2-star		0.33	2	1.0	-1.00	0	1	2	2	1
		$g_{49}$	4-node-2-edge		0.33	1	1.0	1.00	0	1	2	1	0
		$g_{410}$	4-node-1-edge		0.17	1	0.5	1.00	0	1	2	1	0
		$g_{411}$	4-node-independent		0.00	0	0.0	0.00	0	0	1	$\infty$	0
													4
(k = 3) — GRAPHLETS													
		$g_{31}$	triangle		1.00	2	2.0	1.00	1	2	3	1	0
		$g_{32}$	2-star		0.67	2	1.33	-1.00	0	1	2	2	1
		$g_{33}$	3-node-1-edge		0.33	1	0.67	1.00	0	1	2	1	0
		$g_{34}$	3-node-independent		0.00	0	0.00	0.00	0	0	1	$\infty$	0
(k = 2) — GRAPHLETS													
		$g_{21}$	edge		1.00	1	1.0	1.00	0	1	2	1	0
		$g_{22}$	2-node-independent		0.00	0	0.0	0.00	0	0	1	$\infty$	0

to edges. Given that some motifs are also present in other motifs, we further tuned these frequencies by normalizing them in two ways - by the total number of motifs for each node, and by the degree of each node.

- Cumulative frequencies of each motif for the neighbors of each node. These values are directly proportional to the degree of the node, and the degrees of a nodes neighbors, hence we decided to normalize these by the product of these two values.
- Frequency of motif in the ego-network of every nodes. The complete algorithm and proofs for the combinatorial arguments are provided in [1].

#### IV. CLASSIFICATION

##### A. Performance Metrics

In our case a simple default strategy of guessing the majority class would give a predictive accuracy of 99% as shown below. Hence, it is clearly not appropriate in this situation. The nature of the application requires a fairly high rate of correct detection in the minority class and allows for a error in the majority class. In order to achieve this we used two such metrics to get more insight into the accuracy of the model as compared to traditional classification accuracy.

- F1 Score (or F-score):** It considers both the precision  $p$  and the recall  $r$  of the test to compute the score:  $p$  is the number of correct positive results divided by the number of all positive results, and  $r$  is the number of correct positive results divided by the number of positive results that should have been returned. The F1 score can be interpreted as a weighted average of the precision and

#### Algorithm 4 Edge Motifs

```

1: procedure EDGE MOTIF( $G(V, E)$ )
2:   for all  $e \in E(G)$  do
3:     Initialize Array  $X$ 
4:      $Star_u = \emptyset$   $Star_v = \emptyset$   $Trie_e = \emptyset$ 
5:     for all  $w \in N(u)$  do
6:       if  $w = v$  then continue
7:       end if
8:       Add  $w$  to  $Star_u$  and set  $X(w) = 1$ 
9:     end for
10:    for all  $w \in N(v)$  do
11:      if  $w = u$  then continue
12:      end if
13:      if  $X(w) = 1$  then
14:        Add  $w$  to  $Trie_e$  and set  $X(w) = 2$ 
15:        Remove  $w$  from  $Star_u$ 
16:      else Add  $w$  to  $Star_v$  and set  $X(w) = 3$ 
17:      end if
18:    end for
19:     $g_{31} = |Trie_e|$ 
20:     $g_{32} = (|Star_u| + |Star_v|)$ 
21:     $g_{41} = \text{CLIQUECOUNT}(X, Trie_e)$ 
22:     $g_{44} = \text{CYCLECOUNT}(X, Star_u)$ 
23:     $g_{42} = \binom{Trie_e}{2} - g_{41}$ 
24:     $g_{46} = |Star_v| * |Star_v| - g_{44}$ 
25:  end for
26: end procedure

27: procedure CLIQUECOUNT( $X, Trie_e$ )
28:    $cliq_e = 0$ 
29:   for all  $w \in Trie_e$  do
30:     for all  $r \in N(w)$  do
31:       if  $X(r) = 2$  then  $cliq_e + = 1$ 
32:       end if
33:     end for
34:      $X(w) = 0$ 
35:   end for
36:   return  $cliq_e$ 
37: end procedure

38: procedure CYCLECOUNT( $X, Star_u$ )
39:    $cyc_e = 0$ 
40:   for all  $w \in Star_u$  do
41:     for all  $r \in N(w)$  do
42:       if  $X(r) = 3$  then  $cyc_e + = 1$ 
43:       end if
44:     end for
45:      $X(w) = 0$ 
46:   end for
47:   return  $cyc_e$ 
48: end procedure

```

---

**Algorithm 5** Node Motifs

---

```
1: procedure EDGE TO NODE MOTIF( $motif(E)$ )
2:   for all  $v \in V(G)$  do
3:      $motif(v) \leftarrow 0$ 
4:     for all  $e \in E(v)$  do
5:        $motif(v) \leftarrow motif(e)$ 
6:     end for
7:      $nmotif(v) \leftarrow nmotif(v)/degree(v)$ 
8:   end for
9: end procedure
```

---

---

**Algorithm 6** Normalized Neighbor Motifs

---

```
1: procedure NEIGHBOR MOTIF( $G(V,E)$ )
2:   for all  $v \in V(G)$  do
3:      $nmotif(v) \leftarrow 0$ 
4:     for all  $u \in N(v)$  do
5:        $nmotif(v) \leftarrow nmotif(v) + motif(u)$ 
6:     end for
7:      $\mu \leftarrow degree(v) * k_{nn,v}$ 
8:      $nmotif(v) \leftarrow nmotif(v)/\mu$ 
9:   end for
10: end procedure
```

---

recall, where an F1 score reaches its best value at 1 and worst at 0.

- ii. *Kappa (or Cohen's kappa)*: Measures the agreement between two raters who each classify N items into C mutually exclusive categories. It is given by

$$\kappa = 1 - \frac{1 - p_0}{1 - p_e} \quad (1)$$

where  $p_0$  is the relative observed agreement among raters, and  $p_e$  is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly saying each category. If the raters are in complete agreement then  $\kappa = 1$ . If there is no agreement among the raters other than what would be expected by chance (as given by  $p_e$ ),  $\kappa \leq 0$ .

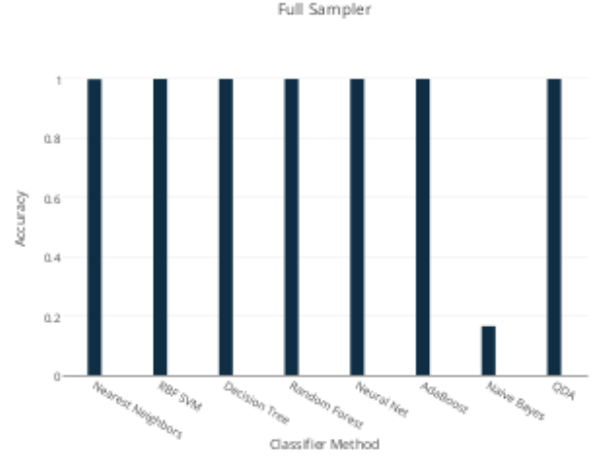


Fig. 17. Accuracy with Complete dataset

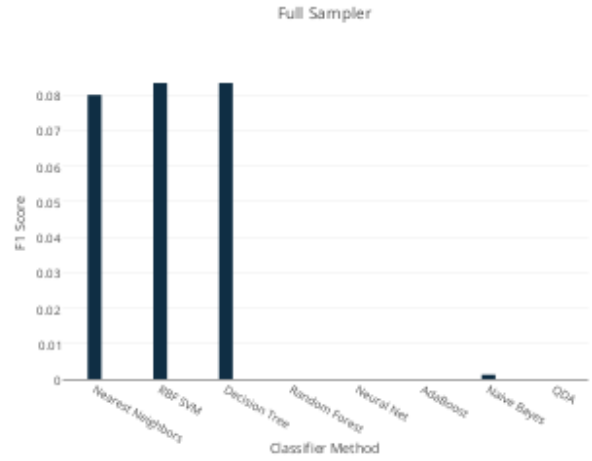


Fig. 18. F1-Scores of models on Complete dataset

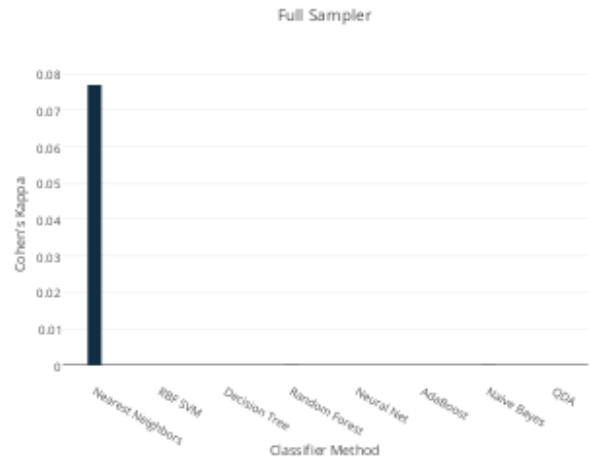


Fig. 19. Cohen's Kappa value of models on Complete dataset



## B. Sampling

Over and under-sampling methodologies have received significant attention to counter the effect of imbalanced data sets. Various studies in imbalanced datasets have used different variants of over and under sampling, and have presented (sometimes conflicting) viewpoints on usefulness of oversampling versus undersampling [4], and [7]. The random under and over sampling methods have their various short-comings. The random undersampling method can potentially remove certain important examples, and random oversampling can lead to overfitting. We changed the dataset that we use to build our predictive model to have more balanced data. We did this using 3 methods:

- We deleted instances from the non-politicians class ( reducing it to 100 )

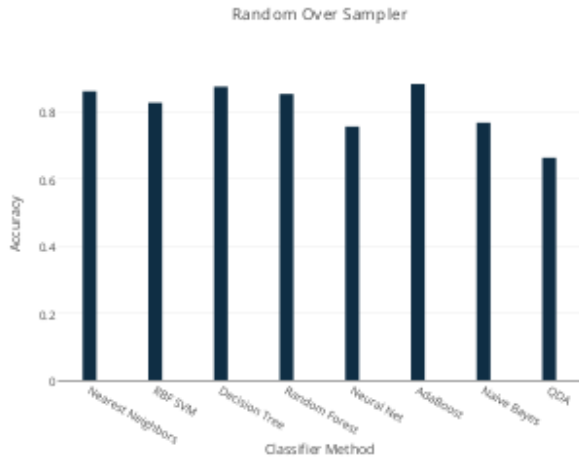


Fig. 20. Classification accuracy for Oversampled data

- We added instances from the politicians class ( increasing them to 500 )

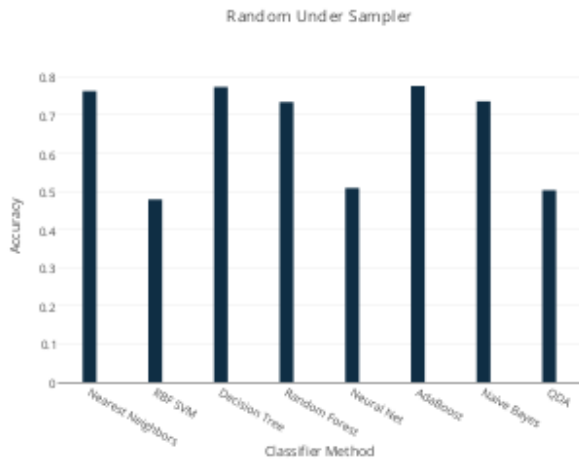


Fig. 21. Classification accuracy for Undersampled data

- Tested different resampled ratios (e.g. instead of a 1:1 ratio in a binary classification problem, tried other ratios

1:100)

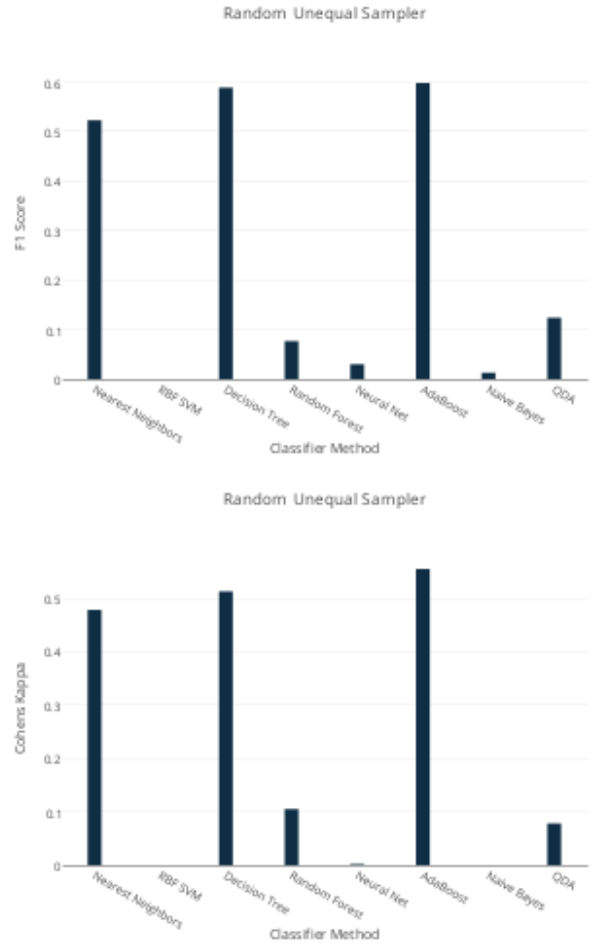


Fig. 22. F1-score & Kappa value of models on moderately imbalanced dataset

## V. EXPERIMENT METHODOLOGY

To benchmark our performance and identify the best classifier, we picked a sampling method - Undersampling, Oversampling, UnEqual sampling, a performance metric - Accuracy, F1, Cohens-Kappa, and ran the following classifiers over it.

- Nearest Neighbors
- RBF Support Vector Machines
- Decision Tree
- Random Forest
- Neural Network
- AdaBoost
- Naive Bayes
- QDA

For most of these we chose default parameters, or commonly used ones. We began by creating 20 samples of train/test data and applying these models to those samples. We then compute the average performance of each model ( because the set of non politicians picked randomly during sampling would vary the feature weights & accuracy ) and use that as an estimate of the models goodness for this problem.

## VI. RESULTS & CONCLUSIONS

As you can see in figure 5, without sampling our dataset our models appear to provide an exceedingly high classification accuracy. But when we investigate further using f1-score and cohens kappa ( Fig 6. and Fig 7. ) we realize its assigning the majority class to all nodes. After sampling, Adaboost & Decision Tree's clearly outperform all other models for every sampling method and over all performance metrics. Random Forests & Neural Nets give a fairly good accuracy for under/oversampled data but clearly overfit and perform poorly when provided with a moderately imbalanced dataset. We believe one of the primary reasons for our high accuracy scores could also be the presence of high correlation between the train and test data. This isn't surprising, the politicians labeled in our dataset clearly exhibited similar characteristics for them to be discovered sooner. Cleaner and more consistent data would add some variability and reliability to the models.

## VII. FUTURE WORK

Due to lack of time, we couldn't try everything we wanted to with our data but we believe that exploring the avenues listed below could yield interesting results.

- i. *Completing and cleaning the dataset.* This can be divided into three parts - the primary being identifying all the politician and non-politician nodes in the dataset. Secondly, we believe our network representation would present a more reliable overview if duplicate nodes were judiciously merged. Finally, given the importance geography plays in offshore money laundering, it would be nice to have country information about all the nodes.
- ii. *Ensemble Sampling.* Although under-sampling is a popular method in dealing with class-imbalance problems, (since its super efficient) many majority class examples are ignored. We think by sampling several subsets from the majority class, training a learner using each of them, and combining the outputs of those learners we can use all our information to get more reliable( if not accurate ) results. Another method of creating an ensemble of samples could be by training the learners sequentially, where in each step the majority class examples which are correctly classified by the current trained learners are removed from further consideration. These methods are further described in [7]
- iii. *Penalized models.* Another interesting model to experiment would be using the imbalanced dataset and just assigning a highest cost to misclassifying the minority class. This could be a way to overcome the effect of class imbalance that pulls down the accuracy of models such as Random Forest and Neural Nets.

## REFERENCES

- [1] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. Efficient graphlet counting for large networks. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 1–10. IEEE, 2015.
- [2] Daniel A. Schult Aric A. Hagberg and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. *Proceedings of the 7th Python in Science Conference (SciPy2008)*, page 11–15, August 2008.
- [3] Thomas Aynaud. Community detection for networkx. <http://perso.crans.org/aynaud/communities/>, 2009–2016.
- [4] Jason Brownlee. 8 tactics to combat imbalanced classes in your machine learning dataset, 2015.
- [5] L da F Costa, Francisco A Rodrigues, Gonzalo Travieso, and Paulino Ribeiro Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in physics*, 56(1):167–242, 2007.
- [6] Iain. Exploring the panama papers network, 2016.
- [7] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.
- [8] The International Consortium of Investigative Journalists. Frequently asked questions: Learn more about the data in the icij offshore leaks database, 2016.
- [9] The International Consortium of Investigative Journalists. Giant leak of offshore financial records exposes global array of crime and corruptionn, 2016.
- [10] Eric W. Weisstein. Tree. From MathWorld—A Wolfram Web Resource.
- [11] Wikipedia. Panama papers, 2016.