8-1-2021

Social Network Analysis: A Machine Learning Approach

Bonaventure Chidube Molokwu
University of Windsor

Social Network Analysis: A Machine Learning Approach

by

Bonaventure Chidube Molokwu

A Dissertation

Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2021

Social Network Analysis: A Machine Learning Approach

by

Bonaventure Chidube Molokwu

APPROVED BY:

E. Bagheri, External Examiner
Ryerson University

D. Martinovic
Faculty of Education

S. Samet
School of Computer Science

J. Lu
School of Computer Science

Z. Kobti, Advisor
School of Computer Science

July 22, 2021

Declaration of Co-Authorship/ Previous Publication

I. Co-Authorship

I hereby declare that this dissertation incorporates material that is the result of my research conducted under the supervision of my advisor, Dr. Ziad Kobti. The research details are covered in chapters 2, 3, 4, and 5 of this dissertation. In all cases, the key ideas, primary contributions, experimental designs, data analysis and interpretation, were performed by Bonaventure Chidube Molokwu (the candidate) and Dr. Ziad Kobti (the supervisor).

In chapters 2 and 4; this dissertation incorporates the outcome of a joint research undertaken in collaboration with Shaon Bhatta Shuvo and Dr. Narayan C. Kar. With regard to these chapters, Shaon Bhatta Shuvo had contributed his time during the validation of our experiments; while Dr. Narayan C. Kar had assisted with proofreading these chapters.

Chapter 3 was co-authored with Dr. Anne Snowdon who supported with the provision of research datasets and resource personnel for conducting our experiments as well as tests. Also, in this chapter, Shaon Bhatta Shuvo has assisted with proofreading the work/chapter prior to its submission to a journal.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my dissertation, and have obtained written permission from each of the co-author(s) to include the above material(s) in my dissertation.

I certify that, with the above qualification, this dissertation, and the research to which it refers, is product of my own work.

II. Previous Publications

This dissertation includes the extended or original version(s) of the papers that have been previously published/submitted for publication in peer reviewed conferences and journals, as follows:

Section	Full Citation	Status
Chapter 2	Bonaventure Chidube Molokwu, Shaon Bhatta Shuvo, Ziad Kobti, and Narayan C. Kar. "ClasReg: A Deep Learning and Heuristic Methodology for Predicting Breakups or Rifts in Social Network Structures." Social Networks (SN). Elsevier, 2021.	Under Review
Chapter 3	Bonaventure Chidube Molokwu, Shaon Bhatta Shuvo, Ziad Kobti, and Anne Snowdon. "A Transfer Learning Framework for COVID-19 Monitoring and the Prediction of PPE Consumption in Community Health Centres." Decision Support Systems (DSS). Elsevier, 2021.	Under Review
	Shaon Bhatta Shuvo, Bonaventure Chidube Molokwu, and Ziad Kobti. "Simulating the Impact of Hospital Capacity and Social Isolation to Minimize the Propagation of Infectious Diseases." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 3451-3457. ACM, 2020.	Published

Chapter 4	Bonaventure Chidube Molokwu, Shaon Bhatta Shuvo, Narayan C. Kar, and Ziad Kobti. "Social Network Analysis using Knowledge-Graph Embeddings and Convolution Operations." The 25th International Conference on Pattern Recognition (ICPR 2021), pp. 6351-6358. IEEE, 2021.	Published
	Bonaventure Chidube Molokwu, Shaon Bhatta Shuvo, Narayan C. Kar, and Ziad Kobti. "Node Classification and Link Prediction in Social Graphs using RLVECN." The 32nd International Conference on Scientific and Statistical Database Management (SSDBM 2020), pp. 1-10. ACM, 2020	Published
	Bonaventure Chidube Molokwu, Shaon Bhatta Shuvo, Narayan C. Kar, and Ziad Kobti. "Node Classification in Complex Social Graphs via Knowledge-Graph Embeddings and Convolutional Neural Network." The 2020 International Conference on Computational Science (ICCS 2020), pp. 183-198. Springer, 2020.	Published
Chapter 5	Bonaventure Chidube Molokwu and Ziad Kobti. "Spatial Event Prediction via Multivariate Time Series Analysis of Neighboring Social Units using Deep Neural Networks." The 2019 International Joint Conference on Neural Networks (IJCNN 2019). IEEE, 2019.	Published
	Bonaventure Chidube Molokwu and Ziad Kobti. "Event Prediction in Complex Social Graphs via Feature Learning of Vertex Embeddings." The 2019 International Conference on Neural Information Processing (ICONIP 2019). Springer, 2019.	Published

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my dissertation. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my dissertation does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my dissertation, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my dissertation.

I declare that this is a true copy of my dissertation, including any final revisions, as approved by my dissertation committee and the Graduate Studies office, and that this dissertation has not been submitted for a higher degree to any other University or Institution.

Abstract

Social Network Analysis (SNA) is an appealing research topic, within the domain of Artificial Intelligence (AI), owing to its widespread application in the real world. In this dissertation, we have proposed effective Machine Learning (ML) and Deep Learning (DL) approaches toward resolving these open problems with regard to SNA, viz: Breakup Prediction, Link Prediction, Node Classification, Event-based Analysis, and Trend/Pattern Analysis. SNA can be employed toward resolving several real-world problems; and ML as well as DL have proven to be very effective methodologies for accomplishing Artificial Intelligence (AI)-related goals.

Existing literature have focused on studying the apparent and latent interactions within social graphs as an n-ary operation, which yields binary outputs comprising positives (friends, likes, etc.) and negatives (foes, dislikes, etc.). Inasmuch as interactions constitute the bedrock of any given Social Network (SN) structure; there exist scenarios where an interaction, which was once considered a positive, transmutes into a negative as a result of one or more indicators which have affected the interaction quality. At present, this transmutation has to be manually executed by the affected actors in the SN. These manual

transmutations can be quite inefficient, ineffective, and a mishap might have been incurred by the constituent actors and the SN structure prior to a resolution. Thus, as part of the research contributions of this dissertation, we have proposed an automatic technique toward flagging positive ties that should be considered for breakups or rifts (negative-tie state), as they tend to pose potential threats to actors and the SN.

Furthermore, in this dissertation, we have proposed DL-based approaches based on edge sampling strategy for resolving the problems of Breakup Prediction, Link Prediction, and Node Classification. Also, we have proposed ML-based approaches for resolving the problems of Event-based Analysis and Trend/Pattern Analysis. We have evaluated our respective approaches against benchmark social graphs, and our results have been comparatively encouraging as documented herein.

Dedication

To Chinemelu, Ify, Amaka, Chukwuka; and my inspirational parents, Dr. and Mrs. C. C. Molokwu.

Acknowledgements

I am grateful to my advisor, Dr. Ziad Kobti, for his support, assistance, and motivation which have jointly guided and steered me through my doctoral program.

Furthermore, I humbly wish to acknowledge members of my doctoral committee comprising: Dr. Saeed Samet, Dr. Jianguo Lu, and Dr. Dragana Martinovic for their constructive feedback and helpful contributions.

In reference to the resources pooled for the successful implementation of our experiments and study herein, I do wish to acknowledge the following, viz: Vector Institute for Artificial Intelligence; International Business Machines (IBM); Canadian Institute of Health Research (CIHR) Operating Grant, [operating grant number VR5 172669]; Shared Hierarchical Academic Research Computing Network (SHARCNET); and Compute Canada (www.computecanada.ca). Also, we acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [RGPIN-2021-03181].

In addition, special appreciation and gratitude goes to the entire staff of the School of Computer Science: Ms. Gloria Mensah, Ms. Melissa Robinet, Ms. Karen Bourdeau, Ms. Christine Weisener, Ms. Margaret Garabon, Ms. Tina Palmer, Mr. Maunzer Batal, Mr.

Sanjay Chitte, and Mr. Robert Mavrinac.

I am indeed very grateful and thankful to all my laboratory colleagues and friends, Shaon Bhatta Shuvo, Akshay Mukundbhai Shah, Kaitav Mehta, Chidinma Oraemesi, etc., for their immense contributions, constructive criticisms, and unswerving support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Social Network Analysis

In recent times, SNA has become a very important and interesting subject matter with regard to AI in that a vast variety of processes, comprising animate and inanimate entities, can be examined by means of SNA. In this regard, SNA has been employed by security agencies for counter-intelligence and law enforcement purposes. SNA has been used in medicine and pharmaceuticals for gaining insights into protein-protein interactions. Also, SNA has been employed in the World Wide Web (WWW) for hyperlink analysis, cybersociety analysis, sentiment analysis, etc. Furthermore, prediction tasks within social network structures have become significant research problems in SNA. Thus, hidden facts and details embedded in social network structures can be effectively and efficiently harnessed for training AI models with the goal of predicting several missing components (such as links/ties, nodes/actors, structure type, etc.) within a given social network. Therefore, important factors such as the individual attributes of spatial social actors, and the underlying patterns of relationship binding these social actors must be taken into consideration; because these factors are

relevant in understanding the nature and dynamics of a given social network structure.

SNA is a subdomain (or research topic) within the domain (or research area) of AI; and several open problems still exist with regard to SNA. Some of these open problems with respect to SNA include, viz: Information Diffusion, Community Detection, Event-Based Analysis, Multi-Layer Network Analysis, Trends and Patterns Analyses, Sentiment Analysis, Collaboration and Knowledge Management, Node Classification, Link Detection, Breakup Prediction, etc. Thus, in this dissertation, we have proposed effective Machine Learning (ML) approaches toward resolving the following SNA (research) problems, namely: Breakup Prediction, Link Prediction, Node Classification, Event-based Analysis, and Trend/Pattern Analysis.

1.2 Motivations and Objectives

Definition 1.1. *Social Network, SN: This is a tuple comprising a graph, $G(V, E)$; a metadata function, f_V, which extends the definition of the vertices' set by mapping it to a given set of attributes, V'; and a metadata function, f_E, which extends the definition of the edges' set by mapping it to a given set of attributes, E'. Thus, $G(V, E) \subset SN$ as expressed via equation 1.1. Also, Figure 1.1 depicts a SN structure.*

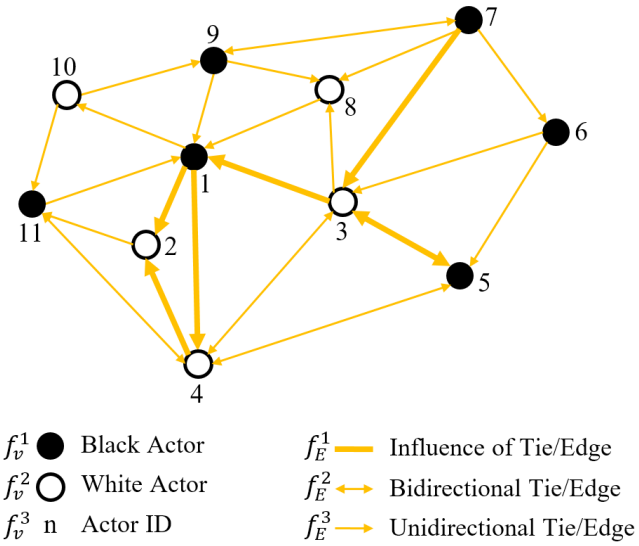

Figure 1.1: Social network (SN) structure.

$$SN = (G, f_V, f_E) \equiv (V, E, f_V, f_E)$$

$$f_V : V \to V' \qquad \text{vertices' metadata function} \tag{1.1}$$

$$f_E : E \to E' \qquad \text{edges' metadata function}$$

Problem 1.1. *Breakup/Rift: The universal set, $E = \{U \times V\}$, represents all possible ties or edges in a social graph, SN. Let $y_i = 1$ be a label denoting that there exists a relationship or edge ($e_i^+ \in E$) connecting vertices, u_m and v_n. Also, let label $y_i = 0$ denote that there is no relationship or edge ($e_i^- \in E$) between vertices, u_m and v_n. A breakup/rift is a function, $f(e_i^+)$, that evaluates all positive ties/edges to determine which e_i^+ (ground-truth edge) needs to transition or transmute to e_i^- (falsehood edge).*

Problem 1.2. *Link Prediction: Consider $G(V, E) \subset SN$ at any instantaneous time, t. The set of actors/vertices is defined via $U \subset V : \{U|u_0, u_1, ..., u_m\} \subset \{V|v_0, v_1, ..., v_m\}$; and the set of ties/edges is defined via $E : (u_i, v_j) \in \{U \times V\}$. The goal of a link-prediction model*

is to train a predictive function, f, that learns the similarity measure, $similarity(U, V)$, between pairs of actors in SN; such that the knowledge gained from the training is used to infer the probability of a tie existence between any valid pair of actors, (u_i, v_j), at time, t.

Problem 1.3. *Node Classification: Given a social network, SN, comprising partially labelled actors (or vertices), $V_{lbl} \subset V : V_{lbl} \rightarrow Y_{lbl}$; and unlabelled vertices defined such that: $V_{ulb} = V - V_{lbl}$. Therefore, a node-classification model aims at training a predictive function, $f : V \rightarrow Y$, that learns to predict the labels, Y, for all actors or vertices, $V \subset SN$, via knowledge harnessed from the mapping: $V_{lbl} \rightarrow Y_{lbl}$.*

Problem 1.4. *Event Prediction: Given a set of indicators, $X \subseteq T \times F$, where T and F denote the domains of Time and Features, respectively. An event prediction model forecasts the occurrence of an event, Y, such that: $Y = f(T, F)$. In other words, it is the conditional probability, $Pr(Y^+|Y^-)$, that a futuristic event, Y^+, will occur given the knowledge that a past event, Y^-, has already occurred.*

Problem 1.5. *Trend/Pattern Analysis: Given a set of feature or independent variables, $X \in \mathbb{R} : x_{i,1}, x_{i,2}, ..., x_{i,j}$, such that the shape of the feature space is an $i \times j$ feature vector; and a set of target or dependent variables, $Y \in \mathbb{Z} : y_{i,1}, y_{i,2}, ..., y_{i,k}$, such that the shape of the target space is an $i \times k$ target vector. A Trend (or Pattern) Analysis framework aims at training a ML function, $f_m : X \rightarrow Y \equiv x_{i,*} \mapsto y_{i,*}$, which learns to effectively and efficiently make predictions about Y based on the patterns of information learnt from X. Thus, $y_{i,*} \in Y = f_m(x_{i,*} \in X)$.*

Taking into consideration the vast application domains where SNA can be applied, and motivated by the fact that several real-world datasets [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] exist for SNA experiments; herein in this dissertation, we have employed ML and DL methodologies toward resolving the aforementioned open research problems.

1.3 Research Contributions

Breakup/Rift Prediction Framework

As SN structures grow and expand with respect to time, their vulnerability to malicious attacks increases. Certain relationships tend to transmute from a positive ($+ve$) state to a negative ($-ve$) state when the relationship is no longer mutually beneficial with respect to the pair of associated actors. In that regard, a breakup is said to have occurred. Breakups can be essential and beneficial to SN structures in that they can either serve as a preventive or control measure for mitigating malicious attacks within a given SN. This dissertation has proposed a unified framework (ClasReg), which employs Deep Learning (DL) techniques and heuristics, toward forecasting breakups in SN structures. Also, our contribution in this regard, increments the awareness and popularity of breakup prediction in SN structures; considering the fact that breakup prediction is a NP-Complete problem [14]. This research contribution was supported by the Vector Institute for Artificial Intelligence.

Link Prediction and Node Classification

Link prediction in SN structures results in the formation of relationships and/or ties which increases the tendency for transitivity in social networks. In addition, classification of nodes

in SN structures results in the formation of cluster(s), and clusters give rise to homophily in social networks. In this dissertation, we have proposed a unique clustering model and semi-supervised framework (RLVECO) based on an iterative learning approach. Our proposed framework herein is a hybrid DL-based model comprising graph-based representation learning layers and convolution operations. RLVECO is capable of learning the non-linear distributed representations enmeshed in a given social graph, and it employs this learning toward the prediction of links and classification of nodes in the social graph. Furthermore, our research contributions toward resolving these open research problems (Link Prediction and Node Classification) in SNA was nominated for the Best Paper Award at the 32^{nd} International Conference on Scientific and Statistical Database Management (SSDBM 2020) [15].

Trend and Pattern Analysis

Taking into consideration the ravaging effect of the novel Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), and the associated impacts of the COVID-19 pandemic; we were prompted to approach the COVID-19 problem from a Data Science perspective. In this regard, we have modelled the COVID-19-pandemic scenario(s) as a trend/pattern analysis problem in SNA. Therefore, in this dissertation herein, we have proposed a unique TL framework which learns via pre-training on knowledge transfer from correlated (source) provinces to a target province/region. The proposed TL framework is capable of monitoring the impacts of SARS-CoV-2 via effective predictions for Infected, Hospitalized, Recovered,

and Death cases. Also, the TL framework is equipped with the capacity to predict PPE demand(s) in CHCs that provide medical treatment to COVID-19 (Hospitalized) patients. Moreover, this timely research was fully funded and supported by the Canadian Institutes of Health Research (CIHR) - Grant Account Number: VR5 172669.

Event-based Analysis

Event prediction in social network structures remains a very interesting research problem with respect to SNA. This impels understanding the intrinsic relationship patterns preserving a given social network structure, based on the study of several structural properties computed on the constituent social units, with respect to space and time. In this regard, tackling problems of this nature is considered NP-Complete [14]. Consequently, herein in this dissertation, we proposed a unique DL approach based on deep-layer stacks of Multi-Layer Perceptron (MLP)s. The proposed model is capable of resolving event-prediction related problems about a target social unit, y, based on the intrinsic patterns of relationship learnt from one or more neighboring social units, x [16]. Additionally, this model is appended with an adjustment-bias (a_b) vector, at its output layer, so as to improve the accuracy and precision of predictions made with respect to the target unit (or node). This research was supported by International Business Machines (IBM) via the provision of computational resources necessary to carry-out our experiments herein.

1.4 Structure of this Dissertation

Thereafter, the subsequent contents of this dissertation are organized as follows.

In Chapter 2, we proposed ClasReg: a methodology based on DL techniques and heuristics. ClasReg is jointly capable of breakup prediction and link prediction in SN structures.

In Chapter 3, we have proposed RLVECO: a hybrid DL framework which possesses the capabilities of effectively resolving link prediction and node classification problems in SN structures.

In Chapter 4, we attempted to address the issue(s) of the COVID-19 pandemic via a SNA approach. To this end, we have introduced a Transfer-Learning framework for impact monitoring of the COVID-19 pandemic as well as for the prediction of PPE consumption/demand in Community Health Centres.

In Chapter 5, we proposed a DL-based approach for resolving spatial event prediction problems in SN structures via learning and knowledge transfer from correlated and/or neighboring social actors.

Finally, in Chapter 6, we have highlighted all the research contributions of this dissertation. Moreover, this chapter provides direction for possible future research or work.

Bibliography

[1] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, pp. 93–106, 2008.

[2] G. Namata, B. London, L. Getoor, and B. Huang, "Query-driven active surveying for collective classification," in *Proceedings of the Workshop on Mining and Learning with Graphs, MLG-2012*, 2012.

[3] V. E. Krebs, "Organizational adaptability quotient," in *IBM Global Services*, 2008.

[4] B. Zhao, P. Sen, and L. Getoor, "Entity and relationship labeling in affiliation networks," in *Proceedings of the 23rd International Conference on Machine Learning, ICML*, 2006.

[5] J. Kunegis, "Konect: the koblenz network collection," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013. [Online]. Available: http://konect.cc/

[6] X. Liang, S. Li, S. Zhang, H. Huang, and S. Chen, "Pm2.5 data reliability, consistency and air quality assessment in five chinese cities†," *Journal of Geophysical Research*,

vol. 121, pp. 10 220–10 236, 2016.

[7] I. Berry, J.-P. R. Soucy, A. Tuite, and D. Fisman, "Open access epidemiologic data and an interactive dashboard to monitor the covid-19 outbreak in canada," *Canadian Medical Association Journal*, vol. 192, pp. E420 – E420, 2020.

[8] Google LLC, "Google covid-19 community mobility reports," https://www.google.com/covid19/mobility/, Mountain View, CA, 2021.

[9] Statistics Canada, "Covid-19 statcan covid-19: Data to insights for a better canada [data tables]," https://www150.statcan.gc.ca/n1/en/catalogue/45280001, Ottawa, Canada, 2021.

[10] Canadian Institute for Health Information, "Covid-19 intervention timeline in canada," https://www.cihi.ca/en/covid-19-intervention-timeline-in-canada, Ottawa, Canada, 2021.

[11] ——, "Access data and reports," https://www.cihi.ca/en/access-data-and-reports, Ottawa, Canada, 2021.

[12] Esri Canada, "Covid-19 open data," https://resources-covid19canada.hub.arcgis.com/pages/open-data, Toronto, Canada, 2021.

[13] Global News, "Coronavirus," https://globalnews.ca/tag/coronavirus/, Toronto, Canada, 2021.

[14] S. A. Cook, "The complexity of theorem-proving procedures," *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, p. 151–158, 1971.

[15] B. Molokwu, S. B. Shuvo, N. Kar, and Z. Kobti, "Node classification and link prediction in social graphs using rlvecn," *32nd International Conference on Scientific and Statistical Database Management*, 2020.

[16] B. C. Molokwu and Z. Kobti, "Spatial event prediction via multivariate time series analysis of neighboring social units using deep neural networks," *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2019.

Chapter 2

ClasReg: A Deep Learning and Heuristic Methodology for Predicting Breakups in Social Network Structures

Herein in Chapter 2, we have studied the problems of Breakup Prediction and Link Prediction in Social Network structures. Hence, we have proposed a 5-layer, bifunctional framework which possesses the capabilities of predicting breakup(s) as well as newer links or ties in a given SN structure. Our proposed framework harnesses the strengths of Deep Learning and logical inferences toward resolving the research problems in this chapter.

2.1 Introduction

The advent and advancement of mobile technologies have immensely fostered the spread of cybersocieties (virtual communities) via social media channels/platforms/services. A Social Network (SN) can be defined as a network of interactions or relationships, where the vertices/nodes consist of actors, and the edges/links consist of the relationships or

interactions between these actors [1]. In the context of SNA, ties (edges or links) are established as a result of unidirectional or bidirectional relationships between actors (vertices or nodes) within a SN. As actors interact and build relationships/ties among themselves; these actions foster transitivity which results in the expansion of the SN structure. Every action carried out by each actor within a given SN is associated with a digital trail or print. SNA, as an interesting research subfield of AI, centers on harnessing these complex digital trails via studies and analyses, with the goal of extracting useful knowledge and/or information for the sustenance of the SN.

The growth or expansion of SN structures increase their vulnerability to malicious attacks such as identity theft, impersonation, phishing, spamming, etc. Additionally, over a period of time, certain relationships tend to transmute from a positive ($+ve$) state to a negative ($-ve$) state. This occurs when a relationship is no longer mutually beneficial with respect to the pair of associated actors. Factors such as the closeness, intensity, frequency, and duration of a relationship are standard indicators, proposed by [2, 3] and analyzed by [4, 5, 6, 7]; which can be implemented to measure how mutually beneficial a relationship is to both connected actors. Breakups are essential and beneficial to SN structures in that they can either serve as a preventive or control measure for mitigating malicious attacks within a given SN. At present, most breakups or rifts are usually implemented as a control measure in SN structures only after a threat and/or violation might have been executed. Also, when and where a breakup is necessary in a SN, this usually has to be manually carried out by

either of the affected actors in the relationship. However, if a breakup is conceptualized as a preemptive measure which will automatically and periodically be suggested to constituent actors of a SN structure; this will effectuate much safer and flourishing cybersocieties. In this regard, we have propounded an inventive and multilayer model (ClasReg) to address the problem.

Basically, ClasReg is a DL model whose framework essentially comprises a preprocessing layer, a Representation Learning (RL) or Feature Learning (FL) layer, a classification layer, a regression layer, and an inference/heuristic engine [8]. ClasReg is a scalable model that exploits its multilayer framework for simultaneously resolving the problems of breakup prediction as well as link prediction in SN structures. At the moment, several SN structures and cybersocieties possess the capability of suggesting newer ties/links to its constituent actors; however, they lack the capability of predicting existent ties which should be broken in a bid to maintain a relatively trustworthy SN. To this end, our ClasReg proposition aims at resolving this existent problem via its breakup prediction capability; and still maintain the relative density (or mass) of the SN structure via its link prediction capability. Breakup prediction in SN structures is an interesting problem which is considered to be NP-Complete [9].

Conventionally, link prediction models are designed to predict newer ties in SN structures as an n-ary operation which generates only binary or dyadic outputs comprising positives (friends, trusts, etc.) and negatives (rivals, distrusts, etc.). However, the hybrid prediction

capabilities of ClasReg predicts or forecasts newer ties as an n-ary operation which renders ternary or triadic outputs composed of positives, negatives, and intersections (*positives* that need to be *negatives*). Furthermore, ClasReg's breakup prediction function is dependent on the logical conjunction (or intersection) between the outputs of its classification and regression layers, and with respect to the regression layer. In this respect, there exist scarcely any published research, in the domain of SNA, that is structurally similar to the model of ClasReg. We have evaluated ClasReg herein against state-of-the-art baselines (or models) for link prediction, and classic models for evaluating the strength of ties in SN structures. It is relevant to point out that breakup or rift prediction problem differs from the problems of link prediction and strength of ties in SNA. Breakups prediction in SN structures focuses on identifying positive ties that ought to be broken in a bid to avoid unforeseeable threats. Link prediction aims at predicting newer relationships that should be established within the SN structure in the near future. Strength of ties essentially evaluates ties using a probability distribution, so as to ascertain which relationships are stronger (approaches 1) and/or weaker (approaches 0).

Our research and findings with respect to ClasReg introduces the following novelties, viz:

1. Proposition of a singular framework/model that possesses the capabilities of link prediction and breakup prediction in SN structures.

2. Proposition of an algorithm, refer to Algorithm 2.1 and equation 2.11 herein, for the

computation of negative (-ve) or falsehood ties to supplement the given positive (+ve) or ground-truth ties.

3. Proposition of a threefold predictor engine composed of classification ($Y^C = \mathbb{B} = \{y_i \in \mathbb{Z} | 0, 1\}$) and regression ($Y^R = [0, 1] = \{y_i \in \mathbb{R} | 0 \leq y_i \leq 1\}$) as well as inference/heuristic subcomponents.

4. Proposition of a classification subcomponent which computes a vector-space embedding, possessing 1×512 feature dimensions, for every constituent tie/link in a given SN structure.

5. Proposition of a prediction-score function ($Y^R_{u,v}$) for the regression subcomponent of the predictor engine. $Y^R_{u,v}$ computes a weighting/score per tie (u, v) in the SN structure.

6. Popularization and awareness to a relatively less popular but interesting research topic (breakup/rift prediction) in SNA.

7. Detailed evaluation reports with respect to classic objective functions used for classification and regression tasks in ML.

The research work presented hereafter is organized as follows: section 2.2 reviews a selected list of related literature. Section 2.3 formally defines the problem statement as well as the details of our proposed framework with respect to ClasReg. Section 2.4 expatiates on the datasets and materials used to facilitate our experiments. Section 2.5 documents

the detailed results of our benchmark experiments. Section 2.6 captures our discussions, with respect to ClasReg's performance, in relation to the benchmark models (or baselines) across standard social graph datasets. Section 2.7 spotlights the potential applications of our proposed methodology (ClasReg). Section 2.8 highlights the known assumptions and/or limitations with regard to our framework and experiments; and our proposed future work.

2.2 Historical Foundation and Related Literature

2.2.1 Strength of Ties

Measuring and evaluating the strength of ties is one of the earliest approaches toward SNA in the 20^{th} century; and it still remains an interesting subtopic in SNA with respect to the 21^{st} century. Strength-of-Ties methodologies have been employed in resolving varying problems in SNA such as Information Diffusion, Churn Prediction, Community Detection, Link Prediction, etc. This approach uses *predictors of tie strength* (such as kinship, neighbor, co-worker, etc.); which are computed based on the evaluation of one or more *indicators of tie strength* (such as the closeness, intensity, frequency, duration, or topics of an interaction) [3]. This approach entails using empirical methods, to evaluate the indicators of tie strength, with the goal of yielding outputs within the domain of real numbers, $x_i : 0 \leq x_i \leq 1$. Thus, x_i denotes the strength of the tie or relationship between a given pair of actors in a SN structure. Ideally, x_i is a probability-distribution value such that a strong tie or relationship is defined as x_i approaches 1; and a weak tie or interaction is defined as x_i approaches 0. Thus, we have employed the following Strength-of-Ties methodologies as our

baselines, viz: Common Neighbor Index [10], Jaccard Coefficient [11], Adamic Adar Index [12], Preferential Attachment [13], Resource Allocation Index [14], and Katz Index [15].

2.2.2 Churn Prediction

Many published research in the domain of SNA have attempted to solve the problem of Churn Prediction using Strength-of-Ties methodology. In this regard, these authors [16, 17, 18, 19, 20] have essentially proposed or used existent metric(s) to quantify the adhesive force binding ties or relationships in SN structures. These metrics are usually probability distributions which generate values, $\{x_i : 0 \leq x_i \leq 1\}$, where x_i denotes the strength existing between a given tie. Thus, a stipulated threshold (e.g. $x_i = 0.5$) is considered as the cut-off point to ascertain which (weak) ties or relationships possess tendencies for churning (e.g. $x_i < 0.5$); and those (strong) ties that may still need to be maintained (e.g. $x_i \geq 0.5$). However, this approach toward Churn Prediction does not seem effective and accurate; because weak ties are actually relevant, influential, beneficial, and advantageous to the pairs of associated actors and the SN structure. Consequently, these literature [21, 7, 22, 4, 5, 23, 2] have studied and highlighted the importance and strength of weak ties in SN structures. Recent approaches [24, 25, 26, 27, 28] to Churn Prediction have employed ML and/or DL techniques toward resolving this problem. In this regard, essential features and patterns are extracted from a set of churned actors; and these extracted features and patterns are used to train a ML or DL classifier, such that it can effectively learn how to flag or identify actors that possess high tendency for churning.

2.2.3 Link Prediction

Link Prediction (LP) still remains an open research problem with regard to SNA. LP involves studying and analyzing SN structures with the goal of predicting newer edges as potential ties or relationships for possible expansion of the SN structure. Over time, several approaches have been proposed toward resolving this problem in SNA. Hence, the following overview highlights the significant breakthroughs in the regard.

LP via Strength of Ties Approach

Strength-of-Ties methodologies [29, 30, 31, 32] is virtually the oldest approach toward resolving link prediction problem in SNA. Applying Strength-of-Ties approach to link prediction problems usually involve setting up a threshold (e.g. $x_i = 0.5$). In this regard, a tie is predicted as a link if its strength is above the threshold. Otherwise, if the strength of a tie is below the given threshold; such a tie is not predicted as a link.

LP via Graph Embeddings (GE) Approach

Over time with respect to the recent advancement in DL, graph embedding [33, 34] approaches have proven to be much more effective, with respect to varying objective functions, in resolving several open problems in SNA (especially the problem of LP). Graph Embeddings are robust DL methodologies which can be applied to several graph structures, irrespective of size and complexity.

LP via GE Approach based on Matrix Factorization Basically, algorithms in this category encode the ties or relationships (existing between actors) into a data matrix such as adjacency matrix, Laplacian matrix, probability matrix, similarity matrix, etc. This data matrix serves as input to the Matrix-Factorization algorithms. Thus, the input data matrix is processed via factorization techniques, such as eigenvalue decomposition, gradient descent technique, etc., to generate an embedding vector for every actor or node contained in the data matrix [35, 36]. Consequently, these generated embedding vectors are used as features of the graph representation for training and validating the Matrix-Factorization baselines or models on link prediction tasks. In this regard, we have employed the following Matrix-Factorization models as benchmark models, viz: Laplacian Eigenmap [37], Singular Value Decomposition [38], Graph Factorization [39], High-Order Proximity preserved Embedding [40], and Graph Representations [41].

LP via GE Approach based on Random Walk(s) These algorithms randomly sample or select ties/paths, within a given social graph, based on the notion of random walks or traversals over the graph. These randomly sampled ties are used as a minimal representation of the whole graph structure. Accordingly, the randomly sampled ties are processed by the Random-Walk algorithms with the goal of generating embedding vectors for the constituent actors or nodes of the graph; and still preserve the inherent properties of the whole graph structure. The generated (graph-preserving) embeddings are used as training features for

the Random-Walk models with respect to link prediction problems. Thus, we have employed the following Random-Walk models as baselines herein, viz: DeepWalk [42], Node2vec [43], and Struc2vec [44].

LP via GE Approach based on Neural Networks Neural networks are the building blocks of DL models. Neural network models comprising a hidden-layer depth or size of 3 and above are considered to be DL models [45, 46]. DL models constitute a subset of ML methodologies and frameworks. In recent times, DL models have been employed to solve complex and challenging problems in several domains (inclusive of SNA). Most DL models, when applied to a graph, absorb the entire graph structure. These DL models, which are essentially dimensionality-reduction and feature-extraction algorithms, reduce the representation of the graphs to low-dimensional vector spaces (embedding vectors). Correspondingly, these generated embedding vectors, which are uniquely mapped to every actor or node in the graph, are used as features of the graph for training the DL models on link prediction tasks over graphs. Hence, we have used the following Neural-Network (or DL) models as our benchmark models, viz: Large-scale Information Network Embedding [47], Structural Deep Network Embedding [48], Graph Auto-Encoders [49], and Representation Learning via Knowledge-Graph Embeddings and ConvNet [50, 51, 52].

2.3 Proposed Framework and Methodology

This section is subdivided into three (3) subsections, namely: subsection 2.3.1 (problem definitions), subsection 2.3.2 (proposed system framework), and subsection 2.3.3 (proposed methodology and algorithms).

2.3.1 Definitions of Problem

Definition 2.1. *Graph, G: This is a function, G, of/over a set of vertices or nodes,* $\{V : v_1, v_2, ..., v_n\}$*, and a set of edges or links,* $\{E : e_1, e_2, ..., e_n\}$*. Thus, a graph* $\equiv G(V, E)$*.*

Definition 2.2. *Shortest Path,* $shortest\_path(u, v)$*: This is the smallest traversal length from a source vertex or node/actor,* $u \in U$*, to a destination vertex or node/actor,* $v \in V$*. In this research, Breadth-First Search algorithm [53] is used for computing 'all_shortest_paths*(u, v)*' in unweighted graphs; while Dijkstra's algorithm [54] has been applied for the computation of 'all_shortest_paths*(u, v)*' in weighted graphs.*

Definition 2.3. *Social Network, SN: This is a tuple comprising a graph,* $G(V, E)$*; a metadata function,* f_V*, which extends the definition of the vertices' set by mapping it to a given set of attributes,* V'*; and a metadata function,* f_E*, which extends the definition of the edges' set by mapping it to a given set of attributes,* E'*. Thus,* $G(V, E) \subset SN$ *as expressed via equation 2.1. Also, Figure 2.1 depicts a SN structure.*

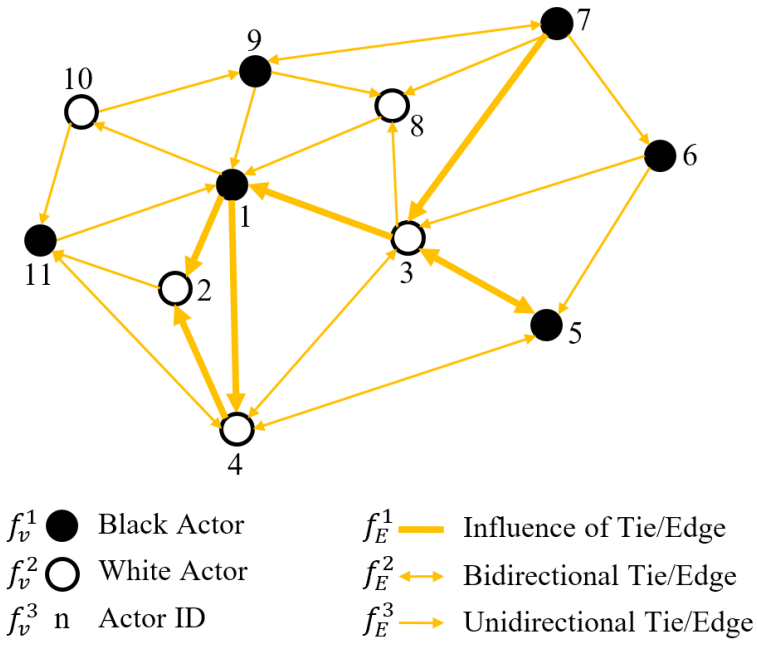

Figure 2.1: Social network (SN) structure.

$$SN = (G, f_V, f_E) \equiv (V, E, f_V, f_E)$$

$$f_V : V \to V' \qquad \text{vertices' metadata function} \tag{2.1}$$

$$f_E : E \to E' \qquad \text{edges' metadata function}$$

Definition 2.4. *Breakup/Rift: The universal set, $E = \{U \times V\}$, represents all possible ties or edges in a social graph, SN. Let $y_i = 1$ be a label denoting that there exists a relationship or edge ($e_i^+ \in E$) connecting vertices, u_m and v_n. Also, let label $y_i = 0$ denote that there is no relationship or edge ($e_i^- \in E$) between vertices, u_m and v_n. A breakup/rift is a function, $f(e_i^+)$, that evaluates all positive ties/edges to determine which e_i^+ (ground-truth edge) needs to transition or transmute to e_i^- (falsehood edge). Figure 2.2 represents the affected ties in the SN structure of Figure 2.1 after a breakup/rift.*

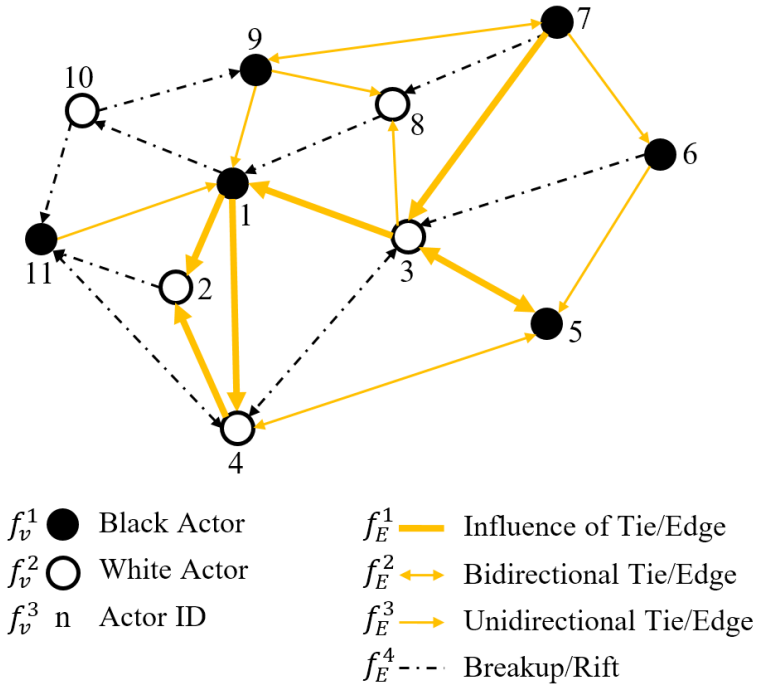

Figure 2.2: Breakup in a social network (SN) structure.

Definition 2.5. *First (1^{st}) Order Proximity: This is the first level of similarity estimation between any pair of vertices/nodes (u_m and v_n). Fundamentally, it is the weight, w_{mn}, of the edge/link existing between the vertices u_m and v_n. Examples include: edge weight(s), adjacency matrix, etc.*

Definition 2.6. *Second (2^{nd}) Order Proximity: This is the second level of similarity estimation between any pair of vertices (u_m and v_n). Let $P_m^1 = \{w_{m1}, ..., w_{m|V|}\}$ denote the 1^{st} order proximity existing between vertex, u_m, and all other vertices in G. Let $P_n^1 = \{w_{n1}, ..., w_{n|V|}\}$ denote the 1^{st} order proximity existing between vertex, v_n, and all other vertices in G. Thus, the 2^{nd} order proximity is the similarity between P_m^1 and P_n^1 which are the neighborhoods of vertices u_m and v_n, respectively. Examples include: Jaccard Coefficient, Cosine Similarity, etc.*

Definition 2.7. *Higher (k^{th}) Order Proximity: This is the k^{th} level of similarity estimation between any pair of vertices/nodes (u_m and v_n). Formally, it is defined as the similarity between P_m^{k-1} and P_n^{k-1} which denote the $k-1^{th}$ neighborhoods for the vertices u_m and v_n, respectively. Examples include: Katz Index, Google^{TM} PageRank, etc.*

Definition 2.8. *Graph Embedding: The embedding vector-space, X, generated by the embedding layer is based on a mapping function, f, expressed via equation 2.2. Thus, f projects the representation of the graph's vertices to a q-dimensional real space, $\mathbb{R}^q$ ($q \ll |V|$). Also, f ensures that the graph's attributes/properties, such as the 1^{st}, 2^{nd}, and some k^{th} order proximities existing between constituent edges (u_m, v_n), remain preserved via the homomorphism from V to X.*

$$f : V \rightarrow X \equiv v_n \mapsto x_n \in \mathbb{R}^q \qquad\qquad q \ll |V| \tag{2.2}$$

Definition 2.9. *Indegree (Indeg) Centrality: The Indegree (Indeg) of a vertex, u_m, is the number of edges that have u_m as their destination vertex. Thus, the Indegree Centrality (Indeg_Centr) of a vertex, u_m, is the standardized Indegree (Indeg) score computed as formalized in equation 2.3.*

$$Indeg\_Centr = \frac{Indeg(u_m)}{|V| - 1} \tag{2.3}$$

Definition 2.10. *Outdegree (Outdeg) Centrality: The Outdegree (Outdeg) of a vertex, u_m, is the number of edges that have u_m as their source vertex. Thus, the Outdegree Centrality (Outdeg_Centr) of a vertex, u_m, is the standardized Outdegree (Outdeg) score computed as formalized in equation 2.4.*

$$Outdeg\_Centr = \frac{Outdeg(u_m)}{|V| - 1} \tag{2.4}$$

Definition 2.11. *(Google^{TM}) PageRank: The PageRank (R) of a vertex is the normalized prestige (or eigenvector centrality) score that denotes the importance and influence of the vertex by evaluating the quality and quantity of its edges. The PageRank, $R(u_m)$, of a vertex, u_m, can be formalized via equation 2.5 such that: L_u is the set of all vertices that have u_m as their destination vertex; and v_n is a n^{th} vertex in L_u. Also, $R(u_m) \in [0, 1]$.*

$$R(u_m) = \sum_{v_n \in L_u} \frac{R(v_n)}{Outdeg(v_n)} \tag{2.5}$$

Definition 2.12. *Euclidean Distance, $d(a, b)$: The is the straight-line proximity measure between point vector, $a = (a_1, ..., a_n)$, and point vector, $b = (b_1, ..., b_n)$, in an Euclidean n-space as defined via equation 2.6.*

$$d(a,b) \equiv d(b,a) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2} \equiv \sqrt{(b_1 - a_1)^2 + ... + (b_n - a_n)^2} \tag{2.6}$$

Definition 2.13. *Cosine Similarity, $sim(a,b)$: This is a measure of similarity, which computes the cosine of the angle, $cos(\Theta)$, between two non-zero vectors, a and b, as defined via equation 2.7. Also, $sim(a,b) \in [0,1]$.*

$$sim(a,b) \equiv cos(\Theta) = \frac{a \cdot b}{\parallel a \parallel \cdot \parallel b \parallel} \equiv \frac{\sum_{i=1}^{n}(a_i \cdot b_i)}{\sqrt{\sum_{i=1}^{n}(a_i^2)} \cdot \sqrt{\sum_{i=1}^{n}(b_i^2)}} \tag{2.7}$$

Definition 2.14. *Kendall's Tau Correlation Coefficient, $\tau(a,b)$: This is a rank-correlation measure which computes the similarity between two nonlinear quantities, a and b, as defined via equation 2.8 [55]. P is the number of concordant pairs; Q is the number of discordant pairs; T is the number of ties only in a; and U is the number of ties only in b. Also, $\tau(a,b) \in [-1,1]$.*

$$\tau(a,b) = \frac{P - Q}{\sqrt{(P + Q + T) \times (P + Q + U)}} \tag{2.8}$$

2.3.2 Proposed System Architecture

Table 2.1 summarizes the system architecture of ClasReg (as depicted via Figure 2.3).

Table 2.1: Summary of ClasReg's system architecture.

Layer	Input	Output	Details
Preprocessing	Social graph	E^- and E^+ in discrete form	Section 2.3.3
RL/FL (Embedding Vector)	E^- and E^+	Vector-space embedding, $x_i \in \mathbb{R}^{256}$, per actor in E^- and E^+	Section 15 (*GE via Neural Networks*)
Classification	Vector-space embedding per actor in E^- and E^+	Ties classification, Y^C, and Link Prediction	Section 15 (*Logistic Regression classifier*)
Regression	Vector-space embedding per actor in E^- and E^+	Weighting/Score, Y^R, per tie in E^- and E^+	Section 15 (*Logistic Regression model*)
Inference or Heuristic Engine	Y^C and Y^R	IDs of breakup ties/edges, $e_i^U = y_i^U \mapsto e_i \in E$	Section 15 (*Computational Logic*)

2.3.3 Proposed Methodology and Algorithms

The basic framework of ClasReg is comprised of a preprocessing layer, a RL or FL (embedding-vector) layer, a classification layer, a regression layer, and an inference/heuristic engine.

Preprocessing Layer

This layer succeeds the input layer with respect to the system architecture of ClasReg. The preprocessing layer is responsible for performing preliminary processing on each input (social graph) dataset, so as to ensure that it is formatted appropriately for machine utilization.

Given an input social network, SN, then let V be the set of all actors/vertices; and let $E \subseteq \{V \times V\}$ be the set of all possible ties/edges in SN. Thus, u_m and v_n denote a source vertex and a destination vertex in E, respectively. The 2-tuple, (u_m, v_n), represents a possible edge comprising a pair of vertices, $V : U \subset V \; \forall \, u_m, v_n \in V$.

Firstly, the preprocessing layer ensures that all constituent actors or vertices of the SN

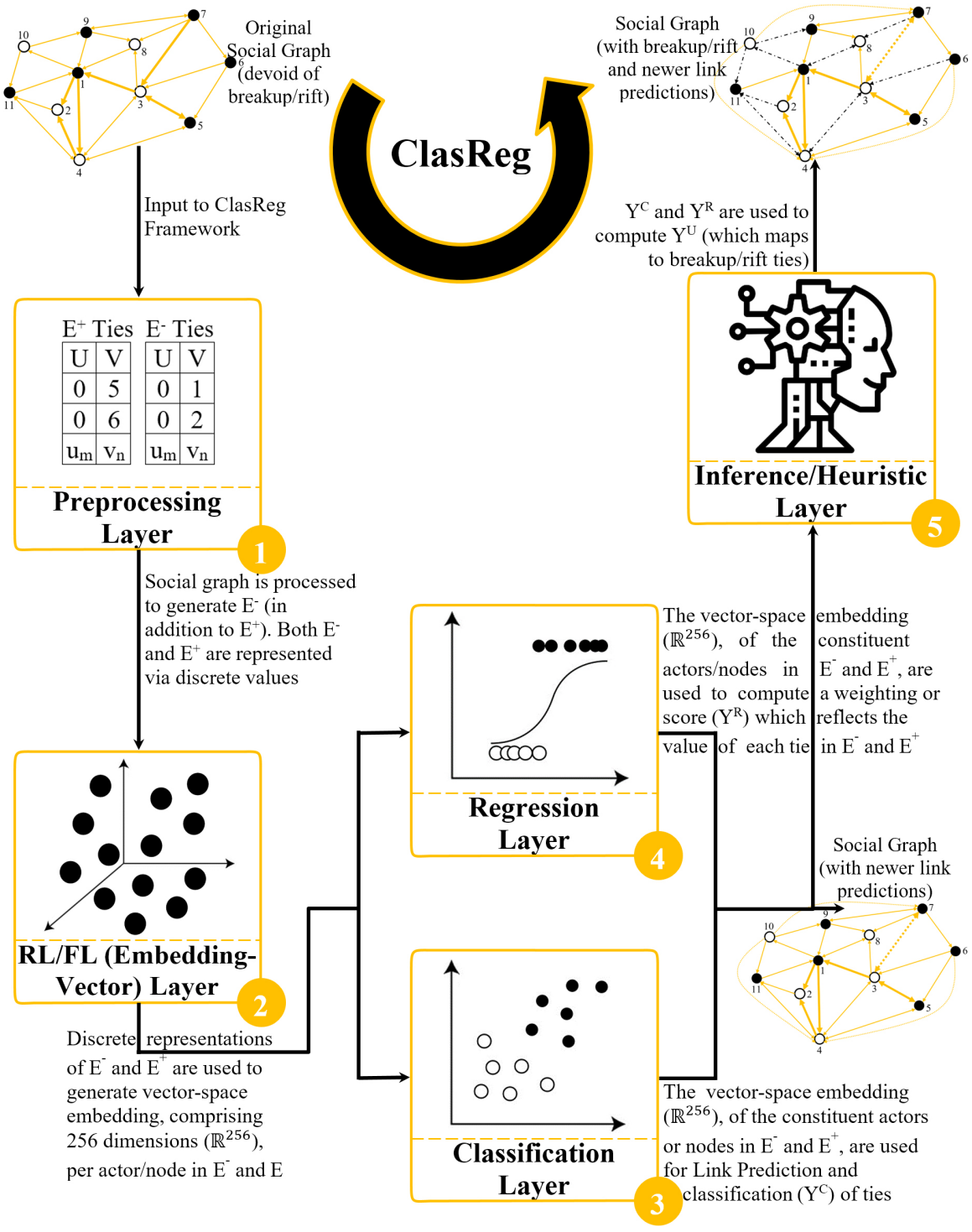

Figure 2.3: System architecture of ClasReg.

structure are represented in numerical form. Most SN datasets, in their respective natural forms, are comprised of data in mixed or several formats (i.e. records may be represented in

both textual and numerical formats). However, ML models can only operate on numerical data. Hence, an injective (one-to-one) operation, f_{in}, is one of the functionality available in the preprocessing layer. This function, f_{in}, maps every unique non-numerical data, present in each SN dataset, to its respective discrete-data (integer) codomain. The injective function, f_{in}, ensures that no semantic loss is incurred during its transcoding operation. f_{in} is formalized via equation 2.9.

$$f_{in} : non\_numeric \longmapsto discrete \tag{2.9}$$

Secondly, the preprocessing layer ensures that the constituent actors or vertices, u_m and v_n, of every tie or edge in the edgelist, $e_i \in E$, is derivable from the actors' list or nodelist, $V : U \subset V$, as expressed in equation 2.10.

$$\begin{aligned} &E = \{E^+, E^-\} \subseteq \{V \times V\} && \text{set of all possible ties in SN} \\ &E : e_i \mapsto (u_m, v_n) \in E && u_m \in U \subset V \text{ and } v_n \in V \\ &V \vdash u_m, v_n \quad \forall\, (u_m, v_n) \in E \end{aligned} \tag{2.10}$$

Thirdly, the preprocessing layer is responsible for generating negative or falsehood (-ve) ties or edges, E^-, to complement the positive or ground-truth (+ve) ties or edges, E^+, with respect to training, testing, and validation in ML and DL models herein. In a bid to appropriately train our ClasReg model, we require falsehood ties (E^-) to supplement the given ground-truth ties (E^+). Equation 2.11 formally describes the computation of these prerequisite ties/edges. Also, Algorithm 2.1 outlines the algorithmic approach towards

computing and/or generating the falsehood ties (E^-). On one hand, the number of shortest paths, $all\_shortest\_paths(u, v)$ existing for a given tie or link (in a graph or SN structure), is computed using Breadth-First Search algorithm [53]; if the graph is an unweighted graph. On the other hand, the number of shortest paths, $all\_shortest\_paths(u, v)$ existing for any given tie in a graph or SN structure, is computed using Dijkstra's algorithm [54]; if the graph is a weighted graph. Refer to Definition 2.2 for additional details with regard to $all\_shortest\_paths(u, v)$.

$$
\begin{aligned}
&E^+ : e_i \mapsto \mathbb{B} = 1 && \text{ground-truth (+ve) ties} \\
&E^+ \subset \{U \times V\} \\
&E^- : e_i \mapsto \mathbb{B} = 0 && \text{falsehood (-ve) ties} \\
&E^- \subset \{V \times V\} : f_o(V \times V) = 0 \\
&f_o = |all\_shortest\_paths(u_m, v_n)| = \begin{cases} 0, & 0 \text{ paths between } u_m \text{ and } v_n \\ \geq 1, & 1/\text{more paths between } u_m \text{ and } v_n \end{cases} \\
&E^- = E^- - E^+ && |E^-| = |E^+| \text{ for training}
\end{aligned}
\tag{2.11}
$$

Therefore, the output of the preprocessing are positive (+ve) ties, E^+, and negative (-ve) ties, E^-, which are transferred to the RL or FL layer.

Algorithm 2.1 Computation of Negative (-ve) or Falsehood Ties

Input: $\{V, E^+\} \equiv$ {Actors, Ground-Truth Ties}
Output: $\{E^-\} \equiv$ {Falsehood Ties}
Data: $SN \equiv$ Social Network dataset

```
Function falseTiesGenerator(V, E+):
    trueTies = E+                                   // Ground-Truth Ties (𝔹 = 1)
    falseTies = array()                  // Array to store Falsehood Ties (𝔹 = 0)
    pathsCount = 0                              // Falsehood Tie:  pathsCount = 0
    for u ∈ V do
        for v ∈ V do
            if ((u, v) ∉ trueTies) and (u ≠ v) then
                k = all_shortest_paths(u, v)                  // See Definition 2.2
                if len(k) <= pathsCount then
                    falseTies.append(u, v)
            if |falseTies| >= |trueTies| then
                break                                  // Break out of inner for loop
        if |falseTies| >= |trueTies| then
            break                                      // Break out of outer for loop
    return falseTies                              // E- ≡ falseTies ≡ Falsehood Ties
```

RL or FL (Embedding-Vector) Layer

This is the second subcomponent of ClasReg's basic framework, and it comes after the preprocessing layer. The inputs to this RL or FL layer are positive (+ve) ties, E^+, and negative (-ve) ties, E^-. The RL or FL layer is responsible for generating vector-space embedding per vertex, $f : v_i \mapsto x_i \in \mathbb{R}^q$, in the graph. ClasReg's embedding-vector layer is based on *Neural Networks* approach as explained in subsection 2.2.3. Also, the hyperparameter configuration of this layer is tabulated in Table 2.8. In this regard, the output of this embedding-vector layer is a vector-space embedding, $x_i \in \mathbb{R}^q$ within a q-dimensional real space, for every actor or vertex in the social graph.

Basically, the embedding-vector layer computes a conditional probability, $Pr(v_n|u_m) \forall (u_m, v_n) \in$

E, which defines the probability of coexistence or correlation between a reference target vertex, v_n, and a given source vertex, u_m. Thus, the goal of the embedding-vector layer is to find the best fit values for the parameters, Θ, such that the conditional probabilities, $Pr(v_n|u_m;\Theta)$ are maximized as defined in expression 2.12.

$$\arg\max_{\theta} \prod_{(u_m,v_n)\in E} Pr(v_n|u_m) \tag{2.12}$$

With regard to expression 2.12, we can model this objective function of the embedding-vector layer using a softargmax (or normalized exponential) function as defined in equation 2.13. The elements constituting the parameters, Θ, include: $u_m, v_n, k, |V|$, etc. Hence, we need to find the best set of Θ that maximizes the product (objective) function in expression 2.12.

$$Pr(v_n|u_m) = \frac{exp(v_n \cdot u_m)}{\sum_{k=1}^{|V|} exp(v_k \cdot u_m)} \tag{2.13}$$

Consequently, we compute the logarithm of the objective function, in expression 2.12, which translates to equation 2.14 upon application of the rule: 'logarithm of a product'. This yields a summation function.

$$\arg\max_{\theta} \prod_{(u_m,v_n)\in E} logPr(v_n|u_m) \;=\; \arg\max_{\theta} \sum_{(u_m,v_n)\in E} logPr(v_n|u_m) \tag{2.14}$$

Furthermore, the summation function in equation 2.14 can be rewritten as expressed in

equation 2.15.

$$\arg\max_{\theta} \sum_{(u_m,v_n)\in E} logPr(v_n|u_m) = \arg\max_{\theta} \sum_{(u_m,v_n)\in E} log\frac{exp(v_n \cdot u_m)}{\sum_{k=1}^{|V|} exp(v_k \cdot u_m)} \tag{2.15}$$

We can further simplify equation 2.15, via application of the rule: 'logarithm of a quotient', to obtain equation 2.16.

$$\arg\max_{\theta} \sum_{(u_m,v_n)\in E} logPr(v_n|u_m) = \sum_{(u_m,v_n)\in E} (log\, exp(v_n \cdot u_m) - log \sum_{k=1}^{|V|} exp(v_k \cdot u_m)) \tag{2.16}$$

However, it is important to observe that it is computationally expensive to maximize the objective function of the embedding-vector layer using either equation 2.15 or 2.16. This is due to the summation expression, $\sum_{k=1}^{|V|} exp(v_k \cdot u_m)$, present in equations 2.15 and 2.16; since $k = 1, ..., |V|$ may span to several thousands [56]. Therefore, to make the computation of the objective function computationally feasible, either hierarchical softmax [57] or negative sampling [57] can be applied.

Classification Layer

This is the third subcomponent of ClasReg's basic framework, and it succeeds the RL or FL layer. The inputs to this layer are real vector-space embedding for each source actor, $X_u : u_m \mapsto x_m$, and its corresponding destination actor, $X_v : v_n \mapsto x_n$, with respect to each tie or edge in $E^+ \subset E$ and $E^- \subset E$. These input embedding vectors, $X_u \in \mathbb{R}^q$ and $X_v \in \mathbb{R}^q$, represent the regressors (or independent/feature variables) of the ClasReg model. Just as the name implies, the classification layer is trained to effectively and efficiently learn how

to categorize ties either as +ve ties, $E^+ \subset E$, or as -ve ties, $E^- \subset E$. Thus, the output of this classification layer lies within a boolean domain, $\{0, 1\}$.

In this layer, each tie (2-tuple comprising a source actor, u_m, and a destination actor, v_n) is represented via the linear concatenation of the vector-space embedding per constituent actor as expressed in equation 2.17.

$$\begin{aligned} X_u : u_m \mapsto x_m \in \mathbb{R}^{256} \qquad & \text{source actor embedding vector} \\ X_v : v_n \mapsto x_n \in \mathbb{R}^{256} \qquad & \text{destination actor embedding vector} \\ x_i^{512} \in X^{512} = X_u^{256} \,||\, X_v^{256} \qquad & \text{concatenation of both embedding vectors} \\ E : x_i^{512} \mapsto e_i \qquad & \text{mapping embedding vector to each tie} \end{aligned} \tag{2.17}$$

The vector-space embedding for each constituent actor of a given tie is expressed in 256 dimensions. The vector-space embedding for each tie, $x_i^{512} \in X^{512}$, constitutes the independent or feature variables of the classification layer. Consequently, the dependent or target variable of the classification layer is a outcome label, $y_i^C \in Y^C = Y_{u,v}^C$, which denotes the signage of each tie or edge. Hence, we have used, $y_i^C = 1$, to be a label denoting that there exists a relationship or edge ($e_i^+ \in E$) connecting vertices, u_m and v_n. Also, we have used the label, $y_i^C = 0$, to denote that there exists no relationship or edge ($e_i^- \in E$) between vertices, u_m and v_n. We have employed Logistic-Regression classifier [58] for the classification tasks herein; and equation 2.18 formalizes the fundamental operations of the

classification layer.

$$x_i^{512} = u_m \mapsto x_m \in \mathbb{R}^{256} \,||\, v_n \mapsto x_n \in \mathbb{R}^{256} \qquad \text{vector concatenation}$$

$$\begin{gathered} y_i^C \equiv y_{u,v}^C = \mathbb{B} = \{y_i^C \in \mathbb{Z} | 0, 1\} = f_C(x_i^{512}) \equiv f_C(x_m^1, ..., x_m^{256}, x_n^1, ..., x_n^{256}) \\ f_C : X^{512} \to Y^C \equiv (x_i^{512}) \mapsto y_i^C \\ f_C(x_i^{512}) = \frac{1}{1 + exp^{x_i^{512}}} \equiv \frac{exp^{x_i^{512}}}{exp^{x_i^{512}} + 1} \end{gathered} \tag{2.18}$$

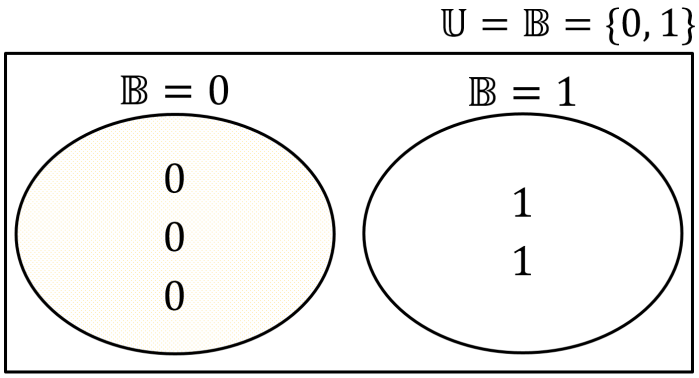

Figure 2.4: Dummy classification result in the classification layer.

In a bid to gain insight into the working principle of the classification layer, consider a dummy SN structure comprising the following ties: $E = \{e_1, e_2, ..., e_i\}$. Thus, each tie is represented via a 2-tuple, (u_m, v_n), such that $e_i \in E : (u_m, v_n) \mapsto y_i^C$. Our explanation is further expressed via Figure 2.4 and Table 2.2.

Table 2.2: Dummy classification result in the classification layer.

E	Classification Task			
	U	V	$U\|\|V$	Y^C
e_1	x_1^{256}	x_1^{256}	x_1^{512}	0
e_2	x_2^{256}	x_2^{256}	x_2^{512}	1
e_3	x_3^{256}	x_3^{256}	x_3^{512}	1
e_4	x_4^{256}	x_4^{256}	x_4^{512}	0
e_5	x_5^{256}	x_5^{256}	x_5^{512}	0
e_i	x_m^{256}	x_n^{256}	x_i^{512}	y_i^C

Regression Layer

This is the fourth subcomponent in ClasReg's basic framework. This layer succeeds the embedding-vector layer, and it operates in parallel with the classification layer. Since this layer is parallel to the classification layer, its inputs are same as that of the classification layer. With regard to every tie in $E^+ \subset E$ and $E^- \subset E$, the vector-space embedding of a source actor, $X_u : u_m \mapsto x_m \in \mathbb{R}^{256}$, and its respective destination actor, $X_v : v_n \mapsto x_n \in \mathbb{R}^{256}$, constitute the inputs to the regression layer. Also, these inputs act as the regressors (or independent/feature variables) of ClasReg. These regressors are employed in the computation of a unique prediction score, $y_i^R \in Y^R = Y_{u,v}^R$, per tie or edge in the SN structure. The regression layer effectively computes and learns how to associate a weighting/score to every tie in the social graph. This prediction score (or weighting) reflects the quality of the relationship existing between any two (2) associated actors in a tie/edge. Therefore, the output of the regression layer lies within a real space, $\mathbb{R}$.

The regression layer, in opposition to the classification layer, represents each tie or edge via the individual 256-dimensional embedding vector corresponding to its constituent source actor ($x_m \in \mathbb{R}^{256}$) and destination actor ($x_n \in \mathbb{R}^{256}$). The independent or feature variables of the regression layer are $x_m \in \mathbb{R}^{256}$ and $x_n \in \mathbb{R}^{256}$. The dependent or target variable, with respect to the regression layer, is a real quantity: $y_i^R \in Y^R$; and it is computed via the objective (or prediction score) function, $Y_{u,v}^R$, of the regression layer. Thus, this is formally expressed via equation 2.19.

$$
\begin{aligned}
g_i &= 1^{st} \text{ Order Influence}(u_m) \times k^{th} \text{ Order Influence}(u_m) \times \text{embedding}(u_m) \\
g_i &= Outdeg\_Centr(u_m) \times PageRank(u_m) \times x_m \\
h_i &= 1^{st} \text{ Order Influence}(v_n) \times k^{th} \text{ Order Influence}(v_n) \times \text{embedding}(v_n) \\
h_i &= Indeg\_Centr(v_n) \times PageRank(v_n) \times x_n \\
y_i^R &= Y_{u,v}^R = \frac{g \cdot h}{\parallel g \parallel \cdot \parallel h \parallel} \equiv \frac{\sum_{i=1}^{\mathbb{Z}} (g_i \cdot h_i)}{\sqrt{\sum_{i=1}^{\mathbb{Z}} (g_i^2)} \cdot \sqrt{\sum_{i=1}^{\mathbb{Z}} (h_i^2)}}
\end{aligned}
\tag{2.19}
$$

One of the novelties of our research work herein lies in the computation of the variables, g and h, with respect to equation 2.19. The embedding vectors, x_m and x_n, tend to preserve (as much as possible) some properties of the SN structure such as the 1^{st} order proximity, 2^{nd} order proximity, and some k^{th} order proximities with regard to the ties/edges and the entire graph structure (as explained in Defintion 2.8). Furthermore, our novelty involves the infusion of 1^{st} order actor/node influence (Degree_Centrality) [59], and k^{th} order actor/node influence (PageRank) to each actor's embedding vector (x_m and x_n). The infusion of these factors of actor/node influence resulted in a relatively better reflection of the tie/relationship quality existing between any given pair of actors in the SN structure. The discussion in section 2.6 herein sheds light, with respect to the results of our experiments, on the importance of the factors of actor/node influence to each embedding vector.

The output of the objective (or prediction score) function, $Y_{u,v}^R$, represents the dependent or target variable of the regression layer. $y_i^R \in Y^R = Y_{u,v}^R$ for any pair of actors, which

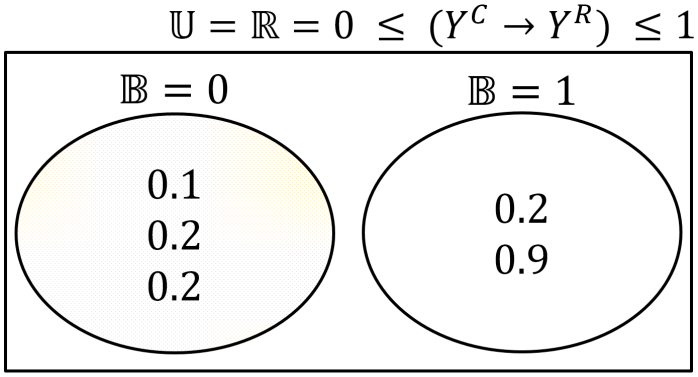

Figure 2.5: Dummy regression result in the regression layer.

constitute a tie in the SN structure, lies within the boundaries of 0 and 1 inclusively, $[0, 1]$. Theoretically, $y_i^R = Y_{u,v}^R$, can be computed as shown in equation 2.19. Empirically, a Logistic-Regression model [58] can be trained for the regression task(s) in this layer. Equation 2.20 shows the primary operations of the regression layer.

$$\begin{aligned} y_i^R \equiv y_{u,v}^R = [0,1] = \{y_i^R \in \mathbb{R} | 0 \leq y_i^R \leq 1\} = f_R(g_i \in \mathbb{R}^{256}, h_i \in \mathbb{R}^{256}) \\ f_R : (g_i \in \mathbb{R}^{256}, h_i \in \mathbb{R}^{256}) \mapsto y_i^R \end{aligned} \tag{2.20}$$

To shed some light on the working principle of the regression layer, consider the dummy SN structure exemplified in the classification layer. Thus, let our dummy SN structure be comprised of a set of ties, $E = \{e_1, e_2, ..., e_i\}$, such that every tie, $e_i \in E : (u_m, v_n) \mapsto y_i^R$. In this regard, Figure 2.5 and Table 2.3 buttresses our explanation.

Table 2.3: Dummy regression result in the regression layer.

E	Regressoin Task					
	U	V	G	H	Y^C	Y^R
e_1	x_1^{256}	x_1^{256}	g_1^{256}	h_1^{256}	0	0.1
e_2	x_2^{256}	x_2^{256}	g_2^{256}	h_2^{256}	1	0.9
e_3	x_3^{256}	x_3^{256}	g_3^{256}	h_3^{256}	1	0.2
e_4	x_4^{256}	x_4^{256}	g_4^{256}	h_4^{256}	0	0.2
e_5	x_5^{256}	x_5^{256}	g_5^{256}	h_5^{256}	0	0.2
e_i	x_m^{256}	x_n^{256}	g_i^{256}	h_i^{256}	y_i^C	y_i^R

Inference/Heuristic (Engine) Layer

This is the fifth and last subcomponent of ClasReg's basic framework. This layer comes after the classification layer and regression layer. The inputs to this layer are the combined outputs from the classification and regression layers, respectively. In other words, the inputs to the inference/heuristic engine, with respect to every tie in $E^+ \subset E$ and $E^- \subset E$, are the outcome labels from the classification layer, $y_i^C \in Y^C = Y_{u,v}^C$; and the prediction scores from the regression layer, $y_i^R \in Y^R = Y_{u,v}^R$. Essentially, the inference or heuristic layer is responsible for flagging ties in $E^+ \subset E$ that should be considered for breakup with respect to the SN structure. Hence, the output of the inference or heuristic layer are negative ties ($E^- \subset E$); which represent the breakup predictions of our proposed framework, ClasReg.

In the inference or heuristic layer, every tie or edge (with respect to the SN structure) is jointly represented via a boolean integer, $Y_{u,v}^C = y_i^C \in Y^C = \mathbb{B} = 0 \vee 1$, and real value, $Y_{u,v}^R = y_i^R \in Y^R = [0, 1]$. Thus, the independent or feature variables in this layer are $Y_{u,v}^C = y_i^C \in Y^C$ and $Y_{u,v}^R = y_i^R \in Y^R$. The dependent or target variable of the inference/heuristic layer is the output variable, $y_i^U \in Y^U$, of a logical operation. $y_i^U \in Y^U$ denotes the prediction score of potential +ve tie(s) that should be considered for breakup. $y_i^U \in Y^U \mapsto e_i \in E$ returns the identity of a tie or edge in $E^+ \subset E$, which has been suggested for breakup. Basically, $y_i^U \in Y^U$ is computed by finding ties or edges, $e_i \in E$, that bear the same prediction score, $Y_{u,v}^R = y_i^R \in Y^R$, but with different outcome labels, $Y_{u,v}^C = y_i^C \in Y^C$. Thereafter, a +ve tie or edge, $e_i^+ \in E^+ \subset E$, is considered for breakup

if the magnitude of the prediction score (dependent/target variable), $y_i^U \in Y^U$, is inclined toward the boolean integer of $\mathbb{B} = 0$. The logical operations of the inference or heuristic layer is explained via equation 2.21 and Algorithm 2.2.

$$
\begin{aligned}
&y_i^C \in Y^C : e_i \in E \mapsto y_{u,v}^C \\
&y_i^R \in Y^R : e_i \in E \mapsto y_{u,v}^R \\
&\quad Y^U = (Y^R \rightarrow (Y^C = \mathbb{B} = 0)) \cap (Y^R \rightarrow (Y^C = \mathbb{B} = 1)) \\
&\quad Y^U = (Y^R | Y^C = 0) \cap (Y^R | Y^C = 1) \\
&y_i^U \in Y^U \Leftrightarrow |Y^R : Y^C = 0| > |Y^R : Y^C = 1| \\
&e_i^U \in E = y_i^U \mapsto e_i \in E \qquad \text{return identity of unlinked tie/edge}
\end{aligned}
\tag{2.21}
$$

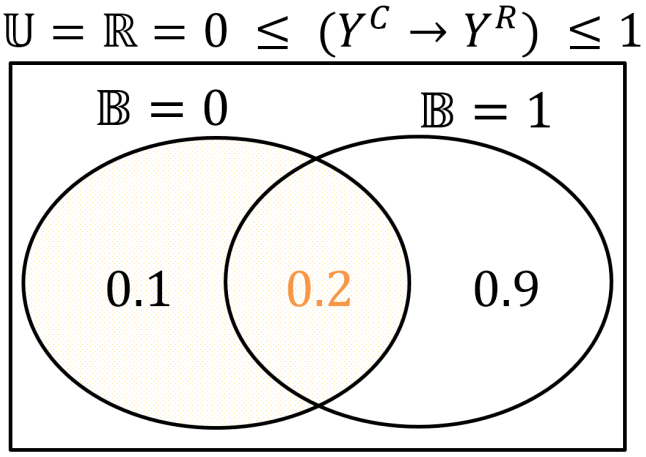

Figure 2.6: Dummy inference/heuristic result in the inference or heuristic layer (type-1).

Furthermore, we have used Figures 2.6 and 2.7 as well as Table 2.4 to depict the working principle of the inference or heuristic layer. Correspondingly, applying the same dummy SN structure, which we have used for exemplification in the classification and regression layers of ClasReg's methodology; the edgelist is defined as: $E = \{e_1, e_2, ..., e_i\}$. Therefore, each

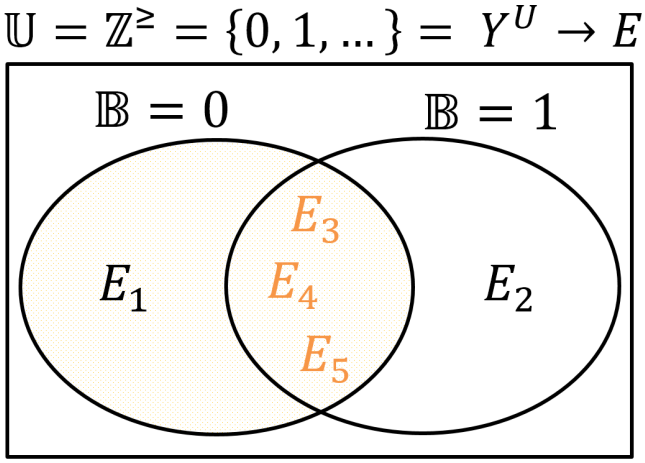

Figure 2.7: Dummy inference/heuristic result in the inference or heuristic layer (type-2).

tie is represented via the expression: $e_i \in E : (y_i^C, y_i^R) \mapsto y_i^U$.

Table 2.4: Dummy inference/heuristic result in the inference or heuristic layer.

E	Inference/Heuristic Task		
	Y^C	Y^R	Y^U
e_1	0	0.1	
e_2	1	0.9	
e_3	1	0.2	0.2
e_4	0	0.2	0.2
e_5	0	0.2	0.2
e_i	y_i^C	y_i^R	y_i^U

Algorithm 2.2 Inference or Heuristic Operations

Input: $\{Y^C, Y^R, E\} \equiv$ {Classifier Labels, Prediction Scores, Ties $(E^+ \cup E^-)$}
Output: $\{e_i^U \in E\} \equiv$ {Breakup/Rift Ties}
Data: $SN \equiv$ Social Network dataset

```
16  Function inferenceOperations(Y^C, Y^R, E):
        /* Array initialization                                          */
17      trueTiesScore = array()                          // Y^R for Ground-Truth Ties
18      falseTiesScore = array()                         // Y^R for Falsehood Ties
19      breakupTiesScore = array()                       // Y^R for Breakup/Rift Ties
20      breakupTies = array()                            // Records of Breakup/Rift Ties
21      /* m = len(E) = |E|                                              */
22      for i = 0 to m do
23          if y_i^C == 0 then
24              falseTiesScore.append(y_i^R)
25          else if y_i^C == 1 then
26              trueTiesScore.append(y_i^R)
27      breakupTiesScore = list(falseTiesScore.intersection(trueTiesScore))
        if |breakupTiesScore| == 0 then
28          breakupTies = NULL
29      else
30          for i = 0 to m do
31              if y_i^R ∈ breakupTiesScore then
32                  if falseTiesScore.count(y_i^R) > trueTiesScore.count(y_i^R) then
33                      breakupTies.append(e_i ∈ E)
34      return breakupTies                          // e_i^U ∈ E ≡ IDs of Breakup/Rift Ties
```

2.4 Materials and Methods

Subsection 2.4.1 as well as Table 2.5 gives a detailed overview of the benchmark social graph datasets employed herein for our experiments and evaluations. Subsection 2.4.2 outlines all the dependencies (libraries and software) which were used to facilitate our research, proposed framework, and experiments. Subsection 2.4.3 clearly lists all the baselines (or benchmark models) employed herein; and the hyperparameter configurations of our proposed model, ClasReg. Subsection 2.4.4 formally defines all the objective functions imple-

mented herein to evaluate our proposed framework, ClasReg, and the benchmark models. Subsection 2.4.5 describes the content as well as the directory structure of our remote (GitHub) code repository, with regard to ClasReg and the baselines employed herein.

2.4.1 Datasets

Table 2.5: Benchmark datasets.

Dataset	Actors	Ties	Classes → {label: 'description'}
CiteSeer	3,312	4,732	{C1: 'Agents', C2: 'Artificial Intelligence', C3: 'Databases', C4: 'Information Retrieval', C5: 'Machine Learning', C6: 'Human-Computer Interaction'}
Cora	2,708	5,429	{C1: 'Case_Based', C2: 'Genetic_Algorithms', C3: 'Neural_Networks', C4: 'Probabilistic_Methods', C5: 'Reinforcement_Learning', C6: 'Rule_Learning', C7: 'Theory'}
Internet-Industry	219	631	{C1: 'Content Sector', C2: 'Infrastructure Sector', C3: 'Commerce Sector'}
PubMed-Diabetes	19,717	44,338	{C1: 'Diabetes Mellitus - Experimental', C2: 'Diabetes Mellitus - Type 1', C3: 'Diabetes Mellitus - Type 2'}
Terrorists-Relation	851	8,592	{C1: 'Colleague', C2: 'Congregate', C3: 'Contact', C4: 'Family'}
Zachary-Karate	34	78	{C1: 'Community 1', C2: 'Community 2', C3: 'Community 3', C4: 'Community 4'}

CiteSeer dataset [60] This is a social network of scientific publications and citations comprising $3,312$ articles, categorized against six (6) classes, with an egdelist comprising $4,732$ citation links. Each CiteSeer publication has a feature set established by a binary (0 or 1) word vector which denotes the absence or presence of the corresponding word in the feature set, respectively. Thus, $3,703$ unique words constitute the CiteSeer words' dictionary [61].

Cora dataset [60] This is a social graph of scientific publications and citations comprising 2,708 publications, distributed over seven (7) classes, with a citation magnitude of 5,429 links. Each publication in the Cora dataset possesses a feature set defined by a binary (0 or 1) word vector which signifies the absence or presence of the corresponding word in the feature set, respectively. The Cora words' dictionary [61] comprises 1,433 unique words.

Internet-Industry Partnership dataset [62] This dataset was made public by Valdis E. Krebs [62] is a social graph representing partnerships amongst 219 companies in the Internet industry. These companies are classified into three (3) categories; and there exist 631 links between the constituent companies of the dataset. Thus, two companies are linked via an edge if they have announced a joint venture, strategic alliance, or other partnership within the timespan of 1998 to 2001 [63].

PubMed-Diabetes dataset [64] This dataset represents a social (network) graph of 19,717 scientific publications which are related to diabetes cases; and they are classified against three (3) classes. This dataset is comprised of 44,338 citation relationships between the selected publications. Every PubMed-Diabetes article is associated with a representation/feature vector, which is defined by a binary (0 or 1) word vector denoting the absence or presence of a diabetes-related word in the representation vector, respectively. Thus, the PubMed-Diabetes words' dictionary is constituted of 500 unique diabetes-related words.

Terrorists-Relationship dataset [65] This consists of data pertaining to ties or relationships amongst selected terrorists. This social network is comprised of 851 unique profiles corresponding to known terrorists, and they are labelled against four (4) categories of relationship between the terrorists. Moreover, the egdelist of this social graph comprises $8,592$ links. Each actor (terrorist) has a representation set established by a binary (0 or 1) feature vector; which denotes the absence or presence of the corresponding feature in the representation set, respectively. Thus, $1,224$ unique features make up the features' dictionary of this dataset.

Zachary-Karate dataset [66] This is one of the earliest and most basic social graph dataset. It is an undirected social network of members whom constitute a university's karate club as compiled by Wayne Zachary [67]. The membership strength of this karate club is 34 actors (nodes). Also, 78 ties/relationships (edges) exist between the club members or actors. Within the Zachary-Karate (club) social network, there exist four (4) distinct communities (classes).

2.4.2 Materials and software

Table 2.6: Materials and software.

Library/Software	Description
TensorFlow and Keras [68]	Deep Learning library
BioNEV [69]	Deep Learning and Graph Embeddings framework
Scikit-Learn [58]	Machine Learning library
EvalNE [70]	Machine Learning and Graph Embeddings framework

The following libraries and/or software outlined in Table 2.6 were employed herein to

facilitate the development of ClasReg, and carry out our benchmark evaluations with respect to the baselines herein.

2.4.3 Baselines and Hyperparameters

Table 2.7: Baselines (or benchmark models).

Task/Job	Materials	Baselines
Strength of Ties	EvalNE [70] Scikit-Learn [58]	Adamic Adar Index (Adamic) [12] Common Neighbor Index (CommNeigh) [10] Jaccard Coefficient (Jaccard) [11] Katz Index (Katz) [15] Preferential Attachment Index (PrefAttach) [13] Resource Allocation Index (ResAlloc) [14]
Link Prediction	BioNEV [69] TensorFlow [68] Keras [68]	DeepWalk (DeepWalk) [42] Graph Auto-Encoders (GAE) [49] Graph Factorization (GraFac) [39] Graph Representations (GraRep) [41] High-Order Proximity preserved Embedding (HOPE) [40] Laplacian Eigenmap (LapEigen) [37] Large-scale Information Network Embedding (LINE) [47] Node2vec (Node2vec) [43] Structural Deep Network Embedding (SDNE) [48] Struc2vec (Struc2vec)[44] Singular Value Decomposition (SVD) [38]

Table 2.7 shows all the baselines employed herein for benchmarking (or evaluating the performance) of ClasReg.

Table 2.8: Configuration of hyperparameters for ClasReg.

Generic Hyperparameters	RL/FL Layer Hyperparameters
Training Set: 80% Test Set: 20% Classifier Optimizer: $lbfgs$	Learning Rate: $1.0 * 10^{-3}$ Optimizer: $Nadam$ Epochs: $1.0 * 10^{2}$ Activation: $sigmoid$ Embedding Dimension: 256 Regularization: $(L1 = 0.0, L2 = 0.0)$

ClasReg, as proposed herein, is a DL and neural-network framework which extends the work of [50, 51, 52]. Table 2.8 specifically outlines the configuration values for the hyperparameters applied with respect to the design, development and implementation of ClasReg.

2.4.4 Benchmark Objective Functions

The following classic objective functions, with respect to ML and DL tasks or jobs, have been applied herein for evaluating the performance of ClasReg against state-of-the-art benchmark models (baselines).

Precision (PC) This objective function is also known as the Positive Predictive Value (PPV); and it is formally defined via equation 2.22. The value of PC lies within the boundaries of 0 (no precision) and 1 (optimum precision), inclusively.

$$\text{PC} = PPV = \frac{\text{TP}}{\text{TP} + \text{FP}} = [0, 1] \tag{2.22}$$

Recall (RC) Other names for this objective function are Hit Rate (HR) or True Positive Rate (TPR). It is formally defined via equation 2.23; and the value of RC falls within the

boundaries of 0 (worst) and 1 (best), inclusively.

$$\text{RC} = TPR = \frac{\text{TP}}{\text{TP} + \text{FN}} = [0, 1] \tag{2.23}$$

Accuracy (AC) This is a measure of how close a predicted value is to the ground truth (actual value). It is formally defined via equation 2.24; such that its value falls within the boundaries of 0 (worst) and 1 (best), inclusively.

$$\text{AC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = [0, 1] \tag{2.24}$$

F-measure or F1-score (F1) This function measures the standardized accuracy of an experiment. Thus, it is the harmonic mean of the PC function and the RC function. Formally, it is defined via equation 2.25. The value of F1 lies within the boundaries of 0 (worst) and 1 (perfect PC and RC), inclusively.

$$\text{F1} = \frac{2}{\text{PC}^{-1} + \text{RC}^{-1}} = 2 \times \frac{\text{PC} \times \text{RC}}{\text{PC} + \text{RC}} = [0, 1] \tag{2.25}$$

Matthews Correlation Coefficient (MCC) This function is also known as the Phi Coefficient; and it measures the quality of any binary classification experiment. MCC is regarded as a balanced measure because it takes into consideration data imbalance in datasets during its evaluation. It is formally defined via equation 2.26; and its value lies within the boundaries of -1 (worst prediction correlation), 0 (neutral prediction correlation),

and +1 (perfect prediction correlation), inclusively.

$$\text{MCC} = \frac{(\text{TP} \times \text{TN}) - (\text{FP} \times \text{FN})}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} = [-1, 0, 1] \tag{2.26}$$

Note: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

Area Under the Receiver Operating Characteristic Curve (RO) Basically, this function computes the Area Under the Receiver Operating Characteristic Curve (ROC AUC) with regard to a classification task/experiment. Formally, it is defined via equation 2.27. The value of RO lies within the boundaries of 0 (worst) and 1 (perfect), inclusively.

$$\begin{aligned} &\text{TPR}(T) : T \rightarrow y\_axis \\ &\text{FPR}(T) : T \rightarrow x\_axis \\ &\text{RO} = \int_{x=0}^{1} \text{TPR}(\text{FPR}^{-1}(x))dx \end{aligned} \tag{2.27}$$

Note: T (Threshold parameter/variable), True Positive Rate (TPR), and False Positive Rate (FPR).

Training Time (TT) This is the average number of seconds (wall-clock time) that elapses during the course/period of passing the training proportion/data, usually 80% of the benchmark dataset, over a ML or DL model.

Table 2.9: Description of our remote source code repository with regard to ClasReg and the baselines (benchmark models) used herein for benchmark experiments.

SN	**Subject:** Remote (GitHub) URL/Link Description
1.	**ClasReg:** https://github.com/bhevencious/ClasReg Homepage of the code repository for ClasReg
2.	**custom_classes:** https://github.com/bhevencious/ClasReg/tree/master/custom_classes Subdirectory containing dependencies (class files) for ClasReg
3.	**breakup_and_link_prediction.py:** https://github.com/bhevencious/ClasReg/blob/master/breakup_and_link_prediction.py ClasReg's source code for breakup prediction and link prediction
4.	**generic_datasets:** https://github.com/bhevencious/ClasReg/tree/master/generic_datasets Subdirectory containing all the benchmark datasets employed herein with regard to our experiments and evaluations
5.	**BioNEV:** https://github.com/bhevencious/BioNEV Library for the graph-embedding baselines (DeepWalk, GAE, GraFac, GraRep, HOPE, LapEigen, LINE, Node2vec, SDNE, Struc2vec, and SVD)
6.	**EvalNE:** https://github.com/bhevencious/EvalNE Library for the strength-of-ties baselines (Adamic, CommNeigh, Jaccard, Katz, PrefAttach, and ResAlloc)

2.4.5 Reproducibility

Table 2.9 herein summarizes the structure of our remote (GitHub) repository, which contains all the source codes, with respect to the experiments and evaluations carried out in this research. Additional details with regard to the installation, usage, and content of ClasReg is available via: https://github.com/bhevencious/ClasReg/blob/master/README.md

2.5 Experiments and Results

Tables 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, and Figure 2.8 represent the results of our experiments.

Table 2.10: Link Prediction experiment results with respect to CiteSeer dataset (objective functions *vs* baselines).

	Model	CiteSeer Dataset							Points
		PC	**RC**	**AC**	**F1**	**MCC**	**RO**	**TT(s)**	
Strength of Ties	Adamic	**0.99**	0.46	0.73	0.63	0.54	0.73	0.08	1
	CommNeigh	**0.99**	0.46	0.73	0.63	0.54	0.73	**0.02**	2
	Jaccard	**0.99**	0.45	0.72	0.62	0.53	0.73	0.10	1
	Katz	0.87	0.02	0.51	0.03	0.07	0.56	0.34	0
	PrefAttach	0.79	0.55	0.70	0.65	0.43	0.79	0.05	0
	ResAlloc	**0.99**	0.45	0.72	0.62	0.53	0.73	0.09	1
Graph Embeddings	DeepWalk	0.63	0.63	0.63	0.63	0.26	0.66	19.36	0
	GAE	0.59	0.64	0.60	0.62	0.20	0.65	19.74	0
	GraFac	0.57	0.57	0.57	0.57	0.14	0.60	16.06	0
	GraRep	0.66	0.64	0.65	0.65	0.31	0.67	16.15	0
	HOPE	0.65	0.57	0.64	0.61	0.27	0.66	20.42	0
	LapEigen	0.56	0.59	0.56	0.57	0.12	0.57	19.07	0
	LINE	0.57	0.53	0.57	0.55	0.13	0.59	17.35	0
	Node2vec	0.63	0.63	0.63	0.63	0.26	0.66	16.82	0
	SDNE	0.67	0.51	0.63	0.58	0.27	0.67	17.29	0
	Struc2vec	0.66	0.54	0.63	0.59	0.26	0.67	23.40	0
	SVD	0.65	0.59	0.64	0.62	0.27	0.66	19.74	0
	ClasReg	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	196.69	6

Table 2.11: Link Prediction experiment results with respect to Cora dataset (objective functions *vs* baselines).

	Model	Cora Dataset							Points
		PC	**RC**	**AC**	**F1**	**MCC**	**RO**	**TT(s)**	
Strength of Ties	Adamic	**0.99**	0.45	0.72	0.62	0.53	0.72	0.20	1
	CommNeigh	**0.99**	0.45	0.72	0.62	0.53	0.72	0.04	1
	Jaccard	**0.99**	0.42	0.71	0.59	0.51	0.72	0.20	1
	Katz	0.65	0.02	0.50	0.03	0.04	0.53	0.48	0
	PrefAttach	0.78	0.47	0.67	0.59	0.37	0.75	**0.03**	1
	ResAlloc	**0.99**	0.42	0.71	0.59	0.51	0.72	0.21	1
Graph Embeddings	DeepWalk	0.61	0.62	0.61	0.61	0.22	0.64	12.65	0
	GAE	0.56	0.69	0.57	0.62	0.14	0.59	12.40	0
	GraFac	0.55	0.53	0.55	0.54	0.10	0.57	11.84	0
	GraRep	0.63	0.60	0.62	0.61	0.24	0.67	12.52	0
	HOPE	0.62	0.57	0.61	0.60	0.22	0.66	12.54	0
	LapEigen	0.59	0.61	0.60	0.60	0.19	0.63	12.23	0
	LINE	0.62	0.56	0.61	0.59	0.21	0.64	13.29	0
	Node2vec	0.59	0.62	0.60	0.61	0.20	0.63	14.19	0
	SDNE	0.67	0.50	0.63	0.57	0.26	0.66	14.78	0
	Struc2vec	0.66	0.55	0.63	0.60	0.27	0.68	16.18	0
	SVD	0.62	0.57	0.61	0.60	0.23	0.64	12.26	0
	ClasReg	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	219.97	6

Table 2.12: Link Prediction experiment results with respect to Internet-Industry dataset (objective functions *vs* baselines).

	Model	Internet-Industry Dataset							Points
		PC	**RC**	**AC**	**F1**	**MCC**	**RO**	**TT(s)**	
Strength of Ties	Adamic	0.78	0.60	0.71	0.68	0.44	0.73	0.03	0
	CommNeigh	0.78	0.60	0.71	0.68	0.44	0.73	**0.01**	1
	Jaccard	0.76	0.54	0.69	0.63	0.39	0.69	0.03	0
	Katz	0.67	0.05	0.51	0.09	0.06	0.66	**0.01**	1
	PrefAttach	**0.91**	0.70	0.81	0.79	0.64	0.88	**0.01**	2
	ResAlloc	0.83	0.52	0.71	0.64	0.44	0.73	0.03	0
Graph Embeddings	DeepWalk	0.74	0.63	0.70	0.68	0.41	0.77	0.16	0
	GAE	0.61	0.73	0.63	0.67	0.27	0.68	0.14	0
	GraFac	0.73	0.71	0.72	0.72	0.44	0.78	0.16	0
	GraRep	0.74	0.67	0.72	0.70	0.44	0.78	0.17	0
	HOPE	0.71	0.68	0.70	0.70	0.41	0.77	0.13	0
	LapEigen	0.71	0.69	0.71	0.70	0.41	0.78	0.11	0
	LINE	0.66	0.63	0.65	0.65	0.31	0.70	0.13	0
	Node2vec	0.73	0.66	0.71	0.69	0.42	0.77	0.14	0
	SDNE	0.74	0.58	0.69	0.65	0.39	0.76	0.17	0
	Struc2vec	0.74	0.67	0.72	0.70	0.44	0.81	0.13	0
	SVD	0.75	0.67	0.72	0.71	0.45	0.79	0.17	0
	ClasReg	0.86	**0.88**	**0.87**	**0.87**	**0.74**	**0.96**	14.17	5

Table 2.13: Link Prediction experiment results with respect to PubMed-Diabetes dataset (objective functions *vs* baselines).

	Model	PubMed-Diabetes Dataset							Points
		PC	RC	AC	F1	MCC	RO	TT(s)	
Strength of Ties	Adamic	**1.00**	0.25	0.63	0.40	0.38	0.63	0.30	1
	CommNeigh	**1.00**	0.25	0.63	0.40	0.38	0.63	0.10	1
	Jaccard	**1.00**	0.25	0.63	0.40	0.38	0.63	0.40	1
	Katz	0.14	0.00	0.50	0.00	-0.03	0.50	7.51	0
	PrefAttach	0.87	0.72	0.81	0.79	0.62	0.91	**0.08**	1
	ResAlloc	**1.00**	0.25	0.63	0.40	0.38	0.63	0.30	1
Graph Embeddings	DeepWalk	0.60	0.55	0.59	0.57	0.18	0.63	2603.06	0
	GAE	0.58	0.71	0.60	0.64	0.21	0.66	1982.52	0
	GraFac	0.56	0.56	0.56	0.56	0.12	0.59	900.13	0
	GraRep	0.69	0.70	0.69	0.69	0.38	0.75	802.93	0
	HOPE	0.68	0.68	0.68	0.68	0.35	0.72	806.98	0
	LapEigen	0.67	0.73	0.69	0.70	0.38	0.73	809.46	0
	LINE	0.74	0.79	0.76	0.77	0.52	0.82	803.01	0
	Node2vec	0.59	0.60	0.59	0.60	0.18	0.62	800.69	0
	SDNE	0.75	0.67	0.72	0.71	0.45	0.80	803.69	0
	Struc2vec	0.76	0.81	0.78	0.79	0.56	0.84	803.83	0
	SVD	0.63	0.59	0.62	0.61	0.25	0.66	802.37	0
	ClasReg	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	3487.22	6

Table 2.14: Link Prediction experiment results with respect to Terrorists-Relation dataset (objective functions *vs* baselines).

	Model	Terrorists-Relation Dataset							Points
		PC	RC	AC	F1	MCC	RO	TT(s)	
Strength of Ties	Adamic	0.98	0.98	0.98	0.98	0.96	**1.00**	0.58	1
	CommNeigh	0.97	0.97	0.97	0.97	0.94	**1.00**	0.07	1
	Jaccard	**1.00**	**0.99**	0.99	0.99	0.98	**1.00**	0.46	3
	Katz	0.69	0.04	0.51	0.08	0.07	0.51	0.10	0
	PrefAttach	0.70	0.70	0.70	0.70	0.39	0.72	**0.04**	1
	ResAlloc	0.99	**0.99**	0.99	0.99	0.98	**1.00**	0.48	2
Graph Embeddings	DeepWalk	0.76	0.83	0.78	0.79	0.57	0.84	1.89	0
	GAE	0.72	0.85	0.76	0.78	0.53	0.79	1.77	0
	GraFac	0.67	0.69	0.67	0.68	0.35	0.73	1.56	0
	GraRep	0.78	0.85	0.80	0.81	0.60	0.85	2.26	0
	HOPE	0.74	0.79	0.76	0.77	0.52	0.83	2.64	0
	LapEigen	0.52	0.56	0.52	0.54	0.05	0.52	1.98	0
	LINE	0.66	0.68	0.66	0.67	0.33	0.71	1.84	0
	Node2vec	0.72	0.79	0.74	0.75	0.48	0.80	1.69	0
	SDNE	0.63	0.83	0.67	0.71	0.36	0.69	1.51	0
	Struc2vec	0.67	0.73	0.69	0.70	0.38	0.73	1.55	0
	SVD	0.77	0.83	0.79	0.80	0.59	0.85	1.50	0
	ClasReg	**1.00**	**0.99**	**1.00**	**1.00**	**0.99**	**1.00**	227.94	6

Table 2.15: Link Prediction experiment results with respect to Zachary-Karate dataset (objective functions *vs* baselines).

	Model	Zachary-Karate Dataset							Points
		PC	**RC**	**AC**	**F1**	**MCC**	**RO**	**TT(s)**	
Strength of Ties	Adamic	0.64	0.60	0.63	0.62	0.27	0.72	0.01	0
	CommNeigh	0.64	0.60	0.63	0.62	0.27	0.67	**0.00**	1
	Jaccard	0.62	0.53	0.60	0.57	0.20	0.57	0.01	0
	Katz	0.00	0.00	0.47	0.00	-0.19	0.39	**0.00**	1
	PrefAttach	0.83	0.67	0.77	0.74	0.54	0.80	**0.00**	1
	ResAlloc	0.82	0.60	0.73	0.69	0.48	0.72	**0.00**	1
Graph Embeddings	DeepWalk	0.77	0.67	0.73	0.71	0.47	0.79	0.02	0
	GAE	0.67	0.53	0.63	0.59	0.27	0.75	0.02	0
	GraFac	0.79	0.73	0.77	0.76	0.53	0.80	0.03	0
	GraRep	0.75	0.60	0.70	0.67	0.41	0.84	0.02	0
	HOPE	0.75	0.60	0.70	0.67	0.41	0.85	**0.00**	1
	LapEigen	**0.85**	0.73	0.80	0.79	0.61	0.83	0.02	1
	LINE	0.75	0.60	0.70	0.67	0.41	0.78	0.02	0
	Node2vec	0.75	0.60	0.70	0.67	0.41	0.76	0.02	0
	SDNE	0.75	0.60	0.70	0.67	0.41	0.87	0.02	0
	Struc2vec	**0.85**	0.73	0.80	0.79	0.61	0.89	0.02	1
	SVD	0.57	0.27	0.53	0.36	0.08	0.76	0.02	0
	ClasReg	0.82	**0.93**	**0.87**	**0.88**	**0.75**	**0.92**	3.15	5

Figure 2.8: Correlation map of the prediction score or weighting proposed herein, Y^R, in comparison to other popular correlation-coefficient scoring functions.

Table 2.16: Breakup/Rift prediction experiment results with respect to the benchmark datasets. The results tabulated herein are based on the entire dataset.

Dataset	**Number of Breakup/Rift Predictions**
CiteSeer	21
Cora	31
Internet-Industry	1
PubMed-Diabetes	281
Terrorists-Relation	15
Zachary-Karate	0

2.6 Discussions

The objective functions employed herein for evaluating the performance of ClasReg, in comparison with state-of-the-art baselines (benchmark models), are described in subsection 2.4.4. Traversing along the rows (row-wise), with regard to Tables 2.10, 2.11, 2.12, 2.13, 2.14, and 2.15; each row represents a baseline or benchmark model and its associated objective-function scores. Also, traversing across the columns (column-wise), with regard to Tables 2.10, 2.11, 2.12, 2.13, 2.14, and 2.15; each column holds the objective-function scores with respect to each baseline or benchmark model (inclusive of ClasReg proposed herein). The values along the rows of each objective-function column (excluding the columns *Model* and *Points*), with regard to Tables 2.10, 2.11, 2.12, 2.13, 2.14, and 2.15; fall within $[0, 1]$ where 0 denotes worst performance/score, and 1 denotes best performance/score. For each objective-function column, the best performance (based on our experiment results) is highlighted with a **bold** font. In this regard, a ranking or scoring technique is employed along the rows of the *Points* column. This is used to aggregate the number of times each

benchmark model (baseline) performed best with respect to the objective functions (PC, RC, AC, F1, MCC, RO, and TT) used for comparative evaluation during our experiments. Thus, the highest possible *Points* attainable by each benchmark model herein is 7 points (that is, 1 point for each objective function). Comparatively, for all the experimental evaluations per benchmark dataset, any baseline that records the highest *Points* signifies an ideal model/technique for Link Prediction task on such dataset(s).

Furthermore, we have proposed a unique weighting formula or prediction score, Y^R, for computing and evaluating the *bond of the relationship* existing between any given pair of actors. Subsection 15 herein contains the formalism and details of our proposed weighting formula or prediction score. Basically, the structural formula for computing Y^R proposed herein, is based on the generic formula for Cosine Similarity as formalized via equation 2.7. However, the formalism of Y^R differs from Cosine Similarity. This difference is seen in the fact that each pair of vectors, embedding-representation vectors: x_m and x_n used in the computation of Y^R, are instilled with additional factors of node influence. Hence, the unique feature of Y^R pertains to the joint introduction of 1^{st} order node influence (via Degree_Centrality), as well as k^{th} order node influence (via PageRank), to the embedding vector representation (x_m and x_n) of each constituent actor/node in a relationship/edge. The embedding vector representation of every actor in a given SN structure, intrinsically/inherently contains and preserves factors of tie proximity existing in the SN structure. Therefore, the significance of our proposed and additional factors of actor/node

influence, with regard to each embedding-representation vector, can be seen in Figure 2.8. The correlation coefficients/values represented in Figure 2.8 are computed based on Spearman's Rank Correlation Coefficient. Hence, a positive relationship (similarity) is denoted as the correlation coefficient approaches +1; and a negative relationship (dissimilarity) is denoted as the correlation coefficient approaches −1. We have evaluated the effectiveness and quality of Y^R, in comparison with Cosine Similarity as well as other popular correlation coefficients used for analyzing and examining nonlinear relationships, against the output of ClasReg's classification layer (Y^C). From the correlation map of Figure 2.8, we can see that our proposed weighting function, Y^R, outperforms Cosine Similarity on CiteSeer and Zachary-Karate datasets. Generally, Y^R slightly surpasses other common correlation-coefficient functions. Although, we believe we can still further improve on the performance and effectiveness of Y^R in these tasks.

From the experiment results tabulated in Table 2.10, Table 2.11, Table 2.12, Table 2.13, Table 2.14, and Table 2.15; with regard to the benchmark datasets employed herein, we can clearly observe that our hybrid model, ClasReg, surpasses other baselines in the aspect of effectiveness (PC, RC, AC, F1, MCC, and RO). However, in the aspect of efficiency (TT), it can be noticed that ClasReg lags behind other baselines. Consequently, there exists a trade-off (effectiveness vs efficiency) with respect to the application of ClasReg for Link Prediction and Breakup Prediction problems in SN structures. Also, we presume that this trade-off may be insignificant, owing to the advancement in computing technologies, with

respect to the widespread availability of computing memory and CPU/GPU resources.

From our observations and findings, we have attributed the encouraging performance of ClasReg to the following factors, viz:

1. The generation of graph embeddings in ClasReg's FL/RL layer, with respect to each constituent actor/node in the input SN structure, is based on *Neural Networks* approach as described in subsection 2.2.3.

2. Only edgelist ties, $E(U, V)$, whose constituent actors are present in the nodelist, $(U, V) : \forall \{u_m, v_n\} \in V \subset G$, were used for training and validating/testing our model as well as other baselines.

3. The dimension of each output embedding-vector generated for every actor/node, with respect to each input SN structure, is exactly 256 dimensions.

4. A Logistic-Regression classifier was employed in the classification layer of ClasReg's framework. Basically, Logistic Regression models are relatively cheaper and easier to train as well as maintain. Moreover, they are relatively less prone to overfitting with respect to ML and DL.

5. In the application of ClasReg for Link Prediction tasks, an edge-sampling technique is employed. Thus, the embedding-vector generated for each tie/edge, is obtained via the linear concatenation of the embedding-vector corresponding to each constituent actor/node of every tie/edge in the SN structure. As a result, this yields an embedding

vector of 512 dimensions, per given tie/edge, with respect to the input SN structure. In this regard, refer to subsection 15 for additional details.

In addition, the graph-embedding baselines employed herein are bifunctional; in that each of them possesses both Node Classification and Link Prediction functionalities. Also, for each baseline (benchmark model) employed herein, the hyperparameter configuration was used in its default mode/standard.

Within the directory of each benchmark dataset, with reference to our code repository (see subsection 2.4.5): the details and log report of the predicted breakups, with respect to the affected ties/relationships in the SN structure, are available and accessible via the *.unlink* file present in the directory of each benchmark dataset. For example, considering Cora benchmark dataset, ClasReg's prediction of breakups is available and accessible via: https://github.com/bhevencious/ClasReg/blob/master/generic_datasets/Cora/Cora.unlink. Additionally, the details and performance report of ClasReg, with respect to link prediction task on each benchmark dataset, are aggregated and available via: https://github.com/bhevencious/ClasReg/blob/master/eval_log.txt. Further details and reports (such as generated embedding vectors, generated false ties/edges, etc.), with regard to the operations of ClasReg, can be accessed via: https://github.com/bhevencious/ClasReg/blob/master/README.md

2.7 Applications

As a result of the widespread diffusion of the Internet, which has directly resulted in the proliferation of virtual communities (or cybersocieties) via social media channels/platforms/services (Facebook^{TM}, Twitter^{TM}, etc.); breakup/rift and link/tie predictions have become vital and relevant with respect to the maintenance of such communities. On one hand, to mention but a few, the following are some advantages or merits derived from the application of breakup prediction(s), viz:

1. It can serve as a preemptive/preventive measure to thwart attacks (such as identity theft, cyberbullying, phishing, malware attack, etc.) in SN structures.

2. It can act as a control measure to mitigate the harm or damage caused by malicious attacks and/or cyberattacks to SN structures.

3. It can be used to control and maintain a relatively standard-sized social network structure.

4. Trustworthiness can be preserved and improved in any given SN structure via the application (1), (2), and/or (3) above.

5. Improve the quality and strength of ties/relationships, via homophily and transitivity, in SN structures.

On the other hand, the following are some selected aspects or areas [71] where link predic-

tion(s) can be applied, viz:

1. Anomaly detection in SN structures.

2. Community detection in SN structures.

3. Event detection in SN structures.

4. Recommendation systems.

5. Influence analysis in social networks.

2.8 Limitations, Conclusion, and Future Work

During our benchmark experiments herein, we did not tune the hyperparameters of the benchmark models (baselines). Thus, we evaluated these baselines using their respective default hyperparameter configurations. In summary, we have proposed a new, bifunctional, hybrid framework (ClasReg) herein; which is simultaneously capable of Breakup Prediction(s) and Link Prediction(s). Also, we have proposed a unique weighting formula, Y^R, for evaluating the quality of any given tie/relationship in a SN structure. Section 2.1 contains a detailed list of the novelties introduced in this research. At the moment, we are working on improving the performance of Y^R with respect to sparse social network structures. In the near future, Y^R shall be improved upon. Also, we shall expand the scope of our benchmark datasets, and our benchmark models (to possibly incorporate Knowledge-Graph Embeddings (VE) models too).

Acknowledgment

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

Bibliography

[1] C. C. Aggarwal, "An introduction to social network data analytics," in *Social Network Data Analytics*. New York, USA: Springer Science+Business Media, 2011, ch. 1, pp. 1–22.

[2] M. S. Granovetter, "The strength of weak ties," *American Journal of Sociology*, vol. 78, pp. 1360–1380, 1973.

[3] P. Marsden and K. E. Campbell, "Measuring tie strength," *Social Forces*, vol. 63, pp. 482–501, 1984.

[4] E. David and K. Jon, "Strong and weak ties," in *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge, England: Cambridge University Press, 2010, ch. 3, pp. 47–84.

[5] H. Memic, "Testing the strength of weak ties theory in small educational social networking websites," *Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces*, pp. 273–278, 2009.

[6] N. Takahashi and N. Inamizu, "Logical weakness of "the strength of weak ties"," *Annals of Business Administrative Science*, vol. 13, pp. 67–76, 2014.

[7] L. Pappalardo, G. Rossetti, and D. Pedreschi, ""how well do we know each other?" detecting tie strength in multidimensional social networks," *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 1040–1045, 2012.

[8] M. Diligenti, M. Gori, and C. Saccà, "Semantic-based regularization for learning and inference," *Artificial Intelligence*, vol. 244, pp. 143–165, 2017.

[9] S. A. Cook, "The complexity of theorem-proving procedures," *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, p. 151–158, 1971.

[10] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the Association for Information Science and Technology*, vol. 58, pp. 1019–1031, 2007.

[11] P. Jaccard, "The distribution of the flora in the alpine zone.1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[12] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Soc. Networks*, vol. 25, pp. 211–230, 2003.

[13] G. Yule, "A mathematical theory of evolution, based on the conclusions of dr. j. c. willis, f.r.s." *Philosophical Transactions of the Royal Society B*, vol. 213, pp. 21–87, 1925.

[14] T. Zhou, L. Lü, and Y. Zhang, "Predicting missing links via local information," *The European Physical Journal B*, vol. 71, pp. 623–630, 2009.

[15] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, pp. 39–43, 1953.

[16] M. A. D. Oliveira, K. Revoredo, and J. E. O. Luna, "Semantic unlink prediction in evolving social networks through probabilistic description logic," *2014 Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 372–377, 2014.

[17] M. N. Abd-Allah, A. Salah, and S. El-Beltagy, "Enhanced customer churn prediction using social network analysis," in *Proceedings of the 3rd Workshop on Data-Driven User Behavioral Modeling and Mining from Social Media (DUBMOD 2014)*, 2014.

[18] J. Perl, J. Kunegis, M. Thimm, and S. Sizov, "Decline - models for decay of links in networks," *ArXiv*, vol. abs/1403.4415, 2014.

[19] R. J. Oentaryo, E. Lim, D. Lo, F. Zhu, and P. K. Prasetyo, "Collective churn prediction in social network," *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 210–214, 2012.

[20] M. Karnstedt, M. Rowe, J. Chan, H. Alani, and C. Hayes, "The effect of user features on churn in social networks," in *Proceedings of the 3rd International Web Science Conference (WebSci 2011)*, 2011.

[21] P. D. Meo, E. Ferrara, G. Fiumara, and A. Provetti, "On facebook, most ties are weak," *Communications of the ACM*, vol. 57, pp. 78–84, 2014.

[22] J. Zhao, J. Wu, and K. Xu, "Weak ties: A subtle role in the information diffusion of online social networks," *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 82 1 Pt 2, p. 016105, 2010.

[23] M. Ruef, "Strong ties, weak ties and islands: structural and cultural predictors of organizational innovation," *Industrial and Corporate Change*, vol. 11, pp. 427–449, 2002.

[24] S. M. Kostic, M. Simic, and M. V. Kostic, "Social network analysis and churn prediction in telecommunications using graph theory," *Entropy*, vol. 22, p. 753, 2020.

[25] A. K. Ahmad, A. Jafar, and K. Aljoumaa, "Customer churn prediction in telecom using machine learning in big data platform," *Journal of Big Data*, vol. 6, pp. 1–24, 2019.

[26] A. Amin, F. Al-Obeidat, B. Shah, A. Adnan, J. Loo, and S. Anwar, "Customer churn prediction in telecommunication industry using data certainty," *Journal of Business Research*, vol. 94, pp. 290–301, 2019.

[27] A. D. Caigny, K. Coussement, and K. W. Bock, "A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees," *Eur. J. Oper. Res.*, vol. 269, pp. 760–772, 2018.

[28] C. Phadke, H. Uzunalioglu, V. Mendiratta, D. Kushnir, and D. Doran, "Prediction of subscriber churn using social network analysis," *Bell Labs Technical Journal*, vol. 17, pp. 63–76, 2013.

[29] D. Sheng, T. Sun, S. Wang, Z. Wang, and M. Zhang, "Measuring strength of ties in social network," in *Asia-Pacific Web Conference (APWeb)*, 2013.

[30] M. Gupte and T. Eliassi-Rad, "Measuring tie strength in implicit social networks," in *Proceedings of the 4th Annual ACM Web Science Conference (WebSci 2012)*, 2012.

[31] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th International Conference on World Wide Web (WWW 2010)*, 2010.

[32] I. Kahanda and J. Neville, "Using transactional information to predict link strength in online social networks," in *3rd International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2009.

[33] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, pp. 2724–2743, 2017.

[34] O.-J. Lee and J. Jung, "Story embedding: Learning distributed representations of stories based on character networks," *Artificial Intelligence*, vol. 281, p. 103235, 2020.

[35] H. Cai, V. Zheng, and K. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 1616–1637, 2018.

[36] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *ArXiv*, vol. abs/1705.02801, 2018.

[37] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, pp. 1373–1396, 2003.

[38] G. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, pp. 403–420, 2007.

[39] A. Ahmed, N. Shervashidze, S. M. Narayanamurthy, V. Josifovski, and A. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd International Conference on World Wide Web (WWW 2013)*, 2013.

[40] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[41] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM 2015)*, 2015.

[42] B. Perozzi, R. Al-Rfou', and S. Skiena, "Deepwalk: online learning of social representations," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, vol. abs/1403.6652, 2014.

[43] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 2016, pp. 855–864, 2016.

[44] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[45] J. Patterson and A. Gibson, Eds., *Deep Learning: A Practitioner's Approach.* Newton, MA: O'Reilly Media, Inc., 2017.

[46] I. G. Goodfellow, Y. Bengio, and A. C. Courville, Eds., *Deep Learning.* Cambridge, MA: MIT Press, 2017.

[47] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, 2015.

[48] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[49] T. Kipf and M. Welling, "Variational graph auto-encoders," *ArXiv*, vol. abs/1611.07308, 2016.

[50] B. Molokwu, S. B. Shuvo, N. Kar, and Z. Kobti, "Node classification and link prediction in social graphs using rlvecn," *32nd International Conference on Scientific and Statistical Database Management*, 2020.

[51] B. Molokwu and Z. Kobti, "Social network analysis using rlvecn: Representation learning via knowledge-graph embeddings and convolutional neural-network," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence, IJCAI*, 2020.

[52] B. Molokwu, S. B. Shuvo, N. Kar, and Z. Kobti, "Node classification in complex social graphs via knowledge-graph embeddings and convolutional neural network," *Computational Science – ICCS 2020*, vol. 12142, pp. 183 – 198, 2020.

[53] S. Skiena, "Sorting and searching," in *The Algorithm Design Manual*. London, England: Springer-Verlag London Limited, 2012, pp. 103–144.

[54] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[55] W. R. Knight, "A computer method for calculating kendall's tau with ungrouped data," *Journal of the American Statistical Association*, vol. 61, pp. 436–439, 1966.

[56] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *ArXiv*, vol. abs/1402.3722, 2014.

[57] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *ArXiv*, vol. abs/1310.4546, 2013.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[59] O. Skibski, T. Rahwan, T. P. Michalak, and M. Yokoo, "Attachment centrality: Measure for connectivity in networks," *Artificial Intelligence*, vol. 274, pp. 151–179, 2019.

[60] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, pp. 93–106, 2008.

[61] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. [Online]. Available: http://networkrepository.com

[62] V. E. Krebs, "Organizational adaptability quotient," in *IBM Global Services*, 2008.

[63] V. Batagelj, P. Doreian, A. Ferligoj, and N. Kejzar, Eds., *Understanding Large Temporal Networks and Spatial Networks: Exploration, Pattern Searching, Visualization and Network Evolution*. Hoboken, NJ: John Wiley & Sons, Inc., 2014.

[64] G. Namata, B. London, L. Getoor, and B. Huang, "Query-driven active surveying for collective classification," in *Proceedings of the Workshop on Mining and Learning with Graphs, MLG-2012*, 2012.

[65] B. Zhao, P. Sen, and L. Getoor, "Entity and relationship labeling in affiliation networks," in *Proceedings of the 23rd International Conference on Machine Learning, ICML*, 2006.

[66] J. Kunegis, "Konect: the koblenz network collection," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013. [Online]. Available: http://konect.cc/

[67] W. W. Zachary, "An information flow model for conflict and fission in small groups1," *Journal of anthropological research*, vol. 33, 11 1977.

[68] F. Chollet, Ed., *Deep Learning with Python*. Shelter Island, NY: Manning Publications, 2017.

[69] X. Yue, Z. Wang, J. Huang, S. Parthasarathy, S. Moosavinasab, Y. Huang, S. Lin, W. Zhang, P. Zhang, and H. Sun, "Graph embedding on biomedical networks: Methods, applications, and evaluations," *Bioinformatics*, 2020.

[70] A. Mara, J. Lijffijt, and T. D. Bie, "Evalne: A framework for evaluating network embeddings on link prediction," *ArXiv*, vol. abs/1901.09691, 2019.

[71] N. N. Daud, S. Hamid, M. Saadoon, F. Sahran, and N. Anuar, "Applications of link prediction in social networks: A review," *Journal of Network and Computer Applications*, vol. 166, p. 102716, 2020.

Chapter 3

Link Prediction & Node Classification in Social Network Analysis using Representation Learning via Knowledge-Graph Embeddings and Convolution Operations (RLVECO)

The research herein in this chapter partially stems from and overlaps with the research contribution in Chapter 2. In this regard, we have jointly examined the problems of Link Prediction and Node Classification in Social Network Analysis from a Deep Learning perspective. Therefore, we have proposed a hybrid framework which primarily harnesses the strengths of Knowledge-Graph Embeddings and Convolution Operations toward resolving the aforementioned problems in SNA. In real-world application scenarios, Link Prediction and Node Classification are essential tasks as they aid in maintaining the social structure and community structure, respectively, of a given society or Social Network structure.

3.1 Introduction

Earth comprises several biosystems and these systems are affected by interactions between a range of biotic and abotic factors that control the dynamics of these biosystems. Interactions within and/or between biosystems is a strategy for survival, and these can be modelled via social networks. With regard to the recent advances in AI, we can effectively model and analyze real-world complex systems as social networks structures using appropriate AI techniques. Considering the impact of Corona Virus Disease 2019 (COVID-19) pandemic, SNA can serve as a handy technique for modelling, analyzing, and predicting the impact of this viral disease. However, social networks are complex and non-static structures which pose analytical challenges to ML and DL models. Hence, analyzing and learning underlying knowledge from communities, comprising social actors and their existent social ties/relationships, using given sets of standard still remain a crucial research problem in SNA. With the goal of solving prediction-based and classification-related problems in social network structures, we have introduced a distinct framework (RLVECO) possessing biform learning layers. RLVECO aims at learning the intrinsic patterns of relationship binding spatial social actors using a twofold RL layer as opposed to most state-of-the-art approaches based on a sole RL layer.

On one hand, the prediction of links brings about correlations and/or ties formation which increases the tendency for transitivity in social networks. On the other hand, the classification of nodes induces the formation of cluster(s), and clusters give rise to homophily in

social networks. Our proposition is a unique clustering model based on an iterative learning approach; and it possesses the ability to learn the non-linear distributed representations [1] enmeshed in a social graph. Primarily, learning in RLVECO is achieved via semi-supervised training. The novelty of our work contains three (3) research contributions as stated below:

- Proposition of a DL-based and hybrid model, RLVECO, aimed at solving link prediction, node classification as well as community detection problems in SNA.
- Detailed benchmarking reports with respect to classic objective functions used for classification tasks.
- Comparative analyses, between RLVECO and state-of-the-art approaches, against standard real-world social networks.

The work presented hereafter is organized into 6 sections. The introductory section is succeeded by the literature review section. Section 3 elucidates our proposed methodology. Section 4 gives an overview of our materials and preprocessing techniques. Section 5 illustrates and discusses our experiments. Section 6 summarizes our research work.

3.2 Brief Review of Related Literature

In reference to link prediction in SNA, several state-of-the-art and VE methodologies have been proposed for resolving this open research problem. [2] proposed an embedding model, ComplEx, based on latent matrix factorization using Hermitian dot product over complex vector space. [3] in their research paper proposed another embedding model, ConvKB,

aimed at solving knowledge base completion problems. ConvKB uses a Convolutional Neural Network (ConvNet) to capture the global relationships and transitional characteristics between the entities and relations in knowledge base triples (head-entity, relation, tail-entity). [4] proposed yet another embedding model, HolE, which learns the compositional vector-space representations of knowledge graphs via circular correlation operations. [5] proposed a neural-embedding approach, DistMult, over knowledge graphs such that low-dimensional vectors are learned from the head and tail entities; while bilinear and/or linear mapping functions are used to implement the relations.

Furthermore, there exist very popular approaches for solving node classification problem in SNA. [6] implemented a semi-supervised DL model, GCN, which operates directly on graphs via localized first-order approximation of spectral graph convolutions. [7]'s 'node2vec' model is based on mapping the constituent nodes of a graph to a low-dimensional feature space while preserving the spatial relationships between the nodes. Node2Vec uses a biased random walk procedure to efficiently explore and learn over the graph structure. Another semi-supervised learning proposition via vector embeddings, SDNE by [8], implements a DL architecture for learning the highly non-linear network relationships in graphs; while still preserving its local and global structures. [9] proposed a vector-embedding model, LINE, which employs an edge-sampling strategy for exploiting and learning features from large-scale graphs; while still preserving the local and global structures of the graph. Finally, [10] proposed and implemented a DL-based approach which applies local information

exploited from truncated random walks to learn the latent representations entangled in graph structures.

3.3 Proposed Framework

3.3.1 Definition of Problem

Definition 3.1. *Social Network, SN: As expressed via equation 3.1 such that SN is a tuple comprising a set of actors/vertices, V; a set of ties/edges, E; a metadata function, f_V, which extends the definition of the vertices' set by mapping it to a given set of attributes, V'; and a metadata function, f_E, which extends the definition of the edges' set by mapping it to a given set of attributes, E'. Thus, a graph function, $G(V, E) \subset SN$*

$$
\begin{aligned}
&SN = (V, E, f_V, f_E) \equiv (G, f_V, f_E) \\
&V : |\{V\}| = M \qquad \text{set of actors/vertices with size, M} \\
&E : E \subset \{U \times V\} \subset \{V \times V\} \text{ set of ties/edges between V} \\
&f_V : V \to V' \qquad \text{vertices' metadata function} \\
&f_E : E \to E' \qquad \text{edges' metadata function}
\end{aligned} \tag{3.1}
$$

Definition 3.2. *Knowledge Graph, KG: $\{E, R\}$ is a set comprising entities, E, and relations, R, between the entities. Thus, a KG [11][12] is defined via a set of triples, $t : \{u, p, v\}$, where $u, v \in E$ and $p \in R$. Also, a KG [13] can be modelled as a social network, SN, such that: $E \to V$ and $R \to E$ and $\{E, R\} \vdash f_V, f_E$.*

Definition 3.3. *Knowledge-Graph (Vector) Embeddings, X: The vector-space embeddings, X, generated by the embedding layer are based on a mapping function, f, expressed via equation 3.2. f projects the representation of the graph's actors to a q-dimensional real space, $\mathbb{R}^q$, such that the existent ties between any given pair of actors, (u_i, v_j), remain preserved via the homomorphism from V to X.*

$$f : V \rightarrow X \in \mathbb{R}^q$$
$$f : (u, p, v) \rightarrow X \in \mathbb{R}^q \text{ Knowledge-Graph Embeddings} \tag{3.2}$$

Definition 3.4. *Link Prediction: A graph function, G, is a proper subset of a social network, $G(V, E) \subset SN$, at any instantaneous time, t. The set of actors/vertices is defined via $U \subset V : \{U|u_0, u_1, ..., u_m\} \subset \{V|v_0, v_1, ..., v_m\}$; and the set of ties/edges is defined via $E : (u_i, v_j) \in \{U \times V\}$. Hence, the goal of a link-prediction model is to train a predictive function, f, that learns the similarity measure, similarity(U, V), between pairs of actors in SN; such that the knowledge gained from the training is used to infer the probability of a tie existence between any valid pair of actors, (u_i, v_j), at time, t.*

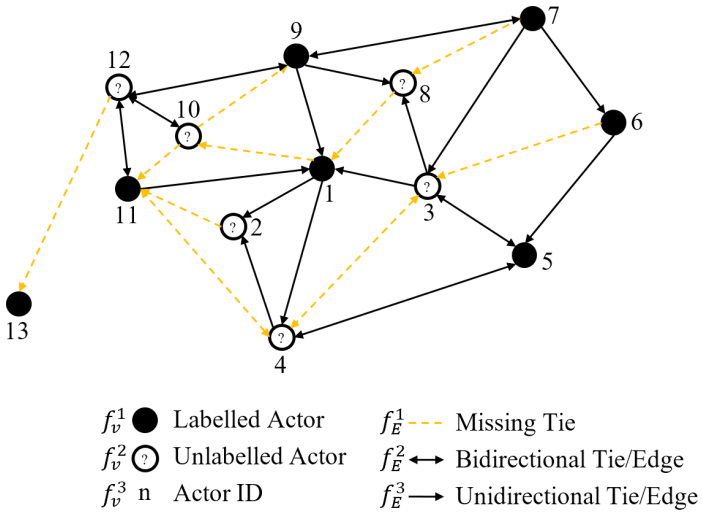

Figure 3.1: Link prediction task in social networks

Definition 3.5. *Node Classification: Given a social network, SN, comprising partially labelled actors (or vertices), $V_{lbl} \subset V : V_{lbl} \rightarrow Y_{lbl}$; and unlabelled vertices defined such that: $V_{ulb} = V - V_{lbl}$. Therefore, a node-classification model aims at training a predictive function, $f : V \rightarrow Y$, that learns to predict the labels, Y, for all actors or vertices, $V \subset SN$, via knowledge harnessed from the mapping: $V_{lbl} \rightarrow Y_{lbl}$.*

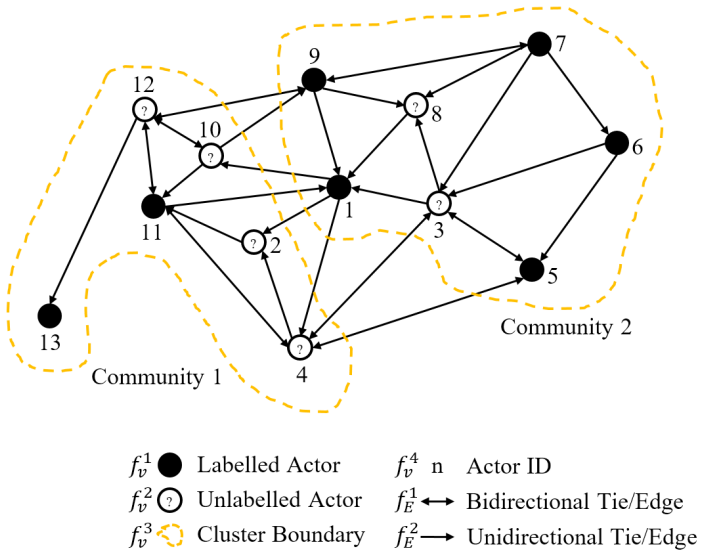

Figure 3.2: Node classification task in social networks

3.3.2 Proposed Methodology

Our proposition, RLVECO, possesses two (2) distinct RL or FL layers, and one (1) classification layer.

Knowledge-Graph Embeddings RL Layer:

In a social network, $V : U \subset V \forall \{u_m, v_m\} \in V$ where $M : m \in M$ denotes the number of unique actors in SN. Hence, $E \subset \{U \times V\}$, such that $u_i \in V$ and $v_j \in V$ represent a source vertex and a target vertex in E.

The goal of the objective function in the vector-embedding layer is to maximize the average logarithmic probability of the source vertex, u_i, being predicted as a correlated actor to the target vertex, v_j, with respect to all training pairs, $\forall (u_i, v_j) \in E$. Formally, this function is expressed via equation 3.3:

$$\mu = \frac{1}{M} \sum_{m=1}^{M} (\sum_{(u_i, v_j) \in E} logPr(u_i|v_j)) \tag{3.3}$$

Consequently, in order to compute $Pr(u_i|v_j)$, we have to quantify the proximity of each target vertex, v_j, with respect to its source vertex, u_i. The vector-embedding model measures this adjacency/proximity as the cosine distance or similarity between, v_j, and its corresponding, u_i. Thus, the cosine distance is calculated as the dot product between the target and the source vertices. Arithmetically, $Pr(u_i|v_j)$ is computed via a softmax function

as defined in equation 3.4:

$$Pr(u_i|v_j) = \frac{exp(u_i \cdot v_j)}{\sum_{m=1}^{M} exp(u_m \cdot v_j)} \tag{3.4}$$

Therefore, the objective function of our vector-embedding layer with respect to a SN is as expressed by equation 3.5:

$$\sum_{(u_i,v_j)\in E} logPr(u_i|v_j) = \sum_{(u_i,v_j)\in E} log\frac{exp(u_i \cdot v_j)}{\sum_{m=1}^{M} exp(u_m \cdot v_j)} \tag{3.5}$$

Convolution Operations RL Layer:

This layer comprises three (3) FL operations, viz: convolution; non-linearity; and pooling operations. RLVECO utilizes a one-dimensional (1D) convolution layer [14] which is sandwiched between the vector-embedding and classification layers. Equation 3.6 denotes the 1D-convolution operation:

$$\begin{aligned} FeatureMap(F) &= 1D\_InputMatrix(X) * Kernel(K) \\ f_i = (X * K)_i &= (K * X)_i = \sum_{j=0}^{J-1} x_j \cdot k_{i-j} = \sum_{j=0}^{J-1} k_j \cdot x_{i-j} \end{aligned} \tag{3.6}$$

where f_i represents a cell/matrix position in the Feature Map; k_j denotes a cell position in the Kernel; and x_{i-j} denotes a cell/matrix position in the 1D-Input (data) matrix.

The non-linearity operation is a rectified linear unit (ReLU) function which introduces non-linearity after the convolution operation since real-world problems usually exist in non-linear form(s). As a result, the rectified feature/activation map is computed via: $r_i \in R = g(f_i \in F) = max(0, F)$.

The pooling operation is responsible for reducing the input width of each rectified activation map while retaining its vital properties. In this regard, the *Max Pooling* function is defined such that the resultant pooled (or downsampled) feature map is generated via: $p_i \in P = h(r_i \in R) = maxPool(R)$.

MLP Classification Layer:

This is the last layer in RLVECO's architecture, and it succeeds the Representation Learning layers. The pooled feature maps, generated by the RL layers, contain low-level representations extracted from the constituent actors of the social network structure. Therefore, the classification layer uses these extracted "low-level representations" for inferring potential ties between constituent actors of a social network as well as for identifying clusters, based on the respective classes, contained in the social network. In this regard, the MLP [15] function, f_c, is defined as a function mapping some set of input values, P, to their respective output labels, Y [16]. Thus, $Y = f_c(P, \Theta)$. Furthermore, Θ, denotes a set of parameters which the MLP function, f_c, learns so as to yield the best decision, Y, approximation for the set of input, P.

Link Prediction and Node Classification Algorithms:

Firstly, RLVECO's RL kernel (post-input layer) comprises a Knowledge-Graph Embeddings (VE) layer and a Convolution Operations (CO) layer; both of which are trained by means of unsupervised learning. These layers are essentially dimensionality-reduction and feature-extraction layers where viable facts are automatically extracted from the social

graph structure [17]. The vector-embedding layer projects the feature representation of the social network structure to a q-dimensional real-number space, $\mathbb{R}^q$. This is accomplished by associating a real-number vector to every distinct actor in the social network; such that the cosine distance of any given tie (a pair of actors) would capture a significant degree of correlation between the two associated actors or nodes. Subsequently, the convolution-operation layer feeds on the output of the vector-embedding layer; and it is responsible for further extraction of latent representations from the given social network structure.

Secondly, a Neural Network (NN) classification layer succeeds the RL layers of RLVECO's architecture; and this layer is trained by means of supervised learning. The classification layer is assembled using deep and multiple layers of stacked perceptrons [18] such that sequential layers of NN units are piled against each other to form a Deep Neural Network (DNN) structure [15]. Every low-dimensional feature (X), extracted by the RL layers, is mapped to a corresponding output label (Y); and these (X, Y) pairs are used to supervise the training of the classifier such that it can effectively and efficiently learn how to predict potential ties, and classify actors in a bid to identify clusters within a given social network structure. Hence, the implementation of RLVECO's link prediction and node classification algorithms are defined via algorithms 3.1 and 3.2 respectively.

3.3.3 Proposed Architecture/Framework

Fig. 3.3 illustrates the architecture of RLVECO proposed herein for experimentation and analyses.

Algorithm 3.1 Proposed Procedure for Link Prediction

Input: $\{V, E, \mathbb{B}_{gTruth}\} \equiv$ {Actors, Ties, Ground-Truth Entities}
Output: $\{\mathbb{B}_{pred}\} \equiv$ {Predicted Entities}

Initialization:
$\mathbb{B}_{gTruth} : \{0, 1\} \equiv$ {C0 : -ve/False tie, C1 : +ve/True tie}
$E = E_{+ves} \cup E_{-ves} = (u_i, v_j) \in \{U \times V\} \subset \{V \times V\}$
// E_{train} : Ground-Truth edgelist
// $E_{pred} : E'_{train}$ = Complement of E_{train}
$E_{train} = E_t : E \rightarrow \mathbb{B}_{gTruth}$ // $|E_{train}| = E - E_{pred}$
$E_{pred} = E - E_{train}$

$f_c \leftarrow$ Initialize // Construct prediction model

Training:
while $E_{train} \neq NULL$ **do**
  $f : E_t \rightarrow [X \in \mathbb{R}^{400}]$ // Embedding operation
  $f_t \in F = (K * X)_t$ // Convolution operation
  $r_t \in R = g(F) = max(0, f_t)$
  $p_t \in P = h(R) = maxPool(r_t)$
  $f_c|\Theta : p_t \rightarrow \mathbb{B}_{gTruth}$ // MLP : $\Theta = similarity()$
end while

return $\mathbb{B}_{pred} = f_c(E_{pred}, \Theta)$

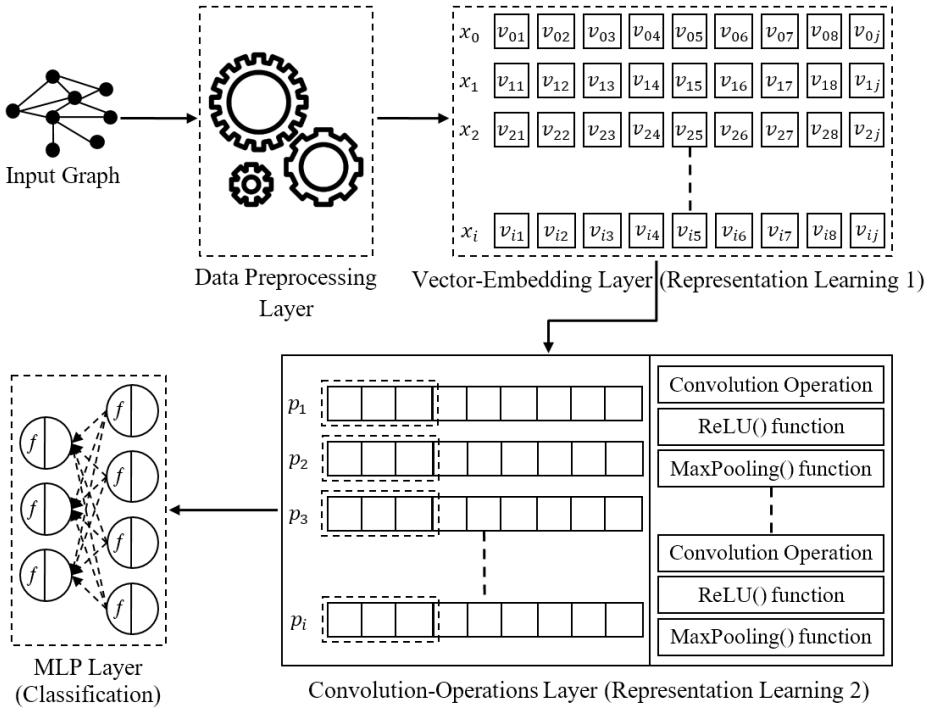

Figure 3.3: Conceptual model of RLVECO

Algorithm 3.2 Proposed Procedure for Node Classification

Input: $\{V, E, Y_{lbl}\} \equiv$ {Actors, Ties, Ground-Truth Labels}
Output: $\{Y_{ulb}\} \equiv$ {Predicted Labels}

Initialization:
// V_{lbl} : Labelled actors // V_{ulb} : Unlabelled actors
$V_{lbl}, V_{ulb} \subset V = V_{lbl} \cup V_{ulb}$
$E : (u_i, v_j) \in \{U \times V\}$ // $(u_i, v_j) \equiv$ (source, target)
// $|E_{train}| = \sum indegree(V_{lbl}) + \sum outdegree(V_{lbl})$
$E_{train} = E_t : u_i, v_j \in V_{lbl}$
$E_{pred} = E_p : u_i, v_j \in V_{ulb}$

$f_c \leftarrow$ Initialize // Construct classifier model

Training:
for $t \leftarrow 0$ **to** $|E_{train}|$ **do**
  $f : E_t \rightarrow [X \in \mathbb{R}^{400}]$ // Embedding operation
  $f_t \in F = (K * X)_t$ // Convolution operation
  $r_t \in R = g(F) = max(0, f_t)$
  $p_t \in P = h(R) = maxPool(r_t)$
  $f_c|\Theta : p_t \rightarrow Y_{lbl}$ // MLP classification operation
end for

return $Y_{ulb} = f_c(E_{pred}, \Theta)$

3.4 Datasets and Materials

3.4.1 Datasets

With reference to Table 3.7 herein, four (4) real-world benchmark social-graph datasets were utilized for experimentation and evaluation, viz: CiteSeer [19] [20], Cora [19] [20], Facebook Page-Page webgraph [21], and PubMed-Diabetes [22].

3.4.2 Data Preprocessing

In their respective original forms, the benchmark datasets (CiteSeer, Cora, Facebook-Page2Page, and PubMed-Diabetes datasets) are made up of nodes and/or edges encoded in mixed formats (categorical and numerical formats). However, it is mandatory for the constituent actors and ties of these benchmark datasets to be represented as discrete data

Table 3.1: Description of source-code repository

Subject	GitHub link/url
Home	https://github.com/bhevencious?tab=repositories
Node-Classification tasks	
RLVECO for Node Classification	https://github.com/bhevencious/RLVECN/blob/master/rlvecn-node_classification.py
RLVECO's experiment results	https://github.com/bhevencious/RLVECN/blob/master/eval_log.txt
DeepWalk, LINE, Node2Vec, and SDNE	https://github.com/bhevencious/Baselines_GraphEmbedding/blob/master/fused_baseline_models.py
DeepWalk, LINE, Node2Vec, and SDNE experiment results	https://github.com/bhevencious/Baselines_GraphEmbedding/blob/master/eval_log.txt
GCN	https://github.com/bhevencious/Baselines_GraphEmbedding/blob/master/kipf_gcn/gcnn_node_classification.py
GCN's experiment results	https://github.com/bhevencious/Baselines_GraphEmbedding/blob/master/kipf_gcn/eval_log.txt
Link-Prediction tasks	
RLVECO for Link Prediction	https://github.com/bhevencious/RLVECN/blob/master/rlvecn-link_prediction.py
RLVECO's experiment results	https://github.com/bhevencious/RLVECN/blob/master/eval_log.txt
ComplEx, ConvKB, DistMult, and HolE	https://github.com/bhevencious/RLVECN/blob/master/fused_ampligraph_models.py
ComplEx, ConvKB, DistMult, and HolE experiment results	https://github.com/bhevencious/RLVECN/blob/master/eval_log.txt

(natural-number format). Therefore, it is imperative to transcode these non-numeric (categorical) entities to their respective discrete (numeric) data representation, without semantic loss, via an injective function that maps each distinct entry in the categorical-entity domain to a distinct numeric value in the discrete-data codomain, $f_m : categorical \rightarrow discrete$.

In this phase, we remodel each social graph, SN, as a knowledge graph, KG [13], such that: $\mathbb{E} \rightarrow V$ and $\mathbb{R} \rightarrow E$ and $(\mathbb{E}, \mathbb{R}) \vdash f_V, f_E$ as elucidated in definition 3.2. Furthermore, only edgelist ties, $E(U, V)$, whose constituent actors are present in the nodelist, (U, V) :

Table 3.2: Link-prediction experiment results over CiteSeer (A), Cora (B), Facebook Page-Page webgraph (C), and PubMed-Diabetes (D) datasets. C0 : $\mathbb{B} = 0$ (-ve/False tie) and C1 : $\mathbb{B} = 1$ (+ve/True tie)

Model	Metric	A		B		C		D			Points
		C0	C1	C0	C1	C0	C1	C0	C1	μ	
RLVECO	PC	1.00	1.00	1.00	0.99	1.00	1.00	1.00	0.99	1.00	15
	RC	0.99	1.00	0.95	1.00	0.80	0.96	0.98	1.00	0.96	
	F1	**0.99**	**1.00**	**0.98**	**1.00**	0.89	**0.98**	**0.99**	**0.99**	0.98	
	AC	0.99	1.00	0.98	1.00	0.97	0.97	0.99	0.99	0.99	
	RO	**0.99**	**1.00**	**0.98**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**	1.00	
	SP	1986	1897	1621	2170	13479	66774	11829	18185	14743	
DistMult	PC	0.89	1.00	0.83	1.00	0.80	1.00	0.86	1.00	0.92	0
	RC	1.00	0.88	1.00	0.85	1.00	0.95	1.00	0.91	0.95	
	F1	0.94	0.94	0.91	0.92	0.89	0.97	0.92	0.95	0.93	
	AC	0.94	0.94	0.91	0.91	0.96	0.96	0.94	0.94	0.94	
	RO	0.94	0.94	0.93	0.93	0.98	0.98	0.96	0.96	0.95	
	SP	1659	1785	1507	2110	12711	66458	9375	17255	14108	
ComplEx	PC	0.88	1.00	0.81	1.00	0.76	1.00	0.84	1.00	0.91	0
	RC	1.00	0.87	1.00	0.84	1.00	0.94	1.00	0.89	0.94	
	F1	0.93	0.93	0.90	0.91	0.86	0.97	0.91	0.94	0.92	
	AC	0.93	0.93	0.90	0.90	0.95	0.95	0.93	0.93	0.93	
	RO	0.94	0.94	0.92	0.92	0.97	0.97	0.95	0.95	0.95	
	SP	1659	1785	1507	2110	12711	66458	9375	17255	14108	
ConvKB	PC	0.86	1.00	0.80	1.00	0.70	1.00	0.77	1.00	0.89	0
	RC	1.00	0.85	1.00	0.83	0.99	0.91	1.00	0.86	0.93	
	F1	0.93	0.92	0.89	0.90	0.82	0.95	0.87	0.92	0.90	
	AC	0.92	0.92	0.90	0.90	0.91	0.93	0.89	0.91	0.91	
	RO	0.93	0.93	0.91	0.91	0.93	0.95	0.91	0.93	0.93	
	SP	1659	1785	1507	2110	12711	66458	9375	17255	14108	
HolE	PC	0.95	0.79	0.92	0.83	0.85	1.00	0.95	0.96	0.91	2
	RC	0.73	0.96	0.73	0.95	1.00	0.97	0.92	0.98	0.91	
	F1	0.82	0.87	0.81	0.89	**0.92**	**0.98**	0.94	0.97	0.90	
	AC	0.85	0.85	0.86	0.86	0.97	0.97	0.96	0.96	0.91	
	RO	0.85	0.85	0.84	0.84	0.98	0.98	0.95	0.95	0.91	
	SP	1659	1785	1507	2110	12711	66458	9375	17255	14108	

$\forall \{u_m, v_m\} \in V \subset G$, were used for training and testing/validating RLVECO as well as other baselines. The numeric representation of all benchmark datasets are normalized, $f_n : discrete \rightarrow continuous$, prior to training against RLVECO and the baselines.

Table 3.3: Classification of actors using CiteSeer dataset with regard to the set apart validation sample - dataset *vs* models.

Model	Metric	CiteSeer Dataset							Points
		C1	C2	C3	C4	C5	C6	μ	
RLVECO	PC	0.76	0.81	0.78	0.43	0.88	0.60	0.71	12
	RC	0.84	0.83	0.79	0.60	0.79	0.65	0.75	
	F1	**0.80**	**0.82**	**0.79**	**0.50**	**0.83**	**0.63**	0.73	
	AC	0.93	0.88	0.92	0.93	0.96	0.89	0.92	
	RO	**0.90**	**0.87**	**0.87**	**0.78**	**0.89**	**0.79**	0.85	
	SP	304	609	377	107	225	275	316	
GCN	PC	0.80	0.78	0.86	0.95	0.91	0.75	0.84	2
	RC	0.76	0.76	0.73	0.08	0.67	0.54	0.59	
	F1	0.78	0.77	**0.79**	0.15	0.77	**0.63**	0.65	
	AC	0.88	0.87	0.88	0.91	0.89	0.83	0.88	
	RO	0.84	0.83	0.83	0.53	0.81	0.72	0.76	
	SP	119	134	140	50	102	118	111	
Node2Vec	PC	0.57	0.55	0.49	0.33	0.55	0.38	0.48	0
	RC	0.55	0.60	0.66	0.06	0.45	0.40	0.45	
	F1	0.56	0.58	0.56	0.10	0.50	0.39	0.45	
	AC	0.85	0.82	0.78	0.92	0.86	0.78	0.84	
	RO	0.73	0.74	0.74	0.53	0.69	0.63	0.68	
	SP	119	134	140	50	102	118	111	
DeepWalk	PC	0.46	0.53	0.43	0.43	0.47	0.33	0.44	0
	RC	0.51	0.54	0.57	0.06	0.41	0.32	0.40	
	F1	0.49	0.54	0.49	0.11	0.44	0.32	0.40	
	AC	0.81	0.81	0.75	0.92	0.84	0.76	0.82	
	RO	0.69	0.71	0.69	0.53	0.66	0.59	0.65	
	SP	119	134	140	50	102	118	111	
SDNE	PC	0.37	0.50	0.24	0.20	0.45	0.31	0.35	0
	RC	0.19	0.27	0.77	0.02	0.14	0.09	0.25	
	F1	0.25	0.35	0.36	0.04	0.21	0.14	0.23	
	AC	0.80	0.80	0.42	0.92	0.84	0.80	0.76	
	RO	0.56	0.60	0.55	0.51	0.55	0.52	0.55	
	SP	119	134	140	50	102	118	111	
LINE	PC	0.18	0.30	0.28	0.60	0.22	0.27	0.31	0
	RC	0.15	0.47	0.39	0.06	0.12	0.21	0.23	
	F1	0.16	0.36	0.32	0.11	0.15	0.24	0.22	
	AC	0.72	0.67	0.65	0.93	0.80	0.76	0.76	
	RO	0.50	0.59	0.56	0.53	0.52	0.55	0.54	
	SP	119	134	140	50	102	118	111	

Table 3.4: Classification of actors using Cora dataset with respect to the set apart validation sample - dataset *vs* models.

Model	Metric	Cora Dataset								Points
		C1	C2	C3	C4	C5	C6	C7	μ	
RLVECO	PC	0.85	0.78	0.80	0.88	0.72	0.90	0.81	0.82	14
	RC	0.86	0.93	0.81	0.87	0.75	0.91	0.78	0.84	
	F1	**0.86**	**0.85**	**0.81**	**0.87**	**0.74**	**0.91**	**0.79**	0.83	
	AC	0.93	0.98	0.96	0.95	0.93	0.97	0.96	0.95	
	RO	**0.90**	**0.96**	**0.90**	**0.92**	**0.85**	**0.95**	**0.88**	0.91	
	SP	541	134	214	405	294	345	237	310	
GCN	PC	0.87	0.95	0.89	0.92	0.85	0.89	0.87	0.89	3
	RC	0.85	0.73	0.65	0.82	0.58	0.85	0.73	0.74	
	F1	**0.86**	0.83	0.75	**0.87**	0.69	0.87	**0.79**	0.81	
	AC	0.89	0.93	0.91	0.92	0.88	0.93	0.91	0.91	
	RO	0.88	0.83	0.80	0.89	0.75	0.90	0.83	0.84	
	SP	164	36	43	85	70	84	60	77	
Node2Vec	PC	0.58	0.78	0.72	0.81	0.80	0.84	0.82	0.76	0
	RC	0.85	0.50	0.53	0.68	0.64	0.74	0.60	0.65	
	F1	0.69	0.61	0.61	0.74	0.71	0.78	0.69	0.69	
	AC	0.77	0.96	0.95	0.92	0.93	0.94	0.94	0.92	
	RO	0.79	0.75	0.76	0.83	0.81	0.86	0.79	0.80	
	SP	164	36	43	85	70	84	60	77	
DeepWalk	PC	0.57	0.58	0.72	0.58	0.68	0.72	0.63	0.64	0
	RC	0.80	0.42	0.42	0.59	0.39	0.63	0.65	0.56	
	F1	0.67	0.48	0.53	0.58	0.49	0.67	0.64	0.58	
	AC	0.76	0.94	0.94	0.87	0.90	0.90	0.92	0.89	
	RO	0.77	0.70	0.70	0.75	0.68	0.79	0.80	0.74	
	SP	164	36	43	85	70	84	60	77	
LINE	PC	0.35	0.86	0.80	0.65	0.50	0.43	0.61	0.60	0
	RC	0.85	0.17	0.19	0.35	0.20	0.15	0.23	0.31	
	F1	0.50	0.28	0.30	0.46	0.29	0.23	0.34	0.34	
	AC	0.48	0.94	0.93	0.87	0.87	0.84	0.90	0.83	
	RO	0.59	0.58	0.59	0.66	0.59	0.56	0.61	0.60	
	SP	164	36	43	85	70	84	60	77	
SDNE	PC	0.37	0.83	0.70	0.60	0.54	0.64	0.64	0.62	0
	RC	0.91	0.14	0.16	0.35	0.20	0.27	0.12	0.31	
	F1	0.53	0.24	0.26	0.44	0.29	0.38	0.20	0.33	
	AC	0.50	0.94	0.93	0.86	0.87	0.86	0.89	0.84	
	RO	0.62	0.57	0.58	0.65	0.59	0.62	0.55	0.60	
	SP	164	36	43	85	70	84	60	77	

3.4.3 Materials

AmpliGraph library [23] was employed to generate the Knowledge-Graph Embeddings. Additionally, the design, development, and implementation of our proposition, RLVECO, was

Table 3.5: Categorization or Classification of actors using Facebook Page-Page webgraph dataset with respect to the reserved validation sample - dataset *vs* models.

Model	Metric	Facebook-Page2Page Dataset					Points
		C1	C2	C3	C4	μ	
RLVECO	PC	0.87	0.95	0.91	0.87	0.90	8
	RC	0.84	0.85	0.85	0.86	0.85	
	F1	**0.85**	**0.90**	**0.88**	**0.86**	0.87	
	AC	0.96	0.90	0.94	0.97	0.94	
	RO	**0.97**	**0.97**	**0.98**	**0.98**	0.98	
	SP	9989	33962	16214	6609	16694	
Node2Vec	PC	0.81	0.84	0.81	0.84	0.83	0
	RC	0.82	0.87	0.85	0.67	0.80	
	F1	0.81	0.85	0.83	0.74	0.81	
	AC	0.89	0.91	0.91	0.93	0.91	
	RO	0.87	0.90	0.89	0.82	0.87	
	SP	1299	1376	1154	665	1124	
DeepWalk	PC	0.75	0.84	0.76	0.75	0.78	0
	RC	0.81	0.85	0.82	0.52	0.75	
	F1	0.78	0.84	0.79	0.62	0.76	
	AC	0.87	0.90	0.89	0.90	0.89	
	RO	0.85	0.89	0.87	0.75	0.84	
	SP	1299	1376	1154	665	1124	
LINE	PC	0.53	0.66	0.72	0.66	0.64	0
	RC	0.72	0.71	0.59	0.29	0.58	
	F1	0.61	0.68	0.65	0.40	0.59	
	AC	0.73	0.80	0.83	0.87	0.81	
	RO	0.73	0.77	0.75	0.63	0.72	
	SP	1299	1376	1154	665	1124	
SDNE	PC	0.49	0.80	0.70	0.65	0.66	0
	RC	0.90	0.63	0.50	0.19	0.56	
	F1	0.64	0.70	0.58	0.29	0.55	
	AC	0.70	0.84	0.82	0.86	0.81	
	RO	0.76	0.78	0.71	0.58	0.71	
	SP	1299	1376	1154	665	1124	

realized by means of Scikit-Learn [24], TensorFlow and Keras [25] libraries.

3.5 Experiment and Discussions

RLVECO's source codes, proposed and implemented herein for resolving both node classification and link prediction tasks, can be accessed via Microsoft's GitHub software de-

Table 3.6: Categorization or Classification of actors over PubMed-Diabetes dataset using the reserved validation sample - dataset *vs* models.

Model	Metric	PubMed-Diabetes Dataset				Points
		C1	C2	C3	μ	
RLVECO	PC	0.76	0.83	0.84	0.81	6
	RC	0.60	0.88	0.91	0.80	
	F1	**0.67**	**0.86**	**0.87**	0.80	
	AC	0.89	0.88	0.90	0.89	
	RO	**0.92**	**0.94**	**0.95**	0.94	
	SP	3300	7715	7170	6062	
DeepWalk	PC	0.65	0.57	0.58	0.60	0
	RC	0.15	0.67	0.71	0.51	
	F1	0.24	0.62	0.63	0.50	
	AC	0.81	0.67	0.68	0.72	
	RO	0.56	0.67	0.69	0.64	
	SP	821	1575	1548	1315	
Node2Vec	PC	0.74	0.47	0.49	0.57	0
	RC	0.03	0.65	0.55	0.41	
	F1	0.05	0.55	0.52	0.37	
	AC	0.80	0.57	0.60	0.66	
	RO	0.51	0.58	0.59	0.56	
	SP	821	1575	1548	1315	
SDNE	PC	0.65	0.43	0.74	0.61	0
	RC	0.05	0.96	0.17	0.39	
	F1	0.10	0.59	0.27	0.32	
	AC	0.80	0.48	0.65	0.64	
	RO	0.52	0.56	0.56	0.55	
	SP	821	1575	1548	1315	
LINE	PC	0.48	0.42	0.44	0.45	0
	RC	0.05	0.60	0.46	0.37	
	F1	0.08	0.50	0.45	0.34	
	AC	0.79	0.51	0.56	0.62	
	RO	0.52	0.53	0.54	0.53	
	SP	821	1575	1548	1315	

velopment version control platform. In addition, Table 3.1 herein describes the directory structure of RLVECO's remote GitHub repository.

With respect to our research proposition, we have represented the SNA research problems solved herein as classification problems. Hence, our results have been compiled using

Table 3.7: Benchmark datasets

Dataset	Classes → {label: 'description'}
CiteSeer [19] [20]	$G(V, E) = G(3312, 4732)$ {C1: 'Agents', C2: 'Artificial Intelligence', C3: 'Databases', C4: 'Information Retrieval', C5: 'Machine Learning', C6: 'Human-Computer Interaction'}
Cora [19] [20]	$G(V, E) = G(2708, 5429)$ {C1: 'Case_Based', C2: 'Genetic_Algorithms', C3: 'Neural_Networks', C4: 'Probabilistic_Methods', C5: 'Reinforcement_Learning', C6: 'Rule_Learning', C7: 'Theory'}
Facebook Page2Page [21]	$G(V, E) = G(22470, 171002)$ {C1: 'Companies', C2: 'Governmental Organizations', C3: 'Politicians', C4: 'Television Shows'}
PubMed Diabetes [22]	$G(V, E) = G(19717, 44338)$ {C1: 'Diabetes Mellitus - Experimental', C2: 'Diabetes Mellitus - Type 1', C3: 'Diabetes Mellitus - Type 2'}

classification-based objective functions as our yardstick. We have applied Categorical Cross Entropy as a measure for the cost/loss function. The fitness/utility of each baseline (or benchmark model) has been evaluated based on the following measurement criteria/metrics: PC, RC, F1, AC, and RO. For each benchmark dataset, we have computed the objective functions with regard to the constituent classes or categories present in each dataset. Thus, the Support (SP) represents the number of ground-truth samples per class for each dataset.

With regard to the link-prediction tasks: Table 3.2 shows the performance scores of RLVECO during comparative analyses with respect to popular VE benchmark models (ComplEx, ConvKB, DistMult, and HolE); and when evaluated over the validation/test samples of the benchmark datasets. $\mathbb{B} = 0$ (C0) denotes the class of -ve/False ties and $\mathbb{B} = 1$ (C1) denotes the class of +ve/True ties. The SPs used in computations by the VE benchmark models appear to be slightly lower in comparison with RLVECO's inas-

much as all models are based on an edge-sampling strategy. In that regard, we discovered that this slight difference in SP is because these VE baselines implement a function, "*filter_unseen_entities*", against their validation/test sets; and this function checks and removes all actors (entities) whose embeddings cannot be computed from the training sets. Algorithm 3.1 explicates RLVECO's edge-sampling approach.

Furthermore, with respect to the node-classification experiments herein: the performance of RLVECO during comparative analyses against popular baselines (DeepWalk, GCN, LINE, Node2Vec, and SDNE); and when evaluated over the validation/test samples of the benchmark datasets are as documented in Table 3.3, Table 3.4, Table 3.5, and Table 3.6. We have applied exactly the same SP to all node-classification benchmark models, RLVECO model inclusive, so as to avoid sample bias across-the-board. However, since our proposition is based on an edge-sampling technique; the SP recorded against RLVECO represent the numbers of edges/ties used for computation as explained in algorithm 3.2.

Table 3.2 reports the link prediction jobs as binary classification tasks; while Tables 3.3, 3.4, 3.5, and 3.6 represent the node classification results based on multi-classification tasks over the benchmark datasets. For each class per dataset, we have laid emphasis on the F1 ($0 \leq \text{F1} \leq 1$) and the RO ($0 \leq \text{RO} \leq 1$) metrics. The F1 and RO scores attain their worst and best values at 0 and 1 respectively. We have spotlighted the model that performed best for each classification task using a **bold font** with respect to its associated F1 and RO metrics. In addition, we have employed a point-based ranking standard to ascertain the

fittest model for each link prediction and node classification task. The model with the best mean (μ) metrics and highest aggregate points signifies the fittest model for the specified task, and so on in descending order of mean (μ) metrics and aggregate points.

Basically, core processing in RLVECO is accomplished via its RL kernel. Every social-graph input undergoes data preprocessing to make certain each constituent actor is represented as discrete data. With respect to the constituent ties of the social graph, the VE layer reduces the dimensionality of the social graph via a mapping function, $f : E_t \rightarrow [X \in \mathbb{R}^{400}]$, which projects each social actor to a real (continuous-data) space. This dimensionality reduction is effectuated while preserving the spatial relationship between neighboring actors. Furthermore, the convolution operation, $F = (K * X)_t$, acts upon the dimensionally reduced representation of the social graph with respect to the edges/ties. By means of unsupervised learning, the convolution operation extracts and learns additional information about the social network structure. Non-linearity, $R = max(0, f_t)$, and pooling, $P = maxPool(r_t)$, operations are applied after the convolution operation. Finally, we train a NN classifier, $Y = f_c(P, \Theta)$, using the features extracted by the RL kernel. Thereafter, this classifier learns to predict ties and classify actors via a supervised training and learning procedure. Our tabular results herein show that our hybrid proposition exhibits remarkable performance, with respect to its aggregate (fitness) points, for the link-prediction and node-classification tasks respectively. With regard to the comparative analyses herein, RLVECO's surpassing performance is attributed to two (2) key factors, namely:

(1) RLVECO's RL kernel comprises two (2) distinct layers of FL, viz: Knowledge-Graph Embeddings (VE) and Convolution Operations (Convolution Operations (CO)) [26]. It's dual RL layers enable it to extract and learn sufficient features of social networks representation during training.

(2) High-quality data preprocessing techniques employed herein with respect to the benchmark datasets. We ensured that the constituent actors of every social graph were transcoded to their respective discrete data representations, without any loss in semantics, and normalized prior to training/testing.

We have applied NN pruning as well as global search methods to our DL architecture [1] [27] so as to improve its efficiency and effectiveness. Hence, our neuron-pruning formula is based on equation 3.7 where N_s, N_i, N_o, and N_m represent the sizes of training set, input layer, output layer, and hidden layer respectively.

$$N_m = \frac{N_s}{4 * (N_i + N_o)} \tag{3.7}$$

Dropout regularization has been implemented within the hidden-layer structure of RLVECO. Also, $L2$ ($L2 = 0.04$) regularization and early stopping [28] were utilized herein as addon regularization techniques to surmount overfitting incurred during the training of RLVECO for the link-prediction and node-classification tasks. The application of early stopping to RLVECO during training over the benchmark datasets are as reported via Table 3.8.

Table 3.8: Early-stopping regularization against datasets

Dataset	Node Classification	Link Prediction
CiteSeer	after 50 epochs	after 35 epochs
Cora	after 50 epochs	after 50 epochs
Facebook Page2Page	after 50 epochs	after 50 epochs
PubMed Diabetes	after 50 epochs	after 50 epochs

3.6 Limitations, Conclusion, and Future Work

The baselines evaluated herein were implemented using their default parameters. GCN [6] was not evaluated against Facebook Page-Page webgraph and PubMed-Diabetes datasets; because each of the aforementioned dataset does not possess a vectorized feature set, which is a prerequisite for executing GCN model successfully over any dataset. RLVECO's strengths, as depicted via the experimentation results herein, are attributed to its biform RL layers and the quality of the preprocessing operations applied to each dataset. In the near future, we intend to expand RLVECO's research and experiment scopes to accommodate other open problems in SNA; and include more benchmark models as well as social network datasets.

Acknowledgements

This research was supported by International Business Machines (IBM) - research was conducted on a high performance IBM Power System S822LC Linux Server. Also, part support was provided by SHARCNET and Compute Canada (www.computecanada.ca).

Bibliography

[1] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. rahman Mohamed, N. Jaitly, A. W. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, 2012.

[2] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of the 33rd International Conference on Machine Learning, ICML*, 2016.

[3] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Q. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, 2018.

[4] M. Nickel, L. Rosasco, and T. A. Poggio, "Holographic embeddings of knowledge graphs," in *30th AAAI Conference on Artificial Intelligence*, 2016.

[5] B. Yang, W. tau Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *International Conference on Learning Representations (ICLR)*, vol. abs/1412.6575, 2015.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

[7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 2016, pp. 855–864, 2016.

[8] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[9] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, 2015.

[10] B. Perozzi, R. Al-Rfou', and S. Skiena, "Deepwalk: online learning of social representations," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, vol. abs/1403.6652, 2014.

[11] P. Tabacof and L. Costabello, "Probability calibration for knowledge graph embedding models," *International Conference on Learning Representations (ICLR)*, 2020.

[12] Q. Zhang, Z. Sun, W. Hu, M. Chen, L. Guo, and Y. Qu, "Multi-view knowledge graph embedding for entity alignment," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[13] S. Yang, J. Tian, H. Zhang, J. Yan, H. He, and Y. Jin, "Transms: Knowledge graph embedding for complex relations by multidirectional semantics," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[14] B. C. Molokwu, "Event prediction in social graphs using 1-dimensional convolutional neural network," in *Canadian Conference on AI*, 2019.

[15] B. C. Molokwu and Z. Kobti, "Spatial event prediction via multivariate time series analysis of neighboring social units using deep neural networks," *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2019.

[16] I. G. Goodfellow, Y. Bengio, and A. C. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[17] B. C. Molokwu and Z. Kobti, "Event prediction in complex social graphs via feature learning of vertex embeddings," in *Neural Information Processing*, T. Gedeon, K. W. Wong, and M. Lee, Eds. Cham: Springer International Publishing, 2019, pp. 573–580.

[18] I. G. Goodfellow, Y. Bengio, and A. C. Courville, Eds., *Deep Learning*. Cambridge, MA: MIT Press, 2017.

[19] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, pp. 93–106, 2008.

[20] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. [Online]. Available: http://networkrepository.com

[21] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *ArXiv*, vol. abs/1909.13021, 2019.

[22] G. Namata, B. London, L. Getoor, and B. Huang, "Query-driven active surveying for collective classification," in *Proceedings of the Workshop on Mining and Learning with Graphs, MLG-2012*, 2012.

[23] L. Costabello, S. Pai, C. L. Van, R. McGrath, N. McCarthy, and P. Tabacof, "AmpliGraph: a Library for Representation Learning on Knowledge Graphs," March 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2595043

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] F. Chollet, Ed., *Deep Learning with Python.* Shelter Island, NY: Manning Publications, 2017.

[26] B. C. Molokwu, "Event prediction in complex social graphs using one-dimensional convolutional neural network," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI*, 2019.

[27] A. Gron, Ed., *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* Newton, MA: O'Reilly Media, Inc., 2017.

[28] J. Patterson and A. Gibson, Eds., *Deep Learning: A Practitioner's Approach.* Newton, MA: O'Reilly Media, Inc., 2017.

Chapter 4

A Transfer Learning Framework for COVID-19 Monitoring and the Prediction of PPE Consumption in Community Health Centres

Considering the global impacts of COVID-19 pandemic in association with the emerging variants of SARS-CoV-2; in this chapter, we have examined and studied a pertinent real-world problem with respect to public health safety and epidemiology. In this regard, we have modelled the impacts of the COVID-19 pandemic as a Trend/Pattern Analysis problem with respect to Social Network Analysis. Therefore, we have employed Transfer Learning methodologies toward understanding and monitoring the impacts of SARS-CoV-2 in selected geographical regions (or provinces). Also, we have proposed a framework herein for forecasting PPE consumption and demand in Community Health Centres within the selected geographical regions. To this end, our research and study has proven that Social Network Analysis has immense real-world applications, as can be seen in its application to

public health care and safety.

4.1 Introduction

The novel SARS-CoV-2, which is responsible for the COVID-19 pandemic, caught the world unawares. The SARS-CoV-2 is a new type of Ribonucleic Acid (RNA) virus that has never been studied or witnessed before December 2019. COVID-19 global epidemiological summaries [1], as at February 2021, report that there have been over $110,384,747$ confirmed/positive cases as well as $2,446,008$ deaths as a result of SARS-CoV-2. In that regard, the American continent is the worst affected by the COVID-19 pandemic. The American continent has recorded over $49,126,365$ confirmed/positive cases (44.5% of the global summary) as well as $1,165,711$ deaths (47.7% of the global summary). Furthermore, these global epidemiological summaries are quite overwhelming; hence, the COVID-19 pandemic calls for exigent and proactive measures from every domain. From a Data Science perspective, it is noteworthy that the world can be conceptualized as one big data problem. With respect to the 21^{st} century, data essentially influences and controls virtually everything that we tend to accomplish. For us, as a human race, to successfully conquer the COVID-19 pandemic; we require sufficient knowledge about the SARS-CoV-2. Thus, the quest for knowledge associated with the SARS-CoV-2 can only be acquired from information available about the virus. In turn, information is a valuable commodity that we can only extract from data. Consequently, there lies the importance of data (as well as Data

Science) with respect to combating the COVID-19 pandemic.

Essentially, epidemiology is a data-driven scientific process, that employs pattern analysis and/or pattern recognition, in the study of certain indicators (such as risk factors or health-related variables that increase the tendency for an individual to contract a disease) with respect to a given population distribution. SARS-CoV-2 is primarily responsible for the outbreak of COVID-19 pandemic. Furthermore, our research herein concentrates and proposes solutions to the following problem statements, namely:

(i) Monitor the effect of SARS-CoV-2 via effective predictions of estimates with respect to Infected, Hospitalized, Recovered, and Death.

(ii) Predict the demand of PPE by CHC that provide medical treatment to COVID-19 patients (Hospitalized cases).

Moreover, considering our experiments and results, we have used the Canadian province of Ontario as our case study.

At the moment, there exist three (3) common variants/strains [2] of SARS-CoV-2 around the world. These strains are, namely: Lineage B.1.1.7 (UK/British variant), Lineage B.1.351 (South African variant), and Lineage P.1 (Brazilian variant). The contributions and novelties of our research herein cannot be downplayed considering the potential consequences of these emerging strains, and the current global impact of COVID-19. From one standpoint, we have been able to identify several factors (Biological, Environmental, Government

Actions, and Human), which influence the spread of SARS-CoV-2, by means of studies, analyses, and experiments. Also, we have been able to forecast, effectively and efficiently, estimates of COVID-19 impact with respect to Infected, Hospitalized, Recovered, and Death cases. Consequently, these contributions can serve as control as well as preventive measures in curtailing the spread of SARS-CoV-2. Likewise, these contributions can be employed in clinical research and trials, with respect to drug and vaccine development, against SARS-CoV-2. From another standpoint, we have proposed a model that forecasts PPE demand in CHC with regard to the COVID-19 pandemic. Considering the safety of our frontline healthcare workers, this contribution serves as a preventive and control measure toward their protection. Similarly, this can serve as a proactive measure to ensure a robust supply network of PPE to CHC.

Additionally, both problem statements herein have been modeled and analyzed as regression-based problems. We have employed TL methodology, which is essentially a ML approach, with reference to the design, development, and implementation of our proposed framework. Our TL model tackles the problem statements herein via pre-training on datasets, which are comprised of $i \times 13$ feature vector (row vector), collected from six (6) distinct Canadian provinces, viz: Alberta, British Columbia, Manitoba, New Brunswick, Quebec, and Saskatchewan. Subsequently, the resultant pre-trained model is referenced, as a source point for the transfer of learning and knowledge, for training another (dedicated) model concerned with the resolution of problems relating to our case study (Ontario province).

The research and findings with respect to our proposed framework and model introduces the following novelties, viz:

(1) The transfer of knowledge and learning from correlated (source) domains to a target domain, via pre-training models and TL frameworks, yields much better generalization results (with relatively low residual errors) with respect to COVID-19 monitoring.

(2) Identification of influential factors, based on four (4) categories (SARS-CoV-2 Biological Factors, Environmental Factors, Government Actions, and Human Factors), which affect the spread of COVID-19.

(3) Human factors (precisely age-group stratification) most significantly influence the rates of Infected cases, Hospitalized cases, and Death cases (see Figures 4.1, 4.2, and 4.4).

(4) Government Actions (precisely vaccination, pandemic wave, etc.) most significantly influence the rate of Recovered cases (see Figure 4.3).

(5) We identified that COVID-19 is most prevalent among males and females, within the age group of 0 to 34, in a given populace (see Figure 4.12).

(6) Proposition of a model for the prediction of PPE demand by CHC with respect to Hospitalized cases of COVID-19.

(7) Detailed evaluation and performance reports, based on classic ML objective functions

used for regression, with reference to our proposed framework.

Our work presented hereafter is organized as follows: section 4.2 reviews a selected list of related literature. Section 4.3 formally defines the problem statement as well as the details of our proposed framework. Section 4.4 expatiates on the datasets and materials used to facilitate our experiments. Section 4.5 documents the detailed results of our benchmark experiments; and it captures our discussions, with respect to the results and performance of our proposed model, on the case study. Section 4.6 spotlights the potential applications of our proposed methodology toward COVID-19 monitoring and management. Section 4.7 highlights our future work; and the known assumptions and/or limitations with regard to our framework and experiments.

4.2 Historical Foundation and Related Literature

Several literature and published work, which aimed at resolving problems related to epidemiology, can be classified into three (3) broad categories, namely: Conceptual Models, Compartmental Models, and Computational Models.

4.2.1 Conceptual Models

Models in this category are essentially high-level representations, based on abstract ideas and notions, which illustrate how these models operate with regard to resolving targeted problems in epidemiology. A common shortcoming of these models is that they are simplistic, abstract, and usually not empirical or verifiable. Thus, [3] [4] [5] [6], have primarily

employed conceptual modeling toward resolving research problems in epidemiology.

4.2.2 Compartmental Models

Basically, these models are Mathematical Models which are based on a series of mathematical equations. They are employed in studying and analyzing how infectious diseases spread and affect different compartments of a given socially-interacting population. Also, they have been used to forecast the potential outcomes of endemics, epidemics, and pandemics. One drawback of these models is that some of them tend to be relatively complex. Common models in this category include, namely: Susceptible-Infectious-Recovered (SIR) model [7], Susceptible-Infectious-Recovered-Deceased (SIRD) model [7], Susceptible-Infectious-Susceptible (SIS) model [8], MaternallyDerivedImmunity-Susceptible-Infectious-Recovered (MSIR) model [9], Susceptible-Exposed-Infectious-Recovered (SEIR) model [9], Susceptible-Exposed-Infectious-Susceptible (SEIS) model [9], Susceptible-UnquarantinedInfected-QuarantinedInfected-ConfirmedInfected (SUQC) model [10], etc.

4.2.3 Computational Models

In this category, there exist two (2) major subcategories of models used for epidemiology-related problems, viz: Agent-based Models and Machine Learning models.

On one hand, an Agent-based Model (ABM) is a computational model that re-creates a system, via simultaneously simulating the interactions of several autonomous agents, with the goal of analyzing and predicting potential event(s) about the given system. A popular downside of these models is that they tend to oversimplify, thereby yielding unauthentic

predictions. Research which employed ABM with regard to the open problems of COVID-19 include, viz: [11], [12], [13], [14], [15], etc.

On the other hand, a ML model is an AI approach such that a computational model is constructed, via learning from sample (or training) data, so as to extract inherent patterns about a given system which will be applied in making predictions and/or decisions about the given system. A common challenge associated with employing these models is the availability of good and sufficient sample data for training these models. Literature which have employed ML approach toward resolving COVID-19-related problems include, viz: [16], [17], etc.

4.3 Proposed Framework and Methodology

This section is subdivided as follows: subsection 4.3.1 (problem definition), subsection 4.3.2 (proposed methodology), and subsection 4.3.3 (proposed system framework and algorithms).

4.3.1 Definition of Problem

Definition 4.1. *COVID-19 Monitoring: Given a set of feature or independent variables,* $X \in \mathbb{R} : x_{i,1}, x_{i,2}, ..., x_{i,j}$*, such that the shape of the feature space is an* $i \times j$ *feature vector; and a set of target or dependent variables,* $Y \in \mathbb{Z} : y_{i,1}, y_{i,2}, ..., y_{i,k}$*, such that the shape of the target space is an* $i \times k$ *target vector. Our COVID-19 Monitoring framework aims at training a ML function,* $f_m : X \rightarrow Y \equiv x_{i,*} \mapsto y_{i,*}$*, which learns to effectively and efficiently make predictions about* Y *based on the patterns of information learnt from* X*.*

Table 4.1: Primary features constituting the feature space of our framework.

	Initial (or Primary) Features			
	Category	Code	Feature Name	Description or Details of Feature
1	Biological Factors	feat_01	Virus Reprod. Index	The Effective Reproduction Number of SARS-CoV-2.
2	Environment Factors	feat_02	Climate	Canadian seasonal periods of the year (1 = Spring, 2 = Summer, 3 = Autumn, 4 = Winter).
3		feat_03	Dry Land	Area (in km^2) of land inhabited by the populace, exclusive of aquatic habitat.
4		feat_04	Region	Numeric encoding of each region/province (0 = Alberta, 1 = British Columbia, 2 = Manitoba, 3 = New Brunswick, 4 = Newfoundland and Labrador, 5 = Nova Scotia, 6 = Ontario, 7 = Prince Edward Island, 8 = Quebec, 9 = Saskatchewan).
5	Government Actions	feat_05	Wave	Pandemic phase (1 = first wave, 2 = second wave).
6		feat_06	Cumm. Vaccine	Cumulative record of inoculated persons.
7		feat_07	Lockdown	Stages of restrictions with regard to public health safety (1 = Lockdown scenario, 2 = Partial/Restricted reopening, 3 = Total/Full reopening).
8		feat_08	Travel Restrict	Implementation of travel restrictions (0 = No restriction, 1 = Federal government restriction, 2 = Provincial government restriction).
9		feat_09	Province Face-Cover	Implementation of compulsory face covering (0 = Not compulsory, 1 = Mandatory/Compulsory).
10		feat_10	Holiday	Effective days of holiday (0 = Workday, 1 = Holiday).
11		feat_11	CHCentres	Total count of Community Health Centres in that region/province.
12	Human Factors	feat_12	retail and recreation change	Deviation from the normal regarding visitations to retail/recreation centres.
13		feat_13	grocery and pharmacy change	Deviation from the normal regarding visitations of grocery/pharmacy centres.
14		feat_14	parks change	Deviation from the normal regarding visitations to camps/parks centres.
15		feat_15	transit stations change	Deviation from the normal regarding visitations to public transit stations.

Thus, $y_{i,*} \in Y = f_m(x_{i,*} \in X)$.

Table 4.2: Primary features constituting the feature space of our framework.

16		feat_16	workplaces change	Deviation from the normal regarding visitations to offices and workplaces.
17		feat_17	residential change	Deviation from the normal regarding visitations to residential apartments and buildings.
18		feat_18	Return Travellers	Number of travelers returning to this region as their destination.
19		feat_19	Employ Rate	Employment rate (%) of the region or province.
20		feat_20	Unemploy Rate	Unemployment rate (%) of the region or province.
21		feat_21	Labor Popln	Eligible workforce for the region or province.
22		feat_22	0 - 34 (M)	Male populace of age range: 0 - 34.
23		feat_23	35 - 69 (M)	Male populace of age range: 35 - 69.
24		feat_24	70 - Above (M)	Male populace of age range: 70 and above.
25		feat_25	0 - 34 (F)	Female populace of age range: 0 - 34.
26		feat_26	35 - 69 (F)	Female populace of age range: 35 - 69.
27		feat_27	70 - Above (F)	Female populace of age range: 70 and above.

4.3.2 Proposed Methodology

Essentially, this encompasses, namely: 4.3.2 (Feature Engineering Layer), 4.3.2 (Feature Extraction Layer), 4.3.2 (Feature Selection Layer), and 4.3.2 (Feature Scaling Layer).

Feature Engineering Layer

The feature space of our TL framework is established with respect to four (4) categories of influential factors, viz: SARS-CoV-2 Biological Factors, Environmental Factors, Government Actions, and Human Factors. These factors tend to affect the spread of COVID-19 with reference to a given populace. Features have been aggregated, based on the aforementioned categories of influential factors, with reference to any given population. Therefore, the initial or primary feature space is an $i \times 27$ feature vector with an elastic sample span. Tables 4.1 and 4.2 present additional details about the primary feature space of our frame-

Table 4.3: Secondary features constituting the feature space of our framework.

	Derived (or Secondary) Features			
	Category	Code	Feature Name	Description or Details of Feature
28	Derived Factors	feat_28	Derived 01	Sum of feat_12 to feat_17 ($\sum_{z=12}^{z=17} feat\_z$).
29		feat_29	Derived 02	Ratio of feat_28 to Total Population (feat_28 : $\sum_{z=22}^{z=27} feat\_z$).
30		feat_30	Derived 03	Ratio of feat_01 to feat_28 (feat_01 : feat_28).
31		feat_31	Derived 04	Ratio of feat_05 to feat_02 (feat_05 : feat_02).
32		feat_32	Derived 05	Ratio of feat_01 to Total Population (feat_01 : $\sum_{z=22}^{z=27} feat\_z$).
33		feat_33	Derived 06	Ratio of feat_01 to feat_18 (feat_01 : feat_18).
34		feat_34	Derived 07	Ratio of feat_01 to feat_22 (feat_01 : feat_22).
35		feat_35	Derived 08	Ratio of feat_01 to feat_23 (feat_01 : feat_23).
36		feat_36	Derived 09	Ratio of feat_01 to feat_24 (feat_01 : feat_24).
37		feat_37	Derived 10	Ratio of feat_01 to feat_25 (feat_01 : feat_25).
38		feat_38	Derived 11	Ratio of feat_01 to feat_26 (feat_01 : feat_26).
39		feat_39	Derived 12	Ratio of feat_01 to feat_27 (feat_01 : feat_27).
40		feat_40	Derived 13	Ratio of feat_06 to Total Population (feat_06 : $\sum_{z=22}^{z=27} feat\_z$).
41		feat_41	Derived 14	Ratio of feat_18 to Total Population (feat_18 : $\sum_{z=22}^{z=27} feat\_z$).
42		feat_42	Derived 15	Ratio of feat_18 to feat_03 (feat_18 : feat_03).
43		feat_43	Derived 16	Ratio of Total Population to feat_03 ($\sum_{z=22}^{z=27} feat\_z$: feat_03).
44		feat_44	Derived 17	Ratio of feat_11 to feat_03 (feat_11: feat_03).

work.

Feature Extraction Layer

Furthermore, based on a basic examination of the initial/primary feature space, we were able to extract derived/secondary features. These derived features were computed via the application of arithmetic ratios and proportions to selected features of the initial/primary feature space. Therefore, the shape of the derived/secondary feature space is an $i \times 17$ feature vector with an elastic sample span. Details of these derived features are contained in Table 4.3.

Table 4.4: Highly relevant features constituting the final feature space of our framework.

	Final (or Highly Relevant) Features						
	Category	Code	Feature Name	Relevance Score (%) per Target Variable			
				Infected	Hospitalized	Recovered	Death
1	Relevant Features or Factors	feat_05	Wave	100%	31%	83%	18%
2		feat_23	35 - 69 (M)	85%	98%	68%	95%
3		feat_21	Labor Popln	83%	93%	66%	90%
4		feat_24	70 - Above (M)	83%	100%	66%	100%
5		feat_27	70 - Above (F)	82%	99%	66%	99%
6		feat_26	35 - 69 (F)	82%	93%	66%	90%
7		feat_22	0 - 34 (M)	81%	88%	65%	84%
8		feat_25	0 - 34 (F)	81%	88%	65%	84%
9		feat_11	CHCentres	76%	80%	62%	72%
10		feat_06	Cumm. Vaccine	68%	54%	100%	42%
11		feat_03	Dry Land	64%	92%	54%	95%
12		feat_17	residential change	51%	60%	52%	60%
13		feat_07	Lockdown	23%	44%	19%	52%

Also, the linear concatenation of the initial (primary) and the derived (secondary) feature spaces, $i \times 27$ and $i \times 17$, temporarily expands our overall feature space to an $i \times 44$ feature vector.

Feature Selection Layer

In this layer, the dimensionality of the overall feature space, $i \times 44$ feature vector, is effectively and efficiently reduced to yield a feature space comprising only highly relevant features. These relevant features possess a high-degree influence with respect to the prediction of the target (or dependent) variables. In view of our study herein and experiment framework, on one hand, the shape of the final feature space of our model is an $i \times 13$ feature vector. Essentially, this means that our proposed model learns to generalize based only on 13 highly relevant features per dataset (for each province or geographical region).

On the other hand, the target or dependent variables comprise, viz:

(1) Predictions of SARS-CoV-2 infections (Infected: $y_{*,1} \subseteq Y^I$).

(2) Predictions of hospitalized COVID-19 patients (Hospitalized: $y_{*,2} \subseteq Y^H$).

(3) Predictions of patients' recoveries from COVID-19 (Recovered: $y_{*,3} \subseteq Y^R$).

(4) Predictions of deaths (mortality) related to COVID-19 (Death: $y_{*,4} \subseteq Y^D$).

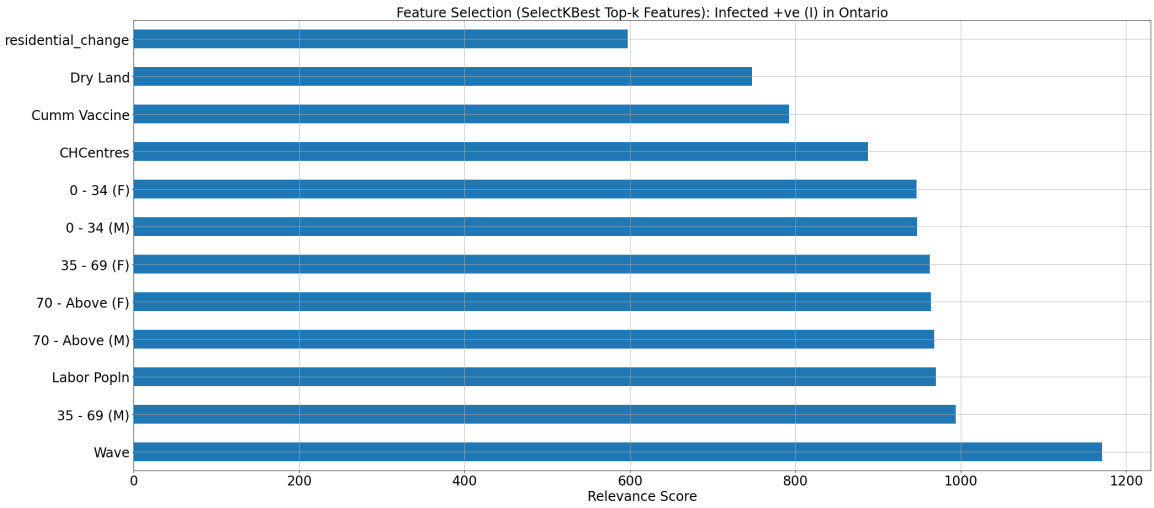

Figure 4.1: Relevant features influencing the infection rate of COVID-19 in Canada.

Table 4.4 provides in detail the 13 highly relevant features which constitute the final feature space. The relevance score with respect to each target or dependent variable is indicated via columns: 'Infected', 'Hospitalized', 'Recovered', and 'Death', respectively. Also, Figures 4.1, 4.2, 4.3, and 4.4 graphically display the relevance score computed against each feature or independent variable based on SelectKBest [18] feature-selection strategy. Thus, only features whose average relevance score exceeds 50% were considered in the final feature space.

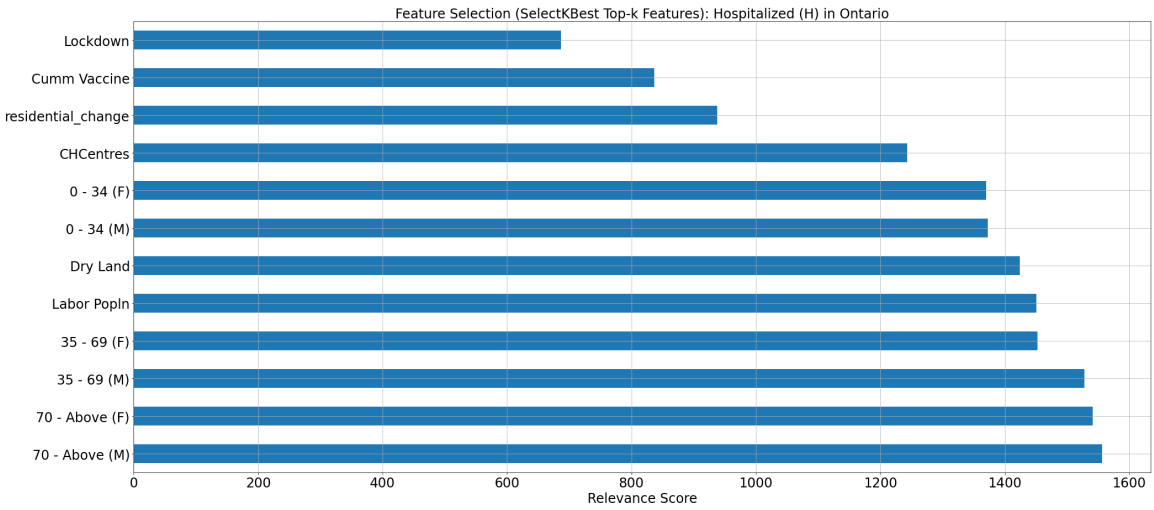

Figure 4.2: Relevant features influencing the hospitalization rate of COVID-19 in Canada.

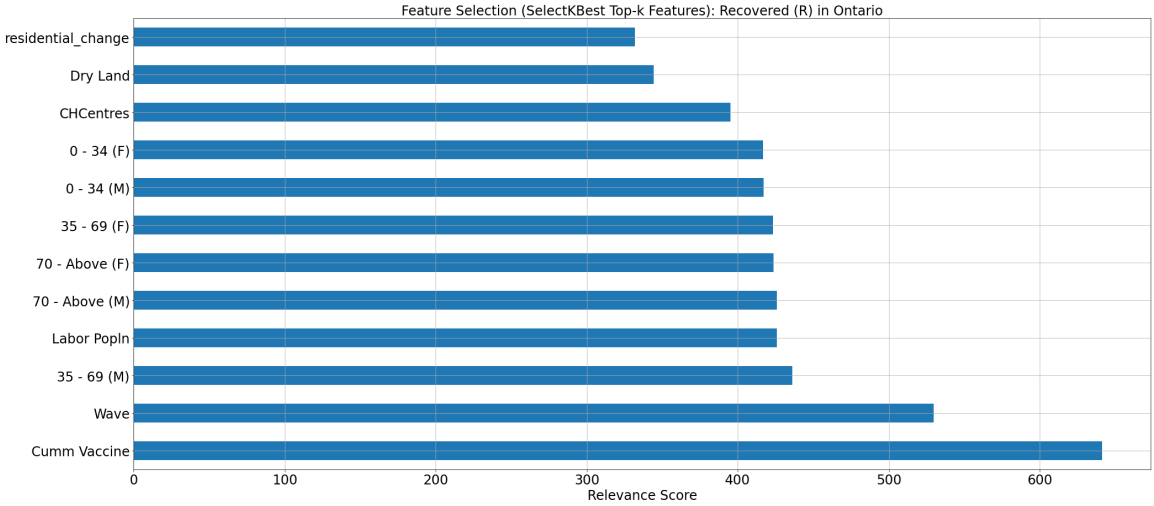

Figure 4.3: Relevant features influencing the recovery rate of COVID-19 in Canada.

Feature Scaling Layer

After reviewing the final feature/independent variables of our data distribution, we noticed a lot of skewness in the data representation of the feature space. This problem of skewness, within the feature space, has to be overcome so as to improve the effectiveness of our model.

The constituent data of every feature variable, in the feature space, has been standardized (column-wise) to a standard normal data-distribution by means of Nonlinear Data

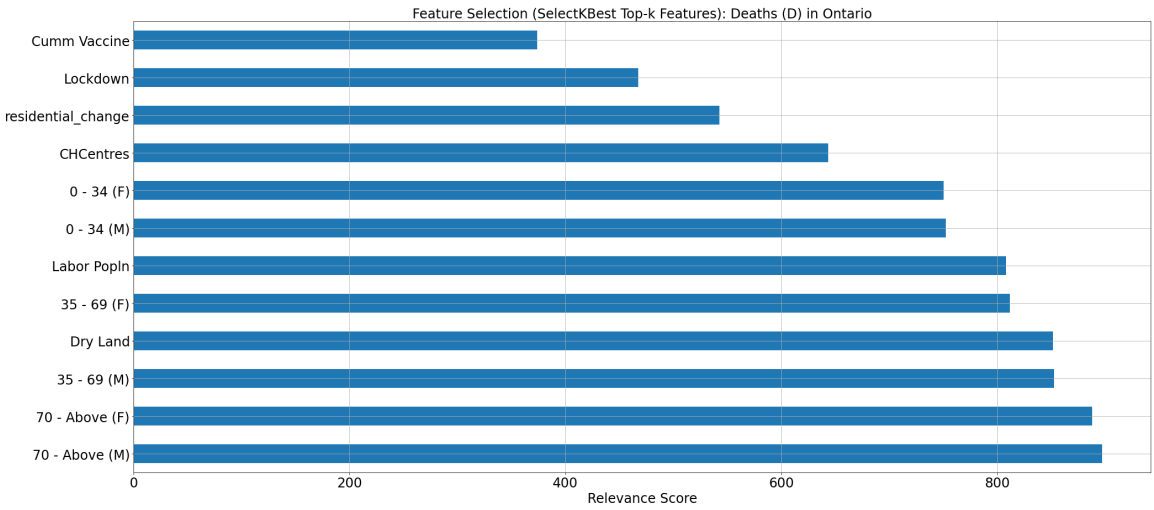

Figure 4.4: Relevant features influencing the mortality rate of COVID-19 in Canada.

Transformation [18] techniques as expressed in equation 4.1.

$$F(x_{*,j}) \equiv \mathbb{P}(x_{*,j} \leq X) = p_{*,j} \in [0,1],\ X \in \mathbb{R} \quad \text{Cumulative Distribution function}$$

$$q_{*,j} \in Q \equiv F^{-1}(p_{*,j}) = min\{x_{*,j} \in \mathbb{R} : F(x_{*,j}) \geq p_{*,j}\} \quad \text{Quantile (Q) function} \tag{4.1}$$

$$z_{*,j} \in Z = \frac{q_{*,j} - \mu}{\sigma} \quad \text{Standard Score (Z) function}$$

In equation 4.1, F and F^{-1} denote the Cumulative Distribution and Quantile functions, respectively. Consequently, the output (Q) of the Quantile function is centered, on a mean ($\mu = 0.0$) and a standard deviation ($\sigma = 1.0$), to yield a standard normal distribution (Z). Thereafter, each standard-normal row/sample, $z_{i,*} \in Z$, of the feature space has been normalized (row-wise) to yield a unit vector via L2-Normalization [19] technique as denoted in equation 4.2. i and j denote the dimensions of the rows and columns per feature vector.

$$\hat{z}_{i,*} \in \mathbb{R} \equiv \sum_{a=1}^{j} (z_{i,a})^2 = (z_{i,1})^2 + (z_{i,2})^2 + ... + (z_{i,j})^2 = 1 \tag{4.2}$$

Taking the dependent variables into consideration, the constituents of the target space have been transformed (column-wise) to a real distribution, $0 \leq \mathbb{R} \leq 1$, by means of MinMaxScaler [18] technique as expressed in equation 4.3.

$$y'_{*,k} \in Y' \equiv G(Y) = \frac{y_{*,k} - min(y_{*,k})}{max(y_{*,k}) - min(y_{*,k})},\ Y \in \mathbb{Z} \quad \text{MinMaxScaler (G)} \tag{4.3}$$

i and k denote the dimensions of the rows and columns per target vector.

4.3.3 Proposed System Architecture and Algorithms

Training a Machine Learning model solely on COVID-19 daily records, which are based on one regional or provincial dataset, has a great tendency for overfitting on the training dataset and/or underfitting on the validation and test datasets. At the moment, COVID-19 daily records spans approximately 400 records (that is 1 record per day). Thus, a training dataset comprising barely 400 records tend to yield a relatively low degree of freedom, with respect to the feature space or independent variables, during ML training. In a bid to overcome these aforementioned challenges, we have adopted a Transfer Learning technique, as represented via Figure 4.5 and Algorithm 4.1, to effectively improve the generalization results with respect to COVID-19 monitoring.

On one hand, we have trained a *Generic* ML-model component on datasets aggregated from several provinces in Canada (exclusive of the province referenced as the case study). On the other hand, we have pre-trained a *Dedicated* ML-model component via *Transfer of Learning* from the *Generic* ML-model component. Subsequently, the *Dedicated* ML-model

component is further trained using datasets acquired from the case study province or region.

Generalizations, with regard to predictions for the case study province, are effectuated via the *Dedicated* ML-model component of our TL framework. Also, the high point of our proposed TL framework is that it can be readily adapted for making generalizations or predictions for any province/region. This is achieved by simply interchanging the regional dataset used for training the *Dedicated* ML-model component with a regional dataset used for training the *Generic* ML-model component.

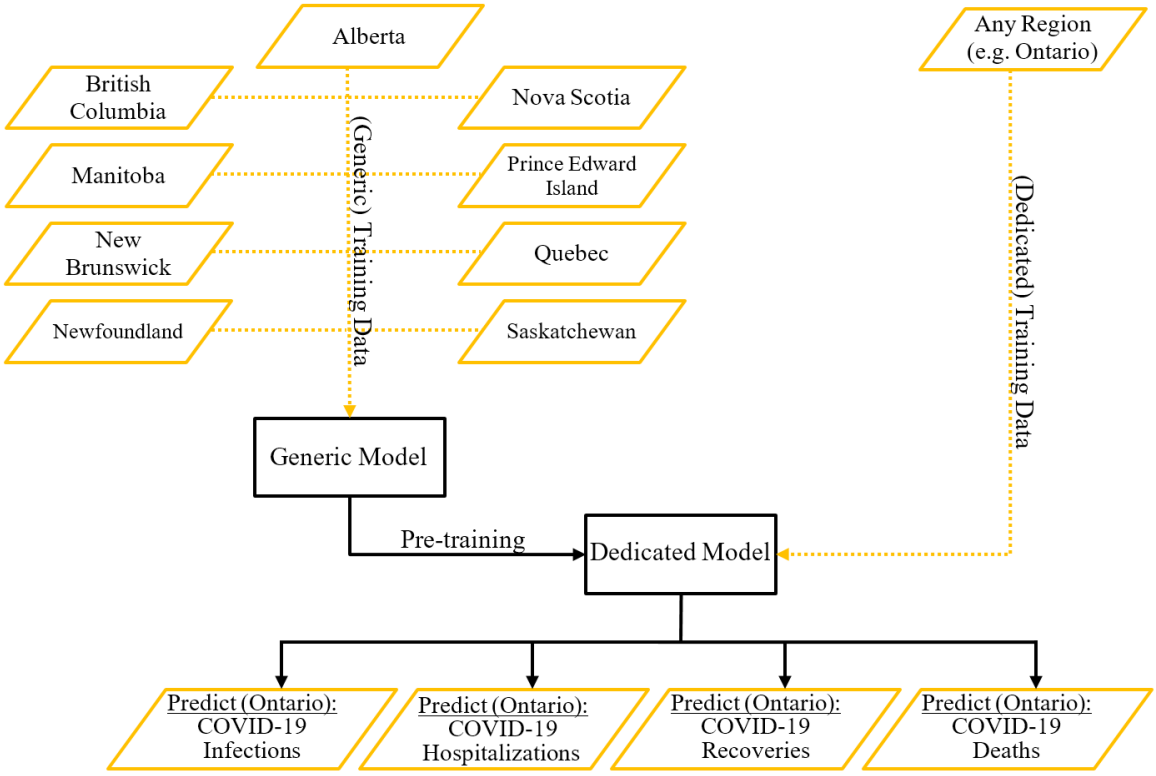

Figure 4.5: Proposed architecture of our TL model for COVID-19 monitoring.

Algorithm 4.1 Transfer Learning Model for COVID-19 Monitoring

`/* See Table 4.4 */`

Input: $\{X : x_{*,1}, x_{*,2}, ..., x_{*,13}\} \equiv \{X^{Generic}_{*,*}, X^{Dedicated}_{*,*}\}$

`/* See subsection 4.3.2 */`

Output: $\{Y : y_{*,1}, y_{*,2}, y_{*,3}, y_{*,4}\} \equiv \{Y^{Generic}_{*,*}, Y^{Dedicated}_{*,*}\}$

Data: Regional datasets containing X and Y `// See Table 4.5`

35 **Program** `Main(`X, Y`):`

`/* Feature Scaling: See subsection 4.3.2 */`

36 **for** *j = 1 to 13* **do**

37 $F(x_{*,j} \in X) \equiv \mathbb{P}(x_{*,j} \leq X) = p_{*,j}$

$q_{*,j} \in Q \equiv F^{-1}(p_{*,j}) = min\{x_{*,j} \in \mathbb{R} : F(x_{*,j}) \geq p_{*,j}\}$

$z_{*,j} \in Z = \dfrac{q_{*,j} - \mu}{\sigma},\ \mu = 0.0,\ \sigma = 1.0$

38 $\hat{z}_{i,*} \in \hat{Z} \in \mathbb{R} \equiv \sum_{a=1}^{j}(z_{i,a})^2 = 1$

$\hat{Z}^{Generic}_{*,*} : \hat{Z} \longrightarrow X^{Generic}_{*,*};\quad \hat{Z}^{Dedicated}_{*,*} : \hat{Z} \longrightarrow X^{Dedicated}_{*,*}$

for *k = 1 to 4* **do**

39 $y'_{*,k} \in Y' \equiv G(y_{*,k} \in Y) = \dfrac{y_{*,k} - min(y_{*,k})}{max(y_{*,k}) - min(y_{*,k})},\ Y \in \mathbb{Z}$

40 $Y'^{Generic}_{*,*} : Y' \longrightarrow Y^{Generic}_{*,*};\quad Y'^{Dedicated}_{*,*} : Y' \longrightarrow Y^{Dedicated}_{*,*}$

`/* Training via Transfer Learning: See Figure 4.5 */`

41 $f_{Generic} : \hat{Z}^{Generic}_{*,*} \longrightarrow Y'^{Generic}_{*,*}$

$f_{Dedicated} = f_{Dedicated} + f_{Generic}$

$f_{Dedicated} : \hat{Z}^{Dedicated}_{*,*} \longrightarrow Y'^{Dedicated}_{*,*}$

$Y'^{Dedicated}_{i,*} = f_{Dedicated}(\hat{Z}^{Dedicated}_{i,*})$

return $Y'^{Dedicated}_{i,*} \approxeq \{y_{i,1} \in Y^I, y_{i,2} \in Y^H, y_{i,3} \in Y^R, y_{i,4} \in Y^D\}$

Figure 4.6 and Algorithm 4.2 showcase our proposed model with regard to the prediction of PPE demand(s), by regional or provincial CHCs, in relation to the COVID-19 pandemic. We have employed an interpolation technique herein, which relies on the predictions of hospitalized COVID-19 patients ($y_{*,2} \subseteq Y^H$).

For each i^{th} day prediction of PPE demand by a CHC, we instantiate Algorithm 4.2 and initialize the following variables:

(i) The i^{th} day prediction of Hospitalized COVID-19 patients ($y_{i,2}$);

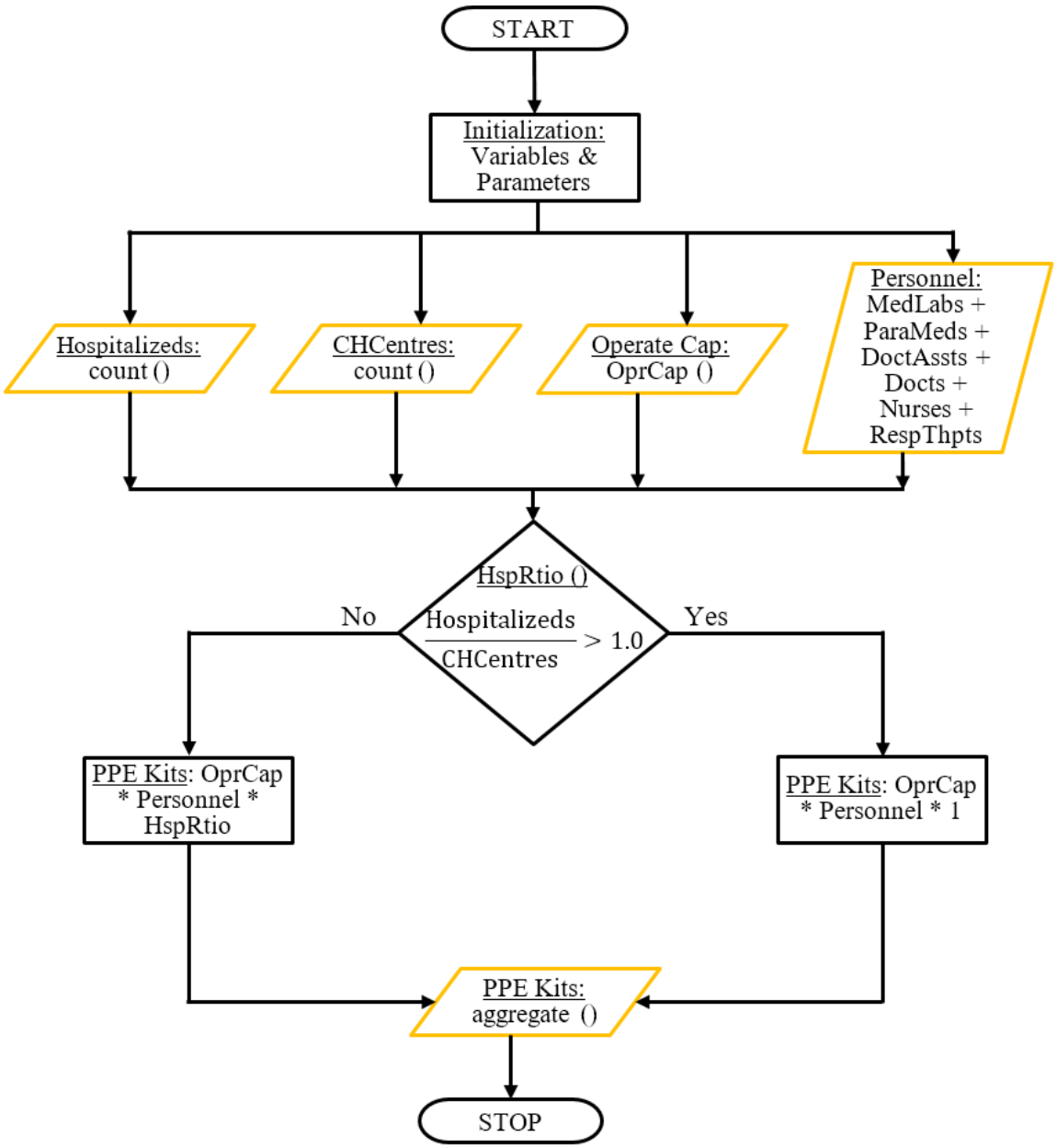

Figure 4.6: Proposed architecture for predicting PPE consumption in CHC.

(ii) The number of operational (regional) CHCs on the i^{th} day ($feat\_11_i$);

(iii) The i^{th} day average operating capacity of the available COVID-19-related workforce in all regional CHCs ($OprCap_i$); and

(iv) The i^{th} day count of available COVID-19-related workforce ($Personnel_i$), whom usu-

Algorithm 4.2 PPE Consumption/Demand Prediction in CHCs

Input: $\{y_{*,2} \subseteq Y^H, feat\_11, OprCap, Personnel\}$ // See Figure 4.6

Output: $\mid PPE\,Kits_* \mid$ // See Figure 4.6

Function ppeDemandPred($Y^H, feat\_11, OprCap, Personnel$):

/* Initialization (Variables and Parameters) */

$y_{i,2} \in Y^H \in \mathbb{Z}$ // Count of hospitalized COVID-19 patients

$feat\_11_i = len(CHCentres_i)$ // Count of regional CHCs

$OprCap_i = 0 \leq \mathbb{R} \leq 1$ // Operating capacity of workforce

$Personnel_i = MedLabs_i + ParaMeds_i + DoctAssts_i + Docts_i + Nurses_i + RespThpts_i$ // Available COVID-19 workforce

$PPE\,Kits_i = 0$ // Count of predicted PPE kit(s)

for *i = 0 to m* **do**

$HspRtio_i = \dfrac{y_{i,2}}{feat\_11_i}$ // Compute hospitalization ratio

if $HspRtio_i > 1.0$ **then**

$PPE\,Kits_i = OprCap_i \times Personnel_i \times 1.0$

else

$PPE\,Kits_i = OprCap_i \times Personnel_i \times HspRtio_i$

return $PPE\,Kits_*$ // $PPE\,Kits_i \approxeq i^{th}$ day PPE-Kit demand

ally come in contact with Hospitalized COVID-19 patients, in all regional CHCs.

Subsequently, we compute the i^{th} day value for the dependent variable, $HspRtio_i$, which denotes the average number of Hospitalized COVID-19 patients per CHC in each region. In other words, $HspRtio_i = y_{i,2} : feat\_11_i$.

Afterwards, if $HspRtio_i \geq 1$ is true, it signifies that there exist at least 1 Hospitalized COVID-19 patient in every CHC for each region or province. Hence, we estimate the PPE-Kit demand for the i^{th} day ($PPE\,Kits_i$) as the product: $OprCap_i \times Personnel_i \times 1.0$.

However, if $HspRtio_i < 1$ is true, it indicates that there exist some CHCs in each region with no/zero (0) Hospitalized COVID-19 patient. In this regard, we estimate the PPE-Kit demand for that i^{th} day ($PPE\,Kits_i$) as the product: $OprCap_i \times Personnel_i \times HspRtio_i$.

4.4 Materials and Methods

Subsection 4.4.1 as well as Table 4.5 gives a detailed overview of the COVID-19 datasets (per province in Canada) employed herein for our experiments and evaluations. Subsection 4.4.2 outlines the hyperparameter configurations of our proposed TL model, and all the dependencies (libraries and software) which were used to facilitate our research, proposed framework, and experiments. Subsection 4.4.3 formally defines all the objective functions implemented herein with respect to the evaluation of our experiments. Subsection 4.4.4 describes the content as well as the directory structure of our remote (GitHub) code repository.

4.4.1 Datasets

The benchmark datasets employed herein are defined in Table 4.5.

Table 4.5: Benchmark datasets.

	Dataset	Start Date	End Date	Description
1	Alberta	January 25^{th}, 2020	January 20^{th}, 2021	Each dataset, with regard to a province in Canada, contains variables of the target space (defined in subsection 4.3.2) and variables of the feature space (defined in Tables 4.1 and 4.2). Furthermore, every regional or provincial dataset comprises 362 rows which represent daily epidemiological records spanning from January 25^{th}, 2020 to January 20^{th}, 2021.
2	British Columbia			
3	Manitoba			
4	New Brunswick			
5	Ontario			
6	Quebec			
7	Saskatchewan			

4.4.2 Materials and software

To facilitate the development, implementation, and benchmarking of our proposed TL framework herein; we have employed the following libraries and/or software, viz: Scikit-Learn [18] and Keras [20]. Moreover, the core regressor function of our proposed TL model is based on k-nearest neighbors [18]. The k-nearest-neighbors regressor has been implemented using its default hyperparameters, as defined in Scikit-Learn [18] library, with exception to the *n_neighbors* and *weights* parameters which we have tuned as: *KNeighborsRegressor(n_neighbors=6, weights='distance')*.

4.4.3 Benchmark Objective Functions

The objective functions employed herein for benchmarking our proposed TL framework are, namely: Coefficient of Determination (R^2), Explained Variance Score (EVS), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Training Time (TT). Taking into account these objective functions defined via equation 4.4; the variables, y and

$\hat{y}$, represent the values of the ground-truth and the prediction, respectively.

$$
\begin{aligned}
R^2(y,\hat{y}) &= 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2};\ \ \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i; \quad R^2 = [-\mathbb{R}, 1.0] = [\text{Worst, Best}] \\
EVS(y,\hat{y}) &= 1 - \frac{\sigma^2(y_i - \hat{y}_i)}{\sigma^2(y_i)}; \qquad\qquad EVS = [0.0, 1.0] = [\text{Worst, Best}] \\
MAE(y,\hat{y}) &= \frac{1}{n}\sum_{i=1}^{n} \mid y_i - \hat{y}_i \mid; \qquad\qquad MAE = [0.0, +\mathbb{R}] = [\text{Best, Worst}] \\
RMSE(y,\hat{y}) &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}; \qquad\qquad RMSE = [0.0, +\mathbb{R}] = [\text{Best, Worst}] \\
TT &= \text{seconds}(\textit{//}) \text{ elapsed during ML training;} \quad TT = [0.0, +\mathbb{R}] = [\text{Best, Worst}]
\end{aligned}
\tag{4.4}
$$

4.4.4 Reproducibility

Table 4.6 herein summarizes the structure of our remote (GitHub) repository, which contains all the source codes, with respect to the experiments and evaluations carried out in this research. Additional details with regard to the installation, usage, and content of our proposed TL framework is available via: https://github.com/bhevencious/COVID-19-Monitor/blob/main/README.md

4.5 Experiments, Results, and Discussions

In our experiments and results stated herein (Tables 4.7, 4.8, 4.9, and 4.10), we have rotated the training dataset for the *Dedicated* ML-model component of our TL framework across seven (7) distinct Canadian provinces (Alberta, British Columbia, Manitoba, New Brunswick, Ontario, Quebec, and Saskatchewan). Thus, any region/province used for training the *Dedicated* ML-model component (of our TL framework) is excluded from the training datasets for the *Generic* ML-model component of our TL framework. The objective

Table 4.6: Description of the remote source code repository with regard to our proposed TL framework implemented herein for benchmark experiments.

SN	**Subject:** Remote (GitHub) URL/Link Description
1.	**COVID-19-Monitor:** https://github.com/bhevencious/COVID-19-Monitor Homepage of the code repository for our proposed TL framework
2.	**custom_classes:** https://github.com/bhevencious/COVID-19-Monitor/tree/main/custom_classes Subdirectory containing dependencies (class files) for our TL framework
3.	**preds.py:** https://github.com/bhevencious/COVID-19-Monitor/blob/main/preds.py Primary source code for COVID-19 monitoring and PPE prediction
4.	**datasets:** https://github.com/bhevencious/COVID-19-Monitor/tree/main/datasets Subdirectory containing all the benchmark datasets employed herein with regard to our experiments and evaluations
5.	**plots_and_data:** https://github.com/bhevencious/COVID-19-Monitor/tree/main/datasets/plots_and_data Subdirectory containing all the graphs and plots with respect to the results obtained from experiments and evaluations

Table 4.7: Experiment results for the prediction of Infected cases, via our proposed TL framework, on a test set comprising 54 randomly sampled days (across 7 provinces).

Province	**R^2**	**EVS**	**MAE**	**RMSE**	**TT(s)**
Alberta	0.828	0.830	114.778	213.486	0.011
British Columbia	0.684	0.686	85.407	192.465	0.012
Manitoba	0.908	0.910	23.852	39.031	0.009
New Brunswick	0.800	0.811	1.815	3.453	0.010
Ontario	0.931	0.933	109.611	180.845	0.010
Quebec	0.963	0.966	101.185	145.749	0.007
Saskatchewan	0.865	0.866	16.778	36.267	0.008

functions R^2, EVS and MAE, RMSE, TT attain their best at the values of 1.0 and 0.0, respectively. Additional details regarding the boundaries of best and worst performance for each objective function employed herein is defined in equation 4.4.

Considering our experiment results in Table 4.7, Table 4.8, Table 4.9, and Table 4.10: as

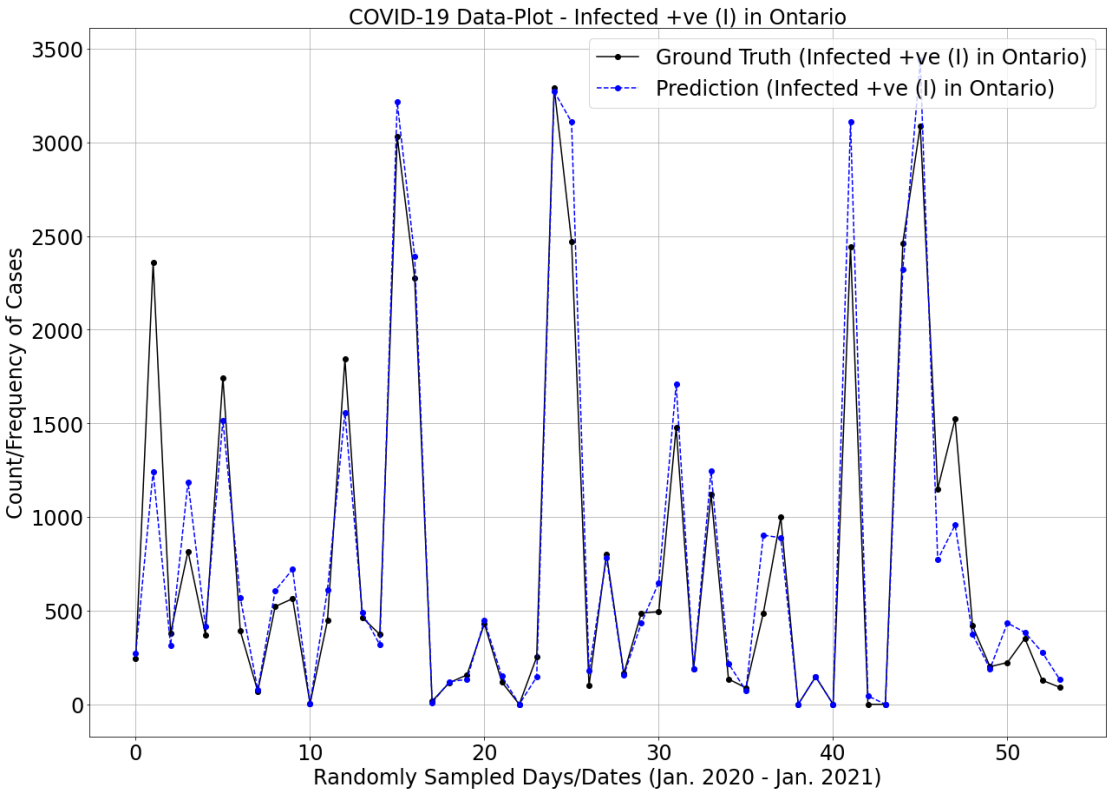

Figure 4.7: Prediction of Infected cases, via our proposed TL framework, on a validation/test set comprising 54 randomly sampled days (Ontario, Canada).

Table 4.8: Experiment results for the prediction of Hospitalized cases, via our proposed TL framework, on a test set comprising 54 randomly sampled days (across 7 provinces).

Province	$\mathbf{R^2}$	**EVS**	**MAE**	**RMSE**	**TT(s)**
Alberta	0.950	0.950	26.796	51.674	0.011
British Columbia	0.917	0.921	20.981	33.417	0.012
Manitoba	0.789	0.792	18.056	52.294	0.009
New Brunswick	0.609	0.609	0.741	1.305	0.009
Ontario	0.908	0.909	65.000	112.352	0.010
Quebec	0.826	0.826	126.593	232.303	0.007
Saskatchewan	0.924	0.926	7.944	15.035	0.007

the respective values for R^2 approach unity (1), they signify that our proposed TL framework fits adequately to the COVID-19 dataset(s); and otherwise, as R^2 approaches zero (0). In a similar fashion, as the respective values for EVS tend to unity (1), they explain to us that our proposed TL model effectively captures and utilizes the data-point variations in

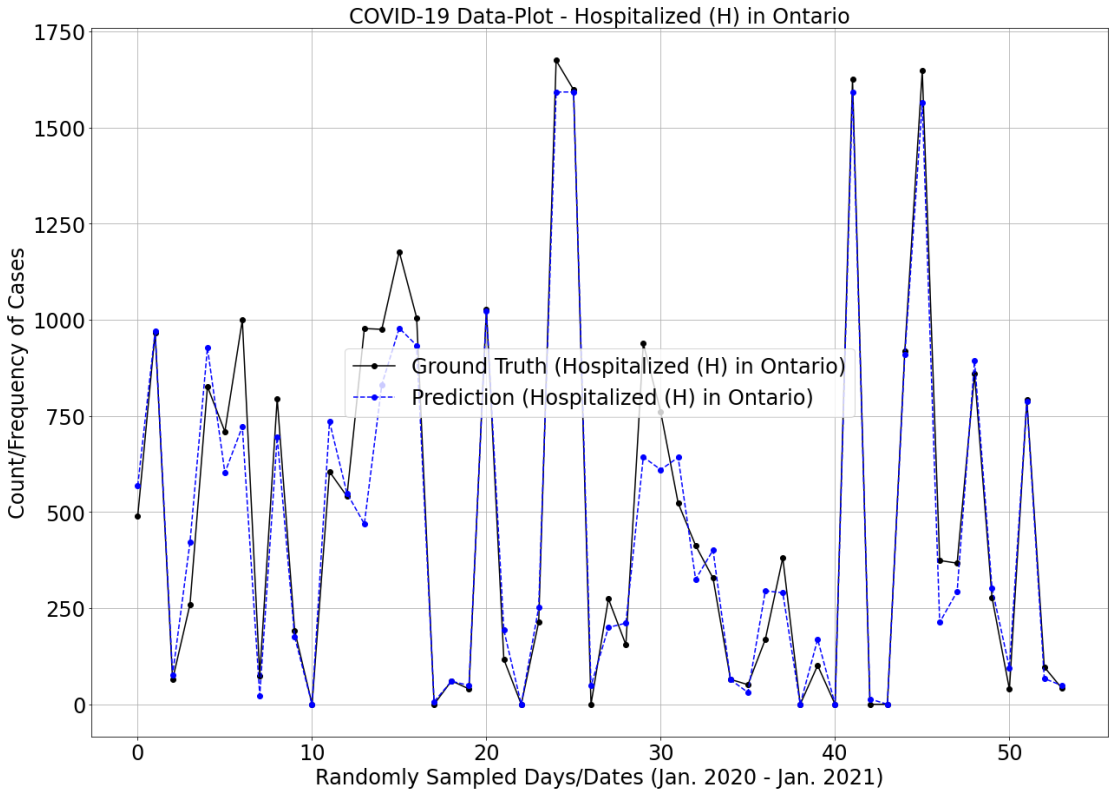

Figure 4.8: Prediction of Hospitalized cases, via our proposed TL framework, on a validation/test set comprising 54 randomly sampled days (Ontario, Canada).

Table 4.9: Experiment results for the prediction of Recovered cases, via our proposed TL framework, on a test set comprising 54 randomly sampled days (across 7 provinces).

Province	$\mathbf{R^2}$	**EVS**	**MAE**	**RMSE**	**TT(s)**
Alberta	0.828	0.835	85.667	181.098	0.011
British Columbia	0.792	0.792	58.278	130.752	0.012
Manitoba	0.726	0.740	20.722	40.928	0.009
New Brunswick	0.578	0.578	1.278	2.305	0.010
Ontario	0.953	0.953	84.519	132.343	0.010
Quebec	0.741	0.742	145.852	329.256	0.007
Saskatchewan	0.758	0.767	19.556	44.653	0.007

the COVID-19 dataset(s). EVS explains otherwise as its respective values tend to zero (0). Moreover, MAE and RMSE compare the predictions from our proposed TL model and the ground truth. As the respective values for MAE and RMSE approach zero (0), they imply that our TL model makes predictions with relatively lower residual error(s).

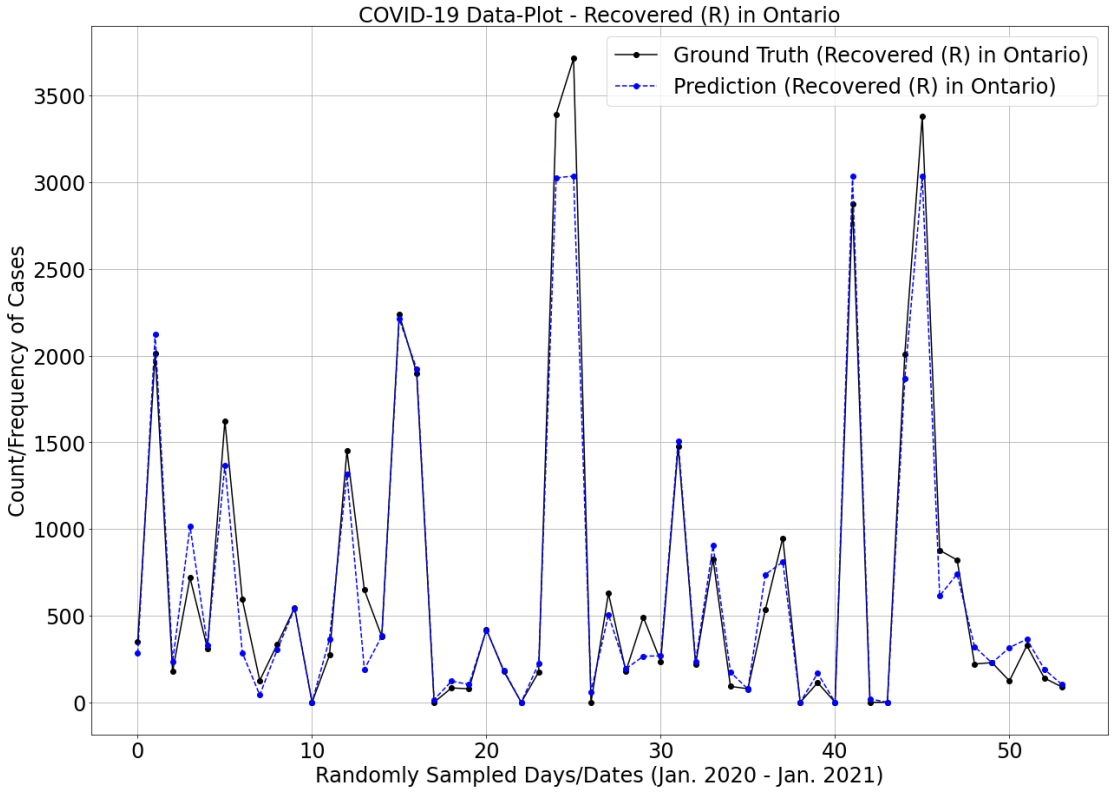

Figure 4.9: Prediction of Recovered cases, via our proposed TL framework, on a validation/test set comprising 54 randomly sampled days (Ontario, Canada).

Table 4.10: Experiment results for the prediction of Death cases, via our proposed TL framework, on a test set comprising 54 randomly sampled days (across 7 provinces).

Province	$\mathbf{R^2}$	**EVS**	**MAE**	**RMSE**	**TT(s)**
Alberta	0.626	0.648	1.704	3.475	0.011
British Columbia	0.602	0.607	2.259	5.128	0.012
Manitoba	0.662	0.669	1.148	2.465	0.009
New Brunswick	0.058	0.048	0.056	0.236	0.009
Ontario	0.658	0.659	5.537	9.869	0.010
Quebec	0.701	0.702	10.389	16.587	0.007
Saskatchewan	0.656	0.665	0.537	1.255	0.007

Taking into consideration our TL framework herein, the constituent independent variables of the final feature space have been optimized for predicting only Infected cases, Hospitalized cases, and Recovered cases. Our TL framework has not been well optimized for the prediction of Death (or Mortality) cases due to the following reasons, viz:

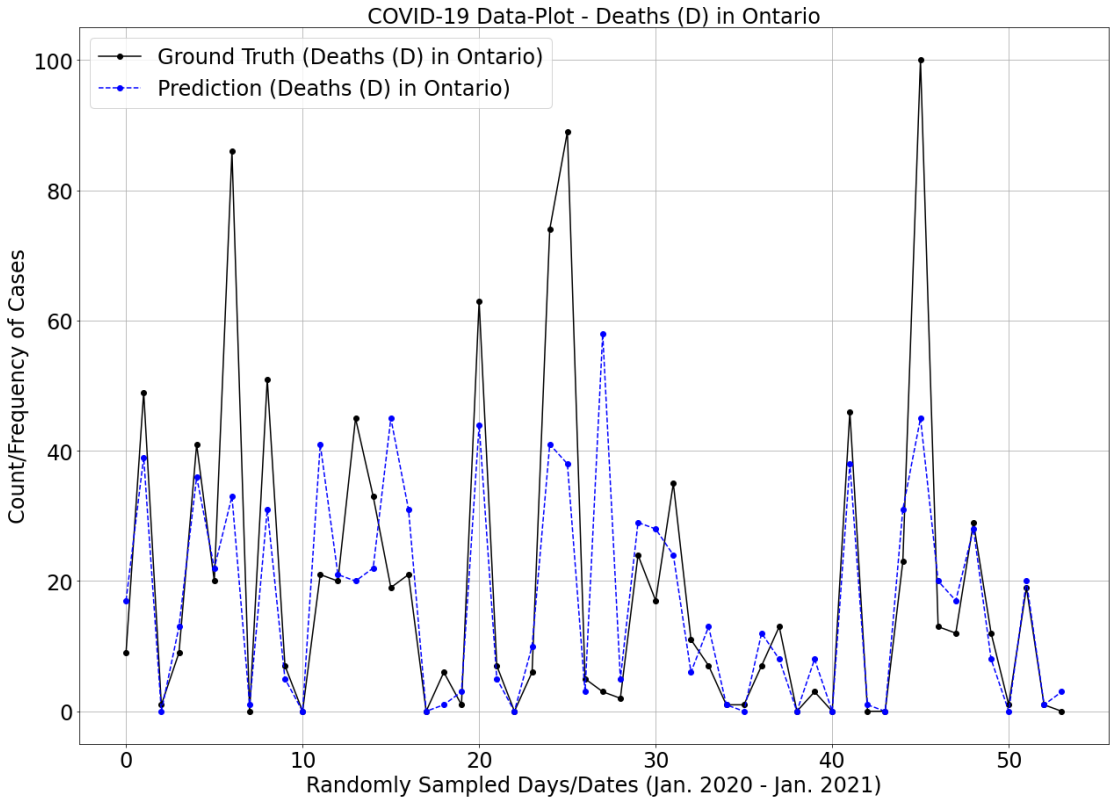

Figure 4.10: Prediction of Death cases, via our proposed TL framework, on a validation/test set comprising 54 randomly sampled days (Ontario, Canada).

(i) Death (or mortality) can occur in COVID-19 patients as a result of several underlying factor(s) and/or risk factor(s) other than SARS-CoV-2.

(ii) Optimizing our TL framework for the prediction of Death cases results in a trade off between the prediction of Infected cases, Hospitalized cases, Recovered cases and the prediction of Death cases. This trade off introduces newer and/or additional parameters (independent variables) which tend to lower the degree of freedom in the training dataset or samples.

Figure 4.7, Figure 4.8, Figure 4.9, and Figure 4.10 depict the generalization results of our TL model (on our province/region of case study) with respect to the prediction of Infected

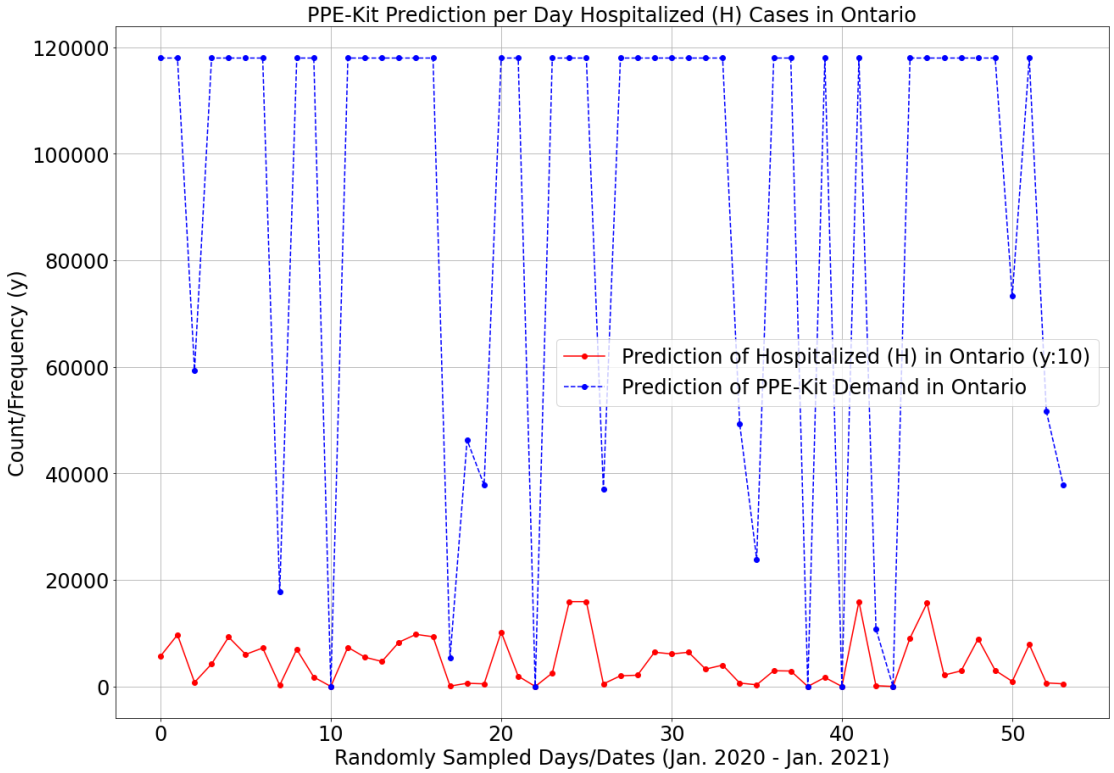

Figure 4.11: Prediction of PPE demand, via our proposed TL framework, on a validation/test set comprising 54 randomly sampled days (Ontario, Canada).

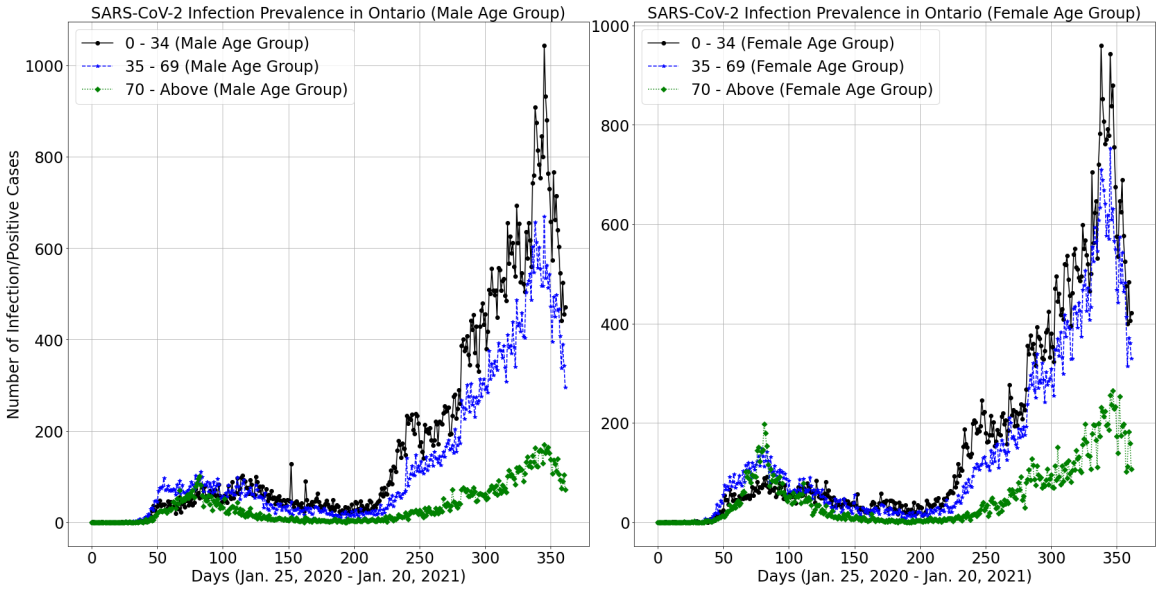

Figure 4.12: COVID-19 prevalence across male and female age groups in Ontario, Canada.

cases, Hospitalized cases, Recovered cases, and Death cases. These generalization results are based on a test dataset comprising 53 randomly selected samples from the province of Ontario (extracted from a date range of January 25^{th}, 2020 to January 20^{th}, 2021). In this regard, the horizontal axes represent the data-points in the test dataset; while the vertical axes represent the case frequencies. For each Figure (4.7, 4.8, 4.9, and 4.10), the black-colored solid line graph represents the ground-truth case frequency per data-point; and the blue-colored dotted line graph represents our TL framework's prediction per data-point. The frequency intervals, between the line graph representing the ground-truth and the line graph representing our model's predictions, indicate the generalization (or residual) error of our proposed TL model. Thus, we hope to effectively minimize this residual error by means of routine incremental learning.

In view of the consumption (or demand) prediction for PPE-kits with reference to our region of case study (province of Ontario), Figure 4.11 represents the generalization results obtained based on Algorithm 4.2 and Figure 4.6. Essentially, Algorithm 4.2 proposed herein effects the prediction for PPE-kits via interpolation into the predictions for Hospitalized cases. Hence, the red-colored dotted line graph corresponds to the prediction for Hospitalized cases; while the blue-colored dotted line graph corresponds to the prediction for PPE-kits consumption (or demand).

In consideration of the feature selection process carried out herein in subsection 4.3.2, we have observed that human-related factors (precisely peer groups) most significantly influence

the rates of Infected cases, Hospitalized cases, and Death cases as can be seen from Figure 4.1, Figure 4.2, and Figure 4.4, respectively. Also, government actions (such as inoculation, pandemic wave, etc.) most significantly influence the rate of Recovered cases as shown in Figure 4.3.

Figure 4.12 represents a distribution plot of COVID-19 prevalence across three major age groups in the province of Ontario (Canada). On one hand, we can see that youths (males and females whom fall within the age group of 0 to 34) greatly influence the spread of SARS-CoV-2 within a given populace. On the other hand, seniors (males and females whom are of age 70 and above) are less likely to influence the spread of SARS-CoV-2 within a given population. However, these seniors (age 70 and above) remain the most susceptible to SARS-CoV-2 due to several age-related risk factors.

4.6 Applications

In view of the virulent nature of SARS-CoV-2, the increase in the number of SARS-CoV-2 strains as a result of mutation, and the global effect of COVID-19 pandemic on businesses and lifestyles; the TL framework proposed herein offers, but not limited to, the following benefits with regard to combating the COVID-19 pandemic, viz:

1. It aids and guides health professionals (epidemiologists, pharmacists, clinicians, etc.) toward understanding several (risk) factors which significantly foster the spread of SARS-CoV-2 within a given populace.

2. It can serve as a validation model to tune and improve the accuracy as well as the precision of ABMs which are prone to oversimplified and/or exaggerated predictions.

3. Our TL framework utilizes and learns from data as situations and conditions evolve; thus, resulting in a much dynamic and flexible solution to COVID-19 monitoring.

4. Prediction of PPE demand and/or consumption with respect to given conditions as a result of Hospitalized COVID-19 patients.

4.7 Limitations, Conclusion, and Future Work

The datasets employed herein for our experiments and analyses were gathered from the date range of January 25^{th}, 2020 to January 20^{th}, 2021. Hence, we assumed that these datasets represent and reflect casualties or cases with reference to the earliest variant of SARS-CoV-2 (UK/British variant). Consecutively, our assumption herein with regard to each unit of PPE-kit is that it comprises five (5) items, namely:

(i) A face shield;

(ii) A N95 respirator or facemask;

(iii) A pair of hand gloves;

(iv) A pair of shoe covers; and

(v) An overall isolation gown.

In summary, we have proposed a TL framework and model for the monitoring COVID-19, based on an engineered feature space, with respect to a given population. Also, we have proposed a model for the prediction of PPE-kit demand(s) and consumption(s) within CHCs. At the moment, we are working on improving the performance of our TL framework in a bid to effectively minimize the generalization (or residual) errors. Moreover, we are still are carrying out extensive validation and testing, on our proposed framework and model, to prove and validate our claims herein. In the near future, we do hope to accomplish the following tasks, viz:

(i) Expand the scope of our benchmark datasets to additional regions and/or populations.

(ii) Benchmark the performance of our proposed TL framework against ABMs.

(iii) Incorporate routine incremental learning to our TL framework and model, with the goal of minimizing the residual errors for improved performance.

(iv) Acquire additional datasets that captures various variants of SARS-CoV-2 (UK/British variant, South African variant, Brazilian variant, etc.).

Acknowledgment

This case has been funded by the Canadian Institute of Health Research (CIHR) Operating Grant: COVID-19 May 2020 Rapid Research Funding Opportunity [operating grant number VR5 172669]; the International Business Machines (IBM) [research was conducted on a high performance IBM Power System S822LC Linux Server].

Bibliography

[1] W. H. Organization, "World health organization (who) coronavirus disease (covid-19) dashboard," Geneva, Switzerland, 2020.

[2] K. Wu, A. P. Werner, J. I. Moliva, M. Koch, A. Choi, G. Stewart-Jones, H. Bennett, S. Boyoglu-Barnum, W. Shi, B. Graham, A. Carfi, K. S. Corbett, R. Seder, and D. K. Edwards, "mrna-1273 vaccine induces neutralizing antibodies against spike mutants from global sars-cov-2 variants," in *bioRxiv*, 2021.

[3] Y. Ben-Shlomo and D. Kuh, "A life course approach to chronic disease epidemiology: conceptual models, empirical challenges and interdisciplinary perspectives." *International Journal of Epidemiology*, vol. 31 2, pp. 285–293, 2002.

[4] H. Burr, M. Formazin, and A. Pohrt, "Methodological and conceptual issues regarding occupational psychosocial coronary heart disease epidemiology." *Scandinavian Journal of Work, Environment and Health*, vol. 42 3, pp. 251–255, 2016.

[5] A. Bernasconi, A. Canakoglu, P. Pinoli, and S. Ceri, "Empowering virus sequences research through conceptual modeling," in *bioRxiv*, 2020.

[6] Q. Lin, S. Zhao, D. Gao, Y. Lou, S. Yang, S. S. Musa, M. Wang, Y. Cai, W. Wang, L. Yang, and D. He, "A conceptual model for the coronavirus disease 2019 (covid-19) outbreak in wuhan, china with individual reaction and governmental action," *International Journal of Infectious Diseases*, vol. 93, pp. 211 – 216, 2020.

[7] W. O. Kermack and A. Mckendrick, "A contribution to the mathematical theory of epidemics," *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 115, pp. 700–721, 1927.

[8] H. Hethcote, "Three basic epidemiological models," in *Applied Mathematical Ecology. Biomathematics, vol 18*, Springer, Berlin, Heidelberg, 1989.

[9] ——, "The mathematics of infectious diseases," *SIAM (Society for Industrial and Applied Mathematics) Review*, vol. 42, pp. 599–653, 2000.

[10] S. Zhao and H. Chen, "Modeling the epidemic dynamics and control of covid-19 outbreak in china," *Quantitative Biology (Beijing, China)*, pp. 1 – 9, 2020.

[11] A. Aleta, D. Martín-Corral, A. P. y Piontti, M. Ajelli, M. Litvinova, M. Chinazzi, N. Dean, M. Halloran, I. Longini, S. Merler, A. Pentland, A. Vespignani, E. Moro, and Y. Moreno, "Modeling the impact of social distancing, testing, contact tracing and household quarantine on second-wave scenarios of the covid-19 epidemic," in *medRxiv*, 2020.

[12] R. Rockett, A. Arnott, C. Lam, R. Sadsad, V. J. Timms, K.-A. Gray, J. Eden, S. Chang, M. Gall, J. Draper, E. Sim, N. Bachmann, I. Carter, K. Basile, R. Byun, M. O'Sullivan, S. C. Chen, S. Maddocks, T. C. Sorrell, D. Dwyer, E. Holmes, J. Kok, M. Prokopenko, and V. Sintchenko, "Revealing covid-19 transmission in australia by sars-cov-2 genome sequencing and agent-based modeling," *Nature Medicine*, pp. 1 – 7, 2020.

[13] C. Kerr, R. M. Stuart, D. Mistry, R. Abeysuriya, G. Hart, K. Rosenfeld, P. Selvaraj, R. C. Nunez, B. Hagedorn, L. George, A. Izzo, A. Palmer, D. Delport, C. Bennette, B. Wagner, S. Chang, J. Cohen, J. Panovska-Griffiths, M. Jastrzebski, A. Oron, E. Wenger, M. Famulare, and D. J. Klein, "Covasim: an agent-based model of covid-19 dynamics and interventions," in *medRxiv*, 2020.

[14] P. C. Silva, P. V. Batista, H. S. Lima, M. A. Alves, F. G. Guimarães, and R. C. Silva, "Covid-abs: An agent-based model of covid-19 epidemic to simulate health and economic effects of social distancing interventions," *Chaos, Solitons & Fractals*, vol. 139, p. 110088, 2020.

[15] S. B. Shuvo, B. C. Molokwu, and Z. Kobti, "Simulating the impact of hospital capacity and social isolation to minimize the propagation of infectious diseases," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[16] D. Zou, L. Wang, P. Xu, J. Chen, W. Zhang, and Q. Gu, "Epidemic model guided machine learning for covid-19 forecasts in the united states," in *medRxiv*, 2020.

[17] M. Kukar, G. Guncar, T. Vovko, S. Podnar, P. Cernelc, M. Brvar, M. Zalaznik, M. Notar, S. Moskon, and M. Notar, "Covid-19 diagnosis by routine blood tests using machine learning," in *ArXiv*, 2020.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products.* San Diego, CA: Academic Press; 7th edition, 2007, ch. 15. Norms.

[20] F. Chollet, Ed., *Deep Learning with Python.* Shelter Island, NY: Manning Publications, 2017.

Chapter 5

Spatial Event Prediction via Multivariate Time Series Analysis of Neighboring Social Units using Deep Neural Networks

Quite similar to our research contribution and application in Chapter 3, in this chapter, we have studied yet another interesting open research problem in Social Network Analysis. Thus, we have examined the problem of Event-based Analysis in SNA using real-world meteorological datasets for our benchmark experiments and comparative analyses. Herein, we have proposed a unique DL-based framework, which is fitted with an adjustment bias. This proposed framework can make effective event predictions about a target social actor, y, based on the analyses and knowledge acquired from correlated and neighboring social actors.

5.1 Introduction

A social network consists of finite set(s) of actors, and the relationship(s) defined between these actors [1]. Identifying and analyzing groups or subgroups of social units using a given set of standards still remains an important research problem in social network analysis (SNA). Thus, the process of developing machine learning models, and training them to predict or forecast spatial events with respect to social units can be very challenging. In fact, they become very difficult when there exist sparse or insufficient records about the activities of the reference actor(s) in the social network; and this problem is what we propose to address herein.

Furthermore, event prediction problem can be expressed as a Satisfiability problem [2] where an event is said to exist if the prediction values for variables necessary to fulfill the event's formal definition reduces it to *true*. Accordingly, the Cook-Levin Theorem [3] has proven that Satisfiability problem is NP-Complete.

Our methodology and approach is relatively different from the conventional event and/or time-series prediction methods. Ideally, our approach is initiated by analyzing a one-mode clique within a social network defined by physically connected cities; and observing the meteorological attributes of each social unit with respect to time and geometric space based on available data. Hence, using the intrinsic patterns and knowledge acquired from the study of each constituent social unit in the actor set; we attempt to predict future meteorological-related events which will affect the social unit, and in turn, influence other related actors

based on the existent social ties within the actor set. Quite simply, we attempt to predict the occurrence of an event in actor-y (which may have poor or insufficient data for lone/independent training) based on the happenings in actor-x (primary data used for training).

Consequently, our proposed framework is based on a neural network architecture assembled using deep layers of stacked MLP. In a bid to develop and train a network that is able to learn the nonlinear distributed representations of a multivariate feature space; we stack several layers of neurons, one layer at a time, to form a deeper network structure [4]. The output layer of our network model is fitted with an adjustment-bias vector which aids the model to make precise and accurate predictions about the nearest neighbor actor. Overall, we evaluated our model against popular methods from Machine (and Deep) Learning as well as Statistical domains.

Our work herein is organized into five sections. It begins with an introductory part; followed by a cross-sectional review of related literature. The third section explains our proposed framework, the adjustment-bias model, data preprocessing techniques, and network training/learning algorithms. The fourth section outlines and discusses our experimentation results based on the evaluation of 10 cross-domain models; and the last section summarizes our research findings (inclusive of our propositions), and future works.

5.2 Related Literature

With regard to the most recent published works in the area of event prediction; the work done by [5] used a Recurrent Neural Network (RNN) model based on Long Short-Term Memory (LSTM) architecture for the prediction of emergency events using datasets acquired through the national police of Guatemala. In the study carried out by [6], the authors proposed an approach based on Gated Recurrent Unit (GRU) for making time series prediction over datasets containing missing data points. Their experiments were carried out using both clinical and synthetic datasets. The individual study by [7] and [8] were based on Multilayer Perceptron (MLP). [7] used MLP for predicting missing data points and the degree of post-operative anemia on real clinical blood samples. [8] designed a MLP model for flood prediction based on the water-elevation level. In addition, the work by [9] used a Random Forest (Rand. Frst.) model for predicting the risks associated with total knee replacements clinical surgeries. The studies carried out by [10] and [11] focused on using Decision Tree (Deci. Tree) models to ascertain rainfall conditions that may induce massive landslides events, and predict vehicular traffic at road intersections respectively. [12] proposed a modified K-Nearest Neighbor (KNN) method for precipitation forecasting using meteorological data records from Beijing, China. Also, [13] proposed a model for forecasting harmful range of indoor/home temperatures during very hot summer conditions which involved experimentations using Autoregressive Moving Average (ARMA) model. Similarly, [14] proposed a model for forecasting the outbreak of breakbone fever in São

Paulo city, and this involved a comparative study using ARMA as one of the benchmark models.

After a comprehensive review of related literature with respect to event prediction in real world, we realized that virtually all event prediction problems were tackled using sufficient data records about the reference object. None of them could make event predictions about a reference object/entity using datasets (or data records) from a nearest-neighbor entity. Moreover, we attempt to classify the approaches proposed in the reviewed literature into 3 categories comprising: Statistical, Machine Learning, and Artificial Neural Network methods.

Beginning with Statistical methods, since we are experimenting with multivariate time series data; these variables have to be expressed as vectors to fit either of Vector Autoregressive (VAR), Vector Moving Average (VMA), or Vector Autoregressive Moving Average (VARMA) models. Accordingly, Vector Autoregressive [15] model of order-p can be formally expressed as:

$$y_t = v + A_1 y_{t-1} + ... + A_p y_{t-p} + u_t \tag{5.1}$$

Also, a q-order Vector Moving Average [16] can be expressed as:

$$\varepsilon_t = u_t + M_1 u_{t-1} + ... + M_q u_{t-q} \tag{5.2}$$

The Vector Autoregressive Moving Average [16] of order-p and order-q, which is a hybridiza-

tion of VAR and VMA models, is as defined below:

$$y_t = v + A_1 y_{t-1} + ... + A_p y_{t-p} + u_t + M_1 u_{t-1} + ... + M_q u_{t-q} \tag{5.3}$$

where t = observation timesteps = $0, \pm 1, \pm 2, ...$; k = variables count = $1, 2, ...$; A_i and M_j are fixed coefficient matrices for the observations; $v = (v_1, ..., v_k)$ denote a fixed vector of intercept terms which allows for the possibility of a non-zero mean; $u = (u_1, ..., u_t)$ is a k-dimensional white noise with zero mean vector; and $y = (y_1, ..., y_t)$ is a multivariate time series of observations.

Secondly, with regard to Machine Learning (ML) methods, models like Linear Regression, Logistic Regression, Stochastic Gradient Descent, etc are well adapted for linear and 1-dimensional regression problems. Decision Tree, Random Forest, and K-Nearest Neighbor models are suited for nonlinear and multiple-target regression problems because they consider the nonlinear correlation existing between multivariate inputs and their corresponding multi-targets. Thus, suppose our dataset is defined by S on p observations: $S = (x_1, y_1), ..., (x_p, y_p)$; formally, the Decision Tree [17] function with respect to Regression Tree [18] implementation in ML can be expressed as:

$$y = f(x, R_n, k_n) = \sum_{n=1}^{N} k_n * I(x \in R_n) \tag{5.4}$$

where N is a partition containing $R_1, ..., R_N$ regions; $I(.)$ is an indicator function returning 1 if its argument is TRUE or 0 if otherwise; and k_n is a constant applied per region. Con-

sequently, "Trees Bagging" or Random Forest [19] is simply the mean of a large collection of de-correlated regression trees; it is mathematically represented as:

$$y = g(x) = \frac{1}{B}\sum_{b=1}^{B} f(x, R_b, k_b) \tag{5.5}$$

where B is the number of generated regression trees; and $f(x, R_b, k_b)$ is the regression tree function. Again, representing our dataset of observations as: $S = (X_1, Y_1), ..., (X_p, Y_p)$; to compute the k-nearest neighbor [20] of x over the S, we have to reorder the dataset, S, according to the increasing Euclidean distances of every X_i to $x = ||X_i - x||$ such that $X_1(x)$ is the 1^{st} nearest neighbor of x amongst X_p. Then the KNN function of x is:

$$y = h(x) = \frac{1}{k}\sum_{i=1}^{k} Y_i(x) \tag{5.6}$$

where k is the number of neighbors.

Thirdly, relating to Artificial Neural Network (ANN) methods, MLP is a quintessential methodology for regression and classification problems. It is defined as a mathematical function, f, mapping some set of input values, x, to output values, y [21]; and it is expressed as:

$$y = f(x, \Theta) \tag{5.7}$$

where Θ is a set of parameters, and the MLP function learns the value(s) of Θ that will result in the best decision approximation. Moreover, RNN models like Basic-RNN (B-

RNN), LSTM, and GRU are still popular techniques for sequence analysis and prediction. Basically, RNNs consist of feedback loops which aid learning within the network; such that each timestep state output, s_t, is computed based on both its current input, x_t, and the output from all other previous time steps, h_{t-n}.

$$\begin{aligned} RNN = S_T &= s_{t-n}, s_{t-2}, s_{t-1}, ..., s_{t+1}, s_{t+n} \\ S_{t-1} &= f(s_{t-2});\ \ S_t = f(s_{t-1});\ \ S_{t+1} = f(s_t); .. \end{aligned} \tag{5.8}$$

Accordingly, a Basic-RNN (B-RNN) [22] whose operation principle is not based on a gated mechanism can be defined as:

$$y_t = \begin{cases} f(x_t, h_{t-1}), & \text{state: } S_t \text{ (simplified)} \\ f(W_{t1}x_t + W_{t2}h_{t-1} + b_t), & \text{state: } S_t \text{ (expressed)} \end{cases} \tag{5.9}$$

Besides, LSTM network is the commonest kind of RNN, and its operation is based on a 3-gate mechanism which enables it to selectively remember or forget any data or information [23]. Formally, the operations with reference to each gate are:

$$LSTM = \begin{cases} f_t : \sigma(W_{f1}x_t + W_{f2}h_{t-1} + b_f), & \text{forget gate} \\ i_t : \sigma(W_{i1}x_t + W_{i2}h_{t-1} + b_i), & \text{input gate} \\ \bar{c}_t : tanh(W_{\bar{c}1}x_t + W_{\bar{c}2}h_{t-1} + b_{\bar{c}}), & \text{input gate} \\ o_t : \sigma(W_{o1}x_t + W_{o2}h_{t-1} + b_o), & \text{output gate} \\ c_t : (f_t * c_{t-1}) + (i_t * \bar{c}_t), & \text{output gate} \\ h_t : o_t * tanh(c_t), & \text{output gate} \end{cases} \tag{5.10}$$

where f_t is the forget-gate operation; i_t and $\bar{c}_t$ are the input-gate operations; and o_t, c_t, h_t are the output-gate operations. Another kind of RNN, similar to LSTM but with fewer gates, is the GRU network. This operates based on a 2-gate mechanism responsible for regulating its memory, and the input/output of data with respect to each unit [24]. Its gate

operations are:

$$GRU = \begin{cases} r_t : \sigma(W_{r1}x_t + W_{r2}h_{t-1} + b_r), & \text{reset gate} \\ \text{update gate:} \\ z_t : \sigma(W_{z1}x_t + W_{z2}h_{t-1} + b_z) \\ \bar{h_t} : tanh(W_{\bar{h}1}x_t + W_{\bar{h}2}(r_t * h_{t-1}) + b_{\bar{h}}) \\ h_t : h_{t-1} * (1 - z_t) + (z_t * \bar{h_t}) \end{cases} \tag{5.11}$$

Thus r_t is the reset-gate operation; and $z_t, \bar{h_t}, h_t$ are update-gate operations. Furthermore, c_{t-1} is the memory from the previous cell or timestep; c_t is the memory of the present timestep; h_{t-1} is the output from the previous cell/timestep; h_t is the output of the present cell; $W_t, W_f, W_i, W_{\bar{c}}, W_o, W_r, W_z, W_{\bar{h}}$ are connection weights associated with respective (gate) inputs; and $b_t, b_f, b_i, b_{\bar{c}}, b_o, b_r, b_z, b_{\bar{h}}$ are respective bias weights.

5.3 Proposed Framework and Methodology

Fig. 5.1 and Table 5.1 illustrate our proposed system structure and machine platform configuration respectively used for our experimentation and analysis.

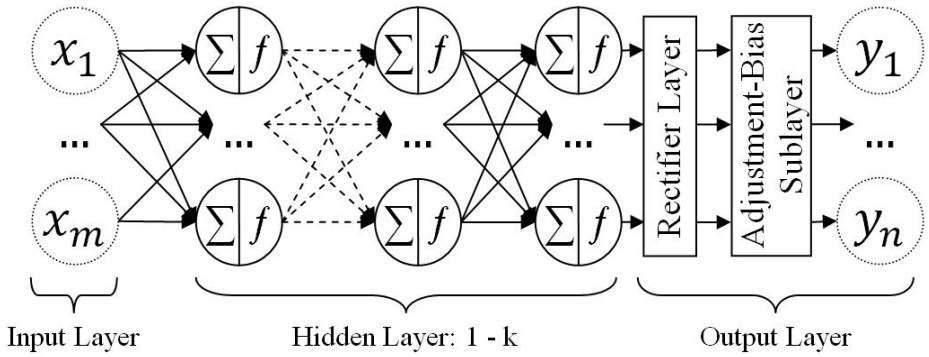

Figure 5.1: Proposed System Architecture

Table 5.1: Experimentation Platform Description

Operating System: *Debian*9.5	**Kernel**: *Linux*4.9.0
CPU Architecture: 64*bits*	**CPUs**: 176
CPU Clock: 3.5GHz	**RAM**: 267GB

5.3.1 Datasets

Data

The sets of data used herein for our experimentation and analysis are real world meteorological records of five major cities in China [25] which are geographically separated by space and time. The readings contained in these datasets were recorded and released by the United States Embassy in Beijing, and its consulates in Chengdu, Guangzhou, Shanghai, and Shenyang respectively. The records in each dataset span from 01-January-2010 00:00 to 31-December-2015 23:00 with exactly 52,584 data rows. In this regard, the feature space of the dataset relevant to our study are, viz: measures of Atmospheric Particles of diameter 0μ to 2.5μ, Season, Dew Point (celsius), Humidity (percentage), Pressure (hectoPascal), Temperature (celsius), Wind-Direction (intercardinal direction), Wind-Speed (metre per second).

Data Preprocessing

Basically, each dataset in its raw form contains several records of missing data points which constitute about 30% of each dataset used herein. Hence, using a ML model which we had earlier developed in our laboratory, we were able to make estimations for every missing data point such that our resultant experimentation dataset comprised 100% complete data

points void of null values. Furthermore, each record of input data was reshaped to a feature space of 1 timestep; while each respective output data record is shaped to 2 timesteps. In this regard, the first output-timestep is the prediction for the current state, s_t, and the second output-timestep is the prediction for next state, s_{t+1}. Thus, for $t+n$ futuristic state predictions, each s_{t+1} output prediction is successively fed back to the model; and retrained as new input, in a bid to improve the model's quality so as to prevent decay, until the target state, s_{t+n}, is attained. Fig. 5.2 depicts the clique structure of the reference cities as the crow flies. We trained our network model using datasets D1, D2, D3, and D4 to make effective meteorological event predictions with respect to the target (or validation) node. The target node is assumed to have very scanty data records of its own; so our goal is to make event predictions with respect to the target node based on its spatial relationship and disposition with neighboring nodes or actors.

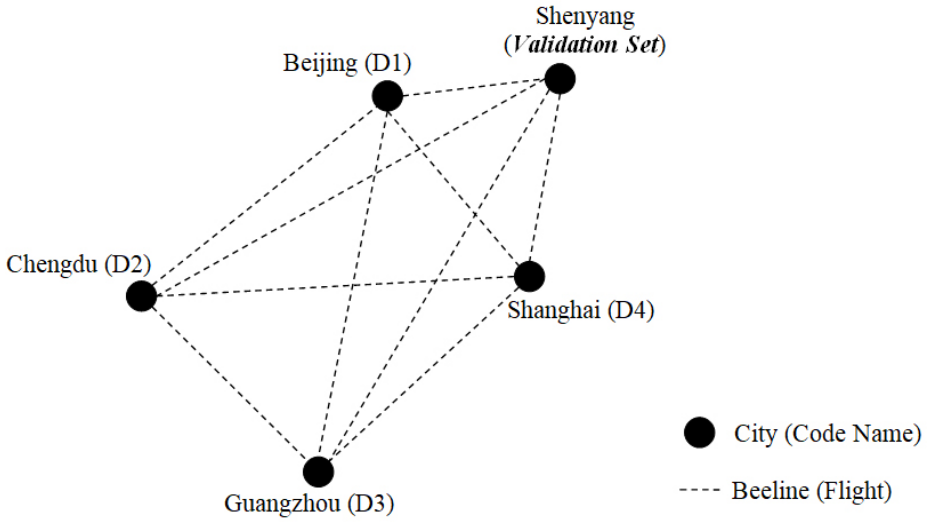

Figure 5.2: Sociogram of the 5-clique Social Network of cities

5.3.2 Training/Learning Algorithms

Feed Forward Pass

This is a chain reaction mechanism that computes the overall output of a neural network (NN) with respect to the net input and activation output generated at each constituent neuron (or node) in the network. Formally, it is computed in a node-per-layer manner as:

$$\begin{aligned} x_i^{l+k} &= w_{ij}^l x_j^l + w_{ij+1}^l x_{j+1}^l + ... + w_{ij+n-1}^l x_{j+n-1}^l + b_i^l \\ \therefore\ x_i^{l+k} &= \sum_{j=1}^{n} [w_{ij}^l x_j^l] + b_i^l \\ y_i^{l+k} &= f(x_i^{l+k}) \end{aligned} \tag{5.12}$$

With respect to the current layer, l; k is an integer increment (usually in steps of 1's) used to reference subsequent layer(s); i is the referenced node number in the subsequent layer $l+k$; j is the source node number in the current layer l; n is the maximum number of nodes in the current layer l; x, y, w, b, and f are placeholders for input(s), output(s), connection weight(s), bias(es), and activation function respectively.

Backpropagation of Errors

This aims at finding the minimum value for the overall cost or loss function (MSE: Mean Squared Error) of a neural network, via derivative of the network MSE, with respect to each connection weight and bias weight in the network. In other words, the slope or derivative of the network error is calculated by propagating the error backwards to the input layer through each connection weight and bias weight existing between pairs of neurons/nodes (inclusive of the bias node) in the network. Mathematically, it is computed as a 3-tuple

$E = (E_1, E_2, E_3)$, and Fig. 5.3 illustrates this concept of backward propagation of the network error, viz:

$$\frac{\delta E}{\delta w_{ij}^{l}} = (\frac{\delta E}{\delta y_i^{l+k}} . \frac{\delta y_i^{l+k}}{\delta x_i^{l+k}} . \frac{\delta x_i^{l+k}}{\delta w_{ij}^{l}}) \tag{5.13}$$

Gradient Descent Algorithm

Fundamentally, this leverages on "Backpropagation" technique to recursively update each neuron's associated weight(w_i) and bias weight(b_i) at each step-down point(p_i) in the loss function, $f(x)$ until a desirable minimum(m) is reached. Fig. 5.4 exemplifies the concept or process of gradient descent algorithm. According to Hinton [26], he proposed that the application of the gradient/slope, $\frac{\delta E}{\delta w_{ij}^{l}}$, of the error function with respect to each connection weight, w_{ij}^{l}, should be multiplied by a learning rate, α. The learning rate determines the speed and pace at which connection weights, w_{ij}^{l}, and bias weights, b_i^{l} change in a neural network training. On one hand, if the learning rate is too high, the gradient descent algorithm may descend steeply and miss the global minimum(m). On the other hand, if the learning rate is too low, the gradient descent algorithm may take much longer time to converge at the global minimum(m). In this regard, we need to strike an effective and efficient balance for the network learning rate (practically: $\alpha < 1$).

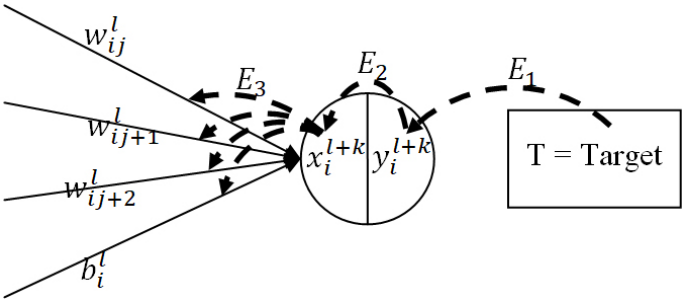

Figure 5.3: Backward Propagation of Network Error to each connection Weight (and bias Weight) per Neuron

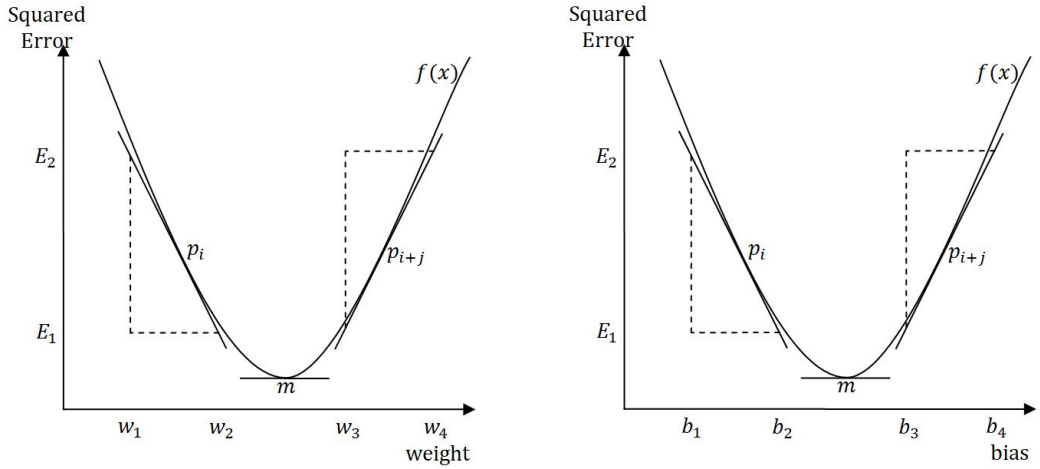

Figure 5.4: Loss (MSE) function Gradient Descent using Backpropagation with respect to each Weight and Bias

5.3.3 Experimental Procedure

Proposed Adjustment Bias

The adjustment bias, a_b, is a 1 by n feature-space dimensional vector which is applied at the output layer to incline respective predictions toward the target node. Let x defined by a feature space, $f_n^x = (f_1^x, f_2^x, ..., f_n^x)$ denote the primary node/actor with very sufficient training samples (row-wise), and is used for model training. Similarly, let y defined by a feature space, $f_n^y = (f_1^y, f_2^y, ..., f_n^y)$ denote the target node/actor with very scanty (insufficient) samples unsuitable for lone/independent model training.

Firstly, considering n, p, and q to be the individual maximum width of the feature space, dataset sample size for the primary-training node, and dataset sample size for the target node; we compute the vector mean $[\bar{f_n^x}]$ and $[\bar{f_n^y}]$ of the primary-training node and target node respectively, viz:

$$\begin{aligned}[\bar{f_n^x}] &= \frac{1}{p}\sum_{i=1}^{p} f_{pn}^x = \frac{1}{p}(\sum_{i=1}^{p} f_{i1}^x, \sum_{i=1}^{p} f_{i2}^x, ..., \sum_{i=1}^{p} f_{in}^x) \\ [\bar{f_n^y}] &= \frac{1}{q}\sum_{i=1}^{q} f_{qn}^y = \frac{1}{q}(\sum_{i=1}^{q} f_{i1}^y, \sum_{i=1}^{q} f_{i2}^y, ..., \sum_{i=1}^{q} f_{in}^y)\end{aligned} \tag{5.14}$$

Secondly, we compute the difference between the vector mean of the primary-training node and target node, viz:

$$\bar{a_b} = [\bar{f_n^x}] - [\bar{f_n^y}] \tag{5.15}$$

Thirdly, we transpose the resultant vector to a 1 by n row vector, viz:

$$a_b = (\bar{a_b})^T = ([\bar{f_n^x}] - [\bar{f_n^y}])^T \tag{5.16}$$

Lastly, we apply the adjustment bias, a_b, to every j^{th} prediction, p, generated by our model (with respect to the primary-training node, x) to give us a corresponding prediction value for the target node, y, viz:

$$p_j^y = p_j^x - a_b \tag{5.17}$$

Experiment Pseudocode

Basic Experimentation Procedure

01: Preprocess datasets with respect to missing data points;
02: Load and transform data to 3-Dimensional shape;
while (hyperparameters != optimal) {
    03: Normalize (or downscale) dataset;
    04: Define and configure neural network model (RNN, MLP, etc);
    05: Compute adjustment bias (a_b) vector;
    06: Train and fit the model using the training dataset;
    07: Make predictions;
    08: Apply the adjustment bias (a_b);
    09: Evaluate model's performance over validation dataset;
    10: Tune hyperparameters where necessary;
}

5.4 Experimentation Results and Discussion

We performed a controlled experiment cutting across the standard methods primarily used in event prediction based on the hyperparameters shown in Table 5.2. The accuracy and efficiency of the models across the board is enumerated in Table 5.5 in ascending order of validation-set RMSE and training time. Each model's performance, grouped based on domain category, is contained in Tables 5.3, 5.4, and 5.6. Also, Fig. 5.5 graphically shows the learning progress curves of the MLP network model (100-neuron width and 4-layer depth) when implemented over datasets D1, D2, D3, and D4.

Table 5.2: Fixed Experimentation Hyperparameters

Training Set: 70%	Test Set: 30%
Batch Size: 1024	Repeats/Runs: 30
Epochs: $6.0 * 10^2$	Optimizer: *Adam*
Dropout: $4.0 * 10^{-1}$	Activation: *ReLU*
Input Timestep: 1	Learning Decay: 0.00
Output Timestep: 2	L1-regularization: 0.01
Network Size: $4 * 10^2$	L2-regularization: 0.00
Samples per Node: $52,584$	Bias Initialization: 1.00

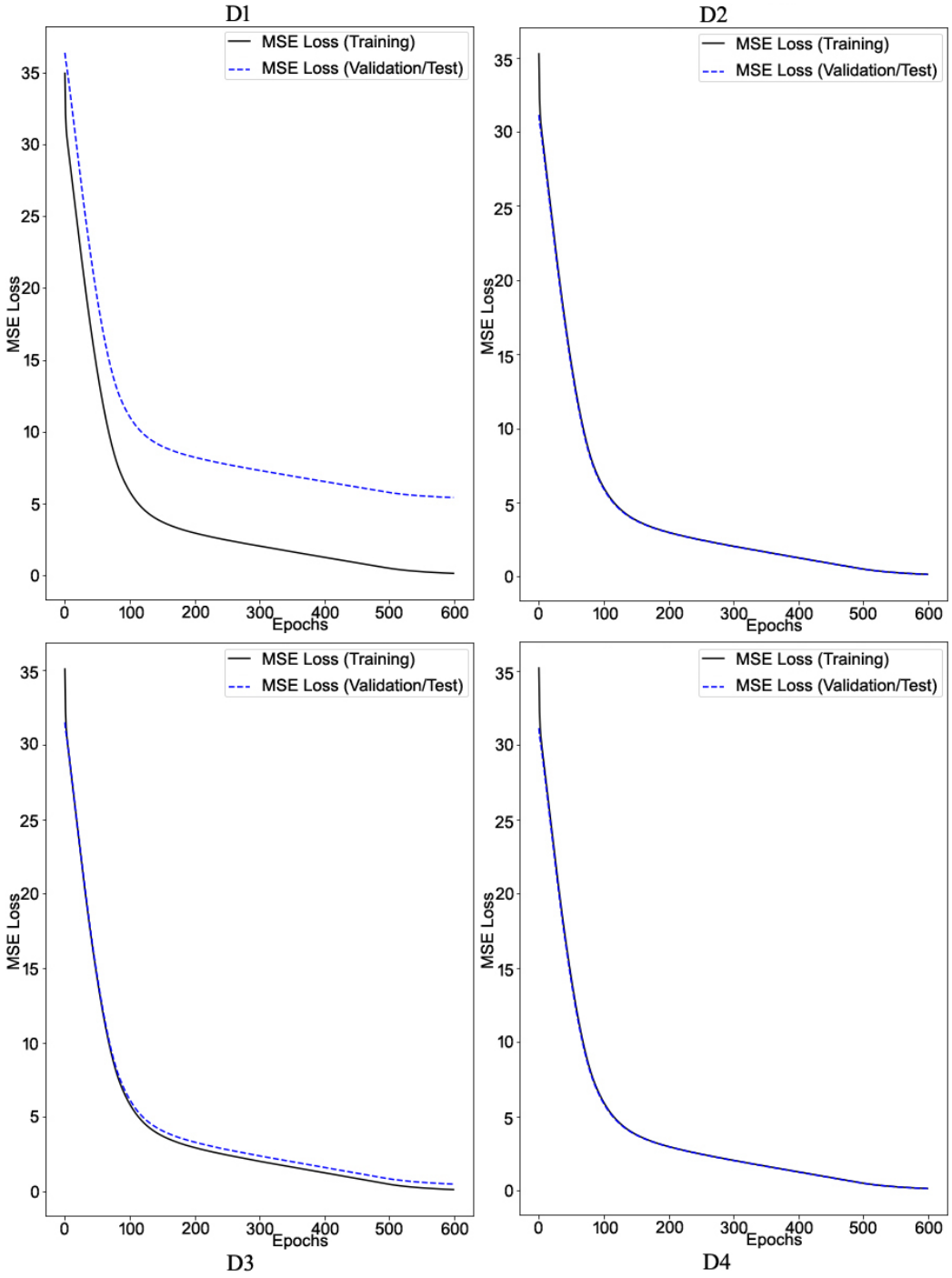

Figure 5.5: Learning curves for the Training (thick black lines) and Validation (dotted blue lines) sets with respect to MLP architecture. Horizontal ($x - axis$) represents the epochs count; and the vertical ($y - axis$) is the MSE loss value.

Table 5.3: Machine Learning - Experimentation Results

Machine Learning Approach				
Dataset	Models	Training		Validation
		Time(s)	RMSE	RMSE
D1	Rand. Frst.	86.35	7.96	17.73
	Deci. Tree	61.52	6.33	20.57
	KNN	62.47	12.37	17.73
D2	Rand. Frst.	92.43	11.95	14.33
	Deci. Tree	60.79	10.20	20.51
	KNN	61.61	18.61	15.55
D3	Rand. Frst.	100.23	15.65	18.13
	Deci. Tree	60.78	1.73	42.24
	KNN	61.76	35.21	22.01
D4	Rand. Frst.	87.40	9.69	13.24
	Deci. Tree	60.83	8.59	18.96
	KNN	61.86	13.38	13.03

Table 5.4: Statistical Methods - Experimentation Results

Statistical Approach				
Dataset	Models	Training		Validation
		Time(s)	RMSE	RMSE
D1	VAR	2310.83	13.85	16.98
	VMA	22893.50	13.94	16.98
	VARMA	14208.01	13.85	16.98
D2	VAR	2324.88	18.84	12.95
	VMA	23115.12	19.47	12.95
	VARMA	13673.28	18.72	12.95
D3	VAR	2316.38	38.78	7.67
	VMA	24176.84	38.82	7.67
	VARMA	7447.98	38.78	7.67
D4	VAR	2326.00	13.96	11.75
	VMA	22925.22	14.23	11.75
	VARMA	13668.11	13.93	11.75

Table 5.5: Average Performance of Models across Datasets

S/N	Category	Models	Validation RMSE	Training Time(s)	Training RMSE
1:	Deep Learning	MLP	9.65	87.64	17.85
2:		B-RNN	9.65	149.28	19.07
3:		GRU	9.65	259.00	20.71
4:		LSTM	9.65	282.17	18.49
5:	Statistical Methods	VAR	12.34	2319.52	21.36
6:		VARMA	12.34	12249.35	21.32
7:		VMA	12.34	23277.67	21.62
8:	Machine Learning	Rand. Frst.	15.86	91.60	11.31
9:		KNN	17.08	61.93	19.89
10:		Deci. Tree	25.57	60.98	6.71

As regards training-time efficiency as the data size increases: ML and DL models perform relatively better while Statistical methods appear to be the most expensive to implement. Also, considering the models' accuracy on the validation set: DL and Statistical methods have proven to have lower RMSE while ML methods demonstrated relatively higher RMSE. To this effect, there exist a trade-off between DL and ML models with respect to training time and accuracy.

Furthermore, and with respect to DL models, we used a mini-batch size of 1024 for training because our data is of time-series type; and we want to ensure that sufficient training patterns are extracted by the model before its network weights are updated. Also, our dropout regularization was implemented in the hidden layers of the network in an alternating pattern such that every other layer sandwiches a dropout regularizer.

A relative comparison of the different network structures (narrow-and-deep, wide-and-

Table 5.6: Deep Learning - Experimentation Results

Deep Learning Approach		Learn Rate: $1.0 * 10^{-3}$ Width: 4 Deep: 100			Learn Rate: $1.0 * 10^{-3}$ Width: 100 Deep: 4			Learn Rate: $1.0 * 10^{-3}$ Width: 20 Deep: 20		
Data	Models	Training Time(s)	Training RMSE	Validtn RMSE	Training Time(s)	Training RMSE	Validtn RMSE	Training Time(s)	Training RMSE	Validtn RMSE
D1	GRU	1335.20	15.07	14.28	262.30	17.29	14.28	452.27	19.64	14.28
	LSTM	1389.36	15.34	14.28	283.17	13.97	14.28	462.03	14.25	14.28
	B-RNN	799.20	14.31	14.28	149.88	14.02	14.28	260.28	14.05	14.28
	MLP	**306.13**	**13.69**	**14.28**	**87.02**	**13.85**	**14.28**	**136.05**	**14.09**	**14.28**
D2	GRU	1349.92	21.26	10.25	257.97	21.26	10.25	455.03	21.32	10.25
	LSTM	1399.39	21.25	10.25	282.71	21.24	10.25	464.79	21.24	10.25
	B-RNN	794.37	21.24	10.25	149.15	21.24	10.25	260.30	21.24	10.25
	MLP	**315.67**	**21.24**	**10.25**	**87.58**	**21.24**	**10.25**	**138.04**	**21.23**	**10.25**
D3	GRU	1348.46	23.23	5.01	257.83	23.26	5.01	453.17	23.16	5.01
	LSTM	1403.56	23.21	5.01	281.63	23.19	5.01	464.54	23.20	5.01
	B-RNN	800.43	23.16	5.01	149.87	23.15	5.01	261.25	23.16	5.01
	MLP	**310.30**	**23.19**	**5.01**	**88.06**	**23.16**	**5.01**	**136.27**	**23.15**	**5.01**
D4	GRU	1338.24	19.83	9.05	257.90	21.03	9.05	454.64	22.47	9.05
	LSTM	1404.05	19.86	9.05	281.17	15.54	9.05	462.70	19.82	9.05
	B-RNN	799.87	17.77	9.05	148.21	17.87	9.05	262.94	16.21	9.05
	MLP	**302.52**	**13.97**	**9.05**	**87.88**	**13.15**	**9.05**	**134.74**	**12.14**	**9.05**

shallow, and wide-and-deep respectively) implemented against our DL models in Table 5.6 shows that with sufficiently large amount of data available for training, a relatively wide-and-shallow network structure will produce approximately same performance as other structures with even lower training time. Although our wide-and-shallow network structure implementation is 4-layer-deep; according to reviewed literature, there exists the problem of poor generalization when using very wide-and-shallow (usually 1-layer-deep) neural structures. In a bid to overcome this problem, sequential layers of neurons are usually stacked adjacent to each other to form a deep network structure [27]. The use of deep-layer structures cannot be downplayed because they are capable of training themselves to recognize and learn hidden features of a representation over consecutive levels of abstraction [28].

However, as the network structure goes deeper, more levels of abstraction to learn are generated; and this results in increased training time. Concurrently, as the network structure goes wider, more hidden parameters are injected into the network for learning, and this amplifies the training time as well. In this regard, there arises the need to strike a balance between the width and depth of the network structure; such that we obtain an efficient/effective model without increasing the chances of underfitting or overfitting in the network.

Apparently, based on our experimentation results, MLPs are still very suitable for event prediction problems as they produce surpassing outcomes with even lower training lags as shown by its prediction accuracy and training time respectively. Therefore, Table 5.7 shows the performance results of an optimized MLP architecture obtained after a grid search on some network hyperparameters. In addition, Fig. 5.6 shows the learning progress curves of the optimized MLP architectural model (15-neuron width and 10-layer depth) evaluated upon datasets D1, D2, D3, and D4. This optimized MLP model can compete favorably with ML models in the aspect of training time.

Table 5.7: Optimized MLP Model - Experiment Results

Optimized Deep Neural Network (MLP) Model				
	Training		Validation	Learning Rate: $3*10^{-3}$
	Time(s)	RMSE	RMSE	Network Size: $1.5*10^{2}$
D1	49.92	13.83	14.28	Dropout: $2.0*10^{-1}$
D2	50.40	21.23	10.25	Epochs: $3.5*10^{2}$
D3	50.71	23.08	5.00	Width: 15
D4	50.09	11.76	9.05	Depth: 10

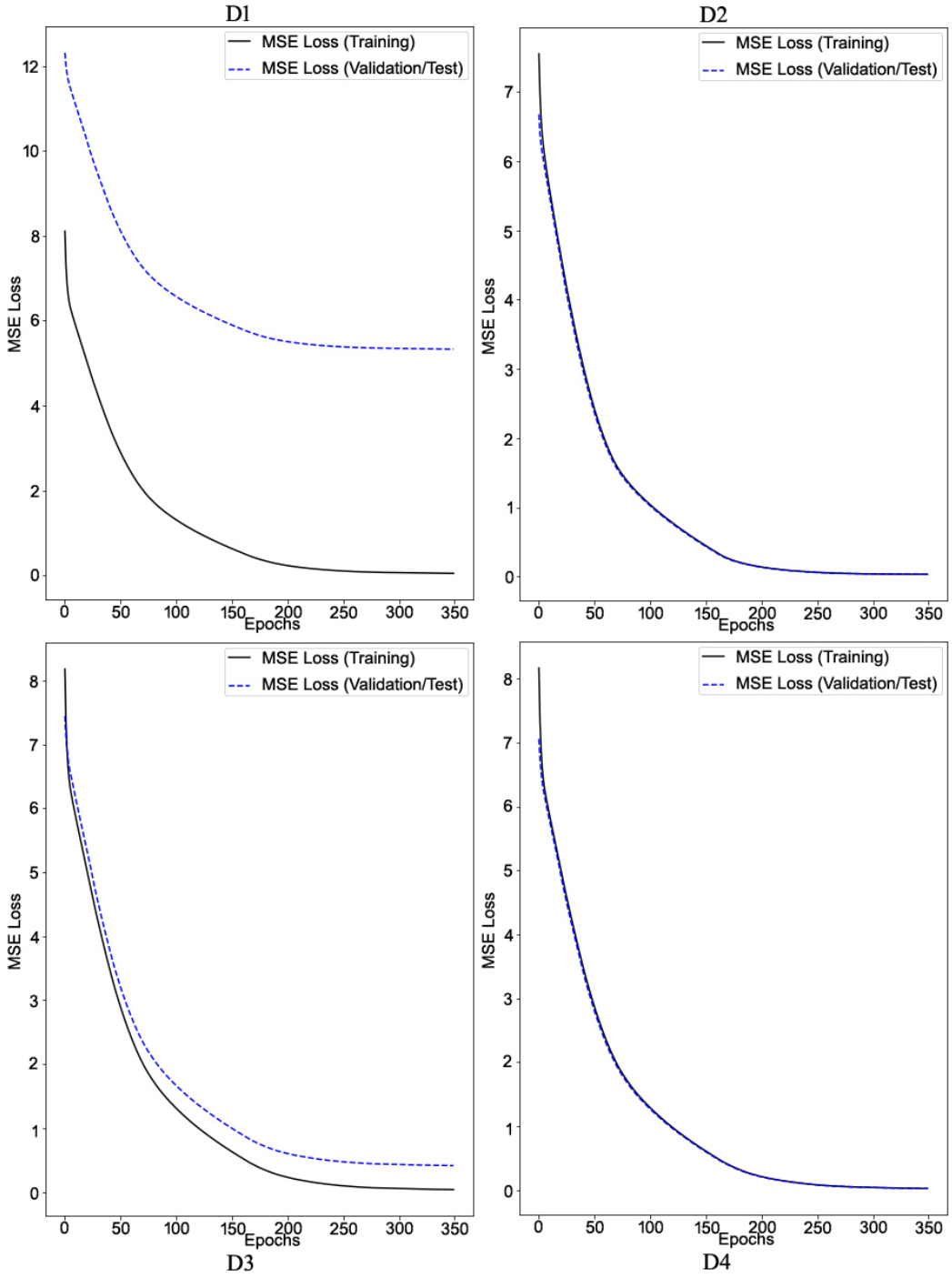

Figure 5.6: Learning curves for the Training (thick black lines) and Validation (dotted blue lines) sets with respect to the Optimized MLP architecture. $x - axis$ represents the epochs count; and the $y - axis$ is the MSE loss value.

5.5 Conclusion and Future Work

We contrived an 'adjustment-bias' function which we applied to our implementation to amplify its prediction accuracy and precision about the target actor or node. Also, based on findings from our experiments and analysis, we propound that: provided other hyperparameters (training samples, network size, etc) remain constant, the shape of both RNN and MLP networks does not significantly affect its prediction accuracy but its training time.

Additionally, we have demonstrated that RNNs are optimal when modelled as wide-and-shallow networks as opposed to narrow-and-deep network architectures. However, RNNs made up of LSTM and/or GRU units are not always the ideal model for every regression and/or forecasting problem because of the relatively complex structure of these constituent neurons with reference to their 'gating' mechanisms. In this regard, networks designed with simpler units like MLPs can compete favorably with RNNs, and even perform better depending on the nature of the task.

Obviously, there are scenarios where the use of some DL, ML, or Statistical models can be quite unbefitting with regards to computational and resource effectiveness as well as efficiency. Hence, proper analysis of a given problem should be conducted before choosing a model/methodology to use.

Lastly, we intend to expand our scope to include more datasets from other application domains; as well as benchmark against more methods/models. We also plan to test the effectiveness and efficiency of model hybridization with respect to spatial event prediction

in social network structures.

Acknowledgment

This research was supported by International Business Machines (IBM) via the provision of computational resources necessary to carry-out our experiments. Research was conducted on a high performance IBM Power System S822LC Linux Server.

Bibliography

[1] J. Scott, Ed., *Social Network Analysis.* Newbury Park, California: SAGE Publications Ltd., 2017.

[2] H. V. Maaren, A. Biere, and T. Walsh, "Handbook of satisfiability," in *Handbook of Satisfiability.* Amsterdam, The Netherlands: IOS Press, 2009.

[3] S. A. Cook, "The complexity of theorem-proving procedures," *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, p. 151–158, 1971.

[4] G. E. Hinton, "Learning multiple layers of representation." *TRENDS in Cognitive Sciences*, vol. 11, no. 10, pp. 428–433, 2007.

[5] B. Cortez, B. Carrera, Y. Kim, and J. Jung, "An architecture for emergency event prediction using lstm recurrent neural networks," *Expert Systems with Applications*, vol. 97, pp. 315–324, 2018.

[6] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, 2018.

[7] C. Yu, M. Bhatnagar, R. Hogen, D. Mao, A. Farzindar, and K. Dhanireddy, "Anemic status prediction using multilayer perceptron neural network model," in *EPiC Series in Computing*, 2017.

[8] I. R. Widiasari, L. Nugroho, and Widyawan, "Deep learning multilayer perceptron (mlp) for flood prediction model using wireless sensor network based hydrology time series data mining," *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, pp. 1–5, 2017.

[9] G. Cafri, L. Li, E. Paxton, and J. Fan, "Predicting risk for adverse health events using random forest," *Journal of Applied Statistics*, vol. 45, pp. 2279 – 2294, 2018.

[10] M. Marjanović, M. Krautblatter, B. Abolmasov, U. Durić, C. Sandić, and V. Nikolić, "The rainfall-induced landsliding in western serbia: A temporal prediction approach using decision tree technique," *Engineering Geology*, vol. 232, pp. 147–159, 2018.

[11] W. Alajali, W. Zhou, S. Wen, and Y. Wang, "Intersection traffic prediction using decision tree models," *Symmetry*, vol. 10, p. 386, 2018.

[12] M. Huang, R. Lin, S. Huang, and T. Xing, "A novel approach for precipitation forecast via improved k-nearest neighbor algorithm," *Adv. Eng. Informatics*, vol. 33, pp. 89–95, 2017.

[13] M. Gustin, R. Mcleod, and K. Lomas, "Forecasting indoor temperatures during heatwaves using time series models," *Building and Environment*, vol. 143, pp. 727–739, 2018.

[14] O. Baquero, L. M. R. Santana, and F. Chiaravalloti-Neto, "Dengue forecasting in são paulo city with generalized additive models, artificial neural networks and seasonal autoregressive integrated moving average models," *PLoS ONE*, vol. 13, 2018.

[15] B. Pfaff, "Var, svar and svec models: Implementation within r package vars," *Journal of Statistical Software*, vol. 27, pp. 1–32, 2008.

[16] H. Ltkepohl, "New introduction to multiple time series analysis," 2007.

[17] L. Torgo, "Inductive learning of tree-based regression models." Porto, Portugal: Dept. of Computer Science, University of Porto, 1999.

[18] R. A. Berk, "An introduction to statistical learning from a regression perspective." Springer Science+Business Media, 2010.

[19] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: Data mining, inference, and prediction, 2nd edition," in *Springer Series in Statistics*, 2009.

[20] T. Laloë, "A k-nearest neighbor approach for functional regression," *Statistics & Probability Letters*, vol. 78, pp. 1189–1193, 2008.

[21] I. G. Goodfellow, Y. Bengio, and A. C. Courville, Eds., *Deep Learning*. Cambridge, MA: MIT Press, 2017.

[22] Y. LeCun, "Deep learning & convolutional networks," *2015 IEEE Hot Chips 27 Symposium (HCS)*, pp. 1–95, 2015.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[24] K. Cho, B. V. Merrienboer, Çaglar Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," *ArXiv*, vol. abs/1406.1078, 2014.

[25] X. Liang, S. Li, S. Zhang, H. Huang, and S. Chen, "Pm2.5 data reliability, consistency and air quality assessment in five chinese cities†," *Journal of Geophysical Research*, vol. 121, pp. 10 220–10 236, 2016.

[26] G. E. Hinton and R. Williams, "Performance analysis of neural network classifier for the different number of hidden units," 2014.

[27] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, pp. 1–113, 2009.

[28] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. rahman Mohamed, N. Jaitly, A. W. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks

for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, 2012.

Chapter 6

Summary and Future Directions

Taking into consideration the outstanding and superb performance of Deep Learning with respect to several open research problems in various fields of Computer Science; we were prompted and motivated to study and employ a range of DL-based architectures toward resolving selected open (research) problems in the domain of Social Network Analysis. Therefore, from a Deep Learning perspective, this dissertation essentially focused on resolving the following SNA problems, viz: Breakup Prediction (in Chapter 2), Trend/Pattern Analysis (in Chapter 3), Link Prediction (in Chapters 2 and 4), Node Classification (in Chapter 4), and Event-based Analysis (in Chapter 5).

In Chapter 2, we jointly attempted to resolve the problems of Breakup Prediction and Link Prediction in social graphs via a unique, bifunctional, and hybrid framework: ClasReg. Primarily, ClasReg is a 5-layer-architecture framework comprising the following layers, viz: a preprocessing layer, a Representation Learning or Feature Learning layer, a classification layer, a regression layer, and a heuristic engine. ClasReg is based on an n-ary input operation which renders ternary or triadic outputs composed of positives, negatives, and intersections

(*positives* that need to transmute to *negatives*). At this juncture, it is relevant to point out that breakup prediction problem differs from the link prediction problem in SNA. On one hand, breakups prediction in SN structures focuses on identifying positive ties that ought to be broken (transmutation to -ve state) in a bid to avoid unforeseeable threats. On the other hand, link prediction aims at predicting newer relationships that should be established within the given SN structure in the near future. Moreover, we have evaluated ClasReg against state-of-the-art baselines for link prediction and classic models for evaluating the strength of ties in SN structures. The comparative benchmark performance of ClasReg has been detailed herein with respect to classic objective functions used for classification and regression tasks in ML.

Furthermore, in Chapter 4, we introduced a hybrid DL-based model: RLVECO. In this regard, our research contribution and publication was recognized for the Best Paper Award at the 32^{nd} International Conference on Scientific and Statistical Database Management (SSDBM 2020). Thus, in this chapter, we have jointly attempted to resolve the problems of Link Prediction and Node Classification in Social Network structures. To this end, we have proposed RLVECO, which is a distinct hybrid model that hybridizes the strengths of Knowledge-Graph Embeddings and Convolution Operations in extracting and learning meaningful features from social graphs via Representation Learning. RLVECO utilizes an edge sampling approach for exploiting features of a social graph via learning the context of each actor with respect to its neighboring actors. In addition, we evaluated the performance

of RLVECO on standard real-world social networks datasets. Thus, several comparative analyses between RLVECO and state-of-the-art approaches for link prediction and node classification have been reported herein in this dissertation.

At present, the world yearns for more effective and efficient mitigation strategies to control as well as prevent the spread of the new SARS-CoV-2. To this end, in Chapter 3, we have proposed a unique Transfer Learning framework aimed at providing insights into the COVID-19 pandemic, and mitigating the impacts of the pandemic by means of preemptive actions based on effective forecasting/predictions. In this regard, we have employed Social Network Analysis toward resolving the current public health and epidemiological problem of the COVID-19. On one hand, monitoring the effect of SARS-CoV-2 via studies, analyses, and predictions enable us understand the nature of SARS-CoV-2 in relation to the factors that promote its growth and spread. Hence, our research work fosters widespread awareness such that the populace can become more proactive and cautious in a bid to mitigate the spread of SARS-CoV-2 infections. On the other hand, understanding and forecasting the demand for Personal Protective Equipment aids in protecting our healthcare workers, whom come in contact with positive COVID-19 and hospitalized cases, in Community Health Centres. As a result of the new and unusual nature of SARS-CoV-2, relatively few literature and research exist in this regard. Furthermore, in this chapter, the results from our experiments indicate that government actions and human factors are the most significant determinants that influence the spread of SARS-CoV-2. Detailed evaluation results of our research and

experiments have been documented in Chapter 3 herein.

In Chapter 5, we have attempted to resolve yet another open problem (Event-based Prediction) in Social Network Analysis. Our proposed approach toward resolving this open (research) problem is based on a DL architecture assembled by means of deep layers of stacked MLPs. Bascially, our approach in this chapter involves applying the intrinsic patterns and knowledge acquired from the study of some selected social actors in a given social graph. Thereafter, we attempt to predict future events about these selected social actors; and in turn, monitor how these selected social actors influence other correlated social actors with regard to their existent social ties within the SN structure. In other words, we attempt to predict the occurrence of an event in actor-y (which may have poor or insufficient data for independent training) based on the happenings in actor-x (primary data used for training). In this chapter, our experiments were carried out using real-world meteorological datasets of five (5) major cities in China which are geographically separated by space and time. Hence, the detaild and comparative benchmark results of our experiments have been documented in Chapter 5 of this dissertation.

In conclusion, the proposed approaches in this dissertation can be extended in the near future to accommodate several graph variants such as: Knowledge Graphs, heterogeneous and multi-layer graphs, etc. Additionally, we can expand the approaches proposed herein to accommodate other open problems in SNA such as, viz: Community Detection, Collaboration and Knowledge Management (e.g. Team Formation), Sentiment Analysis, etc.

Vita Auctoris

NAME:	Bonaventure Chidube Molokwu
PLACE OF BIRTH:	Benin City, Nigeria
EDUCATION:	Ph.D. in Computer Science, University of Windsor, Ontario, Canada, 2021.
	M.Sc. (Hons.) in Computer Science, University of Nigeria, Nsukka, Nigeria, 2015.
	B.Sc. (Hons.) in Computer Science, University of Benin, Benin City, Nigeria, 2011.