

PAPER • OPEN ACCESS

Machine learning-based prediction of Q -voter model in complex networks

To cite this article: Aruane M Pineda *et al* *J. Stat. Mech.* (2023) 123402

View the [article online](#) for updates and enhancements.

You may also like

- [Ordering dynamics of the multi-state voter model](#)
Michele Starnini, Andrea Baronchelli and Romualdo Pastor-Satorras
- [Threshold \$q\$ -voter model with signed relationships](#)
Zhen Lou and Long Guo
- [Noisy voter model for the anomalous diffusion of parliamentary presence](#)
A Kononovicius

PAPER: Interdisciplinary statistical mechanics

Machine learning-based prediction of Q -voter model in complex networks

Aruane M Pineda^{1,2,*}, Paul Kent², Colm Connaughton^{2,3}
and Francisco A Rodrigues¹

¹ Institute of Mathematical and Computer Sciences, University of São Paulo, São Carlos, São Paulo, Brazil

² Mathematics Institute, University of Warwick, Coventry, United Kingdom

³ London Mathematical Laboratory, London, United Kingdom

E-mail: aruane.pineda@usp.br

Received 9 June 2023

Accepted for publication 10 October 2023

Published 4 December 2023



Online at stacks.iop.org/JSTAT/2023/123402
<https://doi.org/10.1088/1742-5468/ad06a6>

Abstract. In this article, we consider machine learning algorithms to accurately predict two variables associated with the Q -voter model in complex networks, i.e. (i) the consensus time and (ii) the frequency of opinion changes. Leveraging nine topological measures of the underlying networks, we verify that the clustering coefficient (C) and information centrality emerge as the most important predictors for these outcomes. Notably, the machine learning algorithms demonstrate accuracy across three distinct initialization methods of the Q -voter model, including random selection and the involvement of high- and low-degree agents with positive opinions. By unraveling the intricate interplay between network structure and dynamics, this research sheds light on the underlying mechanisms responsible for polarization effects and other dynamic patterns in social systems. Adopting a holistic approach that comprehends the complexity of network systems, this study offers insights into the intricate dynamics associated with polarization effects and paves the way for investigating the structure and dynamics of complex systems through modern methods of machine learning.

*Author to whom any correspondence should be addressed.



Original Content from this work may be used under the terms of the [Creative Commons Attribution 4.0 licence](https://creativecommons.org/licenses/by/4.0/). Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Keywords: network dynamics, machine learning, communication, supply and information networks

Contents

1. Introduction 2

2. Methods 6

 2.1. Stochastic simulation of Q -voter model6

 2.2. Networks7

 2.3. Network measurements9

 2.4. Machine learning algorithms 10

3. Results 11

4. Conclusions 17

 Acknowledgments 18

 Appendix A. Construction of complex networks 18

 Appendix B. Network measurement details 22

 B.1. Clustering coefficient (C) 22

 B.2. CLC 23

 B.3. BC 23

 B.4. SPL 24

 B.5. Degree PC coefficient (PC) 24

 B.6. IC 25

 B.7. SC 25

 B.8. Approx. current flow betweenness centrality (AC) 26

 B.9. EC 26

 Appendix C. PCA 27

 Appendix D. Grid search hyperparameter tuning 29

 References 30

1. Introduction

Interactions among the components of a complex system have given rise to properties not present in its isolated parts [1]. For instance, the collective behavior of ants in a colony provides a compelling illustration of emergence. While individually following simple rules, ants exhibit complex behaviors such as efficient food foraging, elaborate nest construction, and coordinated defense [2]. Such an emergence phenomenon significantly

extends beyond the natural world, since it also manifests within our society through intricate interactions among agents, groups, and institutions.

A substantial consequence of emergence is social polarization, according to which agents develop increasingly extreme opinions and display diminished tolerance for opposing viewpoints, ultimately leading to societal divisions. Numerous studies have associated the phenomenon with negative outcomes in political contexts, as seen in the recent elections in both Brazil and the United States [3–6]. In Brazil, heightened polarization culminated in a significant event on 8 January 2023, when key institutions in Brasília, the capital of Brazil, were invaded. This event was the result of escalating tensions stemming from polarized political discourse. The Supreme Federal Court, the National Congress building, and the Presidential Palace were among the targeted institutions. Similarly, the United States also faced its own challenges associated with polarization. A notable incident occurred on 6 January 2021, when a crowd stormed the United States Capitol in an attempt to overturn the results of the presidential election. Therefore, the causes and effects of polarization in social networks must be comprehended so that effective communication strategies and social interventions that mitigate its detrimental impact can be designed [7, 8].

Towards a deeper understanding of social polarization, various mathematical models have been developed [9] and the most sophisticated ones have recently considered the dynamics of interactions between agents and their underlying structure. Indeed, consensus models must be simulated in complex networks to be more realistic, since the network topology heavily influences both their dynamics and the final result of consensus generation [10].

Several models, including the Ising model, the Sznajd model, the voter model, the naming game, the bounded confidence model, and the Q -voter model, address complex phenomena stemming from interactions among individuals in social and physical contexts and are adapted for complex networks. Researchers employ these models to identify conditions fostering consensus emergence and network features facilitating the process. Simulations within complex networks are crucial to achieving a more accurate portrayal of consensus formation. The intricate network topology significantly influences model dynamics and consequently impacts consensus generation outcomes [11]. The Ising model, originating from physics, focuses on material magnetization by representing the magnetic orientation of spins in a three-dimensional lattice. The interaction between neighboring spins aims to minimize the system's energy, leading to phenomena like the Ising phase transition [12–15]. In contrast, the Sznajd model explores how similar opinions can influence others. The premise is that people with coinciding opinions are more likely to persuade others, leading to the formation of opinion clusters [16–18]. Meanwhile, the voter model simplifies decision-making in a population, where individuals adopt the majority opinion of their neighbors, illustrating how social influences can drive convergence towards dominant opinions or polarization [9, 19]. The naming game addresses language evolution, where individuals attempt to communicate and reach a consensus on names for concepts, balancing communicative efficiency and linguistic diversity [9, 20–22]. Bounded confidence explores how opinions change through social interactions, assuming people update their opinions only when the difference from others' opinions falls within a specific limit [23, 24]. Finally, the Q -voter model offers an

approach to simulate collective decisions within groups of individuals. In this model, each agent adopts the opinion of one of its randomly selected Q -neighbors. These Q -neighbors represent a subset of neighboring agents, and the extent of this subset, denoted by Q , significantly influences the dynamics of opinion diffusion. This model investigates how connectivity and information exchange between individuals impact consensus formation. By studying how opinions spread through the Q -voter framework, researchers gain insights into the emergence of consensus, polarization, or the coexistence of diverse viewpoints within a population [9, 25, 26]. In [27], researchers investigate the impact of polarization in the three-state Q -voter model, considering limited confidence and noise. By incorporating these factors, the study reveals how agent interactions lead to the formation of groups with divergent opinions, complicating the convergence to a single opinion. Similarly, [28] examines the role of anticonformity and limited confidence in the Q -voter model. This study demonstrates that anticonformity amplifies polarization and emphasizes the coexistence of groups with similar yet distinct opinions, especially when limited confidence is present. Furthermore, [29] introduces a mathematical model that examines the effects of conformity and anticonformity on opinion polarization. This study investigates a similar opinion dynamics model based on the Q -voter, analyzing how the interplay between these behaviors influences the formation of groups with divergent opinions. These collective studies substantially contribute to a deeper understanding of the underlying dynamics of opinion polarization in social contexts.

Empirical investigations have consistently provided compelling evidence that different network topologies exhibit varying degrees of polarization and consensus formation [30–34]. For example, recent studies have shown the adoption of the Q -voter model within modular networks can result in highly polarized public opinions [30]. On the other hand, in scale-free networks, highly connected agents can expedite the process of consensus formation while potentially amplifying extreme polarization [31]. Furthermore, studies have delved into the influence of network clustering, degree distribution, and other network properties on the dynamics of consensus formation [9], highlighting the crucial role of network topology in the development of realistic models for comprehending consensus formation within complex networks. By considering the intricate interplay between network structure and opinion dynamics, researchers can attain a more comprehensive understanding of the factors that shape the emergence of consensus and polarization.

Given the significant influence of network topology on the emergence of consensus, an essential question is whether it is feasible to develop a machine learning model that can forecast dynamic variables based on network properties. Such an inquiry has been widely explored in various fields, including the prediction of both epidemics in human contact networks [35, 36] and synchronization in coupled oscillators [36, 37]. The investigations have not only demonstrated the possibility of forecasting the behavior of dynamic systems from the network topology but also underscored the importance of comprehending the relationship between network structure and dynamics in those

systems as in the recent article by Brooks and Porter [38]. Both our study and the one by Brooks and Porter share a common focus on complex phenomena within social networks. We employ interdisciplinary approaches that integrate complex network theory, system dynamics, and machine learning. Both studies acknowledge the pivotal role of network structure in shaping social dynamics and investigate opinion dynamics within social networks, although with different emphases. While both studies delve into opinion dynamics, our research primarily centers on utilizing machine learning to predict variables based on the Q -voter model. In contrast, Brooks and Porter's research delves into how media exposure influences ideological content within social networks. This distinction underscores the significance of media in their study, while our research places a strong emphasis on network structure and agent interactions.

The application of machine learning algorithms in the study for the prediction of consensus time and frequency of opinion changes in the Q -voter model offers several advantages. Machine learning promotes the capture of complex patterns, learning from historical data, and adaptation to evolving dynamics; it is a powerful tool for uncovering intricate relationships and enhancing predictive accuracy. Moreover, its use in the context of Q -voter represents a novel approach, pushing the boundaries of traditional analysis and providing new insights into the mechanisms driving opinion dynamics in complex social systems. The consensus is a significant metric that indicates the level of agreement among agents in a network. Conversely, the frequency of opinion changes reflects a network's ability to maintain its beliefs and showcases the level of volatility in the system. Understanding and mitigating the effects of polarization in complex network systems is of utmost importance, as it can significantly impact both the consensus formation process and the stability of opinions within a network. Both metrics play a crucial role in the comprehension of the behavior of social systems and offer insights into the factors contributing to stability or instability within such systems [39].

This study provides valuable insights into the intricate relationship between network structure and social dynamics, highlighting the potential of complex network measures for analyzing dynamic systems. Additionally, it demonstrates the effectiveness of complex network structures in accurately predicting the consensus time and frequency of opinion changes within the Q -voter model using machine learning algorithms. The significance of each network feature in these predictions was evaluated, revealing the clustering coefficient (C) and information centrality (IC) as the most influential measures for predicting these outcomes. Furthermore, the robustness of these predictions was tested using three distinct initialization methods in the Q -voter model, specifically assessing the model's behavior when initialized with high degree, low degree, and a random selection of agents with positive opinions.

The article is organized as follows: section 2 is divided into four parts. The first part introduces the simulated Q -voter model, section 2.2 describes the investigated networks, section 2.3 explains the network measurements, and section 2.4 presents the machine learning algorithms used for prediction. Section 3 provides the results, and section 4 is dedicated to relevant observations and conclusions.

2. Methods

2.1. Stochastic simulation of Q -voter model

In the context of the Q -voter model, a group of Q agents (Q -voters) influences the opinion of a single agent. This interaction determines the number of neighbors considered by an agent for decision-making, as dictated by the parameter Q . This model is particularly interesting for studies of social dynamics since it captures the impact of group influence, conformity, and social reinforcement on opinion dynamics. Furthermore, it exhibits a rich phase-transition behavior, depending on the value of Q and network topology, leading to various outcomes such as consensus, fragmentation, and coexistence of opinions [40–44]. Introduced in [45], its applicability extends to all integer-values of $Q > 0$, meaning that Q can encompass a range of values greater than 0. Furthermore, by setting $Q = 1$, we directly return to the standard voter model. Within this framework, the possibility of repetition is considered, implying that a specific neighbor can be selected multiple times. Thus, when Q is greater than the number of neighbors (the degree of a node), the opinion of the same neighbor will be taken into account more than once.

Consider a network of N voters (also known as agents, nodes, spins, or individuals). Each is defined by a single dynamical binary variable $s(x, t) = j$, where $j = +1$ or $j = -1$, $x = 1, \dots, N$, and t represents time. From a social standpoint, $s(x, t)$ represents a two-point psychometric scale (yes/no, agree/disagree) opinion of an agent placed at node x at time t on a particular subject.

The initial fraction of agents with positive opinions ($p+$) is fixed at the beginning of the simulation and randomly distributed to the network nodes. Parameter ϵ represents the probability of an agent x acting independently of their neighbors, indicating their unwillingness to yield to group pressure. Consequently, $(1 - \epsilon)$ represents conformity, influencing the likelihood of an agent adopting the majority opinion of her/his Q neighbors. Note that the individual opinion of the selected agent x is not taken into account in the probability of opinion change or retention in the dynamics. Table 1 shows the fixed parameters of Q -voter, including the number of nodes in complex networks ($N = 1000$), probability of an agent acting independently ($\epsilon = 0.01$), an initial fraction of agents with positive opinions ($p+ = 0.20$), and the number of neighbors ($Q = 2$). The value of β represents the probability of an agent changing their opinion to the opposite when there is no consensus among their neighbors.

The parameters were fixed toward establishing a consistent baseline for our machine learning-based prediction of consensus time and frequency of opinion changes. Consensus time is the relaxation time of a finite-size system needed to approach a stationary state. By keeping them constant, our exploration can focus on the impact of other variables and a more thorough analysis of our machine learning model's predictive performance regarding the desired outcomes can be conducted. The initial percentage of agents selected was modified to have a positive opinion in three ways: through random agent selection, and by selecting high- and low-degree agents.

Table 1. Q -voter model parameters with default values.

Parameter	Default Value	Description
N	1000	Number of nodes
ϵ	0.01	Probability of an agent acting independently (non-conformity)
Q	2	Neighbor consideration for decision-making
p_+	0.20	Initial fraction of agents with positive opinions
β	0.20	Probability to alter opinion with no consensus among neighbors

Algorithm 1 exemplifies the stochastic simulation, followed by an illustration in figure 1 of the model. In other words, all agents have a binary opinion, represented here by the colors red and blue. Suppose an agent has a red opinion; then, their opinion can be altered based on the following social response: the probability of non-conformity, i.e. reluctance to yield to group pressure, with a probability ϵ , of changing their opinion. Alternatively, conformity ($1-\epsilon$) represents the probability of behaving like their neighbors. If the neighbors share a consensus, meaning they all have the same opinion, the agent will switch to the blue color or remain in the red color. However, if there is no consensus among the neighbors, with a probability β , the agent will switch to the blue color, and with a probability of $1-\beta$, they will maintain their opinion.

Algorithm 1. Q -voter model algorithm.

1. Initialize a complex network of size N representing the agents
2. Assign each agent a binary variable, $s(x,t)$ with $x \in [1,N]$ at time t , whose values $+1$ or -1 representing two opposing opinions ($j = +1$ or $j = -1$)
3. **for** each time step t **do**
4. Randomly select an agent x
5. Randomly choose Q neighbors of agent x (allowing for repetition)
6. **if** all Q neighbors have the same state **then**
7. agent x takes the value of the Q neighbors
8. **else**
9. agent x flips with probability ϵ
10. **end if**
11. Update the time
12. **end for**

2.2. Networks

Nine complex network measures were examined, as discussed in section 2.3. The analysis involved eight distinct topological structures, including Erdős–Rényi [47], Barabási–Albert linear [48], Barabási–Albert nonlinear with $\alpha = 0.5$ and $\alpha = 1.5$ [49], Lancichinetti–Fortunato–Radicchi (LFR) graphs [50], Watts–Strogatz [51], Waxman [52], and path graph [53]. The Erdős–Rényi network model is generated by randomly adding connections between nodes with a uniform probability. In contrast, the non-linear Barabási–Albert model is constructed iteratively, incorporating preferential attachment

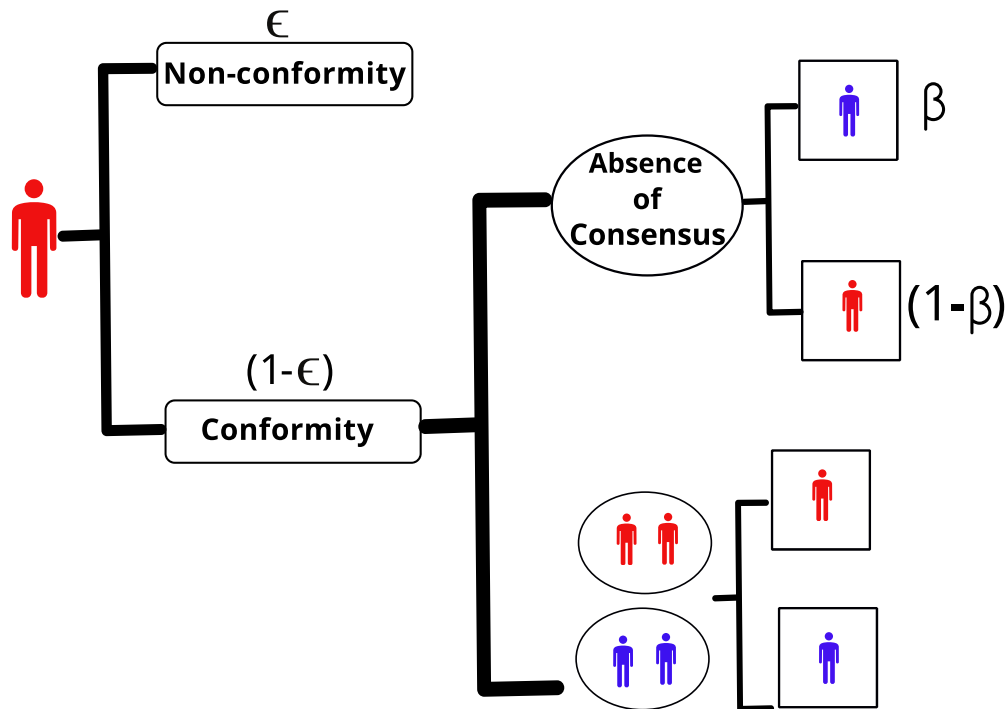


Figure 1. Illustration of the Q -voter model: All agents have a binary opinion, represented here by the colors red and blue. Suppose an agent has a red opinion; then, their opinion can be altered based on the following social response: the probability of non-conformity, i.e. reluctance to yield to group pressure, with a probability ϵ of changing their opinion. Alternatively, there is conformity $(1-\epsilon)$, which represents the probability of acting like their neighbors. If the neighbors have a consensus, meaning they all share the same opinion, the agent will switch to the blue color or remain in the red color. However, if there is no consensus among the neighbors, with a probability β , the agent will switch to the blue color, and with a probability of $1-\beta$, they will maintain their opinion. The figure was created by the authors and is based on [46]. Reproduced with permission from [46].

of new nodes to existing ones through a non-linear function that considers the node's connections. The LFR model is widely employed for creating networks with realistic community structures, assigning nodes to communities based on degree and community size distributions, and establishing connections that consider both intra- and inter-community links. The Watts–Strogatz model introduces the concept of small-world networks by randomly rewiring a portion of links in a regular lattice. A path graph is a specific type of graph in graph theory that consists of a linear sequence of connected nodes, where each node is linked to the next node in the sequence by a single edge. This creates a structure resembling a straight line of nodes, and it is often used as a simple representation of an ordered sequence of elements or events. A path graph is created by defining the nodes in the desired order and connecting them sequentially with edges. Lastly, the Waxman model takes into account geographic proximity and node attractiveness to determine the formation of connections, considering both physical distances

and random appeal. For each of the mentioned networks, appendix A provides details of the Python functions used, and 100 unique instances were generated, with each network consisting of $N = 1000$ nodes and an average degree ranging from 9 to 10, that is, the available dataset comprises 800 instances of complex networks (denoted as i).

2.3. Network measurements

To effectively capture and explain the dataset's predominant variability, a visual representation of the principal component analysis (PCA) plot is provided in appendix C. This PCA plot serves as a powerful tool for gaining insights into the underlying patterns and structures within the dataset. Subsequently, the Q-voter model was simulated in each of these structures to measure the time taken to reach consensus (Y_i) and the total number of opinion changes that occurred in the model (C_i). It is hypothesized that both Y_i and C_i can be predicted using a feature vector derived from the network structure, denoted as $\mathbf{X}_i = X_{i1}, X_{i2}, \dots, X_{ik}$, where X_{ik} represents the k -th measure extracted from network i . The subsequent explanation primarily focuses on the prediction of Y_i , although the same process is applied to the prediction of C_i . Therefore, the learning model is defined by

$$Y_i = f(\mathbf{X}_i) + \delta. \quad (1)$$

The goal is to infer the function $f()$ that relates Y_i to the network measures. Estimating Y_i is treated as a regression problem, where δ represents a random error term independent of \mathbf{X}_i , following a normal distribution with a mean of zero and a standard deviation of σ . While feature selection and model comparison algorithms can be used to identify components of \mathbf{X}_i that contribute to predicting Y_i , this study employed conventional network measures which are presented in table 2.

The first measure utilized in this study was the clustering coefficient (C), a local measure, which quantifies the extent to which nodes in a network tend to form tightly connected clusters. It assesses the likelihood of two neighbors of a node being connected, reflecting the local clustering patterns within the network [51]. Closeness centrality (CLC), another local measure, was employed to calculate the proximity of a node to all other nodes in the network. It reflects the average distance between a node and all other nodes, indicating the efficiency of information or resource flow within the local neighborhood of a node [54]. Betweenness centrality (BC) is a measure that identifies nodes acting as critical intermediaries in the network. BC quantifies the extent to which a node lies on the shortest paths between other pairs of nodes, thus indicating its influence over the flow of information or resources within its vicinity [55]. The shortest path length (SPL) measures the minimum number of edges required to traverse between two nodes in the network, providing insights into network connectivity and the efficiency of information or resource transfer within local regions of the network [56]. Degree Pearson correlation coefficient (PC) examines the correlation between the degrees of connected nodes, capturing the tendency of nodes with similar degrees to connect and indicating the presence of assortativity or disassortativity within the network [57]. IC assesses the importance of a node based on its ability to control the flow of information in the network, considering the number of shortest paths that pass through the node [58].

Table 2. Measures of complex networks used here.

	Network Measures	Acronym
X_1	Clustering coefficient	C
X_2	Closeness centrality	CLC
X_3	Betweenness centrality	BC
X_4	Shortest path length	SPL
X_5	Degree Pearson correlation coefficient	PC
X_6	Information centrality	IC
X_7	Subgraph centrality	SC
X_8	Approx. Current flow betweenness centrality	AC
X_9	Eigenvector centrality	EC

Subgraph centrality (SC) measures the importance of a node within its local subgraph by considering the closed walks that pass through the node, capturing its influence within specific network neighborhoods [59]. Approximate Current Flow Betweenness Centrality (AC) quantifies the extent to which a node controls the flow of electric current in the network, considering the current paths between all pairs of [60]. Finally, Eigenvector centrality (EC) determines a node’s importance based on its neighboring nodes’ centrality, assigning higher importance to nodes connected to other important nodes and capturing the concept of influence [61]. Such measures, collectively used here, provide valuable insights into complex network structures, connectivity, efficiency, influence, and community organization [62]. Details and equations for each of the mentioned measures, along with the Python functions used, are provided in appendix B.

2.4. Machine learning algorithms

The machine learning algorithms utilized are the least absolute shrinkage and selection operator (LASSO), multi-layer perceptron regressor (MLP), random forest (RF), and extreme gradient boosting (XGBoost). Among the several techniques used to improve the proposed machine learning algorithms, nested cross-validation, shuffle, and grid search are highlighted. The former is a multi-round cross-validation procedure adopted in machine learning for model selection and performance assessment [63]. It is a more rigorous model selection and performance evaluation than traditional cross-validation since it reduces the risk of overfitting and provides a more accurate estimate of the model’s performance on unseen data [64]. Its main idea is the existence of an outer loop, which divides the data into training and test sets, and an inner one, which uses cross-validation to determine optimal values for the model’s hyperparameters. Shuffle was employed during nested cross-validation to avoid possible biases in the selection of training and testing data, ensuring the model learned in a balanced way throughout the range of data. Finally, grid search searched for the best model hyperparameters by systematically exploring different combinations of possible values for them. The set of techniques used significantly contributed to the development of a more robust and accurate model. A 5-fold outer shuffle cross-validation and a 5-fold inner cross-validation were also adopted, following similar approaches described in previous studies

[65]. During the inner folds, a grid search hyperparameter optimization was performed—specific details can be found in table D1 in the appendix D.

The coefficient of determination, R^2 , is a metric used to measure how well a regression model fits the data [66]. However, when we add more predictors to the model, R^2 can increase even if these new predictors do not really help explain the variation in the dependent variable. To address this, we use the R^2 adjusted, which considers the number of predictors and penalizes the inclusion of irrelevant ones. This adjustment gives us a more accurate evaluation of how well our model predicts the outcome. In simpler terms, we prefer R^2 adjusted over R^2 because it prevents values from being artificially inflated by including unnecessary predictors. This ensures a more reliable assessment of our model's performance.

The schematic in figure 2 provides an overview of the comprehensive process outlined in this article, which encompasses several steps: (a) generation of complex networks: We generate the eight types of networks under study. (b₁) Calculation of topological measures: In this step, we compute the nine topological measures for all the previously generated complex networks. (b₂) Implementation of the Q -voter model: in this stage, we implement the Q -voter model on each of the complex networks using three distinct initialization methods represented by colored circles: high-degree (purple), low-degree (green), and random selection (orange). This analysis is performed for both Y_i (consensus time) and C_i (frequency of opinion changes). (c) Creation of the dataset: A dataset is constructed containing information from all generated networks. Each row represents a specific network, and the columns contain topological measure calculations. The dataset also includes values for initialization methods (high-degree, low-degree, and random selection) for both Y_i (consensus time) and C_i (frequency of opinion changes). (d) Application of machine learning algorithms: Based on the collected information, machine learning algorithms are used to conduct further analyses and extract significant insights and summary statistics from the generated data.

3. Results

Figure 3 presents boxplots illustrating four machine learning algorithms: LASSO (light brown box), RF (pink box), XGBoost (blue box), and MLP (yellow box) for predicting Y_i . It is worth noting that RF (box 2, pink) and XGBoost (box 3, blue) exhibit the tallest boxes, indicating their tendency to yield higher average adjusted R^2 values compared to the other algorithms. Furthermore, LASSO, RF, and XGBoost consistently produce the best results across all initialization methods, including high degree, low degree, and random selection. These three algorithms were selected for further analysis to predict C_i , and the results are presented in figure 4. Remarkably, LASSO (box 1, light brown) and RF (box 2, pink) emerge as the tallest boxes, suggesting their inclination to yield higher average adjusted R^2 values compared to XGBoost. For this reason, we chose the RF algorithm, which stood out as the best in both figures, to illustrate the following figures (figures 5 and 6).

In figures 5(A) and (B), we refer to the variables Y_i and C_i , respectively, and illustrate the relationship between predicted values (\hat{y}) on the y -axis and their corresponding original values (y) on the x -axis. Each point in the plot represents a specific data

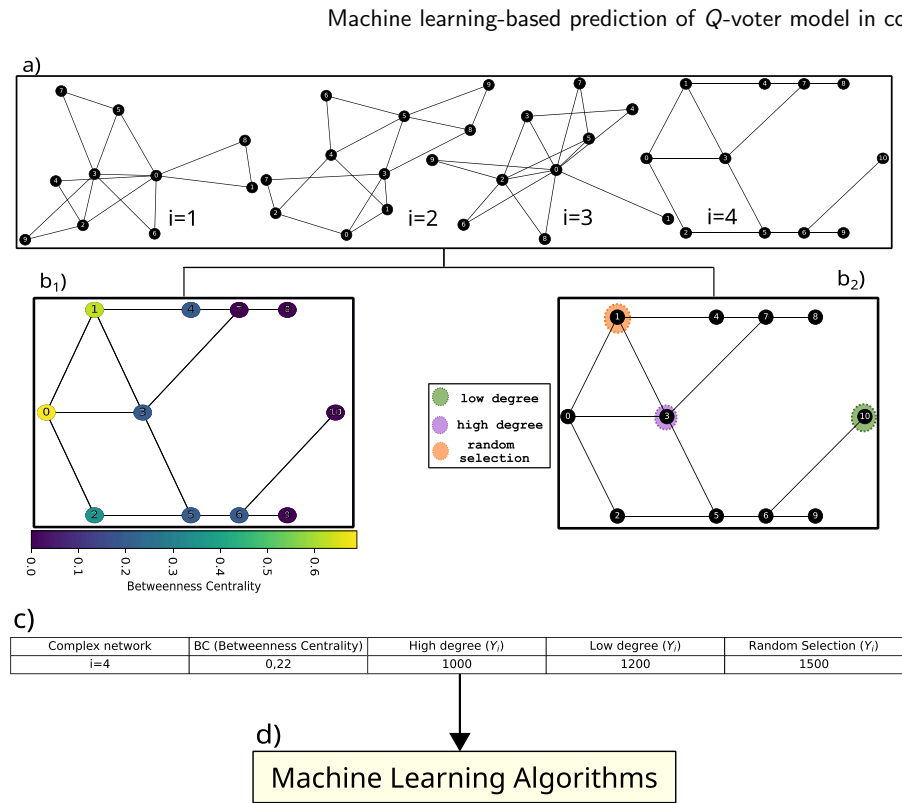


Figure 2. Schematic Overview of the process outlined in this article. The process involves several key steps: (a) Generation of complex networks: In this initial step, we create complex networks for analysis. In this illustrative example, we generate four networks labeled as $i = 1$, $i = 2$, $i = 3$, and $i = 4$, each consisting of a total of 10 nodes. It's worth noting that in our article, we generate a set of 800 complex networks. b_1) Calculation of topological measures: In this step, we compute various topological measures for all the previously generated complex networks. However, for the sake of simplification in this illustration, we focus on a single measure: Betweenness Centrality (BC). We apply this calculation to one of the four networks, specifically network $i = 4$. b_2) Implementation of the Q -voter model: In this stage, we implement the Q -voter model on each of the complex networks using three distinct initialization methods represented by colored circles: high-degree (purple), low-degree (green), and random selection (orange). This analysis is performed for both Y_i (consensus time) and C_i (frequency of opinion changes). For the sake of simplification, we select only network $i = 4$ to illustrate this process. c) Creation of the dataset: In this step, we construct a dataset that contains information from all the generated networks. Each row of the table represents a specific network, and the columns contain the calculations of topological measures for these complex networks. Additionally, we include the corresponding values for initialization methods (high-degree, low-degree, and random selection) regarding Y_i and C_i . For illustration purposes, we present information only for network $i = 4$, including BC and Y_i . However, in the full article, our table encompass 800 rows and 15 columns, comprising nine topological measures, along with three variations of initialization methods for Y_i and C_i . d) Application of machine learning algorithms: Finally, based on the gathered information, we apply machine learning algorithms to conduct further analyses and obtain significant insights and summary statistics from the data generated in the previous steps.

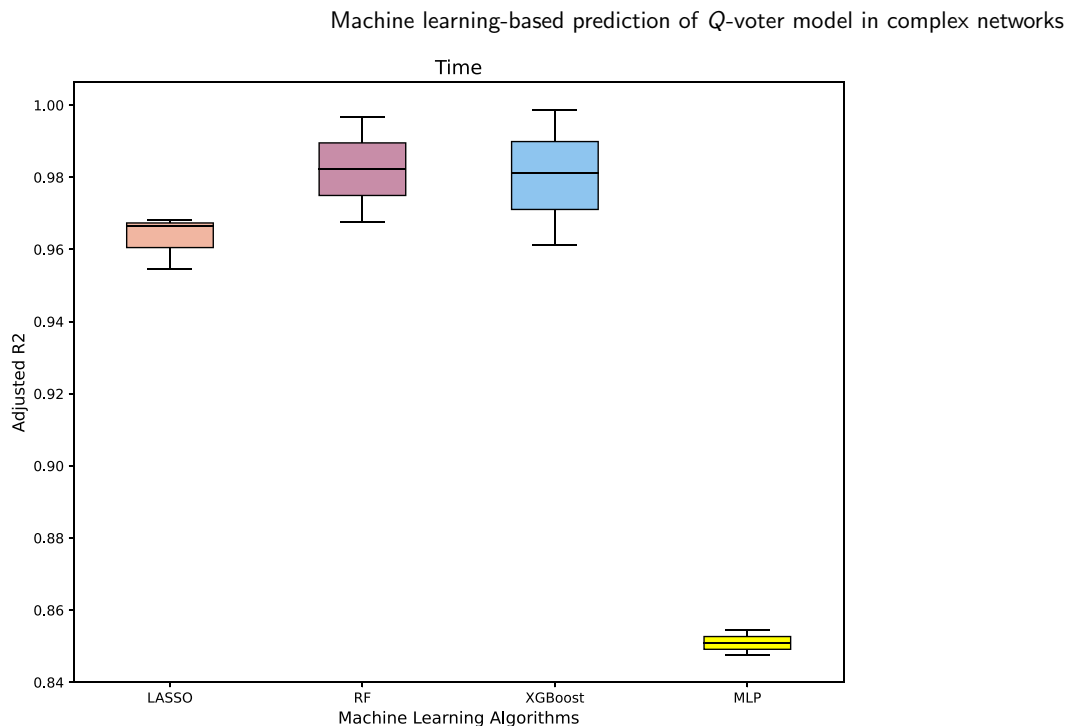


Figure 3. Each boxplot represents the distribution of adjusted R2 values for the corresponding machine learning algorithms (LASSO, RF, XGBoost, and MLP), considering different initialization methods (high degree, low degree, and random selection) to predict Y_i . Among the algorithms, box 2 and box 3 correspond to the RF and XGBoost algorithms, respectively, and show the highest adjusted R2 values. This indicates that, on average, the RF and XGBoost algorithms outperform the other algorithms (LASSO and MLP) in terms of predictive accuracy.

instance, where the x-coordinate indicates the actual value, and the y-coordinate represents the predicted value. The red dotted line represents a linear regression model, which provides an approximation of the overall trend in the data, aiding in the visualization of our model's predictive performance. For Y_i (figure 5(A)), we calculated Pearson's correlation coefficients, resulting in values of 0.998 for high-degree initialization (purple dots), 0.991 for low-degree initialization (green dots), and 0.990 for random selection (orange dots). Additionally, we computed the adjusted R2 values, which were 0.996, 0.982, and 0.968, respectively, for the same initialization methods. For C_i (figure 5(B)), we also calculated Pearson's correlation coefficients, yielding values of 0.999 for high-degree initialization, 0.991 for low-degree initialization, and 0.991 for random selection. The adjusted R2 values were 0.997, 0.983, and 0.945, respectively. These results underscore the correlations observed between the original and predicted values for both Y_i and C_i , regardless of the initialization method used.

The RF algorithm assessed the input variables (network features) in our model. It evaluates the significance of variables by observing the improvement they provide to the model when incorporated into decision trees. The prioritization of network features based on their average importance across different initialization methods, as depicted in figure 6, provides valuable insights into their predictive capabilities. In this analysis, the

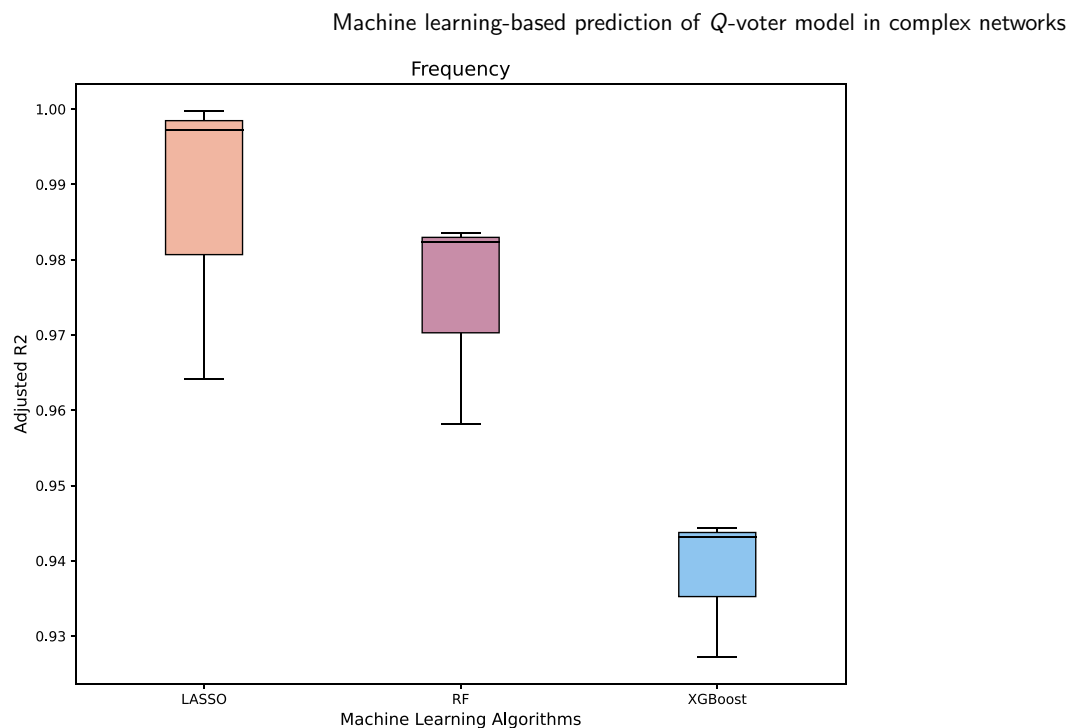


Figure 4. Each boxplot represents the distribution of adjusted R^2 values for the corresponding machine learning algorithm (LASSO, RF, and XGBoost), considering different initialization methods (high degree, low degree, and random selection) to predict C_i . Box 1, which corresponds to the LASSO algorithm, is the highest. This indicates that, on average, the adjusted R^2 values for the LASSO algorithm are higher compared to the other algorithms (RF and XGBoost) considered.

features were ranked according to their average importance, considering three initialization methods: high degree (purple bar), low degree (green bar), and random selection (orange bar). Upon analyzing the bar chart (figure 6), it becomes apparent that network features with higher average importance occupy the top positions. Notably, when attempting to predict Y_i , the clustering coefficient (C) emerges as the most significant measure (figure 6(A)). This indicates that the network's structure, particularly the formation of cohesive groups, plays a crucial role in the speed of consensus attainment within the Q -voter model. In terms of C_i , IC stands out as the most relevant network measure (figure 6(B)). This suggests that the dissemination and influence of information within the network play a fundamental role in the dynamics of opinion changes. These network measures play vital roles in predicting different aspects of the Q -voter model. These inferences underscore the significance of different network aspects concerning the various phenomena under study. While the C focuses on consensus formation, IC pertains to opinion changes. These findings offer valuable insights for comprehending and forecasting the behavior of voter models in broader contexts. In contrast, measures such as EC, Degree PC coefficient (PC), and SC do not exhibit significant predictive capabilities in these scenarios.

Also, note that, individually, the CLC (represented by the purple bar in figure 6(A)) becomes more relevant in networks initialized with a high degree of connectivity, while

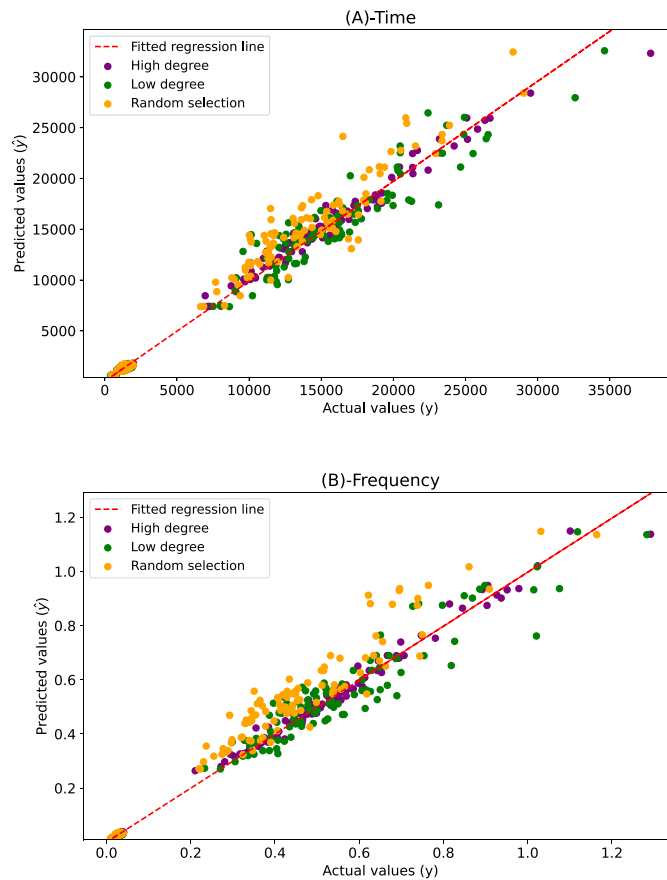
Machine learning-based prediction of Q -voter model in complex networks

Figure 5. Illustration showing the relationship between their corresponding original values (y) and predicted values (\hat{y}) for (A) Time and (B) Frequency regarding the selection of agents with high degree (purple dots), low degree (green dots), and random selection (orange dots) for the initiation of dynamics. This analysis was conducted using the RF algorithm. This analysis was conducted using the RF algorithm.

the AC (indicated by the orange bar in figure 6(A)) is more significant in the randomly initialized networks. CLC gains importance when the dynamics of the Q -voter model are initiated by selecting nodes with a higher degree, as it measures how easily a node can communicate or influence other nodes in the network. When starting the dynamics with high-degree nodes, these high-degree nodes can have a substantial influence on the spread of opinions, and CLC can capture this capacity for influence. Similarly, the significance of the AC centrality measure when initiating the dynamics of the Q -voter model by selecting nodes randomly may be related to the definition of this centrality measure and the dynamics of opinion propagation in the Q -voter model on a network. AC is a measure that reflects the efficiency with which a node can transmit information or influence others in the network. When the dynamics of the Q -voter model are initiated randomly, there is no initial preference for high-degree or low-degree nodes. Therefore, it is crucial to identify nodes that can effectively facilitate the spread of

Machine learning-based prediction of Q-voter model in complex networks

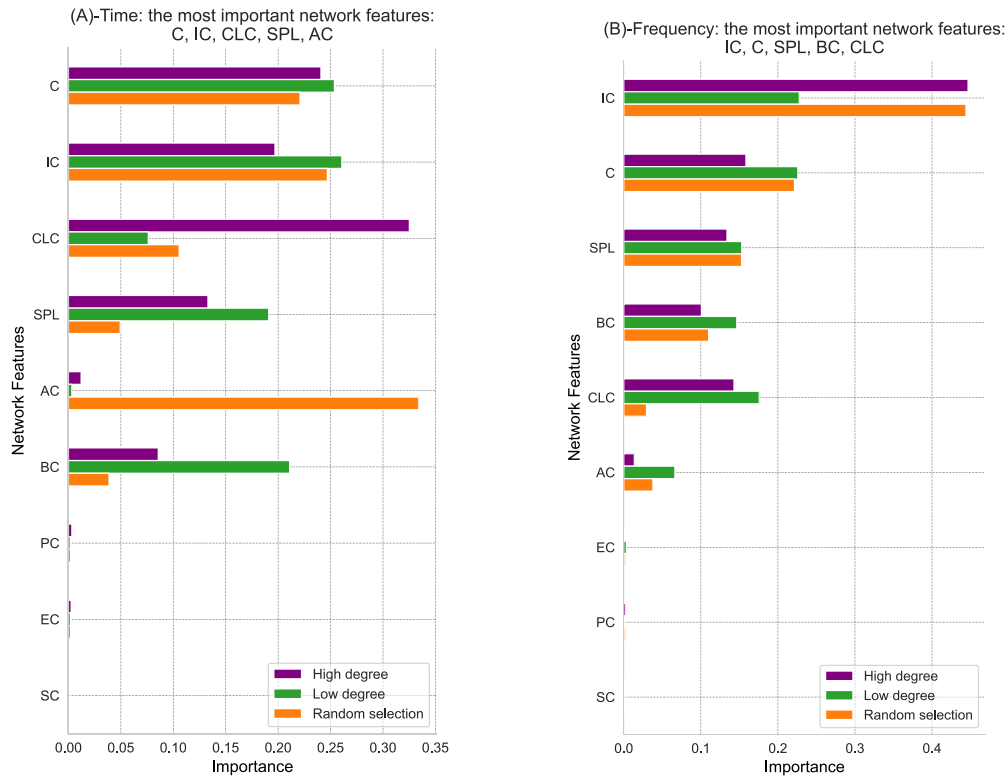


Figure 6. The examination of the most crucial features, which are determined based on the average importance of complex network measures, was conducted to predict both (A) Y_i and (B) C_i using various initialization methods. These methods encompassed the selection of agents with the highest degree (purple bars), lowest degree (green bars), and random selection (orange bars) to initiate the dynamics. Notably, the clustering coefficient (C) and information centrality (IC) consistently emerged as the two most significant measures in both scenarios. This analysis was carried out employing the RF algorithm.

opinions throughout the network, and AC can highlight nodes playing an important role in this regard.

Finally, the learning curve was calculated specifically for the two best results achieved using the high-degree initialization method for Y_i (adjusted $R^2 = 0.996$) and C_i (adjusted $R^2 = 0.997$). By manipulating the size of the training set, the learning curve offers valuable insights into the model’s predictive capabilities [67]. This approach provides the advantage of understanding how the model’s performance improves as more training instances are used, focusing on the most promising initialization methods. The findings depicted in figure D1 indicate that the complete database is not indispensable for achieving the highest level of validation accuracy. Surprisingly, even with a mere 200 training instances, the model demonstrated exceptional performance. These results emphasize that a relatively smaller training set can still yield satisfactory results.

4. Conclusions

In this article, we predicted dynamic variables associated with Q -voter models based on network properties. We verified that the prediction is very accurate and determined which features most contribute to the emergence of polarization. Mainly, we show that the clustering coefficient and IC are the most important measures to quantify these patterns of connections. Moreover, variations in the initialization method, to start the dynamic of the Q -voter model with a positive opinion, were performed to predict consensus of the time (Y_i) and frequency of opinion changes (C_i). Initially, agents were randomly selected, following the original method of the Q -voter model. Subsequently, agents with the highest degree were identified and selected to investigate their potential for strongly influencing the overall opinion dynamics due to their extensive connections. Lastly, agents with the lowest degree of connectivity were considered initiators of the dynamics to explore the potential impact of less influential agents on opinion evolution. Although modifications in the initialization methods of positive opinions affect the results, their impact is relatively small. Indeed, subsequent interactions and information exchange among agents tend to overshadow the influence of the initially selected agents, leading to a consensus of opinions and a limited long-term impact of the initial agent selection. Nonetheless, the exploration of the role of both highly connected and less connected agents provided valuable insights into the complex dynamics of opinion formation and consensus emergence within the Q -voter model.

We found that, regardless of the initialization method used to start the Q -voter model, the initial influence of the selected agents tends to decrease over time. This occurs because, as agents interact and exchange information, their opinions are influenced by others. Over time, opinions begin to converge towards a consensus, and the initial influence of randomly selected, high- or low-connectivity agents becomes equivalent since there is not a significantly superior initialization method over the others; all of them yield equally good results. When we say that the absence of influential agents contributes to a more efficient consensus, we mean that the absence of agents with disproportionate influence in the network means that each agent plays a similar role in shaping the collective opinion. This is important because polarization often occurs when a few extremely influential agents have a disproportionate impact on others' opinions. In the article [68], the authors investigate the influence of highly connected individuals in opinion dynamics. Their research illustrates that a small number of highly connected individuals can significantly influence the polarization of opinions within a network. Furthermore, Sunstein's book 'Republic: Divided Democracy in the Age of Social Media' [69] provides insights into the role of online platforms and highly influential users in shaping public discourse, potentially leading to polarization.

If all agents have similar influence, it is less likely that a few highly influential agents dominate the conversation and pull the collective opinion to opposite extremes. Therefore, the absence of highly influential agents can contribute to a more balanced and less polarized decision-making process.

Expanding our methodology to explore the variance prediction within the Q -voter model can provide further insights into the factors that contribute to diverse outcomes

in social dynamics. Future work in this direction will contribute to a more comprehensive understanding of the complex nature of polarization and its potential implications. By leveraging machine learning algorithms and complex network features, this study can advance research in the field of complex systems and pave the way for future investigations on the dynamics of polarization in various social contexts. Overall, the combination of machine learning algorithms and complex network analysis has the potential to revolutionize our comprehension of social systems, leading to a deeper understanding of human behavior and the development of strategies that promote positive societal outcomes.

Acknowledgments

A M P acknowledges the support of the São Paulo Research Foundation (FAPESP), Grant 2021/13843-2. P K gratefully acknowledges support from the Engineering and Physical Sciences Research Council and Medical Research Council through the Mathematics of Systems I Centre for Doctoral Training at the University of Warwick (Reference EP/L015374/1). F A R acknowledges CNPq (Grant 309266/2019-0) and FAPESP (Grant 19/23293-0) for the financial support given for this research. This research was conducted with the computational resources of the Center for Research in Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP, Grant 2013/07375-0.

Appendix A. Construction of complex networks

In this appendix, parameters involved in network generation are presented in tabular format. The network values were adjusted to ensure that the average degree of all networks fell within the range of 9–10.

- **Erdős-Rényi:** We used the `nx.erdos_renyi_graph` function from NetworkX to create an Erdős-Rényi network [70]. The following table provides information concerning the creation of this network (table A1).

Parameter Descriptions:

- **n:** The number of nodes in the network.
- **p:** The probability for edge creation. The model chooses each of the possible edges with probability p .
- **seed:** Indicator of random number generation state. In our case, it is set to None, which means the default random number generation state is used.
- **directed:** If True, this function returns a directed network. In our case, it is set to False, indicating that the network is undirected.

- **Barabási Linear, Barabási Non-Linear (0.5), Barabási Non-Linear (1.5):** We employed the `graph.Barabasi` function to create networks following the Barabási-Albert model [71]. The subsequent table furnishes specific details regarding the generation of this network (table A2).

Table A1. Parameters for the Erdős-Rényi network model.

Parameter	Value
n	1000
p	0.01
<i>seed</i>	None
<i>directed</i>	False

Table A2. Parameters for the Barabási-Albert Model.

Parameter	Value
n	1000
m	5
<i>outpref</i>	True
<i>directed</i>	False
<i>power</i>	1.0
<i>zero_appeal</i>	1
<i>implementation</i>	psumtree
<i>start_from</i>	None

Parameter Descriptions:

- **n**: The number of nodes in the generated network. In the example, 1000 nodes were created.
- **m**: The number of outgoing edges generated for each node or a list containing the number of outgoing edges for each node explicitly. In the example, each node has 5 outgoing edges.
- **outpref**: A boolean value that determines whether the out-degree of a node affects its citation probability. In the example, it is set to True.
- **directed**: A boolean value that determines whether the generated network is directed. In the example, it is set to False, indicating that the network is undirected.
- **power**: The power constant of the nonlinear model. In the example, the value is 1.0, representing the linear model.
- **zero_appeal**: The attractiveness of nodes with degree zero. In the example, it is set to 1.
- **implementation**: The algorithm used to generate the network. In the example, it is set to psumtree, which uses a partial prefix-sum tree.
- **start_from**: If provided and not None, this parameter uses another network as a starting point for the preferential attachment model. In the example, no starting network is specified (None).

Note that to generate the Barabási networks in a non-linear manner, we modified the power parameter to 0.5 and later to 1.5.

Table A3. Parameters for the LFR Benchmark network model.

Parameter	Value
n	1000
τ_1	3
τ_2	1.5
μ	0.1
<i>average_degree</i>	10
<i>min_degree</i>	None
<i>max_degree</i>	None
<i>min_community</i>	100
<i>max_community</i>	None
<i>tol</i>	1×10^{-7}
<i>max_iters</i>	500
<i>seed</i>	10

- **LFR (LFR Benchmark):** We generated LFR networks using the `LFR_benchmark_graph` function [72]. A table following this one provides information on how this network was generated (table A3).

Parameter Descriptions:

- **n:** Number of nodes in the created network.
- **τ_1 :** Power law exponent for the degree distribution of the created network. This value must be strictly greater than one.
- **τ_2 :** Power law exponent for the community size distribution in the created network. This value must be strictly greater than one.
- **μ :** Fraction of inter-community edges incident to each node. This value must be in the interval $[0, 1]$.
- **average_degree:** Desired average degree of nodes in the created network. This value must be in the interval $[0, n]$.
- **min_degree:** Minimum degree of nodes in the created graph. This value must be in the interval $[0, n]$.
- **max_degree:** Maximum degree of nodes in the created network. If not specified, this is set to n , the total number of nodes in the network.
- **min_community:** Minimum size of communities in the network. If not specified, this is set to `min_degree`.
- **max_community:** Maximum size of communities in the network. If not specified, this is set to n , the total number of nodes in the network.
- **tol:** Tolerance when comparing floats, specifically when comparing average degree values.
- **max_iters (int):** The maximum number of iterations to attempt in order to create community sizes, degree distribution, and community affiliations.

Table A4. Parameters for the Watts-Strogatz network model.

Parameter	Value
n	1000
k	10
p	0.01

Table A5. Parameters for the Waxman network model.

Parameter	Value
n	1000
beta	0.12
alpha	0.1
L	None
domain	(0, 0, 1, 1)
metric	function
seed	None (default)

- **seed(integer, random_state, or None—default):** An indicator of the random number generation state.
- **Watts-Strogatz:** We used the `nx.watts_strogatz_graph` from the NetworkX library to generate a Watts–Strogatz network [73]. The following table contains information about the values of each parameter of this network (table A4).
Parameter Descriptions:
 - **n:** The number of nodes.
 - **k:** Each node is joined with its k nearest neighbors in a ring topology.
 - **p:** The probability of rewiring each edge.
- **Waxman:** We used the `nx.waxman_graph` function from the NetworkX library to generate a Waxman network [74]. The following table contains information to generate this type of network (table A5).
Parameter Descriptions:
 - **n:** Number of nodes.
 - **beta:** Model parameter.
 - **alpha:** Model parameter.
 - **L:** The maximum distance between nodes is set to be the maximum distance between any pair of nodes.
 - **domain:** Domain size, given as a tuple of the form $(x_{min}, y_{min}, x_{max}, y_{max})$.
 - **metric:** Euclidean distance metric is used.
 - **seed (integer, random_state, or None):** Indicator of random number generation state (default is None).

Table A6. Parameters for the Path network model.

Parameter	Value
n	1000

- **Path:** We used the `nx.path_graph` function from the NetworkX library to generate this network [75]. The only parameter for this network is presented in table A6. Note that in our code available on GitHub [here](#) for generating networks, we have added specific lines of code for the path graph to ensure that the average degree falls within the range of 9–10, aligning with the characteristics of the other networks generated.

Parameter Descriptions:

- **n:** Number of nodes.

Appendix B. Network measurement details

B.1. Clustering coefficient (C)

The local clustering coefficient (C) is an important metric in network and graph analysis that quantifies the tendency of neighbors of a node in a network to cluster together. In other words, it measures the degree of connectivity among the direct neighbors of a specific node, which is useful for understanding community structure and cohesion within a network. The mathematical formula for calculating C of a node v in a graph is as follows:

$$C(v) = \frac{2 * E(v)}{k_v * (k_v - 1)} \tag{B1}$$

where:

- $C(v)$ is the local clustering coefficient of node v .
- $E(v)$ is the number of edges between the direct neighbors of v (i.e. the triangles that include node v).
- k_v is the degree of node v , which is the number of direct neighbors it has.

The `transitivity_local_undirected(mode = ‘zero’)` is a Python function commonly employed in network analysis using the Igraph library. This function calculates the C for individual nodes within a graph. It operates in “zero” mode, which specifically considers triangles in the network that share exactly one node with the node being analyzed. The output of this function is a data structure, typically a list or a similar container, containing the C corresponding to each node in the graph. Finally, we calculate the mean to get a final value.

B.2. CLC

Local CLC is a network analysis metric that measures how close a node is to all the other nodes in its local neighborhood within a graph. It quantifies how quickly information can spread from a specific node to its neighboring nodes. Nodes with higher CLC are considered to be more central within their local environment, as they can reach other nodes more efficiently. The mathematical formula for the CLC of a node v is as follows:

$$\text{CLC}(v) = \frac{1}{\sum_{u \neq v} d(v, u)}$$

where:

- $\text{CLC}(v)$ is the local CLC of node v .
- $d(v, u)$ represents the shortest path distance between nodes v and u in the graph. The \sum in the denominator calculates the sum of the shortest path distances from node v to all other nodes u in its local neighborhood.

The **closeness centrality(normalized = True)** function is a commonly used Python function in network analysis using the Igraph library. This function calculates CLC measures for each node in a graph. When we use `normalized = True`, it indicates that we want the CLC values to be normalized. In other words, the values are adjusted to be within the range of 0 to 1, making these measures comparable across different graphs, regardless of the network's size or scale. Finally, by calculating the average of these normalized measures, we obtain a representative value of the average CLC in the network, which is useful for assessing the communication efficiency of nodes within their respective local environments.

B.3. BC

BC is a fundamental metric in network analysis that assesses the importance of nodes as crucial intermediaries in communications within a network. Mathematically, the formula for calculating the BC of a node is as follows:

$$\text{BC}(v) = \sum_{s \neq v \neq t} \frac{\phi_{st}(v)}{\phi_{st}}$$

where:

- $\text{BC}(v)$ is the BC of node v .
- ϕ_{st} is the total number of shortest paths (geodesics) between nodes s and t .
- $\phi_{st}(v)$ is the number of shortest paths between s and t that pass through node v .

The **betweenness centrality()** function is a specific feature of the NetworkX library, widely used for network analysis in Python. This function is responsible for calculating

BC in a graph. Essentially, it assesses the importance of each node within the graph by measuring how often a node acts as a crucial bridge in the shortest paths between other nodes in the network. The result of this function is a dictionary where the keys represent the nodes in the graph, and the corresponding values are the BC measures associated with these nodes. This analysis is valuable for identifying nodes that play a critical role as intermediaries in communication or the transportation of information within a network.

B.4. SPL

The SPL, also known as the length of the shortest path, is a metric that describes the distance between two nodes in a graph, representing the minimum number of edges or weighted edges required to travel from node A to node B within the network. The formula to calculate the SPL between two nodes can be described as:

- $SPL(A, B)$ = the smallest number of edges between nodes A and B.

In Python, we can calculate the SPL libraries such as Igraph. For example, the **average_path_length()** function in calculates the average SPL between nodes in the network, providing a valuable measure for evaluating the efficiency of transportation, communication, and connectivity in a network.

B.5. Degree PC coefficient (PC)

The PC Coefficient for Degrees (PC) is a metric that assesses the linear relationship between the degrees of nodes in a graph. It measures the tendency of nodes with similar degrees to connect or whether they prefer to link to nodes with different degrees. This measure is important for understanding how the network is organized in terms of node degrees, indicating whether there is a tendency for assortativity (positive correlation) or disassortativity (negative correlation) in the network's connectivity. The formula for calculating the PC is given by:

$$PC = \frac{\sum (x_i - \hat{x}) * (y_i - \hat{y})}{\sum (x_i - \hat{x})^2 * \sum (y_i - \hat{y})^2}$$

where:

- $PC(v)$ is the PC Coefficient.
- x_i and y_i are the degrees of the nodes.
- \hat{x} and \hat{y} are the means of the node degrees.

In Python, we can calculate the PC Coefficient for Degrees using libraries such as NetworkX. The functions **degree_pearson_correlation_coefficient()** in NetworkX can

be used to calculate this measure on a graph represented by the respective libraries. The result will inform us about the nature of the network's connectivity about node degrees, which is useful for network analysis and characterization.

B.6. IC

IC is a network metric used to assess the importance of nodes in a graph in terms of how they facilitate the flow of information or communication within the network. This metric is based on the idea that some nodes may act as critical points for the efficient dissemination of information in a network. IC measures the amount of information a node is capable of controlling or transmitting to other nodes in the network. The mathematical formula for IC is defined as:

$$IC(v) = \sum_{u \neq v} \frac{1}{d(v, u)}$$

where:

- $IC(v)$ is the IC of node v .
- \sum represents the sum over all nodes u different from v .
- $d(v, u)$ is the geodesic distance between nodes u and v , i.e. the length of the shortest path between them.

This formula calculates the IC of a node by summing the inverses of the geodesic distances between the node in question v and all other nodes u in the graph. The shorter the path between v and u , the greater the contribution of node u to the IC of v . Therefore, nodes that are closer to v will have a higher contribution to its IC.

In Python, we can use the `information_centrality()` function from NetworkX to calculate the IC for the nodes in a graph. The function returns a dictionary where the keys are the nodes in the graph, and the values are the corresponding IC scores. This allows us to identify the most critical nodes in the network in terms of their ability to influence the flow of information.

B.7. SC

SC is a network centrality (SC) metric that assesses the importance of a node based on how many subgraphs containing that node are connected in the network. In other words, it measures how central a node is in terms of its participation in interconnected subgraphs. The mathematical formula for SC is defined as follows:

$$SC(v) = \sum_{S \subseteq N \setminus \{v\}} \left(\frac{1}{1 + |E(S)|} \right) \quad (B2)$$

where:

- $SC(v)$ is the SC of node v .
- S is a subset of the neighbors of v .
- N is the set of neighbors of v .
- $E(S)$ is the number of edges in the subgraph induced by S .

This formula calculates the SC of a node v by summing the contributions of all subsets of its neighbors. The more subsets contain v , and the more these subsets are interconnected (have fewer edges), the higher the SC of v . In Python, we can use the **subgraph centrality()** function from NetworkX to calculate the SC for the nodes in a graph. The function returns a dictionary where the keys are the nodes in the graph, and the values are the corresponding SC scores. This allows us to identify nodes that play a crucial role in connecting interconnected subgraphs in the network. Keep in mind that the calculation can be computationally expensive in large networks due to the need to evaluate many subsets of neighbors for each node.

B.8. Approx. current flow betweenness centrality (AC)

Approximate current flow betweenness centrality is a metric that assesses the importance of nodes based on their ability to influence the flow of electrical current within a network. Unlike the traditional approach to BC, which precisely calculates exact paths, this methodology employs numerical methods, such as Monte Carlo algorithms, to estimate the flow of current between all pairs of nodes in the network. This approach makes it suitable for large-scale and complex networks. To calculate this centrality metric in Python, we use the **approximate_current_flow_betweenness centrality** function from the NetworkX library. The result is a dictionary that associates each node in the network with its approximate centrality value. This metric plays a vital role in network analysis across various domains, aiding in the identification of key points of control and influence.

B.9. EC

EC is a measure of centrality in a network or graph that assesses the relative importance of a node based on its connections to other nodes in the network. The underlying idea is that nodes connected to other important nodes are themselves important. Therefore, EC takes into account not only the number of connections a node has but also the

importance of the nodes to which it is connected. The mathematical formula to calculate the EC of a node in a graph is defined by the following equation:

$$EC(v) = \frac{1}{\lambda} \sum_{u \in N(v)} w(u, v) \cdot C(u)$$

where:

- $EC(v)$ is the EC of node v .
- λ is the eigenvalue associated with the largest eigenvalue of the adjacency matrix of the graph.
- \sum represents the sum over all nodes u connected to node v .
- $w(u, v)$ is the weight of the edge between nodes u and v .
- $C(u)$ is the EC of node u .

The `eigenvector centrality()` function is part of the Igraph library in Python, used to calculate EC in a graph. EC is a measure that assesses the importance of nodes in a graph based on their connections, taking into account the importance of the nodes to which they are connected. The EC values are not scaled, meaning they reflect the raw measure of importance for each node in the graph. To obtain a single centrality measure for the entire graph, it is common to calculate the average of the centrality values for all nodes.

The Python code used to generate the Q-voter model, as well as the complex networks and measures of complex networks, is available for access at [76].

Appendix C. PCA

The analysis of cumulative explained variance provides valuable insights into the dimensionality reduction achieved by the PCA algorithm. The plot of cumulative explained variance illustrates the amount of information retained as the number of principal components increases (figure C1). This information helps determine the minimum number of principal components required to capture a significant portion of the original data's variability, considering the dataset with 800 rows and 9 columns. This analysis is crucial for making decisions regarding the dimensionality reduction process, in the context of changing network topologies every 100 rows.

On the other hand, the plot of the reduced data using the principal components visually represents the transformed dataset in a lower-dimensional space (figure C1). By visualizing the data in this reduced space, which is particularly important in the case of high-dimensional data with 9 complex network measures, a better understanding

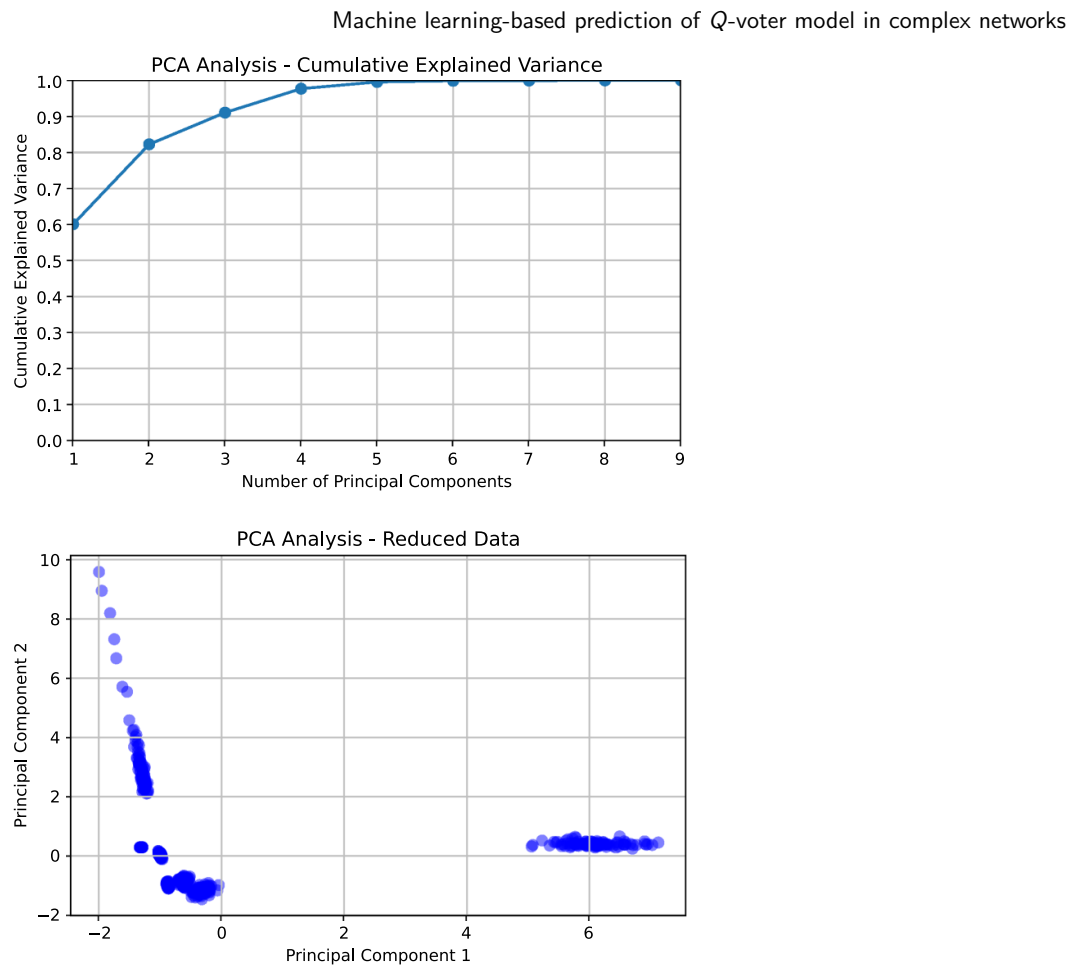


Figure C1. The top figure illustrates the Cumulative Explained Variance in PCA (Principal Component Analysis) Analysis. This plot showcases the cumulative amount of variance in the data explained by each principal component, while the subsequent figure displays the Reduced Data Plot using Principal Components. The reduced data is represented in a lower-dimensional space defined by the principal components, allowing for a simplified representation of the original data while preserving its underlying structure. These figures provide insights into the data used to feed our machine-learning prediction models and demonstrate the effectiveness of PCA in reducing the dimensionality of the input data.

of its structure and potential patterns or clusters that may exist is gained. These plots play a vital role in validating the effectiveness of the PCA algorithm in capturing the most relevant features of the data while reducing its dimensionality, considering the complexity and diversity of the network measures across different network topologies.

Additionally, the proximity of data points in the reduced space reflects the similarity between the models, allowing for the identification of clusters or groupings within each network topology and across different topologies. This further aids in understanding the relationships and similarities among different instances in the dataset, facilitating comparative analysis and identification of common characteristics or trends. Overall, these plots provide valuable insights into the data, aiding in analysis, interpretation,

and model comparison, particularly in the context of complex networks with multiple measures and changing topologies.

Appendix D. Grid search hyperparameter tuning

Table D1 shows the hyperparameters optimized by grid search.

Table D1. Hyperparameters for each machine learning algorithm optimized by grid search optimizer.

Predictor	Hyperparameters and description	Values
RF	- max_depth: Maximum depth of the tree.	[10, 20, 30, 40, 50]
	- max_features: Number of features to be considered toward a best split.	[2, 3, 4]
	- min_samples_leaf : Minimum number of samples required to be at a leaf node.	[1, 2, 4]
	- min_samples_split: Minimum number of samples for the split of an internal node.	[2, 5, 10]
	- n_estimators: Number of trees in the forest.	[100, 200, 300]
LASSO	- regularization parameter.	range 0.0001 to 0.0005
MLP	- activation: Activation function for the hidden layer.	[identity, logistic, tanh, relu]
	- solver: Solver for weight optimization.	[lbfgs, sgd, adam]
	- alpha: L2 penalty (regularization term) parameter.	[0.0001, 1×10^{-5} , 0.01, 0.001]
	- batch_size: Size of minibatches for stochastic optimizers.	[1000, 5000]
	- learning_rate: Learning rate schedule for weight updates.	[constant, invscaling, adaptive]
XGBoost	- learning_rate_init: Initial learning rate used.	[0.001, 0.01, 0.1, 0.2, 0.3]
	- subsample: fraction of observations to be randomly sampled in each tree.	[0.6, 0.8, 1.0]
	-max_depth: maximum depth of each tree.	[3, 4, 5]

J. Stat. Mech. (2023) 123402

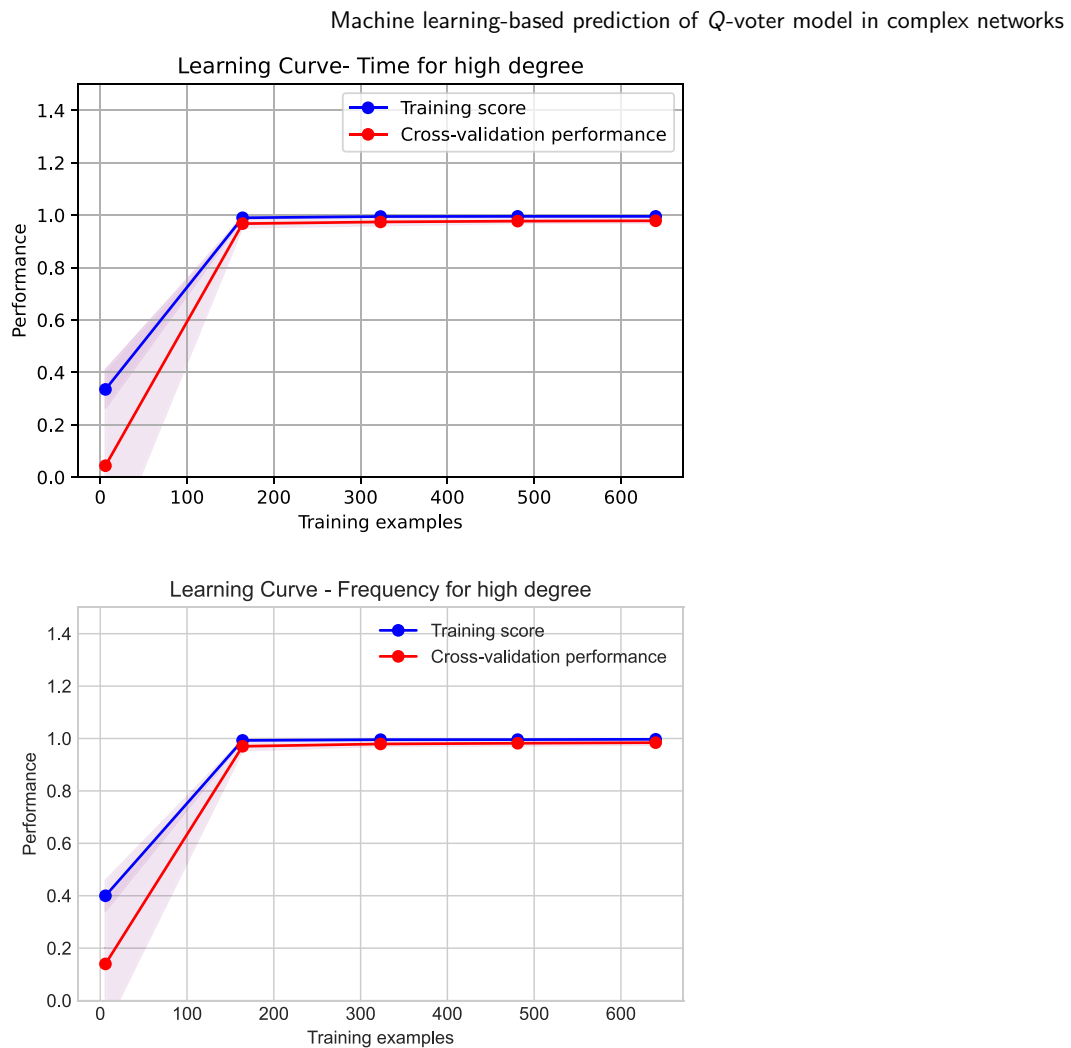


Figure D1. The learning curve displays the training accuracy (represented by the blue curve) and the test accuracy (represented by the red curve) for the high initialization method. The top figure illustrates the learning curve for predicting Y_i , while the subsequent figure shows the learning curve for predicting C_i . These learning curves offer valuable insights into the performance of the model and the effectiveness of the high initialization method. By examining the training and test accuracies, one can evaluate the model's ability to generalize to unseen data and detect potential concerns such as overfitting or underfitting.

References

- [1] Thurner S, Hanel R and Klimek P 2018 *Introduction to the Theory of Complex Systems* (Oxford University Press)
- [2] Boccarà N and Boccarà N 2010 *Modeling Complex Systems* vol 1 (Springer)
- [3] Del Vicario M, Bessi A, Zollo F, Petroni F, Scala A, Caldarelli G, Stanley H E and Quattrocioni W 2016 *Proc. Natl Acad. Sci.* **113** 554–9
- [4] Flaxman S, Goel S and Rao J M 2016 *Publ. Opin. Q.* **80** 298–320
- [5] Barberá P, Jost J T, Nagler J, Tucker J A and Bonneau R 2015 *Psychol. Sci.* **26** 1531–42

- [6] Conover M D, Ratkiewicz J, Francisco M, Goncalves B, Menczer F and Flammini A 2011 Proc. Int. AAAI Conf. on Web and Social Media vol **133** pp 89–96
- [7] Bessi A and Ferrara E 2016 *J. Comput. Mediat. Commun.* **21** 303–20
- [8] Centola D 2018 *Science* **360** 1116–9
- [9] Castellano C, Fortunato S and Loreto V 2009 *Rev. Mod. Phys.* **81** 591
- [10] Soares G, Oliveira S and Carvalho T C 2021 arXiv:2103.14179
- [11] Soares L, de Oliveira J, de Oliveira M and Ferreira S 2021 *Physica A* **573** 125931
- [12] Ising E 1925 *Z. Phys.* **31** 253–8
- [13] Stauffer D and Aharony A 2007 *Introduction to Percolation Theory* vol 91 (CRC Press) pp 46–49
- [14] Newman M E J and Barkema G T 1999 *Monte Carlo Methods in Statistical Physics* vol 1 (Oxford University Press)
- [15] Kuperman M N and Abramson G 2001 *Phys. Rev. Lett.* **86** 2909
- [16] Sznajd-Weron K and Sznajd J 2000 *Int. J. Mod. Phys. C* **11** 1157–65
- [17] Lima F W and Gonçalves S 2007 *Phys. Rev. E* **76** 036117
- [18] Rodrigues F A and Costa L F 2005 *Int. J. Mod. Phys. C* **16** 1785–92
- [19] Liggett T M 1985 *Interacting Particle Systems* (Springer)
- [20] Baronchelli A, Dall'Asta L and Barrat A 2006 *J. Stat. Mech.* p 08014
- [21] Baronchelli A and Dall'Asta L 2006 *Physica A* **356** 294–9
- [22] Baronchelli A and Loreto V 2006 *J. Stat. Mech.* **5** p 08001
- [23] Deffuant G, Neau D, Amblard F and Weisbuch G 2000 *Adv. Complex Syst.* **3** 87–98
- [24] Hegselmann R and Krause U 2002 *J. Artif. Soc. Soc. Simul.* **5** 2
- [25] Malarz K and Kulakowski K 2009 *Phys. Rev. E* **80** 036105
- [26] Pinheiro F L and Santos M A 2015 *Phys. Rev. E* **92** 012812
- [27] Doniec M, Lipiecki A and Sznajd-Weron K 2022 *Entropy* **24** 983
- [28] Lipiecki A and Sznajd-Weron K 2022 *Chaos Solitons Fractals* **165** 112809
- [29] Krueger T, Szwabiński J and Weron T 2017 *Entropy* **19** 371
- [30] Soares R L and Fontanari J F 2021 *Phys. Rev. E* **103** 032312
- [31] Fernandez-Gracia J, Eguíluz V M and Miguel M S 2014 *Phys. Rev. E* **90** 012811
- [32] De Sanctis L F, Gonçalves B and Pinto S S 2013 *Phys. Rev. E* **87** 052811
- [33] Liu Z, Wang J, Ren F and Wu X 2020 *Europhys. Lett.* **130** 20001
- [34] Brugnano G, Chat'e H and Manrubia S C 2018 *Phys. Rev. E* **97** 062309
- [35] Keeling M J and Eames K T 2005 *J. R. Soc. Interface* **2** 295–307
- [36] Rodrigues F A, Peron T, Connaughton C, Kurths J and Moreno Y 2019 arXiv:1910.00544
- [37] Pecora L M and Carroll T L 1998 *Phys. Rev. Lett.* **80** 2109
- [38] Brooks H Z and Porter M A 2020 *Phys. Rev. Res.* **2** 023041
- [39] Moretti P, Liu S, Castellano C and Pastor-Satorras R 2013 *J. Stat. Phys.* **151** 113–30
- [40] Lorenz J 2007 *Int. J. Mod. Phys. C* **18** 1819–38
- [41] Moretti P and Munoz M A 2013 *Nat. Commun.* **4** 1–6
- [42] Guerra R A, Eguíluz V M and Miguel M S 2004 *Phys. Rev. E* **70** 046106
- [43] Mobilia M 2003 *Phys. Rev. Lett.* **91** 028701
- [44] Galam S 2002 *Eur. Phys. J. B* **25** 403–6
- [45] Castellano C, Muñoz M A and Pastor-Satorras R 2009 *Phys. Rev. E* **80** 041129
- [46] Jankowski R 2020 real-q-voter (available at: <https://github.com/robertjankowski/real-q-voter>) (Accessed 6 September 2023)
- [47] Erdős P *et al* 1960 *Publ. Math. Inst. Hung. Acad. Sci.* **5** 17–60
- [48] Barabási A L and Albert R 1999 *Science* **286** 509–12
- [49] Onody R N and Castro P A 2004 *Physica A* **336** 491–502
- [50] Lancichinetti A, Fortunato S and Radicchi F 2008 *Phys. Rev. E* **78** 046110
- [51] Watts D J and Strogatz S H 1998 *Nature* **393** 440–2
- [52] Waxman B M 1988 *IEEE J. Sel. Areas Commun.* **6** 1617–22
- [53] Hagberg A A, Swart P J and Chult D B 2014 Networkx (version 1.9.1) (available at: <https://networkx.github.io/>)
- [54] Freeman L C 1979 *Soc. Netw.* **1** 215–39
- [55] Freeman L C 1977 *Sociometry* **40** 35–41
- [56] Newman M E 2010 *Networks: An Introduction* (Oxford University Press)
- [57] Newman M E 2018 *Networks: An Introduction* (Oxford University Press)
- [58] Stephenson K and Zelen M 1989 *Soc. Netw.* **11** 1–37

- [59] Estrada E and Hatano N 2008 *Phys. Rev. E* **77** 036111
- [60] Brandes U, Delling D, Gaertler M, Goerke R, Hoefer M, Nikoloski Z and Wagner D 2005 *IEEE Trans. Knowl. Data Eng.* **17** 754–67
- [61] Bonacich P 1987 *Am. J. Sociol.* **92** 1170–82
- [62] Costa L F, Rodrigues F A, Traverso G and Villas Boas P R 2007 *Adv. Phys.* **56** 167–242
- [63] Wainer J and Cawley G 2021 *Expert Syst. Appl.* **182** 115222
- [64] Cawley G C and Talbot N L 2010 *J. Mach. Learn. Res.* **11** 2079–107
- [65] Leaver A M, Wade B, Vasavada M, Hellemann G, Joshi S H, Espinoza R and Narr K L 2018 *Frontiers Psychiatry* **9** 92
- [66] Nakagawa S, Johnson P C and Schielzeth H 2017 *J. R. Soc. Interface* **14** 20170213
- [67] Spadon G, Carvalho A C Rodrigues-Jr J F and Alves L G 2019 *Sci. Rep.* **9** 11801
- [68] Centola D and Macy M 2007 *Am. J. Sociol.* **113** 702–34
- [69] Sunstein C 2018 *# Republic: Divided Democracy in the age of Social Media* (Princeton university press)
- [70] NetworkX 2023 Erdős-rényi graphs (available at: https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.erdos_renyi_graph.html)
- [71] Igraph 2023 Barabasi-albert model (available at: https://igraph.org/python/api/latest/igraph._igraph.GraphBase.html#Barabasi)
- [72] NetworkX 2023 LFR benchmark graph (available at: https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.LFR_benchmark_graph.html)
- [73] NetworkX 2023 Watts-strogatz graph (available at: https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.watts_strogatz_graph.html)
- [74] NetworkX 2023 Waxman graph (available at: https://networkx.org/documentation/stable/reference/generated/networkx.generators.geometric.waxman_graph.html)
- [75] NetworkX 2023 Path graph (available at: https://networkx.org/documentation/stable/reference/generated/networkx.generators.classic.path_graph.html)
- [76] Kent A P P 2023 Qvml 2023 (available at: https://github.com/kentwar/QVML_2023/)