DTU Compute

Department of Applied Mathematics and Computer Science

Detecting Online Predatory Behaviour Using Attributed Graph Neural Networks

Bachelor Thesis

Detecting Online Predatory Behaviour Using Attributed Graph Neural Networks

Bachelor Thesis
January, 2023

By:
Alma Fazlagic & Natasha Graae Norsker

Supervised by:
Bjørn Sand Jensen, DTU
Patrick Bours, NTNU & Aiba

Cover photo: Vibeke Hempler, 2012

Approval

This bachelorproject has been prepared over eighteen weeks at the Department of Applied Mathematics and Computer Science of the Technical University of Denmark, DTU, and in cooperation with the company Aiba, specialising in Author Input Behaviour Analysis. This is done as partial fulfilment for the degree *Bachelor of Science in Artificial Intelligence and Data Science*.

It is assumed that the reader has a basic knowledge in the areas of statistics, machine learning and linear algebra.

Alma Fazlagic, s194271	Natasha Graae Norsker, s194270
.....	.....
Signature	*Signature*
.....	.....
Date	*Date*

Abstract

Sexual predators pose a serious threat to children, who often use online gaming platforms. These predators use games as a means of luring and grooming children for the purpose of sexual exploitation. In order ensure the safety of children while playing online games, it is crucial to develop effective methods for detecting and deterring sexual predators.

By examining the social network patterns of users in this online environment, it is possible to identify potentially predatory behaviour. In this thesis, we investigate the effectiveness of unsupervised Graph Neural Networks for detecting sexual predators in MovieStarPlanet, without relying on linguistic analysis. We compare a baseline model that does not use graph structure explicitly with four models that utilise the structure in order to rank the users based on how anomalous their behaviour is.

Our results showed that there was not a significant difference in performance between the baseline model and the other models, and we were unable to conclude that models that utilise the structure of the graphs consistently outperform the model that does not. However, we found that anomalous users exhibit distinct behaviour in terms of their social network behaviour, and the proposed graph features captured a significant amount of useful information for detecting sexually predatory users and other abnormal behaviours. Overall, our research suggests that a graph-based approach can be useful for detecting online sexual predators, but may be enhanced by incorporating linguistic analysis.

Acknowledgements

We would like to express gratitude to our supervisors, Bjørn Sand Jensen and Patrick Bours, for their invaluable guidance throughout this thesis. We are also grateful to the rest of Aiba for the opportunity of conducting this research.

Contents

1 Introduction

In 2020, the National Center for Missing and Exploited Children (NCMEC) reported receiving 21.7 million reports of child sexual exploitation as well as a year-over-year growth in cases by 35% from 2020 to 2021. In 2021 solely they received 16.9 million unique images and 5.1 million videos regarding potential sexual abuse online [NCM21].

Researchers have found that to due the nature of online communication, strangers form intimate relationships much faster than in real life [Bla+15]. Additionally, the possibility of anonymising oneself online increases children's susceptibility to sexual exploitation and grooming. In the worst case, this leads to offline meetings that can result in physical abuse or sexual trafficking. However, even children who have been groomed online without experiencing physical abuse oftentimes suffer numerous negative effects such as anxiety, stress, depression, and substance abuse. Children may suffer long-lasting emotional damage caused by the non-contact sexual abuse [San17].

Predators looking to groom children online often visit social media websites that are popular with young people and will pretend to be of similar age [Kra22]. MovieStarPlanet (MSP) is an online gaming and social network specifically aimed for children aged eight to fifteen [Mov22a]. According to the American Federal Bureau of Investigation (F.B.I.), over 50 percent of the victims of online sexual exploitation are between the ages of twelve and fifteen [Kra22], making MSP's target audience vulnerable to online sexual exploitation.

The primary motivation for this thesis is to detect grooming and other sexual behaviour on MSP. The thesis focuses on specific behaviours such as participating in sexually charged conversations and asking for Personal Identifiable Information (PII). Oftentimes groomers will attempt to move the conversation away from MSP on other platforms with less restrictions and where photo-sharing is possible. Our primary goal is to detect groomers and predators, however, we will refer to users that break MSP's guidelines on harmful behaviour as being anomalous or displaying predatory behaviour.

This thesis aims to determine if anomalous users exhibit distinct behaviours in their social networks in addition to the content of their conversations. This is done by investigating if models that solely use graph-based information, either encoded through attributes or in the graph structure itself, can distinguish between anomalous and non-anomalous users. We further compare graph-based approaches that explicitly uses the structure of the graph to a baseline model that only uses higher-level graph based features. Therefore the modelling approach will strictly focus on graph behavioural patterns.

The graph based approach is motivated by generalising easier across different geographical areas since it does not depend on linguistic features specific to different languages. Furthermore it is

harder for anomalous users to disguise their network behaviour than obstructing specific words when they are writing.

The US MSP servers alone saw more than 250,000 users in the first half of 2022, and the activity in Europe was significantly higher. Obtaining ground truth anomalies in this setting also has many limitations due to the massive volume of traffic, making it infeasible to investigate all users and receive labels for all of them in advance. Therefore, the goal of this approach is to examine how well an unsupervised graph-based approach can rank users based on anomalous behaviour, allowing MSP moderators to focus their attention on users with a higher probability of being anomalous.

The project is written in collaboration with Aiba, a company specialising in Author Input Behaviour Analysis. Aiba develops solutions, that ensure the safety of children by detection of cyber-grooming behaviour in online chat-rooms. They focus their research into early detection and intervention using both linguistic, key-stroke and graph-based approaches.

1.1 Problem Definition

Sexual Predator Detection using Graph Neural Networks on Attributed and Directed Graphs:
Our goal is to detect anomalous behaviour in an attributed and directed network, $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where the nodes represent users chatting on the children's gaming platform, MovieStarPlanet [Mov22b], and the edges connect nodes that have exchanged messages. Specifically, we want to identify nodes that engage in sexually charged or grooming conversations with other users or request Personal Identifiable Information (PII).

Our primary research question is:

> **RQ1:** *Can a purely graph-based approach detect sexual predators in a children's gaming platform?*

To conduct our research in a structured manner, we define the following sub-questions:

> **SQ1:** *Does including the structure of the graph explicitly yield a better performance than only using higher-level attributes derived from the graph-structure?*

> **SQ2:** *Does the sexually predatory users display a different behaviour in terms of their social network graph?*

1.2 Notation

Listed below are the notations used in this thesis.

Notation	Description
$\mathcal{G}$	A graph
$\mathcal{V}$	The set of nodes in a graph
u	A node $u \in \mathcal{V}$
$\mathcal{E}$	The set of edges in a graph
$e_{u,v}$	An edge connected from node u to node v
$\mathcal{G}_u$	Ego graph of node u
$\mathbf{A} \in \mathbb{R}^{n \times n}$	The adjacency matrix of $\mathcal{G}$
$\mathbf{A}^T$	The transpose of matrix of $\mathbf{A}$
$\mathbf{D}$	The degree matrix $\mathbf{A}$
$deg(u)$	The degree of node u
$deg^-(u)$	The in-degree of node u
$deg^+(u)$	The out-degree of node u
N	The number of nodes in $\mathcal{G}$
$\mathcal{N}(u)$	The neighbours of node u
$\mathbf{X} \in \mathbb{R}^{N \times D}$	The attribute matrix of $\mathcal{G}$
$\mathbf{x}_u \in \mathbb{R}^D$	The attribute vector of node u
D	The number of attributes per node in $\mathcal{G}$
$\|.\|$	The cardinality of a set
$\|\|.\|\|_2$	The ℓ_2 norm of a vector
$\|\|.\|\|_F$	The Frobenius norm of a matrix
$\mathbf{z}_u$ (or $\mathbf{h}_u$)	Latent embedding of node u
$\mathbf{m}_{\mathcal{N}(u)}$	Message aggregated from $\mathcal{N}(h)$
$\tilde{\mathbf{A}}$	Adjencency matrix with added self loops
$\bar{\mathbf{A}}$	Symmetrically normalised $\tilde{\mathbf{A}}$, $\bar{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$
$\hat{\mathbf{X}}$	Reconstruction of attribute matrix $\mathbf{X}$
$\hat{\mathbf{x}}_u$	Reconstruction of attribute vector $\mathbf{x}_u$
$\hat{\mathbf{A}}$	Reconstruction of adjacency matrix $\mathbf{A}$

Table 1.1: *Table of notations*

2 Theory

This section will first focus on theory about sexual grooming of children and the psychology and behaviour of groomers are presented. Then, the concept of graphs and graph-based deep learning tools, such as the Graph Neural Network framework, are introduced.

2.1 Grooming Behaviour Patterns

Sexual grooming refers to the process in which an offender forms a relationship with a child or young adult with the purpose of sexual abuse and exploitation. Cravel et al. [CBG06] propose the following definition of sexual grooming that highlights how this relationship is manifested by the offender: *"A process by which a person prepares a child, significant adults and the environment for the abuse of this child. Specific goals include gaining access to the child, gaining the childs compliance and maintaining the childs secrecy to avoid disclosure. This process serves to strengthen the offenders abusive pattern, as it may be used as a means of justifying or denying their actions."*

2.1.1 Grooming Process: Luring Communication Theory

Literature suggests that the appearance of sexual grooming can look very different. The *stages* or *steps* of the sexual grooming have been widely researched by [Ols+07], [WJ16], [Oco03]. Some researchers propose a process ranging between four and seven steps. American scholar of family communication Loreen Olsen proposed in 2007 *the theory of luring communication* (LCT) [Ols+07], which explores the characterisation of different stages of communication used by sexual predators when luring their victims into a sexual relationship. It is important to note that the scope of LCT is limited to male perpetrators due to limits in data collection.

She proposes the following five stages in the luring communication theory [Ols+07], where the first step for the predator is to gain access to the victim. In the second step the predator will try to gain the trust of the child by giving the them special attention and making them feel loved, which could include gift-giving and taking the child on 'dates'. In the third step the predator will use subtle strategies to prepare the child for sexual grooming by verbally and physically desensitising the child to sexual contact, for instance by rubbing the back or neck of the child. They can also try to manipulate the child into thinking that they permitted and encouraged the contact. After preparing them for sexual contact, the predator will start to isolate the victim, which is the fourth step. The isolation can both be physical and emotional, which includes an attempt to increase the victim's dependency on him. The fifth and final step is approaching the child to see if sexual actions are possible. The outcome of this process is the sexual abuse. The predator will convince the child that keeping their relationship a secret is necessary. The perpetrator will often threaten the victim with the loss of their 'special relationship' and try

to convince the them that no one will believe them if they come forward. This can discourage them from disclosing the abuse.

2.1.2 Online Grooming

Black et al. [Bla+15] argue that due to the nature of online communication, it has become more socially acceptable to form friendly and intimate relationships with strangers much quicker than in real life. However, because of the possibility of anonymising oneself online, the risk of forming these relationships with people with bad intentions is much higher. The offender and target may both feel less inhibited and are less cautious about sharing personal information.

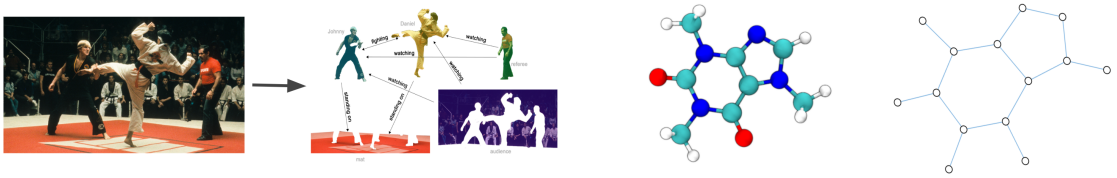
A common type of online exploitation of children is child grooming happening in online chat-forums and social media platforms. This type of grooming typically involves building a trust-relationship with a minor online in order to eventually exchange explicit sexual content or even with a goal of approaching the minor in person. Several social-media platforms and chat-rooms serve as contact points for predators to meet and victimise minors and grooming behaviour online can be more difficult to guard against [CFA14]. Although online grooming differs from the grooming happening in the physical world, several research has shown that grooming stages of online conversations resemble the grooming stages described by LCT [Kon+09], [GKS12].

Kontostathis et al. describe in detail how LCT relates to the online world. In online grooming, a predator can gain access to minors via messaging forums and social networking sites such as Facebook. They will often place themselves in a chat room frequently used by minors and quickly try to exchange personal information such as their location, ages, phone numbers and even pictures of themselves. The predator will also make it a priority to compliment the victim in order to present themselves in a positive light. Furthermore, the predator will use several tactics in order to desensitise the victim to the use of sexual language, for instance by sending sexual images or using sexual slang (e.g using 'cum' instead of 'come') [Kon+09]. When the relationship has progressed, the predator will try to further isolate the victim from his or her family and friends - in terms of physical isolation they will try to ensure that the victim chats without supervision and otherwise convince the victim that he will always be there for them (online and present) when they cannot rely on the people around them. After building the relationship through grooming and isolation, the predator will try to approach the victim by either suggesting a physical meeting to further their sexual connection or request to receive sexual explicit content from the victim in the form of either videos or photographs.

2.2 Graphs as a Data Structure

A vast majority of real-life problems can be modelled through graphs. Graphs consists of a collection of objects as well as a set of interactions between them. We typically understand real world objects in terms of their relation to other things and graphs naturally capture pairwise relations between objects.

The *objects* in a graph are referred to as **nodes** and **vertices** interchangeably in literature. In



(a) *Interactions between objects in a movie scene* **(b)** *Interactions between atoms in a molecule*

Figure 2.1: *Illustration from [San+21] visualising the modelling of different problems using graphs.*

this paper they will be denoted nodes. The *relations* between objects in a graph are represented by **edges**.

The potential of graphs lies both in their ability to model the relationship between objects as well as its generality. Graph data structures have been applied in vastly different domains from drug interactions to interactions between objects in an image.

Node **Definition 1.** *An object in a graph.*

Edge **Definition 2.** *A connection between two objects in a graph.*

The nodes in a graph can have extra information attributed to them. An example of this is the gene expression features found in biological networks or text features in social networks [Ham20]. This is typically stored in a vector $\mathbf{x}_u$ for a given node u. In this case a graph is referred to as an **attributed graph** and otherwise as an **unattributed graph**.

Unattributed Graph **Definition 3.** *An unattributed graph is represented as $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges.*

Attributed Graph **Definition 4.** *An attributed graph is represented as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. $\boldsymbol{X}$ is an attribute matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ where $|\mathcal{V}| = N$ and D is the number of attributes per node.*

Each edge can also have additional information written into it. In that case the edge is referred to as a **weighted edge**. This additional information indicates the strength of the connection between nodes. An example of a weighted edge in a social network graph is the number of text messages between two users in a chat conversation.

Weighted edge **Definition 5.** *And edge with an associated number indicating the strength of the connection between two nodes.*

Given that the edges of a graph are weighted, the graph is a **weighted graph**. In some literature it is referred to as an edge-attributed graph instead.

Aside from adding extra information to the edges, the direction of a connection can be specified. A mutual relationship between a node is represented by an **undirected edge** whereas an one-sided relationship has a **directed edge**. Depending on whether the edges in the graph are

undirected or directed, the graph is referred to as a **directed graph** or an **undirected graph** respectively.

Undirected edge

Definition 6. *The connection between two nodes, u and v, denoted as $e_{u,v} \in \mathcal{E}$. Undirected edges have the property $e_{u,v} \in \mathcal{E} \leftrightarrow e_{v,u} \in \mathcal{E}$.*

Directed edge

Definition 7. *The relationship of a node u to a node v, denoted $e_{u,v} \in \mathcal{E}$. The existence of $e_{u,v} \in \mathcal{E}$ does not imply $e_{v,u} \in \mathcal{E}$.*

A graph can be both **attributed**, **weighted** and **directed** at the same time as well as every combination of these qualities.

2.2.1 Representations of Graphs

Adjacency Matrix

Graphs can be conveniently represented through an *adjacency matrix*, $\mathbf{A} \in \mathbb{R}^{N \times N}$.

The square matrix is constructed such that row and column indexes correspond to nodes in the graph. The connectivity between two nodes, u and v in an unweighted graph are presented by $\mathbf{A}[u, v] = 1$ given that the two nodes are connected and $\mathbf{A}[u, v] = 0$ otherwise. If the graph is undirected, the adjacency matrix will be *symmetric* while it is not necessarily symmetric for a directed graph.

Given a weighted graph, the elements in the adjacency matrix are denoted by the weight of the specific edge.

Adjacency Matrix

Definition 8. *A matrix, $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ which elements describe whether a set of nodes are connected. A set of nodes $\{u, v\}$ that are unconnected will have 0 in the corresponding element $\boldsymbol{A}[u, v]$. Otherwise $\boldsymbol{A}[u, v]$ is equal to the weight of the edge. An unweighted graph has a weight of 1 for all edges.*

Since a graph often consists of many nodes with low inter-connectivity, the adjacency matrix of a graph is often sparse.

Node Degree

The node degree is a measure of how many neighbours a specific node has. It is defined as the count of edges directly related to the node.

Node Degree

Definition 9. *Given a node $u \in \mathcal{V}$, the node degree $deg(u)$ counts the number of edges incident to u:*

$$deg(u) = \sum_{v \in \mathcal{V}} \boldsymbol{A}[u, v]$$

Given a directed graph, the **node degree** is the same as above, but there are additional notions of degrees, namely in- and out-degree.

Given a node u, the **in-degree** is the count of directed in-going edges to u and the **out-degree** is the count of directed out-going edges from u to other nodes.

Importantly, the node degree of a user in a directed graph can be calculated as

$$deg(u) = deg^{+}(u) + deg^{-}(u)$$

where $deg^{+}(u)$ is the out-degree of the node and $deg^{-}(u)$ is the in-degree.

Given a weighted graph, the notion of weighted degrees can be introduced. The **weighted degree** is given by

$$deg(u) = \sum_{v \in \mathcal{V}} w_{u,v}$$

where $\mathbf{A}[u,v] = w_{u,v}$ given the weighted graph.

Additionally the degree measurements can be weighted and directed in the case of a directed weighted graph.

Degree Matrix

The degree of all nodes in the graph can be represented in the degree matrix.

Degree Matrix

Definition 10. *Given an adjacency Matrix, the degree matrix* $\boldsymbol{D}$ *is defined as:*

$$\mathbf{D}_{u,u} = \sum_{v=1}^{n} \mathbf{A}_{u,v}$$

Clustering Coefficient

The clustering coefficient measures how tightly clustered a node's neighbourhood is in a graph [Ham20]. It looks at a specific node and its neighbours, and assesses the degree to which those neighbours are also linked to each other [Bar14].

Clustering Coefficient

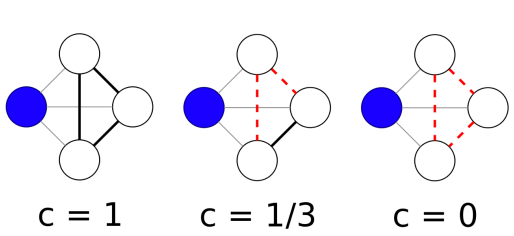
Definition 11. *Given a node* $u \in \mathcal{V}$ *and the neighbourhood of* u, $\mathcal{N}(u)$, *the clustering coefficient for node* u *is defined as:*

$$c_u = \frac{|e_{v_1,v_2} \in \mathcal{E} : v_1, v_2 \in \mathcal{N}(u)|}{\binom{deg(u)}{2}}$$

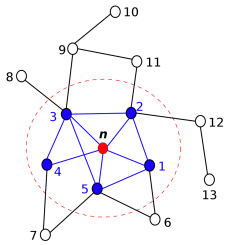
The numerator counts the number of edges that exist between the neighbours of node u and the denominator computes how many pairs of nodes that are possible in u's neighbourhood [Ham20]. This coefficient is a number between 0 and 1. If $c_u = 0$, none of u's neighbours are connected to each other, and if $c_u = 1$, all of u's neighbours are connected to each other, indicating maximal clustering. Figure 2.2a illustrates how the completeness of graphs affects the clustering coefficient.

Ego Graph

An ego-graph is a sub-graph consisting of a single focal node (the ego) and all of its neighbours (alters). It is used for analysing the properties of a graph in the local neighbourhood of a specific node [Aka+18]. The ego-graph consists of alters that are less than a certain path-length from the ego-node.



(a) *Example of different clustering degrees.*



(b) *Example of a 2-hop ego graph.*

Figure 2.2: *Illustration from [Hol22] (left) and [Aka+18] (right).*

Ego-graphs can also be utilised for visualising graphs when they are too large to comprehend and for analysis of the variation between behaviour of different nodes in the graph.

Ego Graph

Definition 12. *A sub-graph consisting of all nodes that are less than a certain distance from a focal node. We denote the ego graph of node u as $\mathcal{G}_u(\mathcal{V}_u, \mathcal{E}_u)$.*

Figure 2.2b illustrates an example of an ego graph, with an ego-node **n** with its 1-hop (blue) and 2-hop (colourless) neighbours.

Homophily assumption in graphs

The majority of graph-based approaches for node classification leverage the connections between nodes to create meaningful embeddings. Many of these approaches rely on the assumption of homophily, which suggests that nodes that are connected in a graph are more likely to have similar attributes.

Homophily

Definition 13. *The tendency for individuals who are socially connected in some way to display certain affinities, such as similarities in demographic background, attitudes, values, and so on [APA22a]*

Heterophily

Definition 14. *Any tendency for individuals who differ from one another in some way to make social connections. It is less common than homophily. Complementarity, which occurs when people with different but complementary characteristics form a relationship, is an example of heterophily. [APA22b]*

2.3 Graph Neural Networks

Wu et al. [Wu+19] describe that modelling complex data structures like graphs is not possible with traditional deep learning methods like convolutional and recurrent neural networks (CNNs and RNNs). These methods are able to extract latent representations from Euclidean data like images and text, however graphs structures arise from non-Euclidean domains where inter-dependency between nodes are present. Wu et al. highlight that the assumption of independent instances is violated for graph data because nodes are inherently related to each other by links, for instance by single interactions or friendships.

Hamilton [Ham20] describes the Graph Neural Network (GNN) as a general framework for defining deep neural networks on attributed graph data. Most GNNs computes and passes

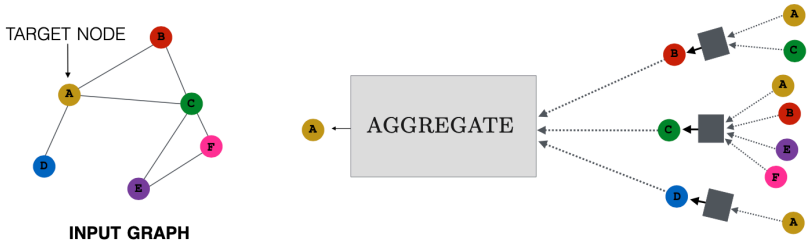


Figure 2.3: *Illustration from [Ham20] which describes how a single node, A, aggregates messages from its local neighbours (nodes B, C and D), whose information also is based on their respective neighbours and so on. This would correspond to a two-layer message passing model.*

information between nodes iteratively. This is formally known as *neural message passing* popularised by Gilmer et al. [Gil+17].

In the following sections, we will first explore a simple version of the message passing framework whose graph-level propagation rule is defined as:

$$\mathbf{H}^{(k)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}\right) \tag{2.1}$$

and investigate the frameworks which build upon this rule. Furthermore, the importance of preserving graph symmetries will be elaborated upon.

The following GNN frameworks will assume an attributed graph input in the form of $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{X})$. Using both topological and node information, the GNN aims to generate meaningful node embeddings $\mathbf{z}_u$ for each node u in $\mathcal{V}$.

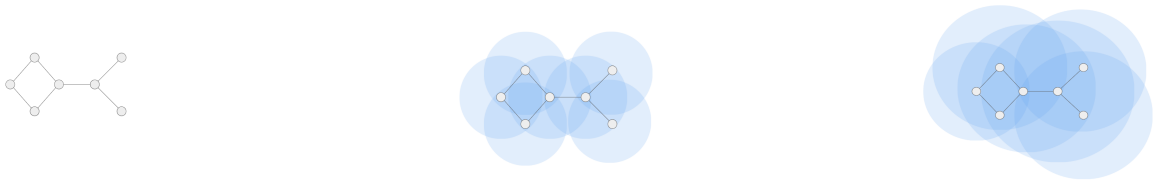
2.3.1 Message Passing

In order to achieve a representation of the belonging of a specific node in a graph, information about its relation to other nodes is needed along with their similarity and connectivity. This can be achieved through message passing, also sometimes referred to as information passing.

The core idea is that during each message passing iteration, the hidden embedding $\mathbf{h}_u^{(k)}$ of a single node u in layer k is updated according to the messages aggregated from u local neighbourhood in the graph, $\mathcal{N}(u)$. This idea is illustrated in Figure 2.3.

The figure illustrates a two-layer message passing, where we consider a target node and retrieve all information from the 1-hop neighbours. This information is aggregated using an arbitrary function and the node representation for the target node is updated.

Following the notations of [Ham20], the process in which the nodes learn information about



(a) *k=0* **(b)** *k=1* **(c)** *k=2*

Figure 2.4: *Expansion of visual perception over time steps using the neural message passing update rule. Inspiration taken from [Mic20].*

each other can formulated by the *general* message passing propagation rule expressed as:

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\right\}\right)\right) \tag{2.2}$$

$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right) \tag{2.3}$$

where the UPDATE *and* AGGREGATE *are arbitrary differentiable functions and* $\boldsymbol{m}_{\mathcal{N}(u)}$ *is the message that is aggregated from the neighbourhood* $\mathcal{N}(u)$ *of the target node.*

The update function integrates the new message $\mathbf{m}_{\mathcal{N}(u)}$ into the old embedding $\mathbf{h}_u^{(k-1)}$ to create a new representation for u, $\mathbf{h}_u^{(k)}$. We note that for $k = 0$, the initial embedding for each node is its input features, $\mathbf{h}_u^{(0)} = \mathbf{x}_u, \forall u \in \mathcal{V}$. After running K iterations of the GNN neural message passing, the final embeddings are computed and denoted as:

$$\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V} \tag{2.4}$$

By this update rule defined in 2.3, each node only knows about itself at iteration $k = 0$ and at each step the field of perception increases. As an example, each node has information about itself at $k = 0$, whereas it has information from its 1-hop neighbours at $k = 1$ and from their 2-hop neighbours at $k = 2$ and so on throughout the graph. This means that after K number of layers, a node has information about nodes K edge steps away from it and eventually a node will have aggregated information from every other node in the graph (given that K is large enough). This expansion of perception is visualised in figure 2.4. The output of the GNN becomes a graph of the same structure as the initial input graph, where each node is encoded with information about its belonging within the graph [Mic20].

Hamilton describes two forms of information being propagated through the algorithm, namely *structural* and *feature-based* information. A node u might receive structural information that contain the degree of its k-hop neighbour nodes or how clustered its k-hop neighbourhood is. Additionally, node u might also receive feature-based information about its neighbours, for instance which text features its neighbourhood nodes include. The manner in which features are aggregated in local graph neighbours is analogous to the way convolutional kernels in CNNs behave.

Before examining how the AGGREGATE function in Equation 2.3 may be defined, the importance of permutation invariance and equivariance in GNNs is elaborated upon.

2.3.2 Permutation invariance and equivariance

Velickovic [Vel21] discusses how traditional image convolution can inform the design of GNNs that preserve certain symmetries. In image convolution, a kernel matrix K is applied to an image I to capture local patterns of interest. CNNs have the property of *translation invariance*, meaning that a pattern is considered interesting regardless of its location in the image (hence the sliding of K across the entire image). Additionally, CNNs capture the locality within an image, as pixels that are closer together tend to be more related than those that are farther apart.

Velickovic considers how these principles can be applied to convolution on graphs. Unlike images, where pixels have a fixed order, graphs have no natural node ordering. Graph isomorphism indeed poses as an initial challenge for the GNN. We wish that our GNN would give us the exact same results on the same graph no matter the node ordering we implicit define when stacking the D features of the node into a feature matrix $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$. The same implicit ordering holds when defining our adjacency matrix $\mathbf{A}$ with respect to the edges. No matter how we choose to permute $\mathbf{X}$ or $\mathbf{A}$, any function f we apply on them must produce the same every time, thus f must be *permutation invariant*:

$$f\left(\mathbf{PX}, \mathbf{PAP}^\top\right) = f(\mathbf{X}, \mathbf{A})$$

where $\mathbf{P}$ is a permutation matrix. In neural message passing, the AGGREGATE function function takes the set $\{\mathbf{h}_v, \forall v \in \mathcal{N}(u)\}$ as input, which has no natural ordering of a nodes neighbours, and the aggregation function must then be permutation invariant. A simple example of a permutation invariant aggregation function is the sum, which is independent of the input order [Bro+21].

Furthermore, Bronstein et al. [Bro+21] describes that if we wish to update the features for every node to obtain a latent set of node features, stacking these features into a matrix $\mathbf{H}$ is no longer permutation invariant. The order of the rows in $\mathbf{H}$ should correspond to the rows in $\mathbf{X}$, so a property more fine-grained than permutation invariance is needed. The function f should consistently permute the input such that we can determine which output node feature corresponds to which input node. This requires f to be *permutation equivariant*:

$$f\left(\mathbf{PX}, \mathbf{PAP}^\top\right) = \mathbf{P}f(\mathbf{X}, \mathbf{A})$$

An example of a permutation equivariant function is a shared linear transform $\mathrm{F}(\mathbf{X}) = \mathbf{XW}$.

The the graph neural network layer can be defined in terms of both permutation invariance and equivariance. The layer $\mathrm{GNN}(\mathbf{X}, \mathbf{A})$ must be a permutation equivariant function constructed by appropriately applying a permutation invariant function, which operates over all sets of

neighbourhoods, $\{\mathbf{h}_v, \forall v \in \mathcal{N}(u)\}$, in $\mathcal{G}$, [Vel21].

2.3.3 The Basic and Self-Loop GNN Approach

Hamilton [Ham20] describes the basic GNN framework as:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}_{\text{self}}^{(k)}\,\mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)}\right) \tag{2.5}$$

where $\boldsymbol{W}_{self}^{(k)}, \boldsymbol{W}_{neigh}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ *are learnable node-wise weights,* $\boldsymbol{b}^k$ *is a bias term and* σ *is a non-linearity like tanh or ReLU.* The idea is that we first sum the information from the incoming neighbours and then combine that neighbourhood information with the previous node embeddings. In this way the aggregation function is the sum:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v$$

and the update function becomes;

$$\text{UPDATE}\left(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}\right) = \sigma\left(\mathbf{W}_{\text{self}}\,\mathbf{h}_u + \mathbf{W}_{\text{neigh}}\,\mathbf{m}_{\mathcal{N}(u)}\right)$$

However, it is common to add self-loops to the input graph in order to omit the update step which combines the two entities described above:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k-1)}\right) \tag{2.6}$$

The update function is thus defined through the aggregation function. The following graph-level update can be achieved:

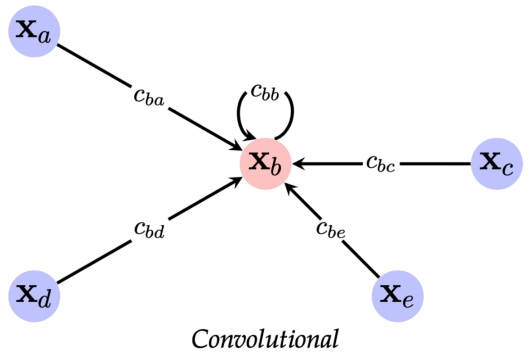
$$\mathbf{H}^{(k)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}\right) \tag{2.7}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self loops.

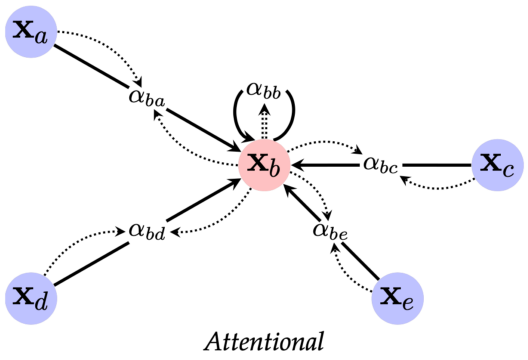
The propagation rule in Equation 2.7 is the foundation of many GNN approaches. Bronstein et al. [Bro+21] argue that the vast majority of GNN frameworks can be derived from three 'flavors', namely the *Convolutional* [KW17], *Attentional* [Vel+18] and *Neural Message Passing* [Gil+17] flavours. The methods we explore in this thesis are based almost entirely on the Convolutional framework and a single on the Attentional, so the Neural Message Passing framework will not be explored further.

2.3.4 Graph Convolutional Networks

Using the sum alone an aggregation function can propose some issues - in some instances it can be unstable and highly sensitive to node degrees [Ham20]. A solution to this is to normalise the aggregation function based on the node degrees. Kipf and Welling [KW17] proposed the



(a) *Message passing in the convolutional framework.*



(b) *Message passing in the attentional framework.*

Figure 2.5: *Visualisation of data flows. Illustration from Bronstein et al. [Bro+21]*

following use of*symmetric normalisation* in 2016[1]:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} \tag{2.8}$$

where each embedding is normalised by the neighbourhood sizes of both the source node u and destination nodes, $v \in \mathcal{N}(u)$. For instance, if we consider a network on a social media platform, and a node u is connected with his or her friends but also follows a few celebrities, who all are followed by millions of people. The coefficient $\frac{1}{\sqrt{|\mathcal{N}(v)|}}$ will then be very small and the information node u can obtain from the celebrities will not be as important or unique because that same information is being propagated to many other neighbourhoods. Symmetrical normalisation ensures that aggregation of embeddings remains numerical stable.

This aggregation method is a key part of the Graph Convolutional Network (GCN) proposed by Kipf and Welling [KW17]. The neural message passing is defined for the GCN as:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right) \tag{2.9}$$

This can more compactly be written as a graph-rule rule:

$$\mathbf{H}^{(k)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}\right) \tag{2.10}$$

where $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$.

Figure 2.5a visualises message passing in GCNs. Here we see that the node features of the sender are multiplied with a constant, c_{uv}, from each neighbour v, as seen in Equation 2.8.

The GCN layer is currently one of the most popular GNN models and is employed in several of methods seen in literature today [Kim+22] and most of the methods used in this thesis.

[1]The paper was revised in early 2017.

2.3.5 Graph Attentional Networks

The Graph Convolutional Network determines the coefficients, c_{uw} for the embedding of node v, indicating how important node v is to node u, based *explicitly* on the structure of the graph. However, the importance of one node to another may not be fully captured by the graph structure alone. An alternative approach is to use an attention function, $attn(\mathbf{h}_u, \mathbf{h}_v)$, which takes into account both nodes u and v to compute a weight that reflects the importance of node v to node u. In this way, the function allows node u to attend to some neighbours more than others even though they have the same neighbourhood sizes.

The Graph Convolutional Network defines its coefficients, c_{uw}, *explicitly* based on the graph structure for defining how important the embedding of node v are to node u. However, the importance of a neighbour to another might not only be defined by the graph structure. We can instead employ an attention function, $attn(\mathbf{h}_u, \mathbf{h}_v)$, which computes this weight that considers both the source node u and destination node v *implicitly*. In this way, the function allows node u to attend to some neighbours more than others even though they have the same neighbourhood sizes. Velickovic et al. [Vel+18] proposed the first GNN model which implements attention weights, namely the Graph Attentional Network. In their paper the attention scores and function are defined as:

$$a_{uv} = attn(\mathbf{h}_u, \mathbf{h}_v) = \text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v\right]\right) \tag{2.11}$$

The attention score a_{uv} is the importance of node u to node v and the attention mechanism $attn$ is single-layer feed-forward neural network parameterised by a trainable attention vector $\mathbf{a}$, a trainable matrix $\mathbf{W}$ and $\oplus$ denotes the concatenation operation [Ham20]. The soft-max function is applied to normalise the raw scores $a_{u,v}$ so they can be used as weights, which they denote as $\alpha_{u,v}$:

$$\alpha_{uv} = \text{softmax}_v\left(a_{uv}\right) = \frac{\exp\left(a_{uv}\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(a_{uv'}\right)} \tag{2.12}$$

We then obtain message passing for the Graph Attention Network:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \alpha_{uv} \mathbf{h}_v\right) \tag{2.13}$$

which is visualised in Figure 2.5b - the messages are weighed with information about both the source and destination node rather than only the destination nodes as seen in the convolutional case. Bronstein et al. also comment on the relationship between these two approaches - in fact, the convolutional approach can be represented by the attentional approach, where the attention mechanism is implemented as a look-up table $\alpha_{uv} = c_{uv}$.

3 Litterature Review

Several papers have previously investigated online grooming detection from a psychological, social or data analysis perspective [Kon+09], [GKS12], [Vil+12], [Che+13]. In this thesis, the primary focus will be on analysing and modelling predatory behaviour from data, and the literature review will reflect this focus.

Previous research on online sexual predator detection using data has primarily focused on analysing linguistic features of conversations. This paper, however, focuses on user behaviour by utilising a graph representation of the social network. While research using natural language processing (NLP) and linguistic feature analysis have been published across many years, the recent popularity of GNNs has led to increased research on graph anomaly detection and behaviour analysis in graphs.

3.1 Detection of online sexual predators using linguistic features

One notable example of research on detecting sexual predators is the 2012 PAN Sexual Predator Identification competition, which aimed to provide researchers with a common framework to test methods for identifying online sexual predators. This competition was a two-part task, with the first task being to identify the predators among all users in the different conversations [IC]. The PAN2012 data set includes three types of conversations: grooming conversations between predators and adults posing as children, consensual sexual conversations between adults, and non-sexual conversations. The training data set consists of around 70,000 conversations produced by nearly 98,000 users. 142 of them were labelled as a predator (0.15%) and around 4.5% of the conversations were labelled as predatory [IC].

The winner among the 16 teams that participated in the competition, was Villatoro et al. [Vil+12], who divided the task of sexual predator detection into a two-step approach. The first step, Suspicious Conversations Identification (SCI) concerned with distinguishing between general chatting and possible cases of online child exploitation and the second step, Victim from Predator disclosure (VFP) was concerned with identifying which of the two users were the predator. On the SCI task, they experimented with a classifier where the conversations was represented by a bag-of-words (BoW) vector which employed either a binary or a TF-IDF weighting scheme. Their best results on this task was a $F_{0.5}$ score of 0.93 with a NN classifier set as a two layer neural network with a single hidden layer of 10 units and the binary BoW representations of chats.

Parapar et al. [PEB12] who also entered in the PAN competition proposed extracting psycho-linguistic features using a Linguistic Inquiry and Word Count (LWIC). LWIC

calculates the degree to which people use different categories of words and Parapar et al. extracted 80 LWIC features for their data set. The text analysis was user-level based, meaning that they concatenated all lines that the user produced in any chat to use as the document-based representation of that user. Asides from text-based features, they analysed features capturing more global aspects of the chat-behaviour. These features include the number of conversations initiated by an individual, average time of day the chat usually starts, the average number of people participating in the conversation etc.

Cheong et al. [Che+13] utilised a NLP-based method to detect sexual predators on MSP. They were provided with text data from MSP that were classified either as unlabelled or labelled as predatory (0.6%). Contrary to the PAN data set, the chat logs contain predatory conversations with actual victims rather than adults posing as children. The researchers created feature vectors for modelling predatory behaviour, which included Bag of Words (BoW), sentiment analysis-based, and rule-breaking features. The BoW features used TD-IDF weighting, similar to what was experimented with in [Vil+12]. Some rule-breaking features consisted of counting the number of blacklisted words and their various spellings in the conversations. Behavioural features, such as the number of non-letter words typed by a user, were also included since users may replace certain letters with symbols or numbers in an attempt to bypass the blacklist (e.g., using "s*x" instead of "sex"). In addition, the number of spaces or consecutive identical letters in the words (e.g., "s e x" or "seeeex", respectively) were counted as additional behavioural features. They experimented with machine learning approaches such as Naive Bayes, Decision Trees, MLPs and the K-nearest neighbour algorithms and achieved their best results with a MLP algortihm on all the features with an $F_{0.5}$-score of 0.86.

The use of behavioural features is also proposed in Milon-Flores et al. [MC22] who describes a set of seven Behavioral Features (BFs) that correlate well with early detecting of grooming behaviour in online conversations. These features include emoticons, correctly-spelled words, sexual topic words, timestamps, intervention words and sentiment and emotional markers.

3.2 Graph Anomaly Detection

Another approach to detecting predators online is to investigate the behaviour of users in terms of their social network patterns. This can involve analysing how users connect to one another, who they connect with, and how frequently they communicate. By looking at these factors, it may be possible to identify patterns of behaviour that are characteristic of predators and use them to develop algorithms or other tools for detecting predatory behaviour. This approach differs from those that rely on the content of text messages, as it focuses more on the underlying social network and communication dynamics rather than the specific content of the messages being exchanged.

Since users on Social Media are connected by their social relationships, which are modelled naturally by graphs, several papers proposes using both the graph structure and features when

detecting various online abnormal behaviour [Wan+], [Ma+21], [Vau].

Anomaly detection is a branch of data mining that focuses on identifying unusual or rare patterns within a data set. A key difference between anomaly detection on tabular and graph data is the assumption of independence and identically distributed (i.i.d.) instances. Outlier detection on tabular data often assumes that the data points are i.i.d., which is violated for graph data.

Akoglu et al. [Ako+15] provides an overview of the state-of-the-art methods for anomaly detection in structured graph data in their paper from 2015. The paper focuses on unsupervised learning techniques. They furthermore highlight key challenges associated with the graph anomaly detection problem that are relevant for our problem as well. One of the challenges is the complexity of the data structure since attributed graphs can be very rich in content such as user demographics, interests, roles, text etc.. which could easily scale to a very large graph containing complex information. Secondly the anomaly detection problem also have a high class-imbalance, which creates a problem when training a classification algorithm. Lastly they highlight that graph structured data often comes with either a lack of labels or very noisy labels due to the inconsistency of human annotators when processing complex information. This is true for our data as well, where the nature of the problem makes it difficult to assign labels to users. For this reason, Akoglu et al. focuses on unsupervised techniques for anomaly detection in their paper.

In 2022, Liu et al.[Liu+22a] published a comprehensive benchmark, BOND, on unsupervised node outlier detection in attributed graphs. They motivate the paper by highlighting the lack of a standard comprehensive setting for evaluating the performance of the numerous algorithms being developed in the field in recent years. The authors focused on detection accuracy, runtime, and memory consumption of multiple graph outlier detection algorithms and also tested four non-graph baselines. Moreover they develop a framework [1] that provides code, APIs and optimisations for the models discussed in the paper. Liu et al. showed that the bench-marked deep graph methods have better average performance than non-deep graph methods.

To compare the performance of different attributed graph outlier detection algorithms, Liu et al. used commonly used outlier detection metrics such as ROC-AUC and average precision. Some of their data sets came with labels, making it easy to evaluate the detection of the unsupervised models. However, they also had unlabelled data sets and proposed an injection method for injecting synthetic outliers into these data sets to evaluate the performance of the algorithms. The proposed injection method could inject either a **structural outlier** (anomalous in the structure of the graph) or a **contextual outlier** (anomalous in the attributes of the graph). However, they argue that organic outliers are more likely to be a combination of the two.

Kim et al. [Kim+22] describes graph outliers as patterns within the graph that do not conform

[1] https://github.com/pygod-team/pygod

to normal patterns expected of the attributes and/or structures of the graph. In the field of graph anomaly detection, researchers often distinguish between *structural* and *contextual* anomalies in attributed graphs. Bandyopadhyay et al. [BVM20] introduces the notion of a **combined outlier** and defines it to be a node that belongs to one community structurally, but belongs to a different community in terms of attribute similarity or vice versa.

Many outlier detection methods use an graph auto-encoder (GAE) structure to learn latent node embeddings in an unsupervised manner. There are various ways to define the encoder and decoder functions, such as using MLP [BVM20], GCN [Yua+21], [Din+19], and GAT [FZL20] layers. GAEs used on attributed graphs often include a structural loss in addition to the attribute reconstruction loss that is typically used in traditional auto-encoders. Hamilton describes in his book on Graph Representation Learning [Ham20] that *pairwise decoders* are commonly used for decoding the graph structure. These decoders are used to reconstruct the relationship or similarity between pairs of nodes and can be defined in multiple ways, with a common approach being the reconstruction of the adjacency matrix.

While unsupervised models can prevent noise from unreliable labels from affecting the model, they also ignore important information about anomalous instances. Unsupervised outlier detection algorithms can identify outliers, but not specific types of outliers. To learn to identify specific types of outliers, the model needs to be provided with examples of what those types of outliers look like. This is a limitation of unsupervised approaches, as they rely solely on the input data and do not incorporate any additional information or labels.

Kumagai et al. [KIF20] proposed a semi-supervised method for detecting anomalous instances on attributed graphs that takes into account the imbalance in the data. To address the problem, they used the differential area under the curve (AUC) loss as a regulariser. This differential approximation has been shown to be effective for class-imbalanced data in previous studies by Iwata et al. [IY] and Kumagai et al. [KIF19]. The model produces node embeddings where normal instances are located close to a centre vector in the embedding space and anomalous instances are farther away. Kumagai et al.'s method outperformed various existing anomaly detection methods in several different settings when tested on five real-world attributed graph data sets. However, their approach is based on undirected graphs.

Other papers, such as those by Yang et al. [YCS16], Kipf and Welling [KW17], and Rong et al. [Ron+20] have also researched variations of semi-supervised learning frameworks for outlier detection in graph structured data.

Semi-supervised outlier detection methods have the advantage of incorporating both labelled and unlabelled data in their analysis, which can improve performance compared to unsupervised methods. However, these approaches do require that some labelled instances be present in the data set, which can be a limitation if such data is not available.

3.2.1 Graph-based Predator Detection

The detection of online predators based solely on the analysis of social graphs is a novel research area. To our knowledge, the NTNU-based research by Anna Aarekol [Aar22] is one of the first studies to investigate the use of graph analysis for this purpose. Aarekol was provided with MSP data containing instances of messages, but not the actual messages. She represented each user as a node and each conversation as a weighted edge between two nodes, and extracted a set of 22 graph-based features from these graphs. These features included degree-based features (e.g., in-degree and out-degree of a node), neighbourhood information (e.g., unweighted and weighted clustering coefficient), and features related to the length of the conversations. The graph data was not labelled, so Aarekol applied unsupervised clustering algorithms such as k-means, Gaussian Mixture Model, BIRCH, Mean Shift, and DBSCAN. By examining the small clusters produced by these algorithms and analysing the nodes within these clusters, she was able to detect predators and other users displaying unwanted behaviour. The BIRCH algorithm was found to be particularly effective, as it resulted in clusters of various lengths that were easier to investigate. Aarekol concluded that it is indeed possible to detect online predators using graph-based analysis.

4 Data

4.1 Retrieval and processing

The data for this study was collected from MSP, an online social game for children and young teenagers between the ages of 8 and 15. In the virtual world of MSP, players take on the roles of movie stars or fashion designers and form online relationships with other users. Communication occurs either in open forums where any user can join the group chat or in private messages between two users. The data for this study contains only private messages. However, the use of open chat rooms allows users to communicate with others outside of their friend circles, resulting in a naturally lower clustering coefficient in the social network.

MSP has implemented several measures to safeguard children on the platform, including advanced filtering by their own content analysis system, trained moderators, and partnerships with law enforcement agencies and non-governmental organisations [Mov22a]. The company operates 16 servers around the world, which are accessible from any location. The chat data used in this thesis was collected from the US server, which include the UTC time stamp, sender, receiver and text ID for each message. A unique ID was also given for each conversation between two users. All user identities have been de-identified and the IDs have been hashed to ensure data safety. Table 4.1 shows a sample of five random entries from the data set.

The data set consists of 121 continuous days of entries from January 7th, 2022 to May 7th. It originally contained 35,735,300 entries with 253,446 unique users before pre-processing. There are 83.870 instances of conversations between two users that were mistakenly recorded as one-sided. This will be referred to as self-loops for the remainder of this thesis. Self-loops have been removed since they can affect the features of the model. After identifying and removing 83,870 instances of self-loops, the data set was left with 35,651,430 entries and 253,397 unique users.

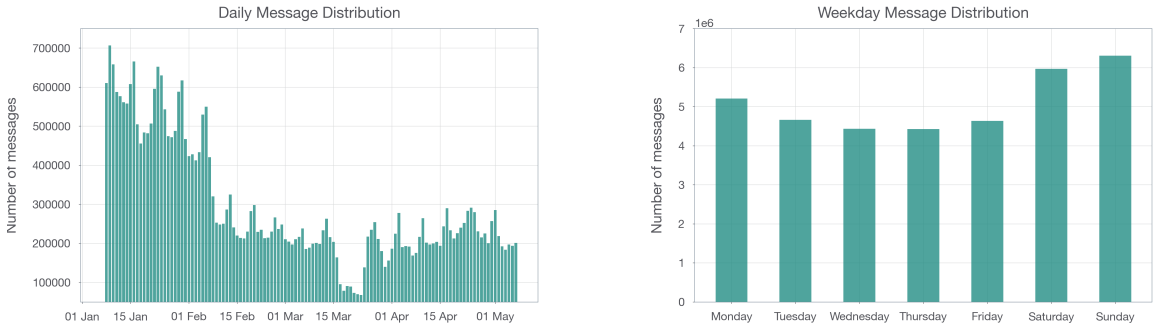
Self-loop

Definition 15. *Messages where the sender and receiver IDs are identical. This is a result of a data collection error.*

We differentiate between all unique and active users, where active users are defined as having

time	*sender ID*	*receiver ID*	*text ID*	*conversation ID*
2022-01-10 15:01:21.227	849419713613	310027196799	299611584147	721776029695
2022-01-27 08:25:19.811	245385583436	157391106515	309933056410	495345148324
2022-02-27 07:45:00.515	287542623032	303707605406	728729767997	305320550430
2022-03-16 09:25:18.797	427946872317	189672978795	258530461919	141406423414
2022-03-27 18:24:10.512	884334592513	859929634128	335131572229	208277685485

Table 4.1: *Sample from the data set*



(a) *Daily distribution of number of messages* **(b)** *Number of messages vs. weekday*

Figure 4.1: *Visualisations of message distributions*

send a message in the given time period. Among these unique users, 74% (187,034) were considered active for the four-month time period.

Active User **Definition 16.** *A user that has send a message to another user in the given time period*

For the purpose of this thesis, which focuses on the frequency of messages and user behaviour rather than text linguistics, only the time, sender and receiver IDs are used for encoding information in the model. The text messages and conversation IDs are only utilised for manual evaluation.

4.1.1 Visualising message distributions

The daily fluctuations in the number of messages are shown in Figure 4.1a. January had the highest level of engagement, while March had the lowest. With the exception of a dip in engagement during the last two weeks of March, the level of engagement remained relatively consistent from mid-February to May.

Figure 4.1b shows that the distribution of messages per weekday is relatively even, with Sunday having the highest level of activity. Since the data is from a US server and the time stamps are in UTC format, it is not very meaningful to include the number of messages per hour. This is because the US has six different time zones and users from around the world can access the US server, which would distort the distribution. Similarly, the distribution of messages per weekday in Figure 4.1b is also skewed by this.

4.2 Groomers and sexting-users

The goal of this study is to identify anomalous users who participate in sexually charged conversations and users who exhibit potential grooming behaviours. We define sexually charged conversations (sexting) as those that involve the explicit use of sexual language or descriptions of sexual acts, and may also involve role play between two users. An example of such a conversation is shown in Table 4.2. Users who may potentially be groomers often participate in sexting conversations, in which the role play may involve a family relationship or a significant age difference between the two roles. Additionally, users who attempt to meet

Sender	Time	Message
A	06:24 AM	**rubbing my /) against your pxxxy **
B	06:24 AM	*>/////<*
A	06:25 AM	*like that huh∼?*
B	06:25 AM	*maybe∼*
A	06:25 AM	*∗rubbing super super fast ∗*
B	06:26 AM	*YEAH∼*
B	06:26 AM	*daddy∼∼∼∼*
A	06:28 AM	*good girl∼∼*
B	06:28 AM	*YES∼*
A	06:29 AM	*∗cxms inside∗*
B	06:29 AM	*more∼∼∼*

Table 4.2: *Example of sexting from the data set*

other users on other social networking sites that allow for photo and video sharing or in real life are also considered potential groomers.

We did not initially have labels for the 187,034 users indicating whether they were groomers or participated in sexting on the platform. However, we were provided with a list of 648 conversation IDs from Aiba that were potential sexually charged conversations. This list was the output of an early NLP-based sexting detection model. Since the list contained false positives, the conversations were manually validated. From these conversation IDs, we calculated how many unique sexting conversations each of the users had participated in, as shown in Table 4.3.

Users who had participated in more than three sexting conversations were manually validated by reading through the conversations to confirm that they were true positives. Aside from already displaying sexual behaviour, we assumed that users who had more than three sexting conversations were more likely to be groomers.

The manual validation of true anomalies was performed based on the definitions of sexting and potential grooming behaviour outlined above. MSP allows users to "date" and engage in mild role play, such as kissing or cuddling, with no sexual acts implied. However, any role play beyond this is considered sexting and is not allowed on the platform. Additionally, the platform's guidelines prohibit the sharing of personal identifying information, such as full names, schools, or other social media accounts [Mov22c]. Users who frequently request this type of information are also considered anomalous.

During the manual validation of sexting conversations, we identified additional users who had

Num. sexting conversations	1	2	3	4	5	6	7	8	9	10	11	12
Num. users	689	107	50	20	10	0	3	1	1	2	0	1

Table 4.3: *Users vs. how many sexting conversation they have participated in.*

been sexting but had not been detected by the text model. In total, we were able to confirm the presence of 346 anomalous users.

4.3 Splitting the data

Using the entire data set for creating a single graph would yield more than 250,000 nodes and 35 million edges. Such a large graph is neither feasible to model, nor reflects the final use case for our methods. In order to make the model useful for detecting anomalous behaviour every few days, the data was divided into sub-sets of seven and three days' worth of data. As shown in Figure 4.1a, January had a higher level of engagement than the other months, so the number of days was chosen based on the size of the graph that was feasible to create for January. For example, the sub-set summarising the first 14 days of January contained over 82,000 nodes and 636,000 edges, which is a relatively large social network. The seven-day split resulted in appropriately sized sub-sets, as shown in Table 4.4. The three-day split was included to investigate whether a longer time period is helpful for more accurately detecting anomalous users.

The seven-day data set was divided into 14 sub-sets, which were assigned to either hyper-parameter search (*tuning*), hyper-parameter metric validation (*validation*), or testing of models (*test*). The assignments and summary of the sub-sets are shown in Table 4.4, which also indicates how many of the manually validated anomalies were active in each sub-set.

The three-day data set is split into 11 sub-sets, where eight of them are assigned for hyper-parameter tuning and the other three are assigned for testing as seen in Table 4.5.

name	*period*	*type*	*conversations*	*users*	*active users*	*anomalies*
Data1	Jan 07 to Jan 13	tuning	328,840	51,933	37,441	163 (0.44%)
Data2	Jan 14 to Jan 20	validation	320,803	51,230	37,008	182 (0.49%)
Data3	Jan 21 to Jan 27	tuning	326,274	50,776	36,688	187 (0.51%)
Data4	Jan 31 to Feb 06	test	280,461	46,921	33,733	171 (0.51%)
Data5	Feb 07 to Feb 13	tuning	209,479	37,558	27,329	166 (0.61%)
Data6	Feb 14 to Feb 20	validation	169,054	28,674	21,298	138 (0.65%)
Data7	Feb 21 to Feb 27	test	165,275	29,097	21,361	140 (0.66%)
Data8	Feb 28 to Mar 06	tuning	142,274	27,922	20,019	137 (0.68%)
Data9	Mar 07 to Mar 13	tuning	154,037	27,096	19,940	132 (0.66%)
Data10	Mar 17 to Mar 23	test	69,186	20,084	11,411	65 (0.57%)
Data11	Mar 25 to Mar 31	tuning	148,213	26,753	19,696	131 (0.67%)
Data12	Apr 01 to Apr 07	tuning	154,076	26,957	19,635	124 (0.63%)
Data13	Apr 15 to Apr 21	tuning	181,718	30,336	22,436	126 (0.56%)
Data14	Apr 30 to May 06	validation	167,420	28,635	21.270	129 (0.61%)

Table 4.4: *Overview of all seven day data sets. We note that the anomalies column refers to the manually validated anomalies that were feasible for us to confirm, the true number of anomalies in each period is unknown and most likely higher. The percentage of anomalies in the sets is based on the number of active users.*

name	*period*	*type*	*conversations*	*users*	*active users*	*anomalies*
Data15	Jan 16 to Jan 18	tuning	155,847	31,599	23,108	146 (0.63%)
Data16	Jan 31 to Feb 02	test	128,790	27,443	19,909	142 (0.71%)
Data17	Feb 09 to Feb 11	tuning	83,723	19,805	14,392	119 (0.83%)
Data18	Feb 15 to Feb 17	tuning	71,852	16,180	12,015	112 (0.93%)
Data19	Feb 24 to Feb 26	test	85,051	18,292	13,591	114 (0.84%)
Data20	Feb 27 to Mar 01	tuning	73,361	17,942	12,913	109 (0.84%)
Data21	Mar 05 to Mar 07	tuning	66,147	16,495	11,942	111 (0.93%)
Data22	Mar 26 to Mar 28	tuning	78,937	17,120	12,818	115 (0.9%)
Data23	Apr 13 to Apr 15	tuning	74,543	16,868	12,347	103 (0.83%)
Data24	Apr 19 to Apr 21	tuning	82,502	18,192	13,542	102 (0.75%)
Data25	May 01 to May 03	test	82,382	17,881	13,400	109 (0.81%)

Table 4.5: *Overview of all three day data sets.*

5 Methodology

The methodology section of this thesis describes the steps taken to address the main research question.

The seven- and three-day data sets are each modelled as an unweighted, directed attributed graph represented as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{X})$. Each node $u \in \mathcal{V}$ represents a user on the platform, and nodes are connected by directed edges $e_{u,v}$ if user u has sent a message to user v. If user v does not send a message back to user u, the edge $e_{v,u}$ does not exist. The node attributes $\mathbf{X}$ consist of 16 graph-based features.

The objective of this study is to compare the effectiveness of different graph auto-encoders in ranking anomalous users in an attributed graph. We aim to learn a function f that assigns a real-valued anomaly score to each node u in the graph $\mathcal{G}$, using only graph information. We also aim to determine if a GAE that utilises both the structure and attributes of the attributed graph performs significantly better than one that only uses the attributes, when the attributes encode higher-level graph information solely.

To achieve these goals, we first present the methodology for generating the high-level features that will be used as attributes in the study. Next, we propose a new method for introducing synthetic combined outliers into an attributed graph. Then, we provide an overview of each of the compared models in this study, including their overall structure, loss, and objective functions. We also explain how batching can be used to scale the algorithms for larger graph data sets. Lastly, we describe the experimental setup of this study.

5.1 Generating Node Attributes

The original data consists of sender IDs, receiver IDs, and timestamps. To model the problem, 16 features were extracted. These features were chosen based on previous research by Aarekol [Aar22] on the subject and domain-specific knowledge gained from conversations with Aiba.

To minimise cold start issues when a new user joins the platform, the chosen features are all calculated on a daily basis. This allows for a fairer comparison between users who have been on the platform for a long time and those who are new.

By calculating the attributes on a daily frequency and aggregating them for use in various-sized graphs, we are able to save computational power and minimise cold start issues when working with longer time periods.

5.1.1 Activity Based Features

To include information about how many hours or days a user has been active, we define the following two features:

Feature Name	Description
num. active hours	*The number of hours the user has been active in a given day*
num. active days	*The number of days where the user has sent at least one message in a given time period*

Table 5.1: *Feature name and descriptions for the features measuring active time in a day*

A user who has only been active for one hour in a seven-day period but has had 80 conversations exhibits abnormal behaviour. Without the active hours feature, this information would not be included in the feature set, as it is more normal to have 80 conversations over the course of seven days than in one hour. The active time features are used to weight the other features.

5.1.2 User Based Hourly Features

Assuming that predatory users aim to maximise the number of users they are chatting with, the following user-based features have been defined:

Feature Name	Description
maximum out hourly	*The maximum amount of unique users the user have written to in a single hour on a given day*
average out hourly	*The average amount of unique users the user the user have written to in an hour on a given day (only based on active hours)*
maximum in hourly	*The maximum amount of unique users the user have received messages from in a single hour on a given day*
average in hourly	*The average amount of unique users the user have received messages from in an hour on a given day (only based on active hours)*

Table 5.2: *Feature name and descriptions for features measuring unique users sent to and received from hourly*

5.1.3 Message Based Hourly Features

To capture the behaviour of a user who has been participating in a sexting conversation, we propose calculating their hourly message frequency and using both the maximum and the average hourly message count as features. This allows us to account for the periods of high message activity that may occur during the conversation, while also considering the user's overall level of activity during the active period.

Feature Name	Description
weighted max. out hourly	*The maximum amount of messages the user has sent out in a single hour on a given day*
weighted avr. out hourly	*The average amount of messages the user sends out in an hour on a given day (only based on active hours)*
weighted max. in hourly	*The maximum amount of messages the user receives in a single hour on a given day*
weighted avr. in hourly	*The average amount of messages the user receives in an hour on a given day (only based on active hours)*

Table 5.3: *Feature name and descriptions for features measuring messages sent and received hourly*

5.1.4 Number of Conversations with X Outgoing Messages

To incorporate information about the length of conversations between users, which is not represented in the graph structure, features incorporating this can be seen in Table 5.4.

Feature Name	Description
out degree 2	*The amount of conversations a user has had in a day where the user has send out less than 2 messages*
out degree 7	*The amount of conversations a user has had in a day where the user has send out between 3 and 7 messages*
out degree 32	*The amount of conversations a user has had in a day where the user has send out between 8 and 32 messages*
out degree 100	*The amount of conversations a user has had in a day where the user has send out between 33 and 100 messages*
out degree 100 plus	*The amount of conversations a user has had in a day where the user has send out more than 100 messages*

Table 5.4: *Feature name and descriptions for the length of conversation bin features*

Conversations are grouped into bins based on the number of messages sent in a single day. Each conversation can only belong to one of these bins. The bins were chosen based on the distribution of message counts in conversations. This allows for comparison of conversations based on the intensity of communication between users.

The distribution shown in Figure 5.1 is created based on the number of messages sent per user per conversation per day. It is calculated based on all data available. The median of the distribution is 2, while the mean is 8 and the maximum number of messages sent in a single conversation by a single user is 2.238 in a day.

However, these features do not take into account if the receiver of the messages respond to the user or not.

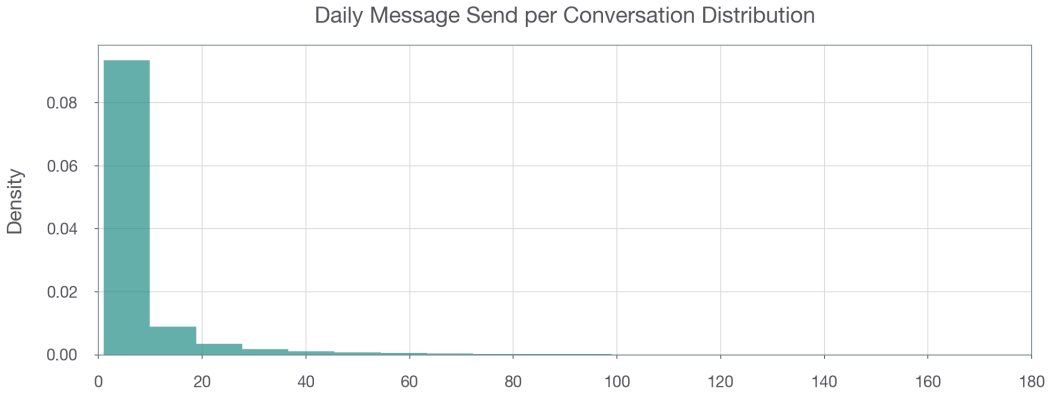


Figure 5.1: *Number of messages sent in a conversation in a day calculated for all 121 days.*

5.1.5 The proportion of sent versus received messages

Assuming that predatory users are more likely to send out more messages than they receive, as their victims may not respond, the following feature has been defined:

Feature Name	Description
weighted proportion outgoing	*The number of messages the user has sent out in a day divided by the total messages in and out of the day. The value is between 0.5 and 1 (> 0.5) if the user sends more messages than he receives. If the user receives more messages than he sends, it is between 0 and 0.5 (< 0.5)*

Table 5.5: *Feature name and description for the proportion feature*

5.2 Aggregating Node Attributes

The pre-calculated daily features are aggregated into either seven or three day graphs either by a sum, maximum or a weighted average based on the active hours. The method used depends on the specific feature being analysed.

The active hours as well as the active days are summed to get the total active time in a given k-day period. The number of active days is in $[0,\ k]$ where $k = 7$ for a seven day graph. The number of active hours is in $[0,\ k \cdot 24]$.

For features that are calculated as a maximum value, we aggregate them by finding the maximum value within the specified time period (either seven or three days). This allows us to identify the highest value of the feature within the period.

The rest of the features, except for the weighted proportion outgoing, are aggregated using a weighted average that takes into account how many days a user has been active in the period as well as weighting the features by number of active hours in a given day.

The weighted average for a specific feature is calculated by:

$$\text{feature}_{agg} = \frac{\sum_{i=1}^{k} \frac{\text{num. active hours}_i}{\text{total active hours}} \cdot \text{feature}_i}{\text{total active days}} \tag{5.1}$$

where i is a given day in the k-day time period, num. active hours$_i$ is the amount of active hours in day i and feature$_i$ is the value of the feature in day i.

By using a weighted average to aggregate the features, we ensure that values from days where the user was more active are given more weight. This allows the features to better reflect the user's activity level in the period.

The *weighted proportion outgoing* feature is aggregated by an average over the k-days. It is not weighted by active hours since it is calculated in a daily frequency and not an hourly frequency. The aggregation is done by:

$$\text{weighted proportion outgoing}_{agg} = \frac{\sum_{i=1}^{i=k} \text{weighted proportion outgoing}_i}{\text{total num. active days}} \tag{5.2}$$

where i is a given day in the k-day time period, num. active hours$_i$ is the amount of active hours in day i and weighted proportion outgoing$_i$ is the value of the feature in day i.

An overview of the aggregation methods for the different features can be seen in Table 9.1 in Appendix 9.1.

Normalisation

Each graph has an attribute matrix of size $N \times 16$. These matrices are column-normalised in a pre-processing step such that each feature has a range from 0 to 1. This helps to ensure that the distance between points is more consistent across different dimensions, which can improve the performance of some models and speed up the convergence of optimisation algorithms like gradient descent by reducing the scale of the optimisation space. In addition, column normalisation can improve interpretability by providing a sense of the relationship between the feature values of all the users. The normalisation is done by min-max scaling the features.

5.3 Generating Synthetic Data

Hyper-parameter tuning for unsupervised methods is typically difficult due to the absence of ground truth labels. In our data set, we confirmed that 346 out of 187,034 active users were anomalous (0.18%). However, these users are not active during the same time periods, which presents an additional challenge when tuning the model. This means that when testing models on certain weeks, the number of true positives may be small, making it difficult to effectively tune the model. To address this issue, we propose generating synthetic attributed graphs to more reliably validate our models.

As mentioned in section 3.2, researchers often distinguish between structural and contextual outliers in graphs. They seek to find nodes that behave very differently from other nodes in

various different ways, whether that be forming certain clusters with other outliers or having a large, but very sparse network. However, in our setting, we are interested in finding nodes that exhibit a specific type of anomalous behaviour, such as participating in sexting and showing signs of potential grooming behaviours. This anomalous behaviour can be observed in both the topology of the node's connections and the node's attributes, such as online activity and message frequency. While methods for injecting synthetic outliers are available (e.g. [Liu+22a]), our focus is on identifying nodes that exhibit this specific anomalous behaviour, rather than finding nodes that simply differ from others in various ways.

Hence, we propose another approach to inject outliers to emulate the real anomalous users who are sexting and show signs of grooming.

5.3.1 Sampling from the anomalous distribution

In order to create synthetic anomalies, both the topology of the graph and node attributes must be considered. To simulate the attributes of already established anomalous users, we create a data set that contains the daily features of all confirmed anomalous users. For each of the 121 days, we find which anomalous users were active in that day and compute their respective daily features. The daily anomalous features are aggregated into either seven day or three day split in the same manner as before.

Alongside each users 16 frequency based features, their in and out degree are also included to incorporate topological information, which is used to change the topology of the synthetic outliers. The anomalous data set is of size $10,261 \times 18$ (16 node attributes and 2 topological attributes).

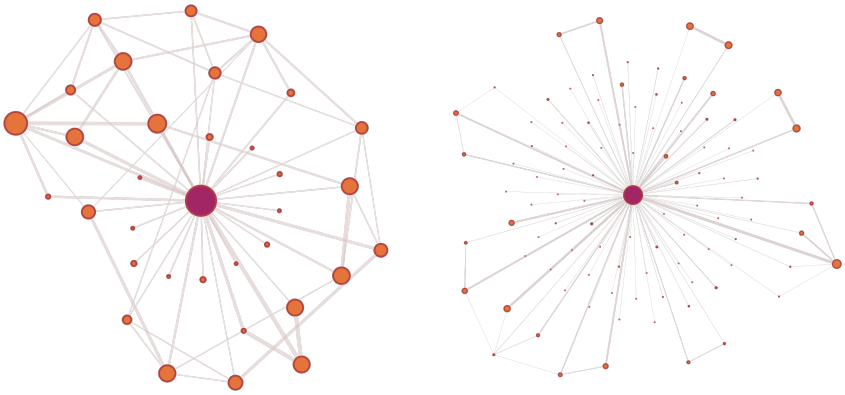
To the best of our knowledge, an approach for generating synthetic combined outliers in attributed graphs has not been proposed before. PyGOD has implemented methods for injecting attribute and structural outliers separately, but not for combining the two types of outliers.

5.3.2 Injecting anomalous users

The ratio of outliers to regular nodes in each synthetic data set is defined as 2.5%. This was proposed before manual evaluation was performed and without statistical tests on the expected percentage, so it is possible that a larger proportion of the population consists of outliers.

To sample the anomalous users, we randomly select 2.5% of all users in each train set without replacement. In order to avoid hyper-parameter tuning on real anomalous users that may also be present in the test set, we identify all anomalous users in each train set and they are included in the 2.5%. We denote the set of anomalous users to be injected as $\mathcal{V}_{anom}$.

For each anomalous user in $\mathcal{V}_{anom}$, we sample 16 new, anomalous attributes and new in- and out-degrees from the anomalous distribution. We start by replacing the old features with the new, sampled ones. Our graph instances are defined by an attribute matrix, $\mathbf{X}$, and an edge index, $\mathcal{E}$, containing all unique edges (or conversations) for a given week. Since the labels of the



(a) *Ego graph for a non-anomalous user in the time period from Data7.*

(b) *Ego graph for a anomalous user in the time period from Data7.*

Figure 5.2: *1-hop ego graphs of a normal node and an anomalous node. The size of the nodes represents the weighted degree of a particular user, which is used only for visualisation purposes. It is worth noting that the anomalous user does not tend to have very long conversations.*

users are encoded to match the node ordering in $\mathbf{X}$, injecting the new features is straightforward. For example, if user 21452 is to be anomalous, we replace the vector at index 21452 of $\mathbf{X}$ with the new, anomalous features.

Changing the edge index is a bit more complex. Along with the 16 new features, the new in and out degree of the user must also be reflected in the graph in terms of adding and removing edges.

For each user $u \in \mathcal{V}_{anom}$, we start by altering its in-edges. The difference between the user's old and new in-degree, $\Delta in = in_{before} - in_{after}$, is calculated. If Δin is positive, Δin edges are randomly sampled and removed from the graph. The same process is applied for out-edges. If Δin is negative, new in-edges are sampled for the new anomalous node u. This is done by firstly determining the set of illegal nodes that the anomalous node already has in-edges to as well as the anomalous node itself, denoted as $\mathcal{V}_{u,illegal} = \mathcal{V}_{u,in} \cup \{u\}$. To ensure that exactly Δin edges are added, the illegal nodes should not be sampled.

If we sampled all new nodes completely random, the clustering coefficient of node u would become very small since there is a smaller chance that the in-nodes also are connected. If this coefficient is too small, it might be too easy for the model to detect u as an outlier. Instead we draw inspiration from Figure 5.2, where 1-hop ego graphs for a confirmed normal and anomalous user are visualised. The ego graph for the normal user displays more clustering than the ego graph for the anomalous user, who tends to connect to users more randomly. Our approach will try to emulate the element of clustering observed in the anomalous users.

To maintain some element of clustering, the sampling of new edges are done in the following way. The ego-graph of u is created, $\mathcal{G}_u(\mathcal{V}_u, \mathcal{E}_u)$ with its 3-hop neighbours, and the legal nodes are sampled from this ego-graph. If $|\Delta in| \leq |\mathcal{V}_{u,legal}|$, we can randomly sample $|\Delta in|$ nodes

from the ego-graph, however, if $|\Delta in| > |\mathcal{V}_{u,legal}|$, we randomly sample the remainder of the nodes from the whole graph such that $|\Delta in|$ new in-edges have been added. The same process is applied for out-edges.

In this way, the sampling of new in-nodes and out-nodes is still somewhat random, since a potential 3-hop neighbour is quite far away, but the network of node u will still maintain some element of clustering.

Pseudo code is written for this procedure in Algorithm 2. The same procedure is implemented for injecting out-edges, whose pseudo code can be found in Appendix 9.3.1.

Algorithm 1 Injecting in-edges for a single anomalous user in the graph

Inputs: $\mathcal{G}(\mathcal{V}, \mathcal{E}), u, \mathcal{G}_u(\mathcal{V}_u, \mathcal{E}_u)$ ▷ *Entire graph $\mathcal{G}$, user node and ego graph for node u*
Output: Updated $\mathcal{G}$

$in_{after} \leftarrow$ sampleFeatures() ▷ *Sample new anomalous in degree*
$\Delta in \leftarrow in_{before} - in_{after}$
if Δin is positive **then**
    $\mathcal{E}_{remove} \leftarrow$ Random sample Δin edges from $\mathcal{E}$
    $\mathcal{E}_u \leftarrow \mathcal{E}_u - \mathcal{E}_{remove}$
else if Δin is negative **then**
    $\mathcal{V}_{u,illegal} \leftarrow \mathcal{V}_{u,in} \cup \{u\}$
    $\mathcal{V}_{u,legal} \leftarrow \mathcal{V}_u - \mathcal{V}_{u,illegal}$
    if $|\Delta in| \leq |\mathcal{V}_{u,legal}|$ **then**
        $\mathcal{E}_{add} \leftarrow$ Randomly sample $|\Delta in|$ nodes $\in \mathcal{V}_{u,legal}$ and create edges to u
        $\mathcal{E}_u \leftarrow \mathcal{E}_u \cup \mathcal{E}_{add}$
    else
        $\mathcal{E}_{add,in} \leftarrow$ Sample all nodes $\in \mathcal{V}_{u,legal}$ and create edges to u
        $toAdd \leftarrow |\Delta in| - |\mathcal{V}_{u,legal}|$
        $\mathcal{E}_{add,random} \leftarrow$ Randomly sample $toAdd$ nodes $\in \mathcal{V}$ and create edges to u
        $\mathcal{E}_u \leftarrow \mathcal{E}_u \cup \mathcal{E}_{add,in} \cup \mathcal{E}_{add,random}$
    end if
else
    pass
end if

5.3.3 Isolated nodes

One problem that can arise when creating synthetic graph data is that a node u in $\mathcal{G}$ may only have a single edge, $e_{u,v}$. If v is randomly selected to be anomalous and requires the removal of in-edges, the edge $e_{u,v}$ may be removed, causing u to become an isolated node with no edges in $\mathcal{G}$. However, u still has an attribute vector in $\mathbf{X}$. Since isolated nodes do not contribute to the social network, we remove them and their attributes from $\mathbf{X}$. The percentage of isolated nodes compared to all active users in the graph is usually very small, so removing them will not have a significant impact on the graph. More information about the synthetic data sets can be found in the results section.

5.4 Mini-batching on Graphs

Mini-batching is a common practice for working with neural networks on large datasets. It involves grouping multiple samples together into a mini-batch, which can make training more efficient and reduce the memory requirements for storing the entire dataset.

While it is relatively straightforward to batch fixed-shaped tensor inputs, graphs can have variable lengths for both nodes and edges and may be very sparse. Hamilton et al. [HYL17] propose a mini-batching algorithm to address these challenges.

A K-layer GNN generates embeddings of a given node using its K-hop neighbourhood structure and features. This idea is utilised by sampling M different nodes along with their K-hop neighbourhood. The mini-batch then consists of M computational graphs that are smaller than the entire graph. If K is not too big and the graph is generally sparsely connected, this means that relatively little information and memory is needed. Figure 5.3 visualises this type of mini-batch. However, the computation graphs become exponentially large with respect to the layer size K and it explodes when it hits a hub node.

The computational graphs can be constructed by sampling at most H neighbours at each hop instead. The neighbours can be sampled in various ways either randomly or by using a way to weight the most important neighbours. This prunes the graphs such that the computation of node embedding becomes more efficient. Although the computation graphs still grows exponentially in regards to K, the fan out of the graphs will be upper bounded by H. This approach is known as neighbour sampling and will be utilised for the mini-batching in this thesis.

There is a trade-off in sampling a value for H, since a smaller H leads to more efficient aggregations, but furthermore results in a more unstable training due to the higher variability. In terms of computation time, increasing the model architecture with one GNN layer will yield a computation time that is H times more expensive.

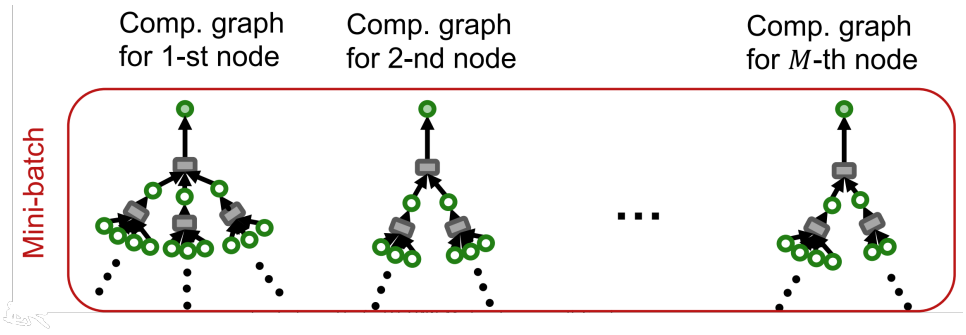


Figure 5.3: *Illustration from [Les22] which visualises the smaller computational graphs in the neighbourhood sampler*

5.5 Baseline Model

The MLPAE model described in this section is considered a baseline model because it only uses the attribute matrix $\mathbf{X}$ for anomaly detection, rather than considering the graph structure represented by $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{X})$.

5.5.1 MLPAE: Multi-Layer Perceptron and Auto-Encoder

In 2014, Sakurada et al. proposed the use of auto-encoders with nonlinear dimensionality reduction for unsupervised anomaly detection [SY14]. The model, referred to as MLPAE in [Liu+22a], is a vanilla auto-encoder with a **M**ulti**L**ayer **P**erceptron (MLP). The MLP serves as both the encoder and decoder. MLPAE only considers the attribute matrix $\mathbf{X}$ to compute an anomaly score for each user.

The model compresses each attribute vector $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ into a lower dimensional latent subspace to reproduce the reconstructed attribute vectors $\{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N\}$.

Activation of each unit i in layer k is given by the following:

$$a_i^{(k)} = \sigma\left(\sum_{j=1}^{n} W_{i,j}^{(k-1)} a_j^{(k-1)} + b_i^{(1)}\right) \tag{5.3}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weights and biases respectively and σ is the non-linear activation function.

The first layer is the input, $\boldsymbol{a}^{(0)} = \mathbf{X}$ and for the output layer, the reconstructions are computed, $\boldsymbol{a}^{(3)} = \hat{\mathbf{X}}$.

Anomaly Score and Loss function

The anomaly score for each user is the MSE loss of original and reconstructed attribute vector:

$$score_u = \frac{1}{D}\|\mathbf{x}_u - \hat{\mathbf{x}}_u\|_2^2 \tag{5.4}$$

where D is the number of features. The MLPAE loss is the average of all anomaly scores:

$$\mathcal{L} = \frac{1}{ND}\sum_{u=1}^{N}\|\mathbf{x}_u - \hat{\mathbf{x}}_u\|_2^2 \tag{5.5}$$

5.6 Graph Auto-Encoder Anomaly Detection Models

The following models are all Graph Auto-encoders, who utilise both the attribute matrix $\mathbf{X}$ and adjacency matrix $\mathbf{A}$ for unsupervised anomaly detection.

To understand how auto-encoders can be used with graph data, the notion of a *graph encoder* and *graph decoder* is introduced.

Graph Encoder

Definition 17. *A **graph encoder** is a function that maps nodes $u \in \mathcal{S}$ to vector embeddings*

$\boldsymbol{z}_u \in \mathbb{R}^d$, *where* $\boldsymbol{z}_u$ *corresponds to the embedding for node* $u \in \mathcal{V}$ [Ham20].

There are many ways of defining the encoder function. The approaches discussed in the following section use three different types of encoders based on MLP, GCN, and GAT layers to create node embeddings.

The decoder takes the generated embeddings as input and reconstructs information about the specific node's neighbourhood. Some decoders predict the set of neighbours for the given node, while others might reconstruct the node's row in the graph adjacency matrix. It is standard practice to define *pairwise decoders*.

Pairwise Graph Decoder

Definition 18. *A* ***pairwise graph decoder*** *is a function that maps pairs of embeddings* $\boldsymbol{z}_u \in \mathbb{R}^d, \boldsymbol{z}_v \in \mathbb{R}^d$ *to a new embedding that corresponds to a graph-based similarity measurement*

They are designed to reconstruct the relationship or similarity between pairs of nodes. Given a graph-based similarity measure $\mathbf{S}[u, v]$ and a pair of embeddings $(\mathbf{z}_u, \mathbf{z}_v)$, the auto-encoder tries to optimise the encoder and decoder so that the difference between the reconstruction and the input is minimised. This can be expressed as:

$$\text{DEC}((\text{ENC}(u), \text{ENC}(v))) = \text{DEC}(\mathbf{z}_u, \mathbf{z}_v) \approx \mathbf{S}[u, v] \tag{5.6}$$

There are many different ways that $\mathbf{S}[u, v]$ can be defined. By letting $\mathbf{S}[u, v] = \mathbf{A}[u, v]$, the measure would be whether two nodes are neighbours in the graph for instance. Other ways of defining $\mathbf{S}[u, v]$ includes utilising centrality measurements.

The objective of the GAE is to minimise the reconstruction loss $\mathcal{L}$ over a set of node pairs in $\mathcal{E}$:

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell(\text{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u, v]) \tag{5.7}$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ *is the loss function, which is as a measurement of the difference between* $DEC(\boldsymbol{z}_u, \boldsymbol{z}_v)$ *and* $\boldsymbol{S}[u, v]$.

Fig. 5.4 visualises a GAE structure for learning node embeddings.

5.6.1 GCNAE: Graph Convolutional Network and Auto-Encoder

The GCNAE model is the graph-equivalent to MLPAE and aims to reconstruct the attribute matrix $\mathbf{X}$. The model is simplified version of Yuan et al.'s GUIDE [Yua+21], which uses the GCNAE structure to reconstruct $\mathbf{X}$ and a GNAAE (**G**raph **N**ode **A**ttention network + **AE**) to reconstruct higher order structures.

It takes as input $\mathbf{X}$ and adjacency matrix $\mathbf{A}$ and utilises graph convolutional layers as both the encoder and decoder with the propagation rule defined in section 2.3.4:

$$\mathbf{H}^{(k)} = \sigma\left(\bar{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}\right) \tag{5.8}$$

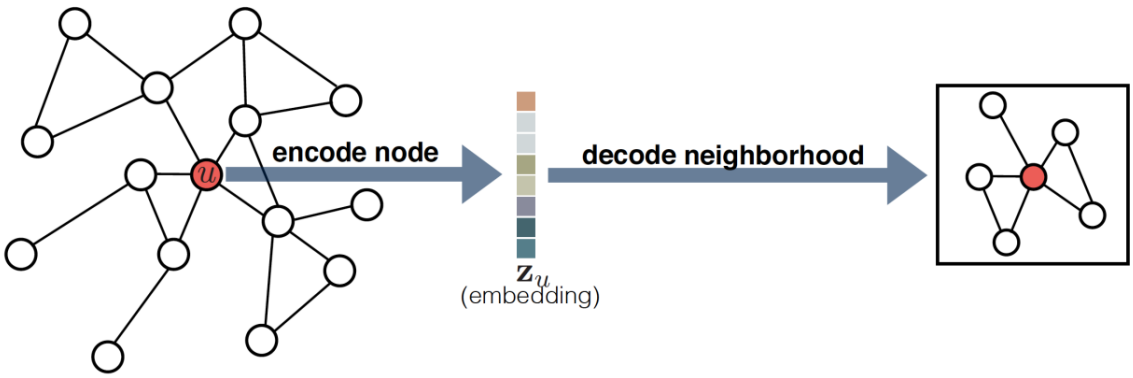


Figure 5.4: *Illustration from [Ham20] which describes how the graph auto-encoder outputs a reconstruction of the node neighbourhood.*

Where $\bar{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ denotes the symmetrical normalisation on the adjacency matrix $\mathbf{A}$ with added self loops.

The encoder aims to learn node embeddings by aggregating 3-hop neighbourhood information:

$$\mathbf{H}^{(1)} = \sigma_{\text{Relu}}\left(\bar{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}\right) \tag{5.9}$$

$$\mathbf{H}^{(2)} = \sigma_{\text{Relu}}\left(\bar{\mathbf{A}}\mathbf{H}^{(1)}\mathbf{W}^{(1)}\right) \tag{5.10}$$

$$\mathbf{Z} = \mathbf{H}^{(3)} = \sigma_{\text{Relu}}\left(\bar{\mathbf{A}}\mathbf{H}^{(2)}\mathbf{W}^{(2)}\right) \tag{5.11}$$

It compresses the node attributes and network topology into a low-dimensional latent representation, $\mathbf{Z}$. To reconstruct the attribute matrix, the decoder leverages a GCN layer to map $\mathbf{Z}$ into the original high-dimensional subspace:

$$\hat{\mathbf{X}} = \sigma_{\text{Relu}}\left(\bar{\mathbf{A}}\mathbf{Z}\mathbf{W}^{(3)}\right) \tag{5.12}$$

The top half of Figure 5.5 visualises the GCNAE model with the 3-layer attribute encoder and 1-layer attribute decoder.

The anomaly score and loss function of GCNAE is the same ones defined for MLPAE in eq. 5.4 and 5.5 respectively.

5.6.2 DOMINANT: Deep Anomaly Detection on Attributed Networks

The DOMINANT model proposed by Ding et al. [Din+19] in 2019 can be seen as an extension of the GCNAE model. The attribute encoder of DOMINANT is equivalent to that of the GCNAE model, however instead of only reconstructing the attribute matrix, the model also aims to reconstruct the adjacency matrix $\mathbf{A}$, to capture structural anomalies. Figure 5.6 visualises the framework of the DOMINANT model.

Attribute Auto-Encoder

The model creates an encoding of the graph by explicitly modelling the topological structure as well as the nodal attributes with three GCN layers as seen in Equation 5.11. This compresses the

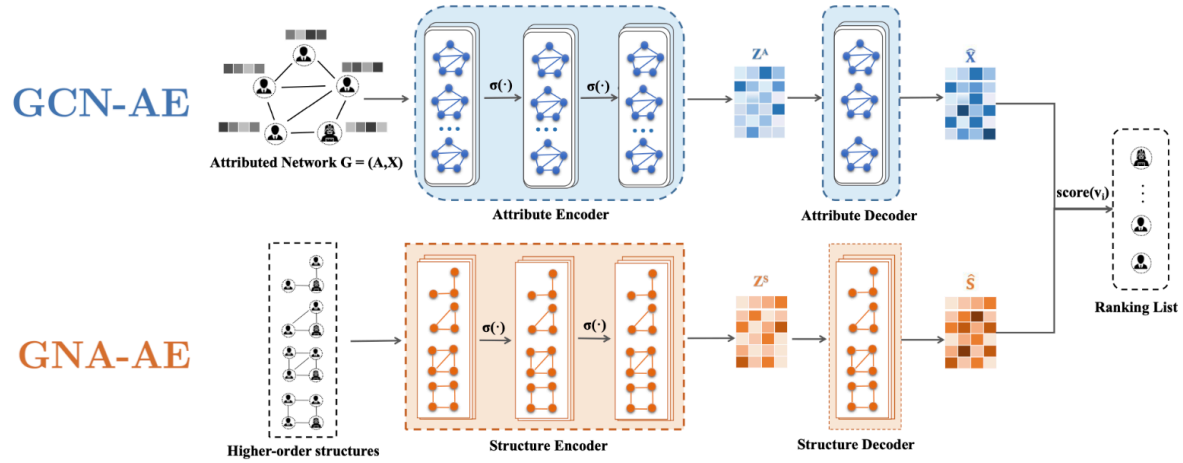


Figure 5.5: *Illustration from [Yua+21] which visualises the GCNAE as part of their GUIDE model.*

input to a lower-dimensional embedding representation, $\mathbf{Z}$, which the decoder uses to reconstruct $\mathbf{X}$ and $\mathbf{A}$ separately. $\mathbf{X}$ is reconstructed by a one-layer GCN attribute decoder in the same manner as the GCNAE:

$$\hat{\mathbf{X}} = \sigma_{\text{Relu}}\left(\bar{\mathbf{A}}\mathbf{Z}\mathbf{W}^{(3)}\right)$$

Structure Auto-Encoder

Furthermore, Ding et al. motivate the use for a structure decoder by arguing that if the structural information of a certain node can be approximated well, the node has a low probability of being anomalous contrary to nodes whose connectivity patterns cannot be well reconstructed. This would imply that the nodes structural information does not conform to the patterns of majority of normal nodes [Din+19]. The decoder computes the inner product of two node embeddings to predict whether there is a link between each pair of two nodes:

$$p\left(\widehat{\mathbf{A}}_{u,v} = 1 \mid \mathbf{z}_u, \mathbf{z}_v\right) = \sigma_{\text{sigmoid}}\left(\mathbf{z}_u, \mathbf{z}_v^\top\right) \tag{5.13}$$

where $\sigma_{\text{sigmoid}}(\cdot)$ *is the logistic sigmoid function.*

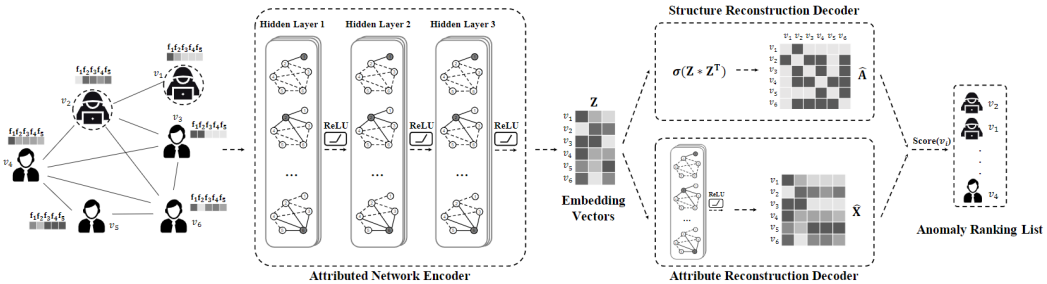


Figure 5.6: *Illustration from [Din+19] which visualises DOMINANT model framework*

Finally, the adjacency reconstruction can be computed by applying this idea over all node embeddings:

$$\widehat{\mathbf{A}} = \sigma_{\text{sigmoid}}\left(\mathbf{Z}\mathbf{Z}^{\top}\right) \tag{5.14}$$

One of the earliest methods that perform link prediction in this way was the GAE (graph auto-encoder) model from Kipf and Welling [KW16], which is the non probabilistic version of their VGAE (variational GAE).

Loss function

The objective function is the weighted reconstruction loss of both the structure and nodal attributes:

$$\mathcal{L} = \alpha\|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 + (1-\alpha)\|\mathbf{A} - \hat{\mathbf{A}}\|_F^2 \tag{5.15}$$

α is an adjustable hyper-parameter which balances the structure reconstruction loss with the attribute reconstruction loss. Given $\alpha = 1$ the model will not use the structural information of the graph and given $\alpha = 0$ the model will not use the attribute information.

The reconstruction loss of the attribute and structure serves as the anomaly score for each user:

$$score_u = \alpha \left\|\mathbf{x}_u - \hat{\mathbf{x}}_u\right\|_2^2 + (1-\alpha)\left\|\mathbf{a}_u - \hat{\mathbf{a}}_u\right\|_2^2 \tag{5.16}$$

5.6.3 AnomalyDAE: Dual Anomaly Detection on Attributed Networks

Fan et al. [FZL20] proposed in 2020 the AnomalyDAE model, which is a deep learning framework for **Anomaly** detection through a **D**ual **A**uto**E**ncoder. Different from DOMINANT which employs a shared encoder for both the attributes and structure, AnomalyDAE defines a unique auto-encoder to reconstruct the structure and attributes respectively. Furthermore, unlike the previous GCN-based models, AnomalyDAE leverages a graph attentional (GAT) based GAE framework to capture more complex interactions between the topology of the graph and nodal attributes to reconstruct the structural information.

Liu et al. [Liu+22a] also refers to AnomalyDAE as an improved version of DOMINANT. Figure 5.7 visualises the structure of the AnomalyDAE model.

Structure Auto-Encoder

The structure auto-encoder maps the original high dimensional attributes $\mathbf{X}$ into a low dimensional latent representation $\widetilde{\mathbf{Z}}^{\mathcal{S}}$ by utilising a fully connected layer with a non-linear activation function σ:

$$\widetilde{\mathbf{Z}}^{\mathcal{S}} = \sigma\left(\mathbf{X}\mathbf{W}^{\mathcal{S}_{(1)}} + \boldsymbol{b}^{\mathcal{S}_{(1)}}\right) \tag{5.17}$$

where $\boldsymbol{W}^{\mathcal{S}_{(1)}}$ *and* $\boldsymbol{b}^{\mathcal{S}_{(1)}}$ *are the weights and biases learned by the encoder.*

Then, a graph attentional layer [Vel+18] detailed in section 2.3.5 is employed to aggregate

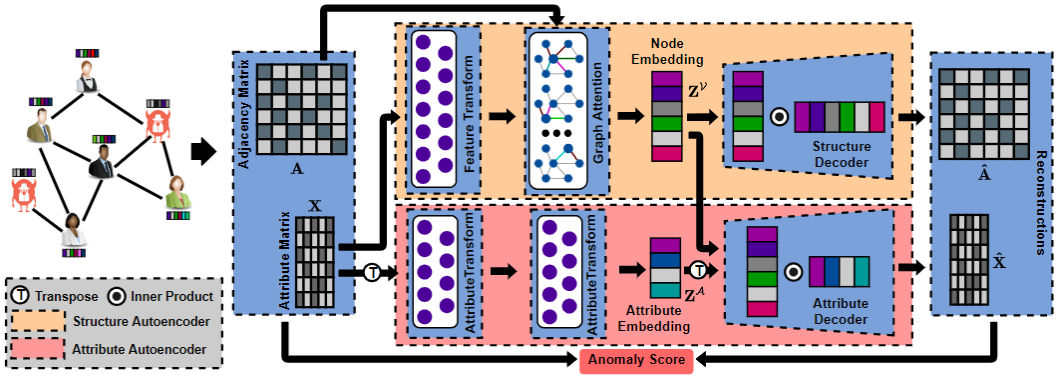


Figure 5.7: *Illustration from [FZL20] which visualises the AnomalyDAE model framework*

neighbourhood information by applying a shared attentional mechanism on the nodes:

$$a_{uv} = attn\left(\widetilde{\mathbf{z}}_u^{\mathcal{S}}, \widetilde{\mathbf{z}}_v^{\mathcal{S}}\right) = \sigma\left(\mathbf{a}^T\left[\mathbf{W}^{\mathcal{V}(2)}\widetilde{\mathbf{z}}_u^{\mathcal{S}} \oplus \mathbf{W}^{\mathcal{V}(2)}\widetilde{\mathbf{z}}_v^{\mathcal{S}}\right]\right) \tag{5.18}$$

where a_{uv} denotes the importance of node u to node v.

The importance scores are normalised through the softmax function:

$$\alpha_{uv} = \text{softmax}_v\left(a_{uv}\right) = \frac{\exp\left(a_{uv}\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(a_{uv'}\right)} \tag{5.19}$$

where information about neighbourhood sizes, $\mathcal{N}(u)$ is provided by the adjacency matrix $\boldsymbol{A}$.

The structural embedding for node u can be obtained as a weighted sum based on the learned importance weights:

$$\mathbf{z}_u^{\mathcal{S}} = \sum_{v \in \mathcal{N}(u)} \alpha_{uv} \cdot \widetilde{\mathbf{z}}_v^{\mathcal{S}} \tag{5.20}$$

Finally, after computing all node embeddings, $\mathbf{Z}^{\mathcal{S}}$, the structure decoder computes the reconstruction of the original network structure, $\mathbf{A}$ in the same manner as DOMINANT:

$$\hat{\mathbf{A}} = \sigma_{\text{sigmoid}}\left(\mathbf{Z}^{\mathcal{S}}\left(\mathbf{Z}^{\mathcal{S}}\right)^T\right) \tag{5.21}$$

Attribute Auto-Encoder

The attribute encoder is employed by stacking two non-linear feature transform layers to the attribute data $\mathbf{X}$ to the latent attribute embedding $\mathbf{Z}^{\mathcal{A}}$:

$$\widetilde{\mathbf{Z}}^{\mathcal{A}} = \sigma\left((\mathbf{X})^{\mathrm{T}}\mathbf{W}^{\mathcal{A}(1)} + \boldsymbol{b}^{\mathcal{A}(1)}\right) \tag{5.22}$$

$$\mathbf{Z}^{\mathcal{A}} = \widetilde{\mathbf{Z}}^{\mathcal{A}}\mathbf{W}^{\mathcal{A}(2)} + \boldsymbol{b}^{\mathcal{A}(2)} \tag{5.23}$$

where $\boldsymbol{W}^{\mathcal{A}(1)}$, $\boldsymbol{b}^{\mathcal{A}(1)}$, $\boldsymbol{W}^{\mathcal{A}(2)}$ and $\boldsymbol{b}^{\mathcal{A}(2)}$ are the weights and biases learned by two layers.

The attribute decoder utilises both the node embeddings, $\mathbf{Z}^{\mathcal{S}}$ learned from the structure encoder

and attribute embeddings $\mathbf{Z}^{\mathcal{A}}$ to compute the reconstruction of $\mathbf{X}$.

$$\hat{\mathbf{X}} = \mathbf{Z}^{\mathcal{S}} \left(\mathbf{Z}^{\mathcal{A}}\right)^{T} \tag{5.24}$$

Fan et al. argues that interactions between the network structure and nodal attributes can be jointly captured in this way. No activation function is utilised for this reconstruction, which allows for arbitrary-valued attributes.

Loss function
Similar to DOMINANT, the loss function is weighted sum of the reconstruction error of both the network structure and attributes:

$$\mathcal{L} = \alpha \|(\mathbf{X} - \hat{\mathbf{X}}) \odot \boldsymbol{\eta}\|_F^2 + (1-\alpha)\|(\mathbf{A} - \hat{\mathbf{A}}) \odot \boldsymbol{\theta}\|_F^2 \tag{5.25}$$

where α again is a parameter which controls the trade off between the structure and attribute reconstruction. However, AnomalyDAE also employs penalties on the reconstruction errors in the form of $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$, which are defined as:

$$\boldsymbol{\eta}_{u,v} = \begin{cases} 1 & \text{if } \mathbf{X}_{u,v} = 0 \\ \eta & \text{otherwise} \end{cases}, \boldsymbol{\theta}_{u,v} = \begin{cases} 1 & \text{if } \mathbf{A}_{u,v} = 0 \\ \theta & \text{otherwise} \end{cases} \tag{5.26}$$

where $\eta, \theta > 1$.

In this way, the model is penalised more on the reconstruction errors of the non-zero elements due to some missing edges or attributes in the real world [FZL20].

The anomaly score of a node u is the reconstruction error of both the network structure and node attributes with the added penalties:

$$score_u = \alpha \left\|(\mathbf{x}_u - \hat{\mathbf{x}}_u) \odot \boldsymbol{\eta}_u\right\|_2^2 + (1-\alpha) \left\|(\mathbf{a}_u - \hat{\mathbf{a}}_u) \odot \boldsymbol{\theta}_u\right\|_2^2 \tag{5.27}$$

5.6.4 AdONE

Bandyopadhyay et al. [BVM20] proposed the **D**eep **O**utlier aware attributed **N**etwork **E**mbedding (DONE) and **A**dversarial **O**utlier aware attributed **N**etwork **E**mbedding model (AdONE) models in their 2020 paper. These models are designed to explicitly handle outlier nodes in order to prevent them from significantly influencing the embedding of regular nodes in the network. This is unique to Bandyopadhyay et al.'s approach.

The implementation of the AdONE model used in this thesis is a simplified version of the original in terms of the input to the structural encoding. While the original model uses the transition matrix $\mathbf{T}$ and random walk to obtain a matrix $\mathbf{S}$ that captures higher-order proximities between the nodes, the implementation used in this thesis uses the adjacency matrix $\mathbf{A}$, like the other models.

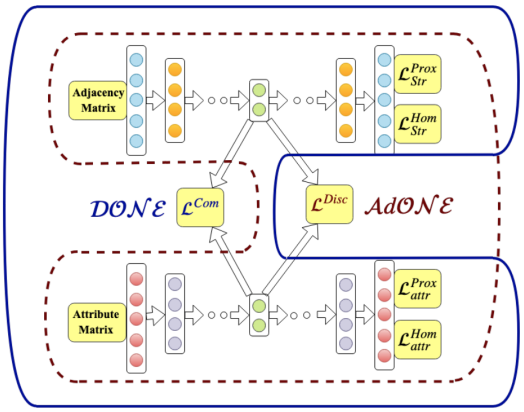


Figure 5.8: *Illustration from [BVM20] which visualises the AdONE model framework*

AdONE uses two parallel auto-encoders to generate embeddings for both the link structure and the attributes of a given node. It employs an adversarial learning approach to align the structural and attribute embeddings, using a discriminator to determine the source of an embedding (structural encoder or attribute encoder) and two generators that aim to deceive the discriminator such that it produces outputs with probabilities as close to 0.5 as possible. The use of adversarial learning serves as a regularisation method for the latent space of the structural and attribute embeddings, making them closer in a semantic sense even if they are not close in an Euclidean space. This is in contrast to other regularisation methods such as L2 regularisation.

Figure 5.8 visualises the structure of the DONE (which is AdONE without a discriminator) and AdONE models respectively.

Encoders and Decoders

Similar to the previous models, AdONE leverages separate auto-encoders to create reconstructions of $\mathbf{X}$ and $\mathbf{A}$ respectively. Both the attribute and structure encoders, Enc^a and Enc^s are composed of two dense layers, thus creating the embeddings $\mathbf{z}_u^A$ and $\mathbf{z}_u^S$ for each node u. The decoders, Dec^a and Dec^s, are composed of two dense layers and compute the reconstructions, $\hat{\mathbf{x}}_u$ and $\hat{\mathbf{a}}_u$ for each node.

Outlier Types

As mentioned previously in Section 3.2, Bandyopadhyay et al. distinguishes between three types of outliers; **structural**, **attribute** and **combined outliers**. AdONE introduces outlier scores ($\in \mathbb{R}$) for each type of outlier and denotes them as o_u^s, o_u^a and o_u^{com} respectively. They assume:

$$\sum_{u=1}^{N} o_u^s = 1, \quad \sum_{u=1}^{N} o_u^a = 1, \quad \sum_{u=1}^{N} o_u^{com} = 1, \quad o_u^s, o_u^a, o_u^{com} > 0 \tag{5.28}$$

Unlike the previous approaches, where the reconstruction loss directly serves as an outlier score for each user, AdONE computes the scores and loss in a slightly different way. Additionally,

it is weighted by the outlier scores to minimise their contribution to the learning process of regular nodes.

Proximity Loss

The proximity losses aim to preserve proximities in the network in by minimising the reconstruction loss of the structure and attribute respectively. The losses are weighted by $\frac{1}{o_u^s}$ and $\frac{1}{o_u^a}$ to reduce the contribution of nodes with a large outlier score, which helps to prevent outlier nodes from having too much influence on the resulting embeddings.

$$\mathcal{L}^{Prox\ str} = \frac{1}{N}\sum_{u=1}^{N}\log\left(\frac{1}{o_u^s}\right)||\mathbf{a}_u - \hat{\mathbf{a}}_u||_2^2 \tag{5.29}$$

$$\mathcal{L}^{Prox\ att} = \frac{1}{N}\sum_{u=1}^{N}\log\left(\frac{1}{o_u^a}\right)||\mathbf{x}_u - \hat{\mathbf{x}}_u||_2^2 \tag{5.30}$$

Homophily Loss

The homophily losses are introduced to preserve homophily in the network, based on the assumption that nodes connected by edges tend to have similar behaviour and should therefore be close in the embedding space. The losses help to ensure that the resulting embeddings reflect the underlying structure and patterns of the network.

$$\mathcal{L}^{Hom\ str} = \frac{1}{N}\sum_{u=1}^{N}\log\left(\frac{1}{o_u^s}\right)\frac{1}{|\mathcal{N}(u)|}\sum_{v\in\mathcal{N}(u)}||\mathbf{z}_u^s - \mathbf{z}_v^s||_2^2 \tag{5.31}$$

$$\mathcal{L}^{Hom\ att} = \frac{1}{N}\sum_{u=1}^{N}\log\left(\frac{1}{o_u^a}\right)\frac{1}{|\mathcal{N}(u)|}\sum_{v\in\mathcal{N}(u)}||\mathbf{z}_u^a - \mathbf{z}_v^a||_2^2 \tag{5.32}$$

The loss for each node is divided by its degree to prevent nodes with high degree from contributing disproportionately more to the overall loss. This is similar to the symmetrical normalisation scheme in the GCN layer seen in Equation 2.8. It is additionally weighted by the respective outlier score similarly to the structural proximity loss.

Discriminator Objective

The input to the discriminator is a pair of embeddings, one from the structural encoder ($\mathbf{z}_u \sim Enc^s$) and one from the attribute encoder ($\mathbf{z}_u \sim Enc^a$). The structural embedding is treated as the positive sample and the attribute embedding as the negative sample. The discriminator aims to **maximise** the following objective function:

$$\mathcal{L}^{Disc}(\Theta_D) = \frac{1}{N}\sum_{u=1}^{N}(\log(D(\mathbf{z}_u^s) + \log(1 - D(\mathbf{z}_u^a)))) \tag{5.33}$$

where Θ_D is the set of parameters to optimise and D is the discriminator function consisting of two dense layers.

Alignment Loss

The encoders aim to minimise the function shown in Equation 5.33 in order to make it more difficult for the discriminator to distinguish between the structural and attribute embeddings. There is an additional weighting with outlier scores to reduce the impact of outliers on the optimisation process.

$$\mathcal{L}^{Alg} = \frac{1}{N}\sum_{u=1}^{N}\log\left(\frac{1}{o_u^{com}}\right)(\log(D(\mathbf{z}_u^s) + \log(1 - D(\mathbf{z}_u^a)))) \tag{5.34}$$

AdONE Loss Function and Outlier Scores

The total AdONE loss function is given by a weighting of the five different loss functions and shown in the equation below:

$$\min_{\Theta,O} \mathcal{L}_{AdONE} = \alpha_1\mathcal{L}^{Prox\ str} + \alpha_2\mathcal{L}^{Hom\ str} + \alpha_3\mathcal{L}^{Prox\ att} + \alpha_4\mathcal{L}^{Prox\ att} + \alpha_5\mathcal{L}^{Alg} \tag{5.35}$$

where $\sum_{i=1}^{5}\alpha_i = 1$.

The AdONE model is trained by pre-training the auto-encoders independently and then running multiple updates of the discriminator. The auto-encoders are then updated to minimise Equation 5.35. This process is repeated until the discriminator converges to output almost equal probability for all samples.

Furthermore, the update rules for the outlier scores can be understood from the losses. The update rule for the structural outliers is defined as:

$$o_u^s = \frac{\alpha_1\left\|\mathbf{a}_u - \hat{\mathbf{a}}_u\right\|_2^2 + \alpha_2\frac{1}{|\mathcal{N}(u)|}\sum_{j\in\mathcal{N}(u)}\left\|\mathbf{z}_u^s - \mathbf{z}_v^s\right\|_2^2}{\sum_{u=1}^{N}\left(\alpha_1\left\|\mathbf{a}_u - \hat{\mathbf{a}}_u\right\|_2^2 + \alpha_2\frac{1}{|\mathcal{N}(u)|}\sum_{j\in\mathcal{N}(u)}\left\|\mathbf{z}_u^s - \mathbf{z}_v^s\right\|_2^2\right)} \tag{5.36}$$

Bandyopadhyay et al. describes that in at each iteration, the structural outlier score is proportional to the weighted sum of the reconstruction loss (Eq. 5.29) and the average structural homophily loss over the neighbourhood of the ith node. Similarly, the update rule for the contextual outliers is obtained as:

$$o_u^a = \frac{\alpha_3\left\|\mathbf{x}_u - \hat{\mathbf{x}}_u\right\|_2^2 + \alpha_4\frac{1}{|\mathcal{N}(u)|}\sum_{j\in\mathcal{N}(u)}\left\|\mathbf{z}_u^a - \mathbf{z}_v^a\right\|_2^2}{\sum_{u=1}^{N}\left(\alpha_3\left\|\mathbf{a}_u - \hat{\mathbf{a}}_u\right\|_2^2 + \alpha_4\frac{1}{|\mathcal{N}(u)|}\sum_{j\in\mathcal{N}(u)}\left\|\mathbf{z}_u^a - \mathbf{z}_v^a\right\|_2^2\right)} \tag{5.37}$$

Finally, the outlier score for the combined outlier type is given by:

$$o_u^{com} = \frac{\left\|\mathbf{z}_u^s - \mathbf{z}_u^a\right\|_2^2}{\sum_{u=1}^{N}\left\|\mathbf{z}_u^s - \mathbf{z}_u^a\right\|_2^2} \tag{5.38}$$

5.7 Evaluation Metrics

In line with the approach taken in several other papers on outlier detection in graphs, such as [Liu+22a; Din+19; FZL20; SY14; KW16], we use the ROC-AUC score, Recall and Precision @ K, and Average Precision (AP), as evaluation metrics for determining the model performance. Due to the highly imbalanced class distribution among anomalous users vs. normal users (in both the synthetic and original data), using accuracy as a metric is not preferable. This is because the model could simply place every user in the largest class and obtain a high, but very misleading accuracy [HSM19].

In addition, we use average overlap to compare the rankings produced by different models. The anomalous class is referred to as the positive class and the normal class is referred to as the negative class in this metric.

5.7.1 ROC-AUC

The Area Under the Receiver Operating Characteristic Curve (ROC-AUC) plots the true positive rate against the false positive rate at various classification thresholds and calculates the area under the resulting curve.

Each model computes an anomaly score, $\hat{y}$, for each user, which can be translated into binary class predictions by introducing a threshold θ. If the score for a node u, $\hat{y}_u$, is above the threshold ($\hat{y}_u > \theta$), the node is classified as anomalous (positive class). If $\hat{y}_u \leq \theta$, the node is classified as normal (negative class). For each threshold, the false positive rate (FPR) and true positive rate (TPR) are calculated, which normalize the false and true positives with their respective imbalanced class sizes to create the ROC curve. The ROC-AUC score is then computed as the area under this curve. It is a measure of how well the model separates the classes and represents the probability that a randomly chosen anomalous node is ranked higher than a normal node [Din+19]. A score of 1 indicates perfect separability of the classes.

5.7.2 Precision @ K

While the ROC-AUC score is a measure of overall separability, it is also of interest to investigate how well the model is at ranking the anomalous users when looking at the top K rankings. In this case, the user ranked as the most anomalous according to the model is at the top of the list, and users are considered "less anomalous" as we move down the list.

Precision @ K measures the proportion of true anomalies that the model has detected in its top K ranked nodes:

$$\text{Precision @ } K = \frac{\#\text{of anomalous users detected in the top } K}{\#\text{of ranked users total in the top } K} \tag{5.39}$$

5.7.3 Recall @ K

Recall @ K measures the proportion of how many true anomalies in the top K that the model as detected compared to the total number of ground truth anomalies:

$$\text{Recall @ } K \ = \frac{\#\text{of anomalous users detected in top } K}{\#\text{of anomalies total}} \tag{5.40}$$

Recall is a good measure when investigating whether all anomalous users are found.

Comparing precision and recall, the precision measures the models ability not to sample a negative instance as a positive one, whereas recall measures the ability to find all positive instances. If we set K equal to the number of anomalies, the precision and recall would be equal.

5.7.4 Average Precision

The Average Precision (AP) is a metric that summarizes the relationship between precision and recall, similar to how the AUC score summarizes the ROC curve. At each threshold θ, the precision and recall are calculated, creating a precision (y) and recall (x) curve. The AP is the weighted mean of precisions achieved at each threshold, where the weight is defined as the increase in recall from the previous threshold [Liu+22a]. This metric can provide a more comprehensive view of the model's performance, as it takes into account the trade-off between precision and recall at various classification thresholds.

$$\text{AP} = \sum_{\theta} \left(R_\theta - R_{\theta-1}\right) P_\theta \tag{5.41}$$

where P_θ and R_θ are the precision and recall at the θ threshold.

The AP score ranges from 0 to 1, with a higher value indicating better performance. A score of 1 indicates that all predictions are classified correctly.

5.7.5 Average Overlap

All users are ranked based on how anomalous the different models find them to be. To assess the similarity of the rankings, we use the Average Overlap (AO) metric. Unlike the Kendall rank correlation coefficient [Ken38], which is applicable only to rankings that consist of the same users and have the same length, the AO can measure the similarity of rankings regardless of their length or the users they include.

Following the notation of Webber et al. [WMZ10], we let S and T be to infinite rankings, where S_i denotes the element at rank i in list S. The set of elements from position c to position d is denoted by $S_{c:d} = \{S_i : x \leq i \leq d\}$. This also gives $S_{:d} = S_{1:d}$ and $S_{c:} = S_{c:\infty}$. At each depth d, the intersection, overlap and agreement of the two lists S and T at depth d are computed:

$$\begin{aligned} I_{S,T,d} &= S_{:d} \cap T_{:d} \quad (\textit{intersection}) \\ X_{S,T,d} &= |I_{S,T,d}| \quad (\textit{overlap}) \\ A_{S,T,d} &= \frac{X_{S,T,d}}{d} \quad (\textit{agreement}) \end{aligned}$$

The average overlap is then defined as:

$$\mathrm{AO}(S, T, k) = \frac{1}{k} \sum_{d=1}^{k} A_d \tag{5.42}$$

where k is the evaluation depth.

The score is range-normalised so that it will always have a value between 0 and 1. It is also symmetric, meaning that $\mathrm{AO}(S, T) = \mathrm{AO}(T, S)$. The AO is often used in information retrieval to evaluate the quality of search results or to compare different ranking algorithms. It is also used in other fields, such as natural language processing and recommendation systems.

5.8 Experimental Setup

The main focus of our research is to determine if a graph-based modelling approach can be effective for detecting sexual predatory users online. In order to structure the findings and discussion, we will address the following questions:

5.8.1 Performance on synthetic data sets

Are the synthetically generated anomalies more similar to the true anomalies than randomly sampled true negative nodes?

To validate the proposed anomaly injection method, we compare the similarity between true negatives, synthetically generated anomalies, and true anomalies for a given time period. The true negatives are manually validated. The comparison is done through visual examination of the marginal distribution of the attributes.

Which combination of hyper-parameters yields the best performance on the synthetic data set?

The hyper parameters of the different models are based on a grid-search over the eight training sets. The parameters shown in Table 5.6 are used for the grid-search.

For the hyper-parameters not tested in the grid search, default values were used. An epoch size of 10 was found to ensure convergence of the models, as shown in Figures 9.1 to 9.9 in Appendix 9.5 and 9.6.

Three different learning rates were tested to determine their effect on model convergence. The hidden dimensions of the embeddings were set to be less than 16, which is the number of features in the embedding set, in order to test under-saturated auto-encoders and potentially generate more meaningful embeddings. Dropout values of either zero or 0.3 were selected to evaluate the usefulness of dropout regularisation.

The number of neighbours refers to the pruning of computational trees created by mini batching. This parameter was tested to determine whether all neighbours are necessary for good performance, as using additional neighbours can slow down the modelling process.

α-values were varied to compare the performance of a model that places a higher weight on attribute reconstruction error (α=0.8) versus one that places a higher weight on structural reconstruction error (α=0.2) or one that uses a balancing approach ($\alpha = \frac{\sigma_A}{\sigma_B+\sigma_A}$).

The grid search consisted of a small number of parameters due to time constraints, and there are many more parameters that could be tested to identify the optimal set of hyper-parameters. The grid search performed resulted in a total of 5,326 model runs.

ADAM was used as the optimisation technique for learning the model parameters, and ReLU was used as the activation function for all models in this thesis.

hyper-parameters in different algorithms		
Algorithm	*hyper-parameter*	*Values*
All	weight decay	$1 \cdot 10^{-5}$, 0.01
	contamination	0.1
	hid. dim	10, 14
	dropout	0, 0.3
	learning rate	0.1, 0.05, 0.001
	epoch	10
	batch size	1000
	num. layers	4
All Graph	num. neighbours	*All*, 15, 30
DOMINANT	α	0.2, 0.8, $*$
AnomalyDAE	α	0.2, 0.8, $*$
	out dim	4
	θ	1.01
	η	1.01
AdONE	$\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$	$0.2, 0.2, 0.2, 0.2, 0.2$
MLPAE	hid. dim	8, 10, 14
	num. layers	4, 6

Table 5.6: *hyper-parameters for different algorithms.* $\alpha = *$ *refers to an automated balancing defined by* $\alpha = \frac{\sigma_A}{\sigma_A+\sigma_X}$

Do DOMINANT and AnomalyDAE models yield higher average precision when the structural reconstruction error is weighted more heavily than the contextual reconstruction error?

DOMINANT and AnomalyDAE have an α-parameter that weights the importance between the structural and the contextual reconstruction error. A sensitivity analysis is done on the α-parameter in order to test if the reconstruction of the adjacency matrix is more more important than the reconstruction of attributes for modelling the problem. The sensitivity analysis is done on the eight synthetic training sets with all parameters constant except for α.

5.8.2 Performance on real data sets

Do the models perform better than random on the real data?

To make sure that the models are performing better than random sampling, they are compared to a random sampling model on six non-synthetic hold-out test data sets where half of the test sets are three days graphs and the other half are seven day graphs.

The performance measure for the experiment is *precision@25*. The models are manually evaluated by reading through the conversations of the 25 highest ranked users for a given model in the given time period. This gives an upper bound of 675 users[1] to evaluate in the given time period. The users are evaluated by reading through their conversations until anomalous behaviour has been seen or all the conversations in the period have been read.

Is there a significant benefit from utilising seven-day graphs rather than three-day graphs?

We explore whether using seven-day graphs instead of three-day graphs leads to an improvement in *precision@25*, and we also consider the run-time of the models in the comparison.

Do the models achieve a better performance when using the graph topology explicitly?

We hypothesise that using the graph structure of the network gives additional useful information for modelling the problem and will yield a better performance than a model that does not use the graph structure. This is tested using the results achieved from the previous question and comparing MLPAE as a baseline model to GCNAE, DOMINANT, AnomalyDAE and AdONE.

[1] $(3 \text{ seven-day test sets} \cdot 4 \text{ models} + 3 \text{ three-day test sets} \cdot 5 \text{ models}) \cdot 25 \text{ users} = 675$

6 Results

The purpose of this results section is to present the findings of our study. We will first present the results from the generation of synthetic graphs and secondly the results needed to answer the questions from Section 5.8.

In order to discuss the results, we introduce the following notations:

Synthetic Positive

Definition 19. *A user that has been synthetically modified to show anomalous behaviour*

True Positive

Definition 20. *A user that was correctly identified by a model to be anomalous and validated through manual evaluation*

False Positive

Definition 21. *A user that was identified as anomalous by a model, but was determined to be a non-anomalous user in the given time period.*

True Negative

Definition 22. *A user that was determined to be non-anomalous user in the given time period.*

6.1 Generated Synthetic Nodes

Table 6.1 presents the graph statistics of the seven-day synthetic graphs. The table presents data on the eleven generated synthetic graphs. In each graph, 2.5% of the total number of nodes were changed to become anomalies. In addition, the number of nodes after the removal of isolated nodes is shown. Isolated nodes represent a small percentage (between 0.1% and 0.5%) of the total number of active users. These nodes were removed from the analysis because graph-based methods cannot be applied to them and they do not contribute to the network. The revised table also includes the updated number of active users, including those that were selected as synthetic anomalies through sampling. Additionally, the number of new edges in each graph is included.

The same statistics for synthetic three-day graphs can be found in Table 9.2 in Appendix 9.3.2.

6.1.1 Visualising Ego Graphs

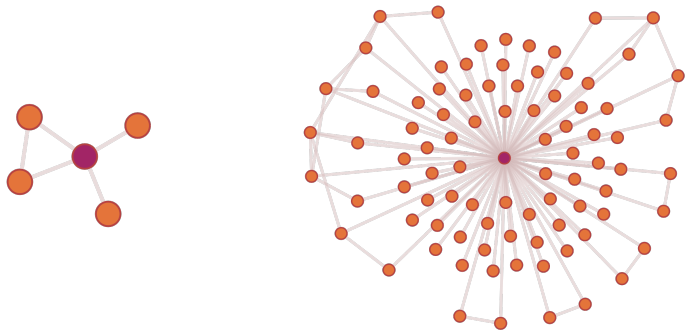
To illustrate the structure of the synthetic anomalies, we present visualisations of the 1-hop ego graphs for a true negative and an anomalous user in Figures 6.1 and 6.2, respectively. The anomalous user in Figure 6.2a was observed engaging in predatory messaging behaviours, including frequently sharing their social media with others and participating in sexting conversations.

We sought to replicate the topology seen in an anomalous node's ego-graph in a true negative user to create the synthetic anomaly. From comparing Figures 6.1a and 6.2a it is apparent that the anomalous user have a higher degree than the true negative. Although there is clustering

name	*type*	*synthetic anoms.*	*isolated nodes*	*nodes after*	*active users after*	*new edges*	*edges after*
synthData1	train	1299	135	51.798	39.952	51.422	380.262
synthData2	validation	1284	124	51.203	39.828	51.059	371.862
synthData3	train	1270	110	50.666	39.408	51.313	377.587
synthData5	train	939	90	37.468	29.331	38.871	248.350
synthData6	validation	717	73	28.601	22.686	27.837	196.891
synthData8	train	699	64	27.858	21.551	30.017	172.291
synthData9	train	678	67	27.029	21.340	26.391	180.428
synthData11	train	669	111	26.642	21.066	25.253	173.466
synthData12	train	674	35	29.188	22.745	24.663	168.770
synthData13	train	759	88	30.248	23.859	30.417	212.135
synthData14	validation	716	60	28.575	22.703	30.523	197.943

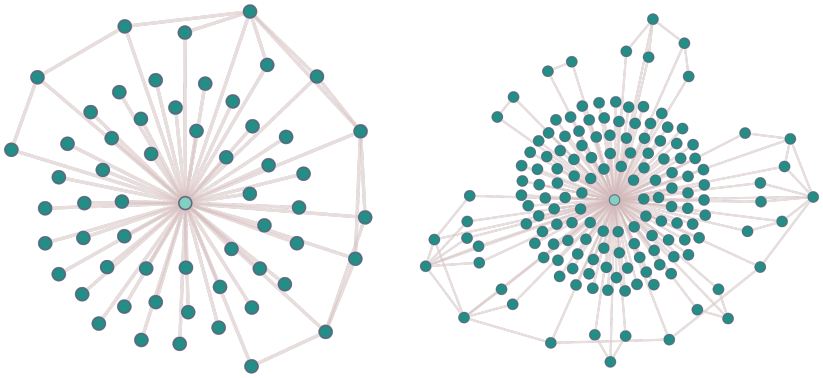
Table 6.1: *Overview of features of the different synthetic graphs*

in the 1-hop neighbourhood of the anomalous user, there is also a large amount of nodes that are not connected. In Figures 6.1b and 6.2a, there appears to be a higher structural similarity between the anomalous and synthetic user, both in terms of the degree and the clustering of their 1-hop neighbourhood. Additionally, when the anomalous user was made synthetic, it communicated with even more users who were not part of the same neighbourhoods. It is important to note that these examples do not necessarily represent the ego graphs of all synthetic anomalies. They have been presented since they offer some insight into the outcomes of our injection algorithm.



(a) *Ego Graph Before Modification* **(b)** *Ego Graph After Modification*

Figure 6.1: *Ego graphs of a normal node before (Data4) and after modification to a synthetic anomalous node (synthData4), respectively.*



(a) *Ego Graph Before Modification* **(b)** *Ego Graph After Modification*

Figure 6.2: *Ego graphs of an anomalous node before (Data4) and after modification to a synthetic anomalous node (synthData4), respectively.*

6.1.2 Visual Comparison of Synthetic Typologies

To show more generally how the topology of synthetic positives compares to the topology of the true positives, we compute the *clustering coefficient*, *in- and out-degree* of the synthetic positives, true positives, and true negatives. The synthetic and true positives are found for each model in each of the seven-day validation sets. A baseline for 75 randomly sampled and manually validated true negative users is shown for comparison. The true positives used for the visualisation are the same non-synthetic anomalies as shown in Table 6.7, and the number of samples the distributions are based upon can be located in Table 9.71 in Appendix 9.8.

Figure 6.3 shows the distributions for *node out degree* and *clustering coefficients*. The *node out degree* refers to the number of unique users a single node has written to across the whole seven days. The distribution of the *node in degree* can be seen in Figure 9.25 in Appendix 9.8.

From Figure 6.3 it can be seen that the true negative users generally have a lower out-degree than the true positive and synthetically generated users. It can furthermore be seen that the variability of the out degree is smaller for the synthetically generated users than it is for the true positives. Table 6.2 summarises the distributions shown in Figure 6.3. The distribution for the out-degree for the true negative users can be summarised by $(Q_{50\%}, \mu, \sigma) = (2.0, 6.28, 10.5)$. The median out-degree for both true positive and synthetic users is much higher than that of the true negatives. While the median for true negatives is 2.0, it ranges from 24 to 254.5 for true positives and from 76.5 to 94.0 for synthetic users. Table 6.2 also shows a much higher standard deviation for the *out-degree* distribution compared to the true negatives.

The distribution for the *clustering coefficient* of the true negative users can be summarised by $(Q_{50\%}, \mu, \sigma) = (0.0, 0.047, 0.15)$. The clustering coefficient distributions of true negative, positive and synthetic show a lot less separation. The median *clustering coefficient* of the true negatives is 0.0 while it is between 0.001 and 0.002 for the synthetic users as seen in Table 6.2.

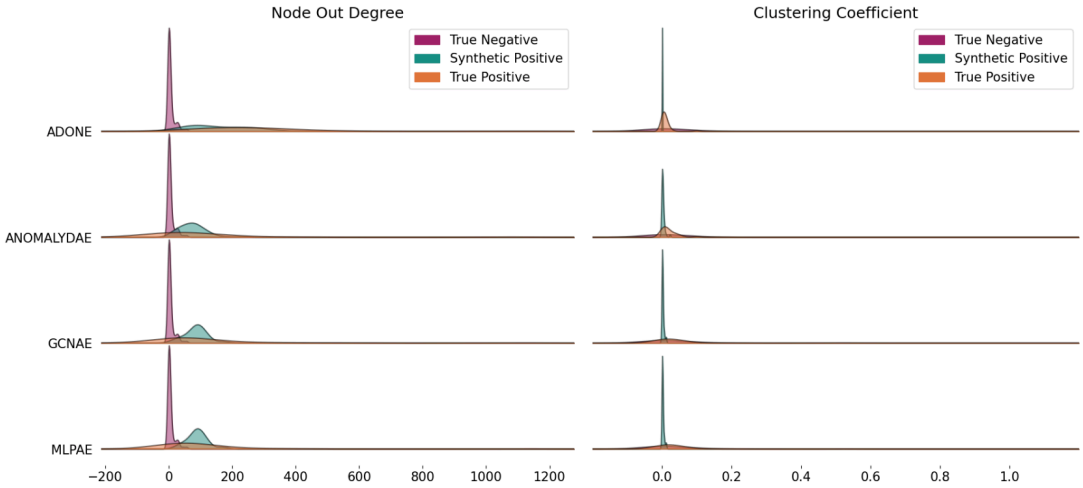


Figure 6.3: *The feature distributions for node out degree and clustering coefficient for each model in the top 25 rankings of the validation data sets*

	Node Out Degree						Clustering Coefficient					
	True Pos			*Syn Pos*			*True Pos*			*Syn Pos*		
	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ
MLPAE	54.0	101.61	178.90	92.0	89.73	31.86	0.016	0.038	0.083	0.002	0.003	0.002
GCNAE	49.5	111.47	195.23	89.5	85.9	35.34	0.016	0.038	0.084	0.002	0.003	0.002
AnomalyDAE	24.0	99.71	209.33	76.5	76.57	44.01	0.01	0.02	0.02	0.002	0.003	0.004
AdONE	254.5	261.6111	187.47	94.0	154.85	90.60	0.006	0.01	0.016	0.001	0.001	0.0003

Table 6.2: *Selected statistics on the true positive and synthetic positive distributions shown in Figure 6.3. It is worth noting that the true negatives are randomly sampled and not based on the models, so their summary statistics are calculated separately*

6.1.3 Visual Comparison of Synthetic Feature Distributions

We furthermore investigate how the attributes of the synthetic anomalies compare to the attributes of the true positives. Figures 6.4 and 6.5 show how selected features are distributed for true positives and synthetic positives found in the validation sets.

Figure 6.4 shows the distributions for *weighted max out hourly* and *weighted max in hourly*. The distributions of synthetic and true positives seems more similar to each other than they are to the true negatives based on visual comparison. The synthetic users tend to have a larger *weighted max out hourly* and *weighted max in hourly* value compared to non-synthetic users.

The majority of the weight of the true negative distribution lies within the range of [0,25] for *weighted max out hourly*, summarised by $(Q_{50\%}, \mu, \sigma) = (4.5, 12.2, 20)$. In contrast, the mean for synthetic and true positives lies between 138 and 256, as shown in Table 6.3.

In terms of the number of active hours, the mean and median are also more similar between the synthetic positives and the true positives than the negatives, which have the summary statistics of $(Q_{50\%}, \mu, \sigma) = (2.0, 6.63, 10.55)$. The *weighted proportion outgoing* feature does not show as much separation between the three distributions as the other features do. It has a median, mean and standard deviation of $(Q_{50\%}, \mu, \sigma) = (0.54, 0.59, 0.2)$ for the true negative

distribution, while both the mean and median for the true positive and synthetic distributions are around 0.54.

As for the rest of the features, the distribution plots can be found in Figures 9.26 to 9.40 in Appendix 9.8.

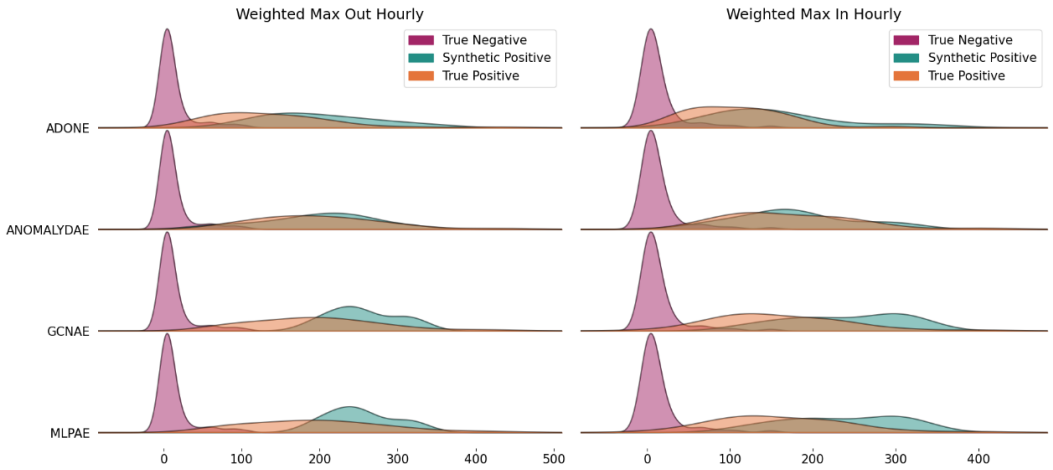


Figure 6.4: *The feature distributions for weighted maximum outgoing and in-going features for each model in the top 25 rankings of the validation data sets*

	Weighted Max Out Hourly						Weighted Max In Hourly					
	True Pos			*Syn Pos*			*True Pos*			*Syn Pos*		
	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ
MLPAE	193.0	197.05	86.76	244.0	253.0	42.22	155.0	161.46	79.48	230.0	235.42	68.46
GCNAE	192.5	197.21	82.66	245.5	255.95	43.16	152.0	160.47	79.13	242.0	239.15	68.69
AnomalyDAE	195.0	198.46	79.44	202.5	196.68	70.5	159.5	173.0	77.38	169.5	178.45	70.92
AdONE	126.5	138.72	80.15	168.0	200.43	70.28	108.0	108.39	57.97	116.0	158.71	77.58

Table 6.3: *Selected statistics on the true positive and synthetic positive distributions shown in Figure 6.4*

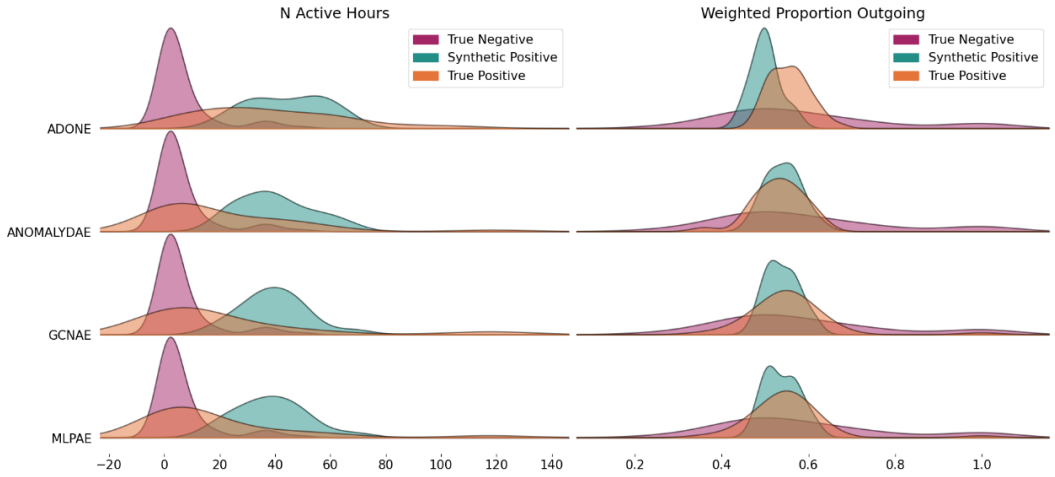


Figure 6.5: *The feature distributions for N. Active Hours and weighted proportion outgoing features for each model in the top 25 rankings of the test data sets*

	N. Active Hours						Weighted Prop. Outgoing					
	True Pos			*Syn Pos*			*True Pos*			*Syn Pos*		
	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ
MLPAE	6.0	19.64	28.74	37.0	38.58	12.78	0.55	0.55	0.09	0.53	0.54	0.04
GCNAE	6.5	21.97	32.42	38.5	39.65	11.82	0.55	0.55	0.1	0.53	0.54	0.04
AnomalyDAE	7.5	21.07	26.47	38.0	39.95	13.36	0.53	0.53	0.06	0.54	0.54	0.04
AdONE	33.0	38.28	26.7	45.0	44.43	13.59	0.55	0.56	0.05	0.5	0.5	0.03

Table 6.4: *Selected statistics on the true positive and synthetic positive distributions shown in Figure 6.5*

6.2 Grid-Search Performance

This section presents the hyper-parameter configurations that yielded the best average precision for each model. The top five configurations for each model based on additional performance metrics, as well as a graph of the top ten configurations ranked by average precision over epochs, can be found in Appendix 9.5 and 9.6. The results of the grid search shows that, considering the standard deviation across eight runs, it is difficult to determine if one model outperforms the others in terms of average precision due to overlap among the models.

In comparing the MLPAE, GCNAE, AnomalyDAE, and AdONE models[1] on the seven-day sets, we find that there is low variability in average performance among the top performing models (Table 6.5). The AnomalyDAE model has the highest average precision score at 50.52% with a standard deviation of 11.66% across the eight data sets. The standard deviation for the top performing models ranges from 11.66% to 15.46%. The best performing AnomalyDAE model

[1]The DOMINANT model was not included in the performance evaluation of the seven-day graph sets due to memory constraints on the GPU. This was partly due to the implementation but also the large size of the January graphs.

Grid-Search Results Average Precision								
Model	*Hyper-parameters*							*Metric*
	α	*dropout*	*hidden dim*	*lr*	*weight decay*	*num. neighbours*	*num. layers*	*Avr. Prec.*
MLPAE	-	0.3	8	0.1	0.01	-	6	45.23 ± 15.46 (80.38)
GCNAE	-	0	14	0.05	0.01	All	4	49.26 ± 13.38 (80.42)
AnomalyDAE	**0.8**	**0**	**14**	**0.05**	**0.01**	**30**	-	$\mathbf{50.52 \pm 11.66}$ **(79.17)**
AdONE	-	0.3	10	0.1	1e-5	15	4	33.99 ± 13.7 (49.51)

Table 6.5: *Average Precision scores and hyper-parameters for best performing seven-day runs in the hyper parameter grid-search.*

Grid-Search Results Average Precision								
Model	*Hyper-parameters*							*Metric*
	α	*dropout*	*hidden dim*	*lr*	*weight decay*	*num. neighbours*	*num. layers*	*Avr. Prec.*
MLPAE	-	0	8	0.001	0.01	-	4	59.78 ± 16.87 (87.62)
GCNAE	-	**0.3**	**10**	**0.05**	**0.01**	**15**	**4**	$\mathbf{79.90 \pm 3.22}$ **(84.54)**
DOMINANT	0.8	0.3	10	0.1	1e-5	All	4	54.56 ± 2.00 (56.3)
AnomalyDAE	0.8	0	10	0.1	1e-5	30	-	79.54 ± 4.75 (85.27)
AdONE	-	0	14	0.1	1e-5	All	4	42.16 ± 11.35 (59.20)

Table 6.6: *Average Precision scores and hyper-parameters for best performing three-day runs in the hyper parameter grid-search.*

uses an α value of 0.8, indicating that attribute reconstruction is given more weight than graph structure reconstruction.

From the results of grid-search on three-day sets shown in Figure 6.6, the best performing configurations of AnomalyDAE and DOMINANT also use a higher weight for the attribute rather than the structural reconstruction loss, similar to the results on the seven-day sets.

The best performing MLPAE model on the seven-day set used six layers, in contrast to the best performing MLPAE model on the three-day sets that used only four layers.

Interestingly, not all the best performing models are using all the possible number of neighbours in the mini-batching pruning process.

It is difficult to definitively conclude which type of data set period performs better due to the small sample size and the standard deviations, even though the mean average precision is generally higher for the three-day grid-search when comparing the scores from the three-day and seven-day sets.

6.3 Validation of Grid-Search Metrics

Table 6.7 shows the *precision@25* scores for three synthetic hold-out data sets, as evaluated both automatically and manually. The automatic evaluation measures the *precision@25* in detecting synthetically generated anomalies, while the manual evaluation is based on the manual labelling of non-synthetic nodes in the top 25 rankings of each model. The table shows a significant difference between the automatic and manual evaluations, suggesting that the models

Precision @ 25									
	Data2			*Data6*			*Data14*		
	Automatic	*Manual*	*↑ Difference*	*Automatic*	*Manual*	*↑ Difference*	*Automatic*	*Manual*	*↑ Difference*
MLPAE	16%	68%	52%	16%	84%	68%	64%	80%	16%
GCNAE	20%	68%	48%	16%	84%	68%	64%	80%	16%
AnomalyDAE	60%	80%	20%	32%	84%	52%	92%	100%	8%
AdONE	4%	48%	44%	12%	68%	56%	16%	56%	40%

Table 6.7: *Manually evaluated performance versus automatically evaluated performance on a synthetic seven-day set for precision@25*

are also ranking organic anomalies among the top 25, in addition to the synthetic ones. The performance of MLPAE and GCNAE increased by up to 68% on Data6 when using both organic and synthetic anomalies in the validation set, compared to using synthetic anomalies only. All models demonstrated an increase in performance of 50% or more on this data set when using a combination of organic and synthetic anomalies. The lowest increase was seen for AnomalyDAE in Data14, which had ranked 24 anomalies and one organic outlier in the top 25 and thereby increased the *precision@25* from 92% to 100%.

6.4 Parameter Sensitivity

The following section presents the average precision score for eight seven-day sets using different values of α. The tested values of α range from 0 to 1 with an interval of 0.05. A value of 1 indicates that only the attribute reconstruction error is considered while the graph reconstruction error is not taken into account.

Figure 6.6 shows that there is high variability in the best performance for AnomalyDAE across the eight data sets. In general, the average precision is higher for larger α-values. The figure also suggests that α-values higher than 0.8 generally perform better, meaning that an AnomalyDAE model with $\alpha = 1$ may achieve better test performance than the $\alpha = 0.8$ model used in the test and validation section. However, some lower performance scores can be observed for specific α-values on certain data sets, as seen in Figure 6.6.

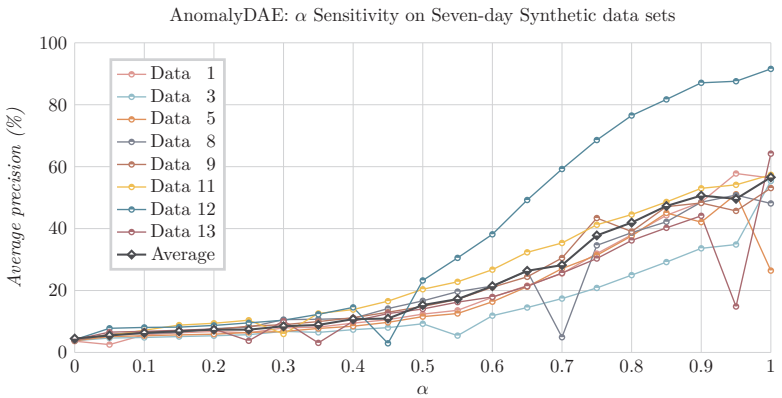


Figure 6.6: *Impact of different values of α on the average precision values of the AnomalyDAE model on seven-day synthetic data sets.*

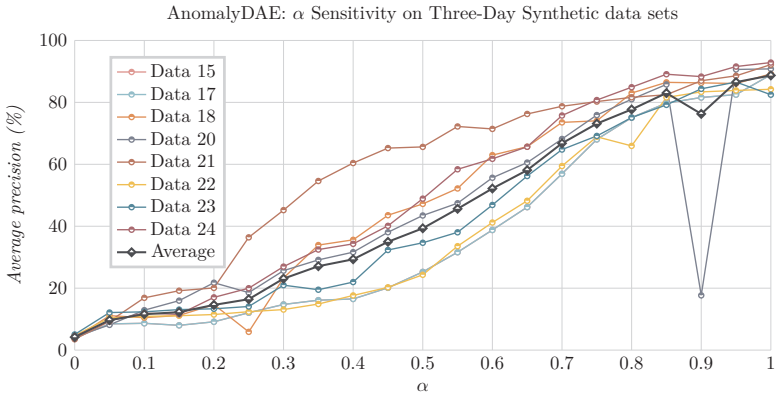


Figure 6.7: *Impact of different values of α on the average precision values of the AnomalyDAE model on three-day synthetic data sets.*

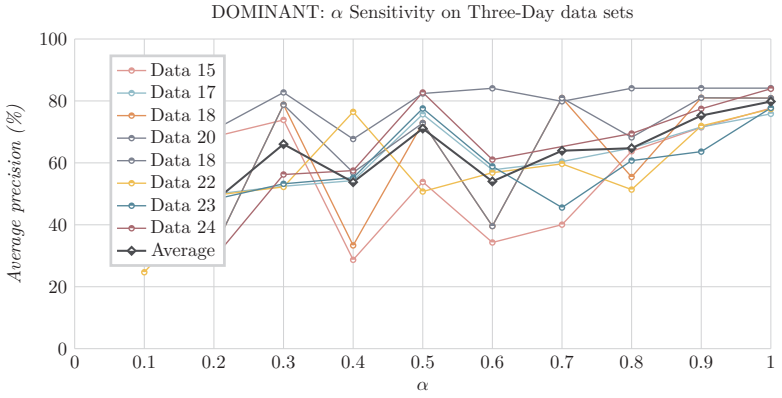


Figure 6.8: *Impact of different values of α on the average precision values of the DOMINANT model on three-day synthetic data sets.*

Figure 6.7 shows that the average precision scores for the three-day sets are generally higher than those for the seven-day sets. Less variability can be observed in the best average precision score across three-day compared to seven-day sets. While there is a general tendency for the average performance to be higher when α is higher, there is a significant drop in performance for $\alpha = 0.9$ in Data13.

The DOMINANT model seems to be more sensitive to the α parameter than AnomalyDAE with a higher volatility in the performance, as seen in Figure 6.8. However, the model generally performs better with a higher α-value similar to AnomalyDAE.

6.5 Test Performance

6.5.1 Test performance on seven-day sets

The models' performance was evaluated using *precision@25* on three non-synthetic hold-out data sets. Table 6.8 displays the *precision@25* scores, as well as the average and standard deviation across the three data sets. By randomly sampling 25 users from each data set, the models detected an average of 12% of users who violated MSP guidelines, indicating that the actual percentage of anomalous users in the data is higher than previously assumed. Among

Precision @ 25				
	Data4	*Data7*	*Data10*	average
Random	12%	20%	4%	12.0%±8.0%
MLPAE	52%	32%	40%	41.3%±10.1%
GCNAE	52%	36%	**48%**	45.3%±8.3%
AnomalyDAE	56%	**48%**	40%	48.0%±8.0%
AdONE	**60%**	**48%**	44%	**50.7%±8.3%**

Table 6.8: *Manually evaluated performance on the non-synthetic seven-day set for precision@25*

Precision @ 25				
	Data16	*Data19*	*Data25*	average
Random	8%	16%	16 %	13.3%±4.6%
MLPAE	**80%**	64%	48%	64.0%±16.0%
GCNAE	76%	**68%**	56%	66.7%±10.0%
DOMINANT	76%	**68%**	56%	66.7%±10.0%
AnomalyDAE	68%	60%	**72%**	**66.7%±6.1%**
AdONE	**80%**	52%	48%	60%±17.4%

Table 6.9: *Manually evaluated performance on the non-synthetic three-day set for precision@25*

the models, AdONE had the highest average *precision@25* score, at 50.7%. The highest score on any data set was 60%, achieved by AdONE on Data4, while the lowest score, except for the random model, was 32%, achieved by MLPAE.

6.5.2 Test performance on three-day sets

The test performance on the three-day sets has a higher overall *Precision@25* score compared to the seven-day sets. The average *Precision@25* score is the same for the DOMINANT, AnomalyDAE, and GCNAE models, however, the AnomalyDAE model performs the best when considering the variability across the three test data sets. All models perform better than random.

Average Overlap of Test Model Top 25 Rankings

The results of the study show that among the top 25 ranked users in the seven-day sets, there are 172 unique users, while in the three-day sets there are 173 unique users. This means that there is an overlap in the ranked users, since it is possible to find 300 and 375 unique users respectively. An overlap means that the models are similar in the ranking of outliers and indicates that the models model the data in a more similar fashion than if no overlap was seen.

The results shown in Figure 6.9 indicate that GCNAE and MLPAE have the highest average overlap in their top 25 rankings. MLPAE also has a high overlap with AnomalyDAE, but a low overlap with AdONE. AdONE generally does not rank users in the top 25 as similarly as some of the other models. The highest overlap is seen between MLPAE and GCNAE in Data7, where the overlap value is 0.93.

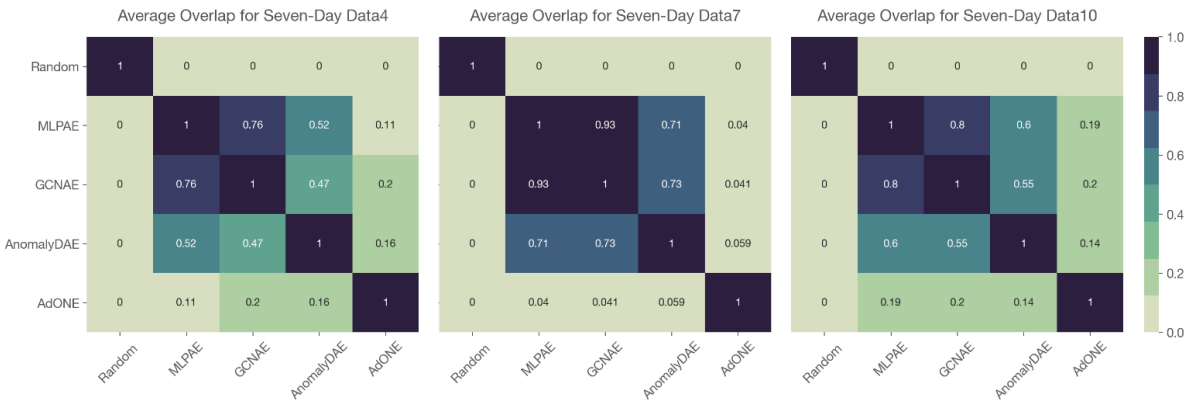


Figure 6.9: *AO of the top 25 ranked users on the seven-day test data sets*

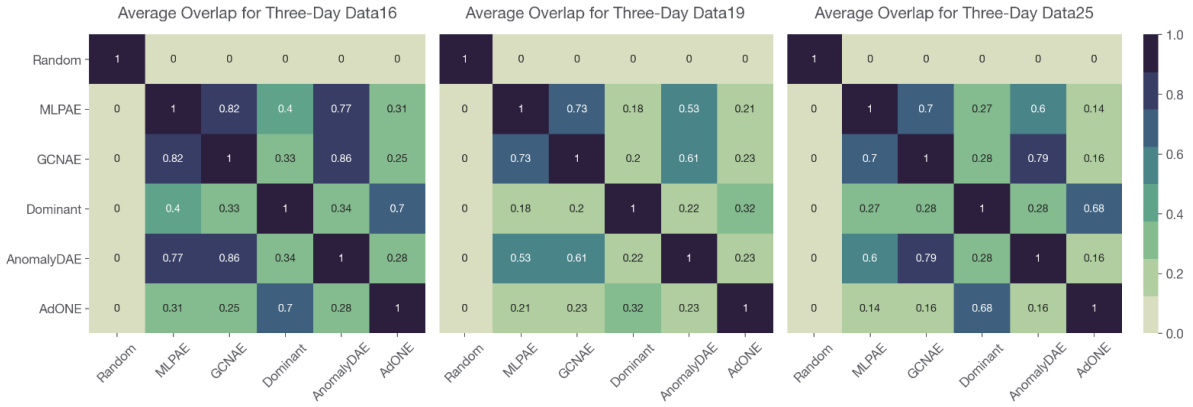


Figure 6.10: *AO of the top 25 ranked users on the three-day test data sets*

On the other hand, the average overlap between the randomly selected users and the users selected by the models is 0. Asides from this, the lowest overlap score is 0.04, as seen between MLPAE and AdONE in Data7.

For the three-day set, the average overlap measurements shown in Figure 6.10 display the same general tendencies as in Figure 6.9, with GCNAE and MLPAE having a high average overlap and AdONE having a lower average overlap with MLPAE, GCNAE, and AnomalyDAE. AdONE does, however, have a higher overlap with DOMINANT than it does with the other models, with an overlap of 0.7, 0.32, and 0.68. DOMINANT has the highest overlap with AdONE.

AnomalyDAE has a high overlap with MLPAE and GCNAE. The highest overlap seen in the three-day user rankings is between GCNAE and AnomalyDAE, with an overlap of 0.86.

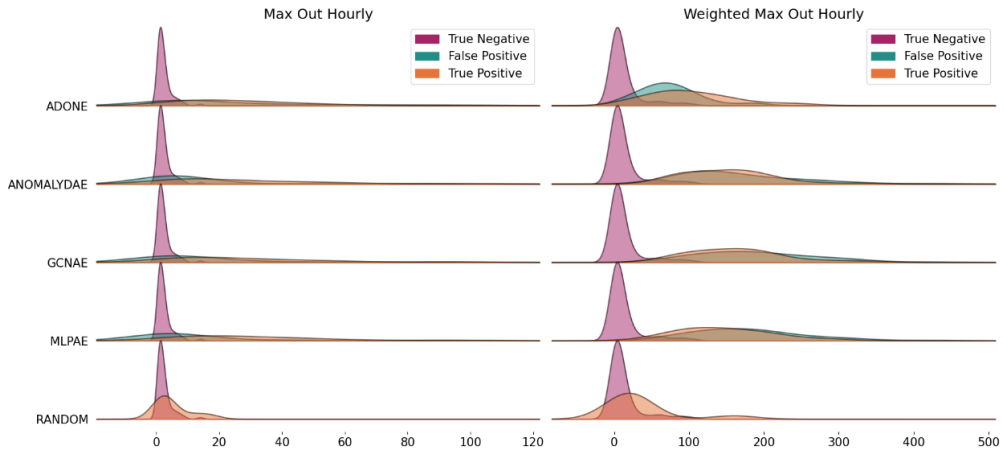


Figure 6.11: *The feature distributions for max out and weighted max out features for each model in the top 25 rankings of the test data sets*

	max out hourly						weighted max out hourly					
	True Pos			*False Pos*			*True Pos*			*False Pos*		
	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ	$Q_{50\%}$	μ	σ
Random	3.0	5.22	5.56	-	-	-	21.0	35.89	47.54	-	-	-
MLPAE	24.0	30.97	24.98	4.5	15.61	25.83	140.0	148.39	63.08	178.0	181.43	76.18
GCNAE	21.0	29.0	25.16	7.0	17.8	26.28	161.0	164.06	63.65	189.0	190.61	79.94
AnomalyDAE	19.0	28.0	25.29	5.0	14.05	22.78	153.5	153.14	59.4	143.0	167.1	79.9
AdONE	22.5	27.18	21.0	11.5	25.14	28.09	91.0	105.13	59.65	74.5	74.86	42.12

Table 6.10: *Selected statistics on the true positive and synthetic positive distributions shown in Figure 6.11*

6.5.3 Visual Comparison of Test Feature distributions

The same true negative distribution used in Figure 6.4 and 6.5 is used as a visual baseline in Figure 6.11.

From Figure 6.11 it can be seen that the true and false positive distributions are more similar to each other than to the true negatives. The summary statistics of the true negative distribution for the *weighted out hourly* is $(Q_{50\%}, \mu, \sigma) = (4.5, 12.2, 20)$. In contrast, the means for false and true positives all lie between 74 and 191, as shown in Table 6.10. Similarly, the true negative distribution for *max out hourly* $(Q_{50\%}, \mu, \sigma) = (2.0, 2.42, 2.26)$, while the means of the false and true negative distributions are above 15.

Similar figures of the other features can be seen in the Appendix 9.7 in Figures 9.10 to 9.24.

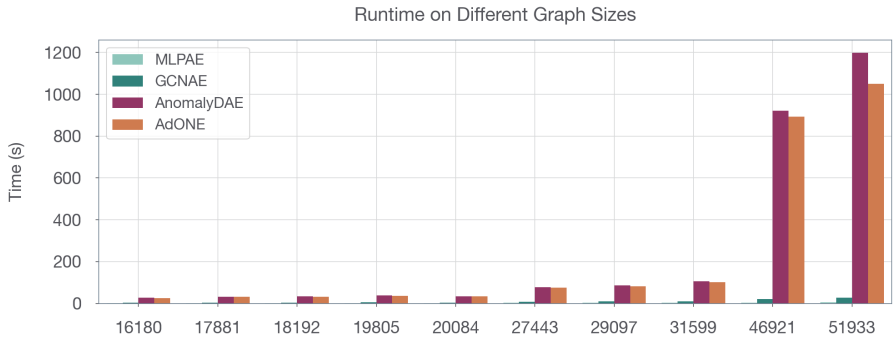


Figure 6.12: *Run-time vs graph size*

6.6 Efficiency and Scalability of Models

Figure 6.12 shows the run-time in seconds for various graphs of different sizes. The DOMINANT model is not included because it is not able to run on the GPU. AnomalyDAE and AdONE have slower performance times compared to MLPAE and GCNAE and also scale worse for larger graphs compared to MLPAE and GCNAE.

6.7 Detected Anomalies

Across the different types of data sets, 600 (seven-day validation and test-sets)[2] and 375 (three-day test-set)[3] non-unique users were ranked in the top 25, giving a total of 975 users. Of these, 443 users were found to be unique. Conversations from 395 unique users were read in total, since synthetically generated users were not manually validated. The time taken to evaluate one data set for the five models took around a working day, although evaluation of three-day sets is quicker than for seven-day sets due to shorter conversations.

Table 6.11 summarises the number of unique, detected anomalies found in each type of data set, as well as the total number of anomalies found when aggregating the three types of data set. We distinguish between new anomalies and already labelled anomalies (which were part of the original 346). A total of 527 unique anomalies were detected in the project, including the 346 anomalies labelled in the data preparation and the 181 new anomalies.

It is worth noting that the 181 new anomalies were identified when evaluating the top 25 most anomalous users per model and data set. We expect that if we had evaluated the top 50 or top 100 users, the number of detected anomalies would be higher.

[2] $25 \cdot 4 \cdot 3 \cdot 2 = 600$
[3] $25 \cdot 5 \cdot 3 = 375$

	New Anoms.	*Already Lab.*	*Total*
seven-day Test	74	7	81
seven-day Validation	64	9	73
three-day Test	82	15	97
Overall	181	29	210

Table 6.11: *Number of detected anomalies in each type of data set as well as overall.*

6.7.1 Conversations from users with highest anomaly scores

Through manual evaluation of the conversations between the top 25 highest-ranked users for each model and data set, we found that the majority of the top-ranked users had one or more of six distinct conversation types discussed in the following subsections.

Data Error Conversations

There is a data collection error that affects the sender and receiver IDs, particularly in March. In many conversations, it appears that only one user is writing to the other, but when reading through the conversations, it is clear both users are writing to each other. Table 6.12 shows an example of this.

Spamming Conversations

The models also detected users who were spamming other users. These users would often send the same message to multiple different users, and the conversations would usually only involve a few messages back and forth. Table 6.13 shows an example of spamming. Only users who consistently sent spam containing threats or bullying would be identified as anomalies.

Trading and Selling Conversations

On MSP users can trade and sell items in their inventory (clothes, accessories etc.). Often, users who promote that they are selling items in group forums, will be approached about their items. Table 6.14 shows an example of an user being approached about trading.

Sexting Conversations

The majority of detected anomalous users consistently participated in sexting conversations (defined in Section 4.2). Sexting conversations are often characterised by a high volume of messages being exchanged between the users in a short period of time, as the users are 'acting' out the role play in real time.

Snapchat Conversations

Snapchat (often abbreviated as 'sc') is a social media platform that allows users to send pictures and videos to each other. Users can set a time limit for the visibility of these photos. Users that consistently ask for other users' Snapchat handles often asks to "trade" nude pictures. There are examples of anomalous users who ask for other users' Snapchat handles in Table 6.16. Even if a user does not explicitly request nude pictures, asking for other social media accounts on MSP is still prohibited and could lead to inappropriate conversations or exchanges once the conversation is moved to Snapchat.

Sender	Time	Message
A	8:24 PM	*At least one of us needs to be happy*
A	8:24 PM	*It's fine like I said <3*
A	8:24 PM	*I might still be able to convince my dad*
A	8:25 PM	*It's still 80 for me lol how did you get it cheap*
A	8:25 PM	*Aw thank you!*
A	8:25 PM	*Wow they never gave me the offer lol*
A	8:26 PM	*On mobile it never gave me tat option lol*
A	8:27 PM	*It's fine like I said it's just life*
A	8:27 PM	*Don't worry about me lol be happy*
A	8:30 PM	*Yep normal is good lol*
A	8:31 PM	*My dad still hasn't come home for dinner yet lol and I didn't even have lunch xd I had a poptart and some Doritos*

Table 6.12: *Example of a data error conversation*

Potential Grooming Conversations

Some users displayed behaviour that not only violated MSP's guidelines, but also potentially violated laws related to age of consent. We can assume that users exhibiting behaviour similar to that shown in Table 6.17 would be considered groomers. These conversations often include requests for inappropriate images and sexually charged questions. In some instances, the potential groomer would ask the user to portray a minor when sexting.

An example of how the different models ranked the top 25 users can be seen in the Appendix 9.9.1 in Tables 9.72 and 9.76 along with comments we made while evaluating if a user showed anomalous behaviour.

Sender	Time	Message
B	04:43 AM	*Hi Please Subscribe to — on YouTube and put on post notifications on for all and like and comment on all her videos thanks :)*
C	04:43 AM	*Absolutely not.*
B	04:43 AM	*Hi Please Subscribe to — on YouTube and put on post notifications on for all and like and comment on all her videos thanks :)*
B	04:43 AM	*Hi Please Subscribe to — on YouTube and put on post notifications on for all and like and comment on all her videos thanks :)*
B	04:43 AM	*Hi Please Subscribe to — on YouTube and put on post notifications on for all and like and comment on all her videos thanks :)*
B	04:43 AM	*Hi Please Subscribe to — on YouTube and put on post notifications on for all and like and comment on all her videos thanks :)*
B	04:43 AM	*Hi Please Subscribe to — on YouTube and put on post notifications on for all and like and comment on all her videos thanks :)*
C	04:43 AM	*STOPP*
B	04:43 AM	*Hi Please Subscribe to — on YouTube and put on post notifications on for all and like and comment on all her videos thanks :)*

Table 6.13: *Example of a spamming conversation. The username is anonymized and marked by "—".*

Sender	Time	Message
D	11:56 PM	*what for xoxo dress*
E	11:56 PM	*depends on your offer !:)*
D	00:00 AM	*look at my ab I willing to overpay yo*
E	00:01 AM	*alr ill go look (:*
F	04:43 AM	*Could i trade something for the red christmas dress?*
E	04:43 AM	*yes*
F	04:43 AM	*i have ab called, "offers?" and there is some rares, and one dpack in there if ur interested!*
E	04:43 AM	*I dont fly see anything*
E	04:43 AM	*rly*
F	04:43 AM	*oh ok, you mean you can't find the ab? or you can't find anything you like?*

Table 6.14: *Example of a trading user*

Sender	Time	Message
G	5:25 PM	**kisses you**
H	5:25 PM	**kisses more**
G	5:25 PM	**kisses**
H	5:26 PM	**tvzakes off robe**
G	5:26 PM	**looks at your bxxvbsx**
H	5:26 PM	**puts my arms around you**
G	5:27 PM	**grabs your asvvsx**
H	5:28 PM	**gets on you**
G	5:28 PM	**looks at your bxxvbsx**
H	5:29 PM	**eats your psvvxyv ouxxtvx**
G	5:29 PM	**moxvns**
I	4:01 AM	**closes door**
G	4:02 AM	**grips your thigh and drvvesvx**
I	4:03 AM	*:o Haha dont grip to hard*
G	4:03 AM	*your psvvxyv is gonna grip around my dvvxkvxvx soon *kisses you**
I	5:26 PM	*I guess your right *kisses back**

Table 6.15: *Example of a sexting user*

Sender	Time	Message
K	6:37 PM	*wyd*
J	6:37 PM	*getting stabbed in bed*
J	6:37 PM	*you?*
K	6:38 PM	*sound fun and just chillin*
J	6:40 PM	*lol at least some thinks so*
K	6:42 PM	*whats your name?*
J	6:42 PM	*—*
K	6:43 PM	*okay interesting way to spell it*
K	6:44 PM	*Im —*
K	6:45 PM	*my mom picked it i like ur name*
J	6:45 PM	*nice name and thanks*
J	6:46 PM	*uhh u have sc*
L	2:12 PM	*hey*
M	2:12 PM	*sc*
L	2:13 PM	*no*
M	2:13 PM	*get it*
L	2:14 PM	*for what*
M	2:14 PM	*wanna trade*
L	2:14 PM	*trade ?*
M	2:14 PM	*pictures*
L	2:15 PM	*umm let me think*
M	2:15 PM	*ok*
L	2:17 PM	*you really want to trade*
M	2:17 PM	*yes*
M	2:18 PM	*im hcorny*

Table 6.16: *Example of different users asking for Snapchat. The user-names are anonymised by "—".*

Sender	Time	Message
N	8:47 PM	*are you freaky*
O	8:48 PM	*how old are you lol*
N	8:49 PM	*20*
O	8:50 PM	*oh im 15*
N	8:50 PM	*tbh i would still be down but you probably wouldnt*
O	8:51 PM	*why do you think i wouldnt be down*
N	8:51 PM	*cause Im older unless ur into that*
N	8:51 PM	*tbh I want pics tho*
O	8:53 PM	*pics of me?*
N	8:53 PM	*yeah naughty ones*
O	8:53 PM	*ive never really done anything like this*
N	8:53 PM	*sc*? do u have it*
O	8:54 PM	*yeah, but my mom has my phone*
N	8:54 PM	*too bad*
O	8:55 PM	*do you have a lot of girls on here send you pics*
N	8:56 PM	*yeah*
O	8:56 PM	*a lot of minors?*
N	8:56 PM	*yeah*

Table 6.17: *Example of a potential groomer conversation*

7 Discussion

In this section, we will examine the results in more detail, searching for patterns and trends in the data. We will also consider the implications of these findings and discuss the limitations of the study. The goal is to provide a comprehensive evaluation of the results and their relevance to the research questions posed earlier. When discussing the limitations of the study, we will also identify areas for improvement and suggest ideas for future research that could build upon our findings.

First, we will address the questions proposed in the experimental setup and address the main research question and two sub-questions. Next, we will discuss the limitations and considerations of the features, metrics, and statistics used in this study.

7.1 Synthetic Data

7.1.1 Are the synthetically generated anomalies more similar to the true anomalies than randomly sampled true negative nodes?

We are interested in examining whether sexually predatory users show distinct behaviour in their social network graphs. By comparing the distribution of features and graph structures between synthetic anomalies to true positives and negatives, we can determine if there are differences in the social networks of anomalous and non-anomalous users. The synthetic anomalies are based on 346 confirmed anomalous users, so this comparison allows us to investigate whether the features and high-level graph structure of the synthetic anomalies accurately represent those of true anomalous users.

Additionally, to utilise synthetic anomalies effectively for hyper-parameter selection, it is crucial that they mimic both the features and graph structures of true anomalies. This is necessary to ensure the effectiveness of the models when applied to non-synthetic data.

Based on the visualisations of negative and positive users before and after the synthetic modification, shown in Figures 6.1 and 6.2, the synthetic anomalies resembled the positive user more than the negative in the structure of the 1-hop neighbourhood.

The structural properties of true and synthetic anomalies, such as clustering coefficients and node in- and out-degrees have been further analysed. The distributions of these graph measurements for true and synthetic anomalies are shown in Figure 6.3, and the results showed that the distributions for the true and synthetic anomalies are closer to each other than the distribution for non-anomalous users, indicating that the topology of the synthetic anomalies more closely resembles that of true anomalies.

The distributions of *clustering coefficients* are not as separated, indicating that the assumption

that the anomalous users have a smaller *clustering coefficient* might be wrong. The results showed that true anomalies have a higher degree of clustering than synthetic anomalies as well as higher variability in their distributions. Based on this, the *clustering coefficient* should be higher for synthetically generated users. This could be done by sampling from the 2-hop neighbours in the injection algorithm when adding new edges, though this could potentially lead to overly large clustering coefficients and is left for further research.

Furthermore, the similarities between the attributes of are investigated. Resemblance between synthetic and true anomalies was found in features like *weighted max in- and out hourly*, where the distributions were largely separated from the true negatives. However in features like the *number of active hours*, the non-synthetic user distributions resembled each other more than the synthetic ones. This could have the effect that the synthetic anomalies become too easy to detect. Based on the marginal distributions of the *weighted proportion outgoing* attribute, it does not appear to effectively separate the classes.

As mentioned in section 4.2, we identified a total of 346 users who engaged in sexting at least three times over the four-month period. These users served as the basis for generating synthetic anomalies with respect to both topology and node attributes. Therefore, it is possible that our final test models may be biased towards users who engage in sexting. Additionally, since we did not have any previously identified users exhibiting potential grooming behaviour, it was not possible to create synthetic anomalies that accurately represent these users. However, the synthetic anomalies could potentially be improved upon by incorporating the newly discovered users who may be potential groomers.

Overall, the synthetic anomalies appear more similar to the true anomalies than to non-anomalous users in terms of both structural indicators (such as clustering coefficient and out-degree) and attributes in terms of their marginal distributions. However, it is difficult to determine if the synthetic anomalies also resemble the true anomalies in terms of joint distributions of these characteristics. Additionally, there are differences in the behaviours of anomalous and non-anomalous users in some of the marginal distributions of their proposed features as well as for graph measurements such as *node out degree*.

7.1.2 Do DOMINANT and AnomalyDAE models yield higher average precision when the structural reconstruction error is weighted more heavily than the contextual reconstruction error?

We conducted an α-parameter sensitivity analysis on AnomalyDAE and DOMINANT, as shown in Figures 6.6, 6.7, and 6.8. It was found that the models generally performed better in terms of average precision on the synthetic sets when using a higher value of α. This suggests that the models place greater importance on correctly reconstructing the attribute matrix rather than the adjacency matrix.

The preference towards weighting the attribute reconstruction loss higher than the structure

reconstruction loss could indicate that the attributes contain more information regarding whether a user is anomalous than the structure does. However, it should be noted that for all graph models, the structure is also used to create node embeddings, so an α-value of one does not mean that the structure is not used at all.

As previously discussed, the synthetic anomalies in the data tended to have feature distributions that were more similar to those of true anomalies than true negatives. Based on this, it can be hypothesised that the results of the α-parameter sensitivity analysis on the synthetic data would be similar to the results of the analysis on non-synthetic data. However, as can be seen from the validation of the *precision@25* metric seen in Table 6.7, the results based on the grid search and the sensitivity analysis done on the synthetic data cannot be generalised directly to the non-synthetic data. We propose testing more models on non-synthetic data using different values of α. The current test results are based on the models with $\alpha = 0.8$, as this value performed the best based on the grid search. However, the α-sensitivity analysis showed that these models performed better on average when $\alpha = 1$, so it would be worthwhile to see if this is also the case on the non-synthetic data.

To summarise, the two models did not perform better when the structural reconstruction error was weighted more heavily than the contextual reconstruction error when evaluated on synthetic data. On the contrary, the average precision was higher when the contextual loss was weighted more heavily. These results cannot be directly translated to non-synthetic data, so we propose running tests on non-synthetic data with $\alpha = 1$.

7.2 Non-synthetic Data

7.2.1 Do the models perform better than random on the real data?

The proposed models were compared to a baseline by using a random sampling of active users in the test graphs. The results showed that the proposed models had a higher *precision@25* than the baseline. The baseline had an average *precision@25* of 12.0% across the three test data sets, with a standard deviation of 8.0%. In contrast, the proposed models had average *precision@25* values ranging from 41.3% to 50.7%, with standard deviations lower than 11%. It is important to note that the standard deviations are based on three samples and may not accurately represent the true variation in performance. These results can be seen in Table 6.8.

To further investigate the performance of the models compared to baseline, the average overlap (AO) of the top 25 rankings of the users was calculated for each model combination across all test data sets. The results, shown in Figures 6.9 and 6.10 for the seven-day and three-day test data sets respectively, indicated that the randomly selected users had no average overlap with the users selected by any of the models. This was expected, since the probability of randomly selecting an identical user as one of the models, is between 3.4% and 1.3% for a population between 17,881 and 46,921 users[1].

[1]This is calculated using the hyper-geometric distribution.

As shown in Figure 6.11, the randomly sampled true positives do not have a feature distribution in terms of the *max out hourly* and *weighted max out hourly* features that differs from the true negative distribution, unlike the users selected by the models. This indicates that the users selected by the models are larger outliers in terms of the marginal distributions of *max out hourly* and *weighted max out hourly*.

Upon manual evaluation of the randomly sampled true anomalies, we also noticed that the anomalous behaviour was generally less consistent compared to the users selected by the model. While the manual evaluation did not consider the severity and consistency of anomalous behaviour was in the labelling process, the comments from the manual evaluation, included in Appendix 9.9.1, indicate that the offences of the randomly sampled users were generally less severe and consistent than those of the users selected by the model. The severity is in this case based upon different levels of sanctions described in MSP moderator guidelines.

According to the guidelines for moderators on MovieStarPlanet, there are several actions that can be taken based on the severity of the offence. These actions range from sending help links and issuing warnings to temporarily or permanently locking the account, along with an IP address lock. In order to rank users based on the severity of their offences, we propose using these sanctions as a measure of severity, such that a user who receives the most severe sanction would be ranked higher than a user who receives a less severe sanction. Building on this suggestion, a semi-supervised conversation classifier could be researched, where different classes could be based on sanctions, allowing the classifier to recommend a specific level of sanction based on the attributes of the conversation. Alternatively, the classes could also be based on the types of conversations identified in Section 6.7.1, such as spamming, grooming, and sexting conversations.

To summarise, the proposed models performed better than random in terms of *precision@25*. There was no overlap between the top 25 rankings of the models and randomly selected users. The users selected by the models also generally showed more consistent and severe anomalous behaviour compared to the randomly selected true anomalies, as determined by manual evaluation. This supports that a purely graph-based approach can detect sexual predators in data from MSP.

7.2.2 Is there a significant benefit from utilising seven-day graphs rather than three-day graphs?

We tested the performance of all models on both the seven and three day data sets, except for the DOMINANT model that was not able to run on the seven-day data set. Tables 6.8 and 6.9 show that the average *precision@25* was slightly higher for the three-day tests, but the standard deviations make it difficult to conclude that one data set period performs consistently better than the other.

In terms of computational resources, it is generally more efficient to use smaller graphs. As

shown in Tables 4.4 and 4.5, the seven-day graphs are larger than the three-day graphs, with an average number of nodes of 34.569 and 19.801, respectively. Figure 6.12 also demonstrates that the AnomalyDAE and AdONE models do not scale well to larger graphs, while MLPAE and GCNAE scale significantly better. In addition to these models' ability to scale, the resources needed to calculate the proposed high-level graph features for a seven-day period are significantly greater than those needed for a three-day period. Based on these considerations, we recommend using the three-day graphs rather than the seven-day graphs, as we did not observe a clear improvement in performance on the seven-day graphs.

Although we used seven-day and three-day periods, the main difference between the periods was more evident in the structure of the graph rather than the attributes. This is because the attributes were created based on daily frequency and then aggregated over the periods, resulting in both the seven-day and three-day graphs having attributes that reflect the "average day" across the respective periods. To better capture the differences between periods, it would be useful to try a non-aggregated approach where the features are calculated specifically for the given graph and not aggregated at all. For example, the features could be the proportion of outgoing messages in a week, rather than a day. Additionally, it would be beneficial to introduce features that have some level of weekly seasonality, as it can be assumed that children between the ages of 8-15 have different schedules and availability to use MSP due to regular school schedules than older users do. In the data section, a distribution plot of the number of messages per weekday was included. However, we only had access to timestamps in the UTC format, which does not accurately reflect the local time experienced by the user when sending the message. To incorporate more temporal features into the attributes, it is necessary to obtain the timestamp in the user's local timezone.

We did not investigate the use of one-day graphs instead of seven- or three-day graphs in our analysis. The decision not to explore one-day graphs was based on the assumption that a single day may not accurately reflect a user's true and consistent behaviour, as there may be greater variability in how a user behaves within a single day compared to an average over a longer period. However, our results showed that using seven-day sets did not perform better than three-day data sets. As a result, we propose further investigation into whether one-day graphs may be more effective. This approach has the advantage of using non-aggregated data and requiring less computation time due to smaller graph sizes. We also suggest the possibility of using even more granular data, such as hourly behaviour, which would require adapting the proposed features but could potentially lead to faster response times.

Based on our analysis, we found that models using seven-day data sets did not perform better than those using three-day data sets. Therefore, we recommend using the three-day data sets due to their lower resource usage. We also propose further research into the potential benefits of using more granular data sets, such as daily or hourly graphs, as they may offer the same performance while requiring fewer resources and providing faster response times.

7.2.3 Do the models achieve a better performance when using the graph topology explicitly?

We research whether models that use the graph structure perform better than models that do not use said graph structure. We have evaluated five models, MLPAE, which does not use the graph structure at all and GCNAE, DOMINANT, AnomalyDAE, and AdONE, which use the structure in various ways to calculate the final outlier score (as described in Section 5.6).

Upon comparing the performance of MLPAE to the other models, we do not observe a significant difference in performance and therefore cannot conclude that models that utilize the structure of the graphs consistently outperform the baseline MLPAE. This is also reflected in the average overlap scores for both the seven-day and three-day sets. The top 25 users for MLPAE are often also found in the top 25 for GCNAE and AnomalyDAE, and the average overlap scores suggest that the models rank the users very similarly.

It is important to note that we encode a lot of information about the higher level graph structure into the attributes implicitly. The graph we are using is not weighted, so although the connections between two users are directed, the structure does not convey information about the balance of the connection. For example, as shown in Table 6.13, the conversation between users B and C is very imbalanced, with B sending many messages to C without receiving many in return. While we try to encode an aggregated version of this information into each user's features (such as through a combination of degree features and the *weighted proportion outgoing* feature), this information cannot be fully represented in the structure of an unweighted graph.

The approach using graph neural networks did not perform significantly better than the MLPAE baseline and required more computation time and processing power. Therefore, it may be more effective to encode additional graph information into the attributes rather than using the graph structure directly. For example, adding the *clustering coefficient* and *weighted and unweighted in- and out-degree* to the attributes could improve performance. This approach also has the added benefit of increased interpretability and reduced resource usage. Additionally, the field of tabular outlier detection is more mature than that of graph outlier detection and is therefore better suited for a production environment.

It would be interesting to conduct this research again using a weighted graph, since the information could be encoded on a per-relation basis rather than being aggregated. However, to the best of our knowledge, there has not been any research on outlier detection in weighted, attributed and directed graphs. In addition, it would be valuable to conduct research on classifying anomalous relationships between users, also known as edge-classification. This could be explored in the tabular data approach as well. It could be achieved by creating conversation-based attributes, such as the length of the conversation, the number of messages, and the proportion of sent messages to received messages. Additional features such as

linguistic or chat-based features could be included to improve performance. These attributes could be used to identify conversations that deviate from the norm and may be considered outliers. Classification of conversations, rather than outlier detection, could also be considered, though it would likely result in a data set with high class imbalance.

In conclusion, our results do not support the notion that models that utilise the structure of the graphs explicitly perform better than those that do not. However, further research may reveal ways to effectively utilise the graph structure to encode useful information. Additionally, exploration of conversation outlier detection or classification in both graph and tabular data may be valuable areas of research.

7.3 True and False Positive Behaviour

The goal of this investigation was to determine if sexual predatory users exhibit different behaviour in their social network graphs. Our findings indicate that it is possible to detect predatory behaviour using only graph-based approaches, which suggests that predatory users exhibit distinct patterns of behaviour within their social networks.

To understand which of the proposed graph-based features showed a difference in the behaviour of anomalous and non-anomalous users, we presented the distribution of the *max in- and out hourly* features for true positives, true negatives, and false positives in the results section. As shown in Figure 6.11, the false positive feature distributions were closer to those of the true positives than true negatives, suggesting that false positives may exhibit behaviour that is more similar to anomalous users than typical non-anomalous users. We argue that false positives are classified as anomalous due to their abnormal behaviour, even though they are not necessarily predatory in nature. In the following section, we will discuss how different types of anomalous behavior described in Section 6.7.1 may exhibit similar patterns in social network graphs, even for groupings that are not sexually predatory.

We identified different categories of highly ranked users, such as spammers, sexters, and traders, based on a manual evaluation of over 600 unique users. One of these categories was data error conversations, which occurred due to a collection error that caused some users to be wrongly labelled as false positives because they appeared to send high volumes of messages without receiving any responses. In reality, both users were participating in the conversation, but the error was not detected until the conversations were manually reviewed and cannot be fixed automatically through pre- or post-processing of the data. This highly skews the performance results of the models and should be investigated further before production.

Spammers were identified as sending messages to many different users, requesting gifts and likes in the game. Anomalous users also tend to connect with many different users, often to request Snapchat information for the purpose of trading nude pictures. Both types of users tend to randomly contact others, which may make spammers more similar to anomalous users in terms of their social network behaviour. They may also be similar in the frequency and volume of

messages. However, the content of their conversations and common linguistic features may differ. Although they are not sexually predatory in nature, they do display behaviour that might be unwanted from MovieStarPlanet's side.

Users who are selling or trading items may not always initiate contact with other users, but we expect their neighbourhood to be less clustered due to the high volume of potentially unconnected users approaching them for trades. Additionally, we suspect that the network behaviour of these users may be unusual in terms of conversation length. Although these conversations have not been evaluated as predatory, it may be useful to rank them highly due to the potential exchanges of money or passwords that may occur in order for the user to receive their items.

Potential groomers were mainly found engaging in two types of conversations. The first involved sexting, in which the groomer would explicitly request that the other user portray a child in the role play and ask questions such as "What's the youngest you'll do?" The second type of conversation involved the potential groomer asking for the other user's social media accounts in order to move the conversation off of MovieStarPlanet, often with the explicit intention of obtaining nude pictures. This leads to two potentially different user behaviours in term of the graph. The first might have a very high volume and frequency of messages with a potentially lower amount of degree since they are preoccupied with having long conversations with one user. The other type may however write to a lot of users and either be turned down or move the conversation to another platform. This could result in a high degree with a lower conversation length.

These two types of users would often ask questions about the sex life of the other user and how 'experienced' they are. When reading through some of their conversations, it is evident that they are lying about their age, since they will often ask about the age of the other user first and match their answer. In the manual evaluation of the chats, a user was seen to immediately ask if the other user sends nude pictures and how old they are when starting the conversation. Based on the answer, the user would say that his or her age ranged from 15 to 19 years old. Some of the users they were approaching were as low as 14 years old. Users who frequently lie about their age in conversation with others can potentially be identified using a linguistic approaches which could be used for identifying instances where users mention their age, and to flag or rank users who give multiple different ages higher.

In summary, our research demonstrated that predatory users exhibit patterns of behaviour within their social networks that are distinct enough to be effectively used for identifying anomalous users using graph-based approaches. We further show that there is a difference in some of the marginal feature-based distributions when comparing highly ranked anomalous users to randomly sampled non-anomalous users. Overall, these results suggest that it is possible to identify predatory users and distinguish them from non-predatory users using their social network behaviour.

7.4 Statistical Considerations

One limitation of this research is that we did not use statistical models to directly compare the performance of the models on the seven-day and three-day data sets. This was due in part to the time-consuming nature of manual evaluation and the complexity of the data, which has both temporal and interdependent properties. While it would have been possible to use bonferroni-adjusted paired t-tests [BMA18] to compare the models this approach would not have been appropriate because the assumption of normality would be violated due to the small sample size of three per data set.

Additionally, we observed that model performance varied across different time periods. A two-way ANOVA [BMA18] could have been used to investigate whether there was a significant difference in model performance and to determine whether the time period had an impact on performance. However, the assumptions of this test, including normality of the samples and independence between them, would not have been met in this case.

We also considered using McNemar's test [HSM19] to compare the models on the test data, but this method requires that the two classifiers be tested on the same exact data splits in order to calculate where they agree and disagree. Although we had the labels for the top 25 rankings for each model, the models did not contain the same exact users in their respective top 25 rankings, so it was not feasible to perform this test. This was also the reason why it was not possible to calculate Kendall's correlation coefficient between the rankings. Although it would have been statistically valid to calculate the correlation coefficient for the entire rankings of the models, we argued that it would not be useful for this comparison because the probability of the models ranking the users similarly across the entire ranking is very low. As a result, the resulting correlation coefficient would likely be very small and not accurately reflect the similarity between the rankings. Additionally it is not meaningful for moderators that cannot feasibly evaluate such a large amount of people.

7.5 Metric and Objective Considerations

We modeled the problem as an outlier detection task, motivated by the assumption that predatory behaviour is highly abnormal. In our results section, we evaluated model performance using labels generated by manual evaluation of predatory behaviour. There is an important distinction between users that are outliers and users behaving in a predatory manner and the latter is only a subsection of the former, and the model's objective may not necessarily align with our own. This is evident when comparing the objective loss per epoch with the average precision per epoch, which is shown in Figures 9.1 to 9.9 in Appendix 9.5 and 9.6. While the loss per epoch consistently decreases and converges towards the best value seen, the average precision does not necessarily converge on the highest value.

Furthermore, the most abnormal users on the platform may not necessarily be the ones exhibiting predatory behaviour. For example, inactive users may be considered abnormal on a

very active platform. To address this issue, it would be valuable to explore semi-supervised outlier detection methods, which require some true positives and true negatives as examples of normal and abnormal behaviour. Our research has generated these resources as a byproduct.

The results of the random sampling in Tables 6.8 and 6.9 also suggest that the proportion of anomalous users may be significantly higher than originally assumed, as one to five anomalous users were identified in each of the six random samples of 25 users. This indicates that anomalous behaviour may be less abnormal than initially thought, and it would be worthwhile to investigate whether outlier detection is an appropriate approach for this problem.

In the results section, we evaluated model performance using average precision in the grid search and *precision@25* in the test and validation performance. The *precision@25* measurement was chosen for practical reasons based on the time frame available for evaluation. When implementing a solution for MSP, it would be important to carefully consider which metric is most appropriate for the specific platform, taking into account factors such as the number of users a moderator can investigate in a day and the relative importance of precision versus recall.

For the grid search, we calculated a range of metrics, including ROC-AUC, average precision, *precision@50*, *recall@50*, *precision@100*, *recall@100*, and *precision@num.anomalies*. The grid search performance for these metrics are shown in the Appendix in Section 9.5.

7.6 Feature Considerations

In this thesis, we proposed sixteen features to be extracted from the sender ID, receiver ID, and timestamp data. These features aim to capture higher-level information about the relationships between users that cannot be represented in the structure of the directed, unweighted graph alone. This includes temporal information such as message frequency, as well as the proportion of inbound and outbound messages. Our models were able to correctly identify between eight and fifteen anomalous users in the top 25 rankings for seven-day graphs, and between twelve and twenty anomalous users in the three-day rankings, as can be seen from the samples shown in Section 6.5.

The test performance suggests that the proposed graph structure and attributes capture a significant amount of useful information, although some additional information may be missing.

The fitted distributions in Figures 6.5 and 6.11 show a separation between anomalous and non-anomalous users in several features. This suggests that anomalous users exhibit distinct behaviour in terms of these features compared to non-anomalous users, particularly in the features *weighted max out hourly* and *weighted max in hourly* (Figures 6.4 and 6.11). The separation is less pronounced in the feature *weighted proportion outgoing*. It's worth noting that these conclusions can be influenced by at least two confounders. First, the features are only compared visually in their marginal distributions, not in combination as they are used in our modelling. Second, the anomalous users shown are selected from the top 25 most highly ranked

users, which means the model's rankings may influence the appearance of the true anomalous distribution. To address this confounder, we would need to randomly select and manually label anomalous users to compare them to non-anomalous users, but this is a time-intensive task and will be left for future research.

The original data includes temporal information, some of which has been incorporated into frequency-based attributes. However, it would be interesting to explore attributes that capture information about a user's activity levels at different times of the day. This could help distinguish between children and adults, who may have different online schedules. For example, we could consider features that indicate the hours of the day when a user is most active or the proportion of total messages sent during a given time interval. However, as previously mentioned in Section 7.2.2, these types of features cannot be accurately calculated from the UTC timestamps alone, as they do not take into account the sender's local time.

Our research focused on the higher-level graph information and the results showed that anomalous and non-anomalous users generally exhibit different social network behaviours. However, considering the nature of the chat-based problem, it would be beneficial to also include information from the actual chat interactions between users, as this could potentially lead to a more robust and accurate performance. We hypothesise that a combination of network information and chat data would provide a better understanding of anomalous behaviour.

Based on the roughly 6.000 conversations we have read during the course of the project, we observed recurring differences between sexually charged conversations and normal conversations. Specifically, sexting chats tended to have a higher proportion of non-alphanumeric characters, which can also be seen in Table 4.2. This is especially true for role-play conversations where the chat often begin and ends with the character ''. An example of this is shown in Tables 4.2 and 6.15. Generally the sexually charged words are often obscured by double or triple consonants as well. We have included a list of such words in Appendix in Table 9.77 These findings suggest that counting non-alphanumeric characters and number of consonants, double consonants and triple consonants may be a useful feature for identifying sexually charged conversations. These attributes as well as the network-based ones have the added benefit of being able to generalise to conversations in different languages. The length of the sentences could be another useful feature to consider.

To summarise, the proposed graph structure and attributes capture a significant amount of useful information although given the false positives, the models could benefit from additional information to be able to better distinguish between sexually predatory users and other abnormal behaviours. One solution is to use the content of the conversations to create linguistically-based features. Based on our analysis of roughly 6,000 conversations, we propose several behavioural features derived from the chats that would make the modelling approach more robust.

One potential approach is to perform semi-supervised conversation classification on the tabular data. The different classes could be based on sanctions, as previously mentioned, allowing the classifier to recommend a specific level of sanction based on the attributes of the conversation. Alternatively, the classes could be based on the types of conversations identified in Section 6.7.1, such as spamming, grooming, and sexting conversations.

8 Conclusion

In this thesis, we have explored the use of a graph-based approach that models users' social network patterns as a means of detecting online sexual predators, without relying on linguistic analysis. This approach differs from the typical method, which often involves analysing language and text-based communication patterns to identify potentially predatory behaviour. Our research aims to determine the effectiveness of this alternative approach.

In conclusion, our results do not support the notion that models that utilise the structure of the graphs explicitly perform better than those that do not. However, the researched showed that the sexually predatory users display a different behaviour in terms of their social network graph, which allowed us to use a purely graph-based modelling approach to detect sexual predators in MovieStarPlanet.

Our analysis of the data led us to propose sixteen features that capture higher-level, graph-based information about the social networks of users on MovieStarPlanet. The fitted marginal feature distributions show a separation between anomalous and non-anomalous users in several of these features. Some of the features showing high separation includes the maximum messages sent and received hourly.

A new way of injecting synthetic outliers that differ both in terms of their attributes and their structure combined was proposed. The generation of synthetically combined outliers was essential for selecting appropriate hyper-parameters for our models. The synthetic anomalies appear more similar to the true anomalies than to non-anomalous users in terms of both structural indicators, such as clustering coefficient and out-degree as well as attributes in terms of their marginal distributions.

One of the main questions, **SQ1**, in our research is whether models that incorporate graph structure perform better than those that do not. To answer this question, we compared a baseline model that does not use graph structure with four models that utilise the structure in different ways to calculate the final outlier score. The results showed that there was not a significant difference in performance between the baseline model and the other models. Therefore, we cannot conclude that models that utilise the structure of the graphs consistently outperform models that do not. In addition, a sensitivity analysis revealed that the average precision was improved when the attribute reconstruction loss was given more weight compared to the structural reconstruction loss, although the attributes still incorporate structural information.

The proposed graph auto-encoder models performed better than random in terms of *precision@25* and the users selected by the models also generally showed more consistent and severe anomalous behaviour compared to the randomly selected true anomalies, as determined

by manual evaluation. The highest seen *precision@25* score was 80% for a dataset consisting of around 20,000 active users, while we saw an average *precision@25* across different time-periods and graph-sizes of around 60%. Throughout the duration of the project we have identified 527 unique users showing predatory behaviour. This, along with the separation in the marginal distributions of the proposed features, shows that anomalous users exhibit distinct behaviour in terms of their social network behaviour.

Overall, the proposed graph structure and attributes capture a significant amount of useful information for detecting sexually predatory users and other abnormal behaviours. However, the model's performance could be improved by adding more information. One way to do this is to use the content of the conversations to create linguistically-based features. Based on our analysis of approximately 6,000 conversations, we propose several behavioural features derived from the chats that would enhance the modelling approach, such as a count of non-alphanumeric characters and the number of single, double, and triple consonants. The advantage of both the graph-based approach as well as these proposed features is that they are language-independent.

Bibliography

[NCM21] NCMEC. *CyberTipline Data*. 2021. URL: https://www.missingkids.org/gethelpnow/cybertipline/cybertiplinedata.

[Bla+15] Pamela J. Black et al. "A linguistic analysis of grooming strategies of online child sex offenders: Implications for our understanding of predatory sexual behavior in an increasingly computer-mediated world". In: *Child Abuse and Neglect* 44 (June 2015), pp. 140–149. ISSN: 18737757. DOI: 10.1016/J.CHIABU.2014.12.004.

[San17] Sandra Marchenko. *Web of Darkness: Groomed, Manipulated, Coerced, and Abused In Minutes - Biometrica Systems, Inc.* 2017. URL: https://www.biometrica.com/icmec-online-grooming/.

[Kra22] Michael Kraut. *Children and Grooming / Online Predators | Child Crime Prevention & Safety Center*. 2022. URL: https://childsafety.losangelescriminallawyer.pro/children-and-grooming-online-predators.html.

[Mov22a] MovieStarPlanet. *Safeguarding Children MovieStarPlanet*. 2022. URL: https://moviestarplanet.zendesk.com/hc/en-gb/articles/115000385689-Safeguarding-Children.

[Mov22b] MovieStarPlanet. *MovieStarPlanet*. 2022. URL: https://www.moviestarplanet.dk/.

[CBG06] Samantha Craven, Sarah Brown, and Elizabeth Gilchrist. "Sexual grooming of children: Review of literature and theoretical considerations". In: (2006). DOI: 10.1080/13552600601069414.

[Ols+07] Loreen N. Olson et al. "Entrapping the Innocent: Toward a Theory of Child Sexual Predators Luring Communication". In: *Communication Theory* 17.3 (Aug. 2007), pp. 231–251. ISSN: 1468-2885. DOI: 10.1111/J.1468-2885.2007.00294.X. URL: https://onlinelibrary.wiley.com/doi/full/10.1111/j.1468-2885.2007.00294.x.

[WJ16] Georgia M. Winters and Elizabeth L. Jeglic. "Stages of Sexual Grooming: Recognizing Potentially Predatory Behaviors of Child Molesters". In: *https://doi.org/10.1080/01639625.2016.1197656* 38.6 (June 2016), pp. 724–733. ISSN: 15210456. DOI: 10.1080/01639625.2016.1197656. URL: https://www.tandfonline.com/doi/abs/10.1080/01639625.2016.1197656.

[Oco03] Rachel O'connell. "A TYPOLOGY OF CHILD CYBERSEXPLOITATION AND ONLINE GROOMING PRACTICES". In: (2003). URL: www.fkbko.net.

[CFA14] Amparo Elizabeth Cano, Miriam Fernandez, and Harith Alani. "Detecting child grooming behaviour patterns on social media". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8851 (2014), pp. 412–427. ISSN: 16113349. DOI:

10.1007/978-3-319-13734-6{_}30. URL: https://www.researchgate.net/publication/265850346_Detecting_Child_Grooming_Behaviour_Patterns_on_Social_Media.

[Kon+09] April Kontostathis et al. *(PDF) Comparison of Rule-based to Human Analysis of Chat Logs*. 2009. URL: https://www.researchgate.net/publication/239929144_Comparison_of_Rule-based_to_Human_Analysis_of_Chat_Logs.

[GKS12] Aditi Gupta, Ponnurangam Kumaraguru, and Ashish Sureka. "Characterizing Pedophile Conversations on the Internet using Online Grooming". In: (Aug. 2012). DOI: 10.48550/arxiv.1208.4324. URL: https://arxiv.org/abs/1208.4324v1.

[San+21] Benjamin Sanchez-Lengeling et al. "A Gentle Introduction to Graph Neural Networks". In: *Distill* 6.9 (Sept. 2021), e33. ISSN: 2476-0757. DOI: 10.23915/DISTILL.00033. URL: https://distill.pub/2021/gnn-intro.

[Ham20] William L Hamilton. "Graph Representation Learning". In: 14.3 (2020).

[Bar14] Albert-Laszlo Barabasi. "NETWORK SCIENCE". In: (2014), p. 26.

[Hol22] Michael Holroyd. *Clustering coefficient - Wikipedia*. 2022. URL: https://en.wikipedia.org/wiki/Clustering_coefficient.

[Aka+18] Ademar T. Akabane et al. "Distributed egocentric betweenness measure as a vehicle selection mechanism in VANETs: A performance evaluation study". In: *Sensors (Switzerland)* 18.8 (Aug. 2018). ISSN: 14248220. DOI: 10.3390/S18082731. URL: https://www.researchgate.net/publication/327121659_Distributed_Egocentric_Betweenness_Measure_as_a_Vehicle_Selection_Mechanism_in_VANETs_A_Performance_Evaluation_Study.

[APA22a] APA Psychology Dictionary. *APA Dictionary of Psychology*. 2022. URL: https://dictionary.apa.org/homophily.

[APA22b] APA Dictionary of Psychology. *Heterophily*. 2022. URL: https://dictionary.apa.org/heterophily.

[Wu+19] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: (Jan. 2019). DOI: 10.1109/tnnls.2020.2978386. URL: https://arxiv.org/abs/1901.00596.

[Gil+17] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: (Apr. 2017). DOI: 10.48550/arxiv.1704.01212. URL: https://arxiv.org/abs/1704.01212.

[Mic20] Microsoft Research. *An Introduction to Graph Neural Networks: Models and Applications - YouTube*. 2020. URL: https://www.youtube.com/watch?v=zCEYiCxrL_0&t=1076s.

[Vel21] Petar Velickovic. *University of Cambridge Lecture - "Theoretical Foundations of Graph Neural Networks"*. Feb. 2021. URL: https://www.youtube.com/watch?v=uF53xsT7mjc&t=553s.

[Bro+21] Michael M Bronstein et al. "Geometric Deep Learning Grids, Groups, Graphs, Geodesics, and Gauges". In: (2021).

[KW17] Thomas N Kipf and Max Welling. *SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS*. Tech. rep. 2017.

[Vel+18] Petar Velikovic et al. “GRAPH ATTENTION NETWORKS”. In: (2018).

[Kim+22] Hwan Kim et al. “Graph Anomaly Detection with Graph Neural Networks: Current Status and Challenges”. In: (Sept. 2022). URL: http://arxiv.org/abs/2209.14930.

[Vil+12] Esaú Villatoro-Tello et al. “A Two-step Approach for Effective Detection of Misbehaving Users in Chats Notebook for PAN at CLEF 2012”. In: (2012). URL: http://www.perverted-justice.com/.

[Che+13] Yun-Gyung Cheong et al. “Detecting Predatory Behaviour in Game Chats”. In: (2013). URL: http://info.moviestarplanet.co.uk/terms-conditions.aspx.

[IC] Giacomo Inches and Fabio Crestani. “Overview of the International Sexual Predator Identification Competition at PAN-2012”. In: *2012* (). URL: http://www.oocities.org/urgrl21f/,.

[PEB12] Javier Parapar, David E. Losada, and Alvaro Barreiro. *(PDF) A learning-based approach for the identification of sexual predators in chat logs*. 2012. URL: https://www.researchgate.net/publication/269103611_A_learning-based_approach_for_the_identification_of_sexual_predators_in_chat_logs.

[MC22] Daniela F. Milon-Flores and Robson L.F. Cordeiro. “How to take advantage of behavioral features for the early detection of grooming in online conversations”. In: *Knowledge-Based Systems* 240 (Mar. 2022). ISSN: 09507051. DOI: 10.1016/J.KNOSYS.2021.108017.

[Wan+] Daixin Wang et al. “A Semi-supervised Graph Attentive Network for Financial Fraud Detection”. In: (). URL: https://www.altexsoft.com/whitepapers/fraud-detection-how-machine-.

[Ma+21] Xiaoxiao Ma et al. “A Comprehensive Survey on Graph Anomaly Detection with Deep Learning”. In: (2021), p. 1. URL: https://www.zdnet.com/article/online-fake-news-costing-us-78-billion-.

[Vau] Rémi Vaudaine. “Detection and Explanation of Contextual Anomalies in Attributed Graphs”. In: (). URL: https://theses.hal.science/tel-03726521.

[Ako+15] Leman Akoglu et al. “Graph based anomaly detection and description: a survey”. In: *Data Min Knowl Disc* 29 (2015), pp. 626–688. DOI: 10.1007/s10618-014-0365-y.

[Liu+22a] Kay Liu et al. “BOND: Benchmarking Unsupervised Outlier Node Detection on Static Attributed Graphs”. In: (June 2022). DOI: 10.48550/arxiv.2206.10071. URL: http://arxiv.org/abs/2206.10071.

[BVM20] Sambaran Bandyopadhyay, Saley Vishal Vivek, and M N Murty. “Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding”. In: (2020). DOI: 10.1145/3336191.3371788. URL: https://doi.org/10.1145/3336191.3371788.

[Yua+21] Xu Yuan et al. *Higher-order Structure Based Anomaly Detection on Attributed Networks*. 2021. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9671990.

[Din+19] Kaize Ding et al. “Deep Anomaly Detection on Attributed Networks”. In: (2019).

[FZL20] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. “ANOMALYDAE: DUAL AUTOENCODER FOR ANOMALY DETECTION ON ATTRIBUTED NETWORKS”. In: (2020). URL: https://github.com/haoyfan/AnomalyDAE.

[KIF20] Atsutoshi Kumagai, Tomoharu Iwata, and Yasuhiro Fujiwara. *Semi-supervised Anomaly Detection on Attributed Graphs*. Tech. rep. 2020.

[IY] Tomoharu Iwata and Yuki Yamanaka. *Supervised Anomaly Detection based on Deep Autoregressive Density Estimators*. Tech. rep.

[KIF19] Atsutoshi Kumagai, Tomoharu Iwata, and Yasuhiro Fujiwara. “Transfer Anomaly Detection by Inferring Latent Domain Representations”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[YCS16] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. *Revisiting Semi-Supervised Learning with Graph Embeddings*. Tech. rep. 2016.

[Ron+20] Yu Rong et al. *DropEdge: Towards Deep Graph Convolutional Networks on Node Classification*. Mar. 2020. URL: https://github.com/DropEdge/DropEdge..

[Aar22] Anna Fridtun Aarekol. “A graph theoretical approach to online predator detection”. In: (2022). URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3014533.

[Mov22c] MovieStarPlanet. *What are the Rules of MovieStarPlanet?* 2022. URL: https://moviestarplanet.zendesk.com/hc/en-us/articles/209992885-What-are-the-Rules-of-MovieStarPlanet-.

[HYL17] William L. Hamilton, Rex Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *Advances in Neural Information Processing Systems* 2017-December (June 2017), pp. 1025–1035. ISSN: 10495258. DOI: 10.48550/arxiv.1706.02216. URL: https://arxiv.org/abs/1706.02216v4.

[Les22] Jure Lescovec. *CS224W*. 2022. URL: http://web.stanford.edu/class/cs224w/index.html#content.

[SY14] Mayu Sakurada and Takehisa Yairi. “Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction”. In: (2014). DOI: 10.1145/2689746.2689747. URL: http://dx.doi.org/10.1145/2689746.2689747.

[KW16] Thomas N Kipf and Max Welling. “Variational Graph Auto-Encoders 1 A latent variable model for graph-structured data”. In: (2016).

[HSM19] Tue Herlau, Mikkel N Schmidt, and Morten Mørup. “Introduction to Machine Learning and Data Mining”. In: (2019).

[Ken38] Maurice Kendall. *A new measure of rank correlation*. 1938. URL: https://www.jstor.org/stable/pdf/2332226.pdf.

[WMZ10] William Webber, Alistair Moffat, and Justin Zobel. “A Similarity Measure for Indefinite Rankings”. In: (2010). DOI: 10.1145/1852102.1852106. URL: http://dx.doi.org/10.1145/1852102.1852106..

[BMA18] Per B Brockhoff, Jan K Møller, and Elisabeth W Andersen. “Introduction to Statistics at DTU”. In: (2018), pp. 368–376.

[Liu+22b] Kay Liu et al. “PyGOD: A Python Library for Graph Outlier Detection”. In: (Apr. 2022). DOI: 10.48550/arxiv.2204.12095. URL: https://arxiv.org/abs/2204.12095.

9 Appendix

9.1 Feature Aggregation

Feature Name	**Aggregation**	Value Range
num. active hours	sum	$\in [0, k]$
num. active days	sum	$\in [0,\ k \cdot 24]$
maximum out hourly	maximum	$\in [0,\ \infty[$
average out hourly	weighted average based on active hours	$\in [0,\ \infty[$ *value <= max. out hourly*
maximum in hourly	maximum	$\in [0,\ \infty[$
average in hourly	weighted average based on active hours	$\in [0,\ \infty[$ *value <= max. in hourly*
weighted max. out hourly	maximum	$\in [0,\ \infty[$
weighted average out hourly	weighted average based on active hours	$\in [0,\ \infty[$ *value <= weighted max. out hourly*
weighted max. in hourly	maximum	$\in [0,\ \infty[$
weighted avr. in hourly	weighted average based on active hours	$\in [0,\ \infty[$ *value <= weighted max. in hourly*
out degree 2	weighted average based on active hours	$\in [0,\ \infty[$ *value <= total conversations*
out degree 7	weighted average based on active hours	$\in [0,\ \infty[$ *value <= total conversations*
out degree 32	weighted average based on active hours	$\in [0,\ \infty[$ *value <= total conversations*
out degree 100	weighted average based on active hours	$\in [0,\ \infty[$ *value <= total conversations*
out degree 100 plus	weighted average based on active hours	$\in [0,\ \infty[$ *value <= total conversations*
weighted proportion outgoing	average based on active days	$\in [0,\ 1]$

Table 9.1: *Aggregation method and value range for each feature*

9.2 Implementation

The PyGOD library by Liu et al. [Liu+22b] is utilized for the model implementations. PyGOD (**Py**thon library for **g**raph **o**utlier **d**etection) is built for Python 3.6+ and depends on Pytorch and Pytorch Geometric (PyG) for effective graph learning on CPUs and GPUs.

We use the NetworkX library and PyTorch Geometric to build and implement the graphs. The graphs are represented as PyG Data objects, which consist of an attribute matrix $\mathbf{X}$ and an edge index $\mathcal{E}$.

Packages for python 3.6+

Package	Version
torch	>=1.10
pytorch geometric	>=2.0.3
networkx	>=2.6.3
numpy	>=1.19.4
scikit-learn	>=0.22.1
scipy	>=1.5.2

9.2.1 Hardware configuration

All gridsearch and test experiments were performed on a Azure Standard NV12s v3 (12 cores, 112 GB RAM, 336 GB disc) machine, expect Dominant which was run on a Azure Standard DS12 v2 (4 cores, 28 GB RAM, 56 GB disc) machine. All computation of daily features and graphs were also run on the latter machine.

9.3 Synthetic Graphs

9.3.1 Injection of Out-edges

Algorithm 2 Injecting out-edges for a single anomalous user in the graph

Inputs: $\mathcal{G}(\mathcal{V}, \mathcal{E}), u, \mathcal{G}_u(\mathcal{V}_u, \mathcal{E}_u)$ ▷ *Entire graph $\mathcal{G}$, user node and ego graph for node u*
Output: Updated $\mathcal{G}$

$out_{after} \leftarrow$ sampleFeatures() ▷ *Sample new anomalous out degree*
$\Delta out \leftarrow out_{before} - out_{after}$
if Δout is positive **then**
    $\mathcal{E}_{remove} \leftarrow$ Random sample Δout edges from $\mathcal{E}$
    $\mathcal{E}_u \leftarrow \mathcal{E}_u - \mathcal{E}_{remove}$
else if Δout is negative **then**
    $\mathcal{V}_{u,illegal} \leftarrow \mathcal{V}_{u,out} \cup \{u\}$
    $\mathcal{V}_{u,legal} \leftarrow \mathcal{V}_u - \mathcal{V}_{u,illegal}$
    if $|\Delta out| \leq |\mathcal{V}_{u,legal}|$ **then**
        $\mathcal{E}_{add} \leftarrow$ Randomly sample $|\Delta out|$ nodes $\mathcal{V}_{u,legal}$ and create edges to u
        $\mathcal{E}_u \leftarrow \mathcal{E}_u \cup \mathcal{E}_{add}$
    else
        $\mathcal{E}_{add,out} \leftarrow$ Sample all nodes $\mathcal{V}_{u,legal}$ and create edges to u
        $toAdd \leftarrow |\Delta out| - |\mathcal{V}_{u,legal}|$
        $\mathcal{E}_{add,random} \leftarrow$ Randomly sample $toAdd$ nodes $\mathcal{V}$ and create edges to u
        $\mathcal{E}_u \leftarrow \mathcal{E}_u \cup \mathcal{E}_{add,out} \cup \mathcal{E}_{add,random}$
    end if
else
    pass
end if

9.3.2 Three Day Synthetic Data Sets

name	*type*	*synthetic anoms.*	*isolated nodes*	*nodes after*	*active users after*	*new edges*	*edges after*
synthData15	train	790	75	31.524	24.766	33.578	189.425
synthData17	train	496	46	19.759	15.457	21.885	105.608
synthData18	train	405	31	16.149	12.929	18.825	90.677
synthData20	train	449	15	17.927	14.106	21.557	94.918
synthData21	train	413	20	16.475	12.940	19.303	85.450
synthData22	train	428	56	17.064	13.741	18.193	97.130
synthData23	train	422	35	16.833	13.352	19.694	94.237
synthData24	train	455	35	18.157	14.545	22.042	104.544

Table 9.2: *Overview of features of the different three-day synthetic graphs*

9.4 Gridsearch Results Summaries

9.4.1 Seven-Day Summaries

Grid-Search Results ROC-AUC								
Model	*Hyper parameters*							*Metric*
	α	*dropout*	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*num. layers*	*ROC-AUC*
MLPAE	-	0.3	8	0.1	1e-5	-	6	97.08 ± 0.73 (99.68)
GCNAE	-	**0**	**14**	**0.05**	**0.01**	**30**	**4**	$\mathbf{98.17 \pm 0.72}$ **(99.69)**
AnomalyDAE	0.8	0	14	0.05	0.01	30	-	96.85 ± 1.23 (99.20)
AdONE	-	0.3	10	0.1	1e-5	15	4	96.41 ± 1.71 (98.26)

Table 9.3: *ROC-AUC scores and hyper parameters for best performing seven-day runs in the hyper parameter grid search. Non-applicable parameters for a given model are marked by '-'*

Grid-Search Results Precision @ num. anomalies								
Model	*Hyper parameters*							*Metric*
	α	*dropout*	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*num. layers*	*Prec.@n. anom.*
MLPAE	-	0.3	8	0.1	1e-5	-	6	51.83 ± 14.66 (86.05)
GCNAE	-	0.3	10	0.05	0.01	30	4	52.37 ± 14.44 (86.18)
AnomalyDAE	**0.8**	**0**	**14**	**0.05**	**0.01**	**30**	-	$\mathbf{53.77 \pm 11.94}$ **(82.63)**
AdONE	-	0.3	10	0.1	1e-5	15	4	35.06 ± 14.95 (51.71)

Table 9.4: *Precision @ number of anomalies scores and hyper parameters for best performing seven-day runs in the hyper parameter grid search.*

9.4.2 Three-Day Summaries

Grid-Search Results ROC-AUC								
Model	*Hyper parameters*							*Metric*
	α	*dropout*	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*num. layers*	*ROC-AUC*
MLPAE	-	0.3	10	0.1	0.01	-	6	99.63 ± 0.01 (99.74)
GCNAE	-	**0.3**	**10**	**0.05**	**0.01**	**15**	**4**	$\mathbf{99.65 \pm 0.08}$ **(99.74)**
DOMINANT	0.8	0.3	10	0.1	1e-5	All	4	98.45 ± 0.16 (99.20)
AnomalyDAE	0.8	0	10	0.05	0.01	30	-	98.15 ± 1.33 (99.30)
AdONE	-	0.3	14	0.1	0.01	All	4	96.47 ± 1.46 (98.65)

Table 9.5: *ROC-AUC scores and hyper parameters for best performing three-day runs in the hyper parameter grid search.*

Grid-Search Results Precision @ num. anomalies								
Model	*Hyper parameters*							*Metric*
	α	*dropout*	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*num. layers*	*Prec.@n. anom.*
MLPAE	-	0.3	10	0.1	0.01	-	6	83.06 ± 3.55 (87.65)
GCNAE	-	**0**	**10**	**0.1**	**0.01**	**All**	**4**	$\mathbf{83.52 \pm 3.31}$ **(87.90)**
DOMINANT	0.8	0.3	10	0.1	1e-5	All	4	58.00 ± 3.48 (60.70)
AnomalyDAE	0.8	0	10	0.1	1e-5	30	-	80.96 ± 7.22 (89.90)
AdONE	-	0.3	10	0.001	0.01	15	4	44.33 ± 13.20 (71.80)

Table 9.6: *Precision @ number of anomalies scores and hyper parameters for best performing three-day runs in the hyper parameter grid search.*

9.5 Gridsearch Results for 7-Day Graphs

9.5.1 MLPAE

MLPAE Grid-Search Results ROC-AUC						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*ROC-AUC*
qpzgfsql	0.3	8	0.1	1e-5	6	97.08 ± 0.73 (99.68)
2g5alnhd	0	10	0.1	0.01	6	88.66 ± 3.33 (99.69)
3ums24rf	**0.3**	**8**	**0.1**	**0.01**	**6**	$\mathbf{89.80 \pm 3.07}$ **(99.69)**
2mpb89io	0.3	10	0.1	1e-5	4	80.09 ± 6.43 (99.68)
2h6meqkb	0.3	14	0.05	1e-5	6	78.47 ± 8.70 (99.68)

Table 9.7: *ROC-AUC scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Average Precision						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Avr. Prec.*
3ums24rf	**0.3**	**8**	**0.1**	**0.01**	**6**	$\mathbf{45.23 \pm 15.46}$ **(80.38)**
2g5alnhd	0	10	0.1	0.01	6	42.30 ± 17.69 (80.37)
1z7t1y3k	0.3	14	0.001	1e-5	6	39.61 ± 18.31 (81.90)
3t7rau1z	0.3	10	0.001	0.01	6	37.14 ± 17.92 (80.41)
2h6meqkb	0.3	14	0.05	1e-5	6	33.34 ± 22.94 (80.36)

Table 9.8: *Average Precision scores for top 5 best performing MLPAE models*

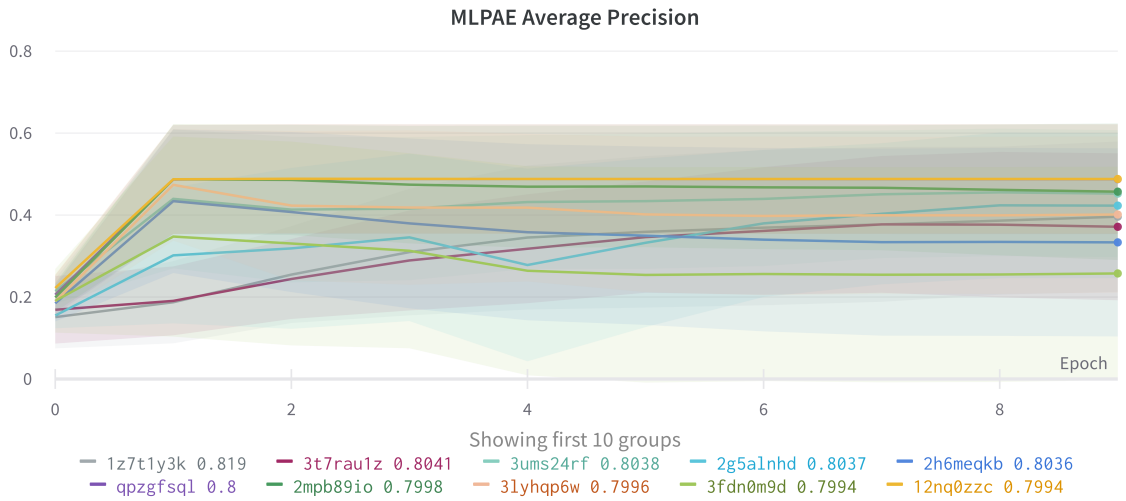


Figure 9.1: *Average Precision scores for top 10 best performing MLPAE models*

MLPAE Grid-Search Results Precision @ 50						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Prec. @ 50*
3t7rau1z	0.3	10	0.001	0.01	6	39.25 ± 16.52 (78.00)
1y741320	0.3	8	0.001	0.01	4	35.00 ± 15.78 (62.00)
1z7t1y3k	0.3	14	0.001	1e-5	6	34.50 ± 20.02 (70.00)
28nvqu2b	0.3	14	0.001	0.01	6	34.50 ± 16.38 (68.00)
294hbxw	0.3	8	0.001	0.01	6	31.50 ± 21.00 (74.00)

Table 9.9: *Precision @ 50 scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Recall @ 50						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Rec. @ 50*
3t7rau1z	0.3	10	0.001	0.01	6	2.43 ± 1.10 (5.34)
28nvqu2b	0.3	14	0.001	0.01	6	2.19 ± 1.28 (4.65)
1y741320	0.3	8	0.001	0.01	4	2.17 ± 1.15 (4.44)
1z7t1y3k	0.3	14	0.001	1e-5	6	2.09 ± 1.49 (4.79)
294h5bxw	0.3	8	0.001	0.01	6	2.03 ± 1.52 (5.06)

Table 9.10: *Recall @ 50 scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Precision @ 100						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Prec. @ 100*
28nvqu2b	0.3	14	0.001	0.01	6	39.88 ± 14.37 (72)
3t7rau1z	0.3	10	0.001	0.01	6	39.37 ± 16.18 (760)
3cvynhbb	0.3	14	0.1	0.01	6	38.50 ± 15.27 (69)
1z7t1y3k	0.3	14	0.001	1e-5	6	35.88 ± 19.87 (73)
294h5bxw	0.3	8	0.001	0.01	6	34.12 ± 18.92 (74)

Table 9.11: *Precision @ 100 scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Recall @ 100						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Rec. @ 100*
28nvqu2b	0.3	14	0.001	0.01	6	4.97 ± 2.42 (9.85)
3t7rau1z	0.3	10	0.001	0.01	6	4.91 ± 2.63 (10.4)
3cvynhbb	0.3	14	0.1	0.01	6	4.86 ± 2.45 (9.44)
1z7t1y3k	0.3	14	0.001	1e-5	6	4.47 ± 3.05 (9.99)
294h5bxw	0.3	8	0.001	0.01	6	4.37 ± 2.88 (10.12)

Table 9.12: *Recall @ 100 scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Precision @ num. anomalies						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Prec.@n. anom.*
qpzgfsql	0.3	8	0.1	1e-5	6	51.83 ± 14.66 (86.05)
3ums24rf	**0.3**	**8**	**0.1**	**0.01**	**6**	$\mathbf{51.08 \pm 14.91}$ **(86.18)**
2mpb89io	0.3	10	0.1	1e-5	4	49.94 ± 15.99 (86.05)
2g5alnhd	0	10	0.1	0.01	6	45.62 ± 18.12 (86.32)
2h6meqkb	0.3	14	0.05	1e-5	6	40.27 ± 21.12 (85.77)

Table 9.13: *Precision @ n. anomalies scores for top 5 best performing MLPAE models*

9.5.2 GCNAE

GCNAE Grid-Search Results ROC-AUC						
Model	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*ROC-AUC*
hce90n97	0	14	0.05	0.01	30	98.17 ± 0.72 (99.69)
rrwtx0xo	0	14	0.05	0.01	15	98.17 ± 0.72 (99.69)
1pxj3sso	0	14	0.05	0.01	-1	98.17 ± 0.72 (99.69)
ahjwkc5r	0	10	0.05	0.01	-1	98.17 ± 0.72 (99.69)
10xvb5bj	0.3	10	0.05	0.01	30	98.17 ± 0.72 (99.69)

Table 9.14: *ROC-AUC scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Average Precision						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Avr. Prec.*
2rne4j2u	**0**	**14**	**0.05**	**0.01**	**-1**	**49.26 ± 13.38 (80.42)**
hce90n97	0	14	0.05	0.01	30	49.26 ± 13.40 (80.44)
1pxj3sso	0.3	14	0.05	0.01	-1	49.24 ± 13.39 (80.42)
ahjwkc5r	0	10	0.05	0.01	-1	49.24 ± 13.40 (80.42)
10xvb5bj	0.3	10	0.05	0.01	30	49.22 ± 13.41 (80.43)

Table 9.15: *Average Precision scores for top 5 best performing GCNAE models*

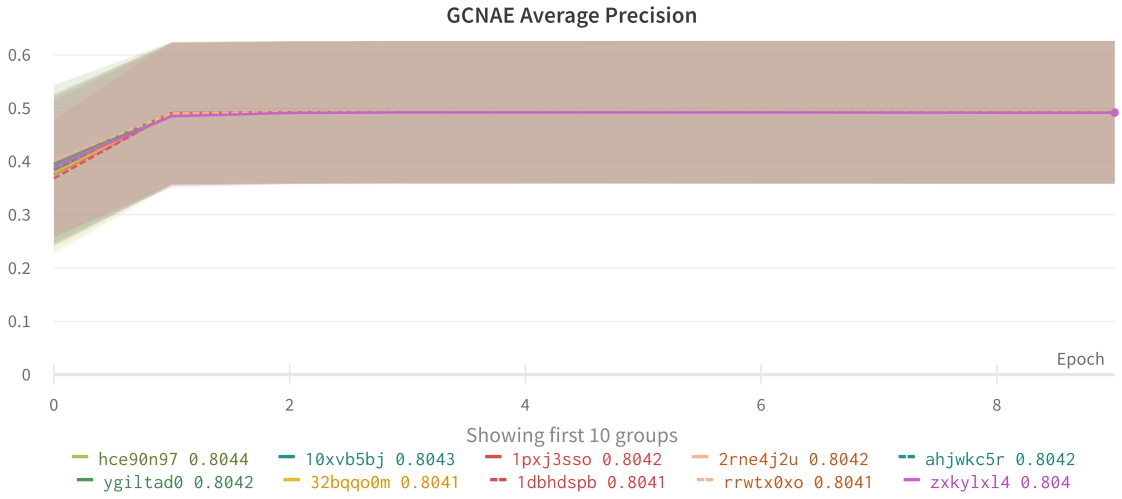


Figure 9.2: *Average Precision scores for top 10 best performing GCNAE models*

GCNAE Grid-Search Results Precision @ 50						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Prec. @ 50*
18reiohh	0	10	0.05	0.01	30	36.75 ± 15.56 (58)
10xvb5bj	0.3	10	0.05	0.01	30	35.75 ± 15.40 (58)
1dbhdspb	0.3	14	0.05	0.01	15	35.75 ± 15.40 (58)
1c70ejtb	0.3	14	0.1	0.01	30	35.50 ± 15.00 (58)
1ne4p81k	0.3	10	0.05	1e-5	-1	12.25 ± 18.71 (58)

Table 9.16: *Precision @ 50 scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Recall @ 50						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Rec. @ 50*
18reiohh	0	10	0.05	0.01	30	2.35 ± 1.17 (3.97)
10xvb5bj	0.3	10	0.05	0.01	30	2.29 ± 1.17 (3.97)
1dbhdspb	0.3	14	0.05	0.01	15	2.29 ± 1.17 (3.97)
1c70ejtb	0.3	14	0.1	0.01	30	2.25 ± 1.12 (3.97)
1ne4p81k	0.3	10	0.05	1e-5	-1	0.81 ± 1.29 (3.97)

Table 9.17: *Recall @ 50 scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Precision @ 100						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Prec. @ 100*
32bqqo0m	0.3	14	0.05	0.01	30	40.88 ± 14.87 (69)
2nxh6dmc	0.3	10	0.05	0.01	15	40.63 ± 14.84 (69)
10xvb5bj	0.3	10	0.05	0.01	30	40.37 ± 14.87 (69)
1pxj3sso	0.3	14	0.05	0.01	-1	40.37 ± 14.84 (69)
1ne4p81k	0.3	10	0.05	1e-5	-1	18.25 ± 21.76 (69)

Table 9.18: *Precision @ 100 scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Recall @ 100						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Rec. @ 100*
32bqqo0m	0.3	14	0.05	0.01	30	5.15 ± 2.44 (9.44)
2nxh6dmc	0.3	10	0.05	0.01	15	5.12 ± 2.43 (9.44)
1pxj3sso	0.3	14	0.05	0.01	-1	5.09 ± 2.44 (9.44)
10xvb5bj	0.3	10	0.05	0.01	30	5.08 ± 2.43 (9.44)
1ne4p81k	0.3	10	0.05	1e-5	-1	2.37 ± 3.03 (9.44)

Table 9.19: *Recall @ 100 scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Precision @ num. anomalies						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Prec.@n. anom.*
10xvb5bj	0.3	10	0.05	0.01	30	52.37 ± 14.44 (86.18)
2nxh6dmc	0.3	10	0.05	0.01	15	52.34 ± 14.50 (86.32)
ygiltad0	0	10	0.05	0.01	15	52.31 ± 14.54 (86.32)
hce90n97	0	14	0.05	0.01	30	52.27 ± 14.53 (86.32)
1dphdspb	0.3	14	0.05	0.01	15	52.24 ± 14.56 (86.32)

Table 9.20: *Precision/Recall @ number of anomalies scores for top 5 best performing GCNAE models*

9.5.3 AnomalyDAE

AnomalyDAE Grid-Search Results ROC-AUC							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*ROC-AUC*
gml85dyg	**0.8**	**0**	**14**	**0.05**	**30**	**0.01**	$\mathbf{96.85 \pm 1.23}$ **(99.20)**
3abjk5c1	0.8	0	10	0.05	30	0.01	96.68 ± 1.49 (99.21)
2lo9jm4i	0.8	0.3	10	0.1	30	1e-5	95.64 ± 2.14 (98.93)
evruipyb	0.8	0	10	0.1	30	1e-5	95.52 ± 1.82 (98.93)
288fdx7c	0.8	0.3	14	0.05	30	1e-5	95.43 ± 1.96 (98.97)

Table 9.21: *ROC-AUC scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Average Precision							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Avr. Prec.*
gml85dyg	**0.8**	**0**	**14**	**0.05**	**30**	**0.01**	$\mathbf{50.52 \pm 11.66}$ **(79.17)**
3abjk5cl	0.8	0	10	0.05	30	0.01	50.04 ± 11.82 (79.13)
1y12542c	0.8	0.3	10	0.05	30	1e-5	42.23 ± 15.43 (77.64)
288fdx7c	0.8	0.3	14	0.05	30	1e-5	42.06 ± 15.3 (76.91)
1fvcrms	0.8	0	14	0.05	-1	1e-5	34.74 ± 19.78 (77.02)

Table 9.22: *Average Precision scores for top 5 best performing AnomalyDAE models*

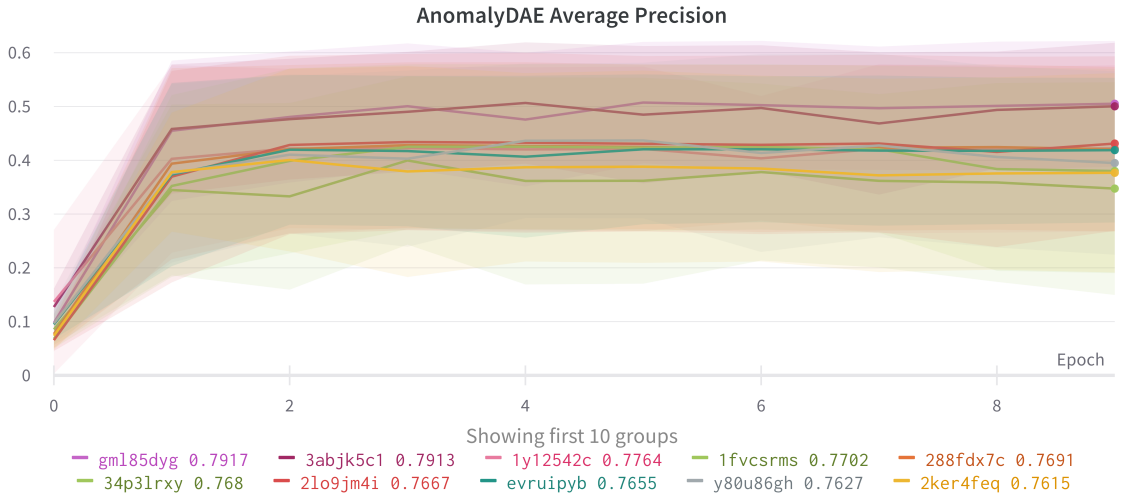


Figure 9.3: *Average Precision scores for top 10 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Precision @ 50							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Prec. @ 50*
1ewr2j7t	0.2	0	14	0.05	-1	0.01	51.25 ± 12.91 (78.0)
g6j5qvfk	0.2	0.3	14	0.05	-1	1e-5	50.25 ± 16.44 (88.0)
2k7ux7wm	0.2	0.3	10	0.05	-1	1e-5	49.43 ± 14.46 (78.0)
1w0zs8m3	0.2	0	10	0.1	-1	1e-5	43.75 ± 20.77 (78.0)
1r2yjwgc	0.2	0	10	0.05	-1	1e-5	39.5 ± 27.08 (78.0)

Table 9.23: *Precision @ 50 scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Recall @ 50							
Model	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Rec. @ 50*
lewr2j7t	0.2	0	14	0.05	-1	0.01	3.16 ± 1.23 (5.34)
g6j5qvfk	0.2	0.3	14	0.05	-1	1e-5	3.11 ± 1.43 (6.02)
2k7ux7wm	0.2	0.3	10	0.05	-1	1e-5	2.99 ± 1.35 (5.34)
1w0zs8m3	0.2	0	10	0.1	-1	1e-5	2.75 ± 1.60 (5.34)
1r2yjwgc	0.2	0	10	0.05	-1	1e-5	2.32 ± 1.88 (5.34)

Table 9.24: *Recall @ 50 scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Precision @ 100							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Prec. @ 100*
5psezf77	0.8	0	10	0.05	30	1e-5	53.22 ± 15.1 (81.0)
y80u86gh	0.8	0	14	0.05	30	1e-5	51.78 ± 21.55 (84.0)
34n4q1fr	0.8	0	10	0.1	-1	1e-5	45.3 ± 24.56 (79.0)
2ker4feq	0.8	0	14	0.1	30	1e-5	44.67 ± 27.14 (79.0)
299181r5	0.8	0.3	14	0.05	30	0.01	38.75 ± 25.45 (79.0)

Table 9.25: *Precision@100 scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Recall @ 100							
Model	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Rec. @ 100*
y80u86gh	0.8	0	14	0.05	30	1e-5	6.76 ± 3.24 (11.49)
5psezf77	0.8	0	10	0.05	30	1e-5	6.69 ± 2.71 (11.08)
34n4q1fr	0.8	0	10	0.1	-1	1e-5	5.94 ± 3.48 (10.81)
2ker4feq	0.8	0	14	0.1	30	1e-5	5.74 ± 3.79 (11.08)
299181r5	0.8	0.3	14	0.05	30	0.01	4.81 ± 3.46 (10.81)

Table 9.26: *Recall @ 100 scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Precision @ num. anomalies							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Prec.@n. anom.*
gml85dyg	**0.8**	**0**	**14**	**0.05**	**30**	**0.01**	$\mathbf{53.77 \pm 11.94}$ **(82.63)**
3abjk5c1	0.8	0	10	0.05	30	0.01	53.27 ± 11.82 (82.76)
5psezf77	0.8	0	10	0.05	30	1e-5	46.54 ± 14.24 (79.21)
y80u86gh	0.8	0	14	0.05	30	1e-5	41.56 ± 19.88 (79.34)
2ker4feq	0.8	0	14	0.1	30	1e-5	39.3 ± 22.08 (79.21)

Table 9.27: *Precision @ n. anomalies scores for top 5 best performing AnomalyDAE models*

9.5.4 AdONE

AdONE Grid-Search Results ROC-AUC							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*ROC-AUC*
2e5oh830	**0.3**	**10**	**0.1**	**15**	**1e-5**	**4**	$\mathbf{96.41 \pm 1.71}$ **(98.26)**
zxkg62ze	0	10	0.1	30	1e-5	4	95.27 ± 2.02 (98.26)
3slxgzn0	0.3	10	0.1	-1	0.01	4	95.27 ± 1.47 (98.43)
15qe89ki	0	14	0.1	15	1e-5	4	94.73 ± 1.90 (98.25)
34ummcmz	0	10	0.1	-1	1e-5	4	94.54 ± 3.83 (98.25)

Table 9.28: *ROC-AUC scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Avr. Precision							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Avr. Prec*
2e5oh830	**0.3**	**10**	**0.1**	**15**	**1e-5**	**4**	**33.99 ± 13.7 (49.51)**
zxkg62ze	0	10	0.1	30	1e-5	4	28.87 ± 10.48 (49.58)
3slxgzn0	0.3	10	0.1	-1	0.01	4	27.58 ± 10.82 (51.81)
34ummcmz	0	10	0.1	-1	1e-5	4	27.25 ± 10.34 (49.53)
15qe89ki	0	14	0.1	15	1e-5	4	26.36 ± 10.40 (49.50)

Table 9.29: *Average Precision scores for top 5 best performing AdONE models*

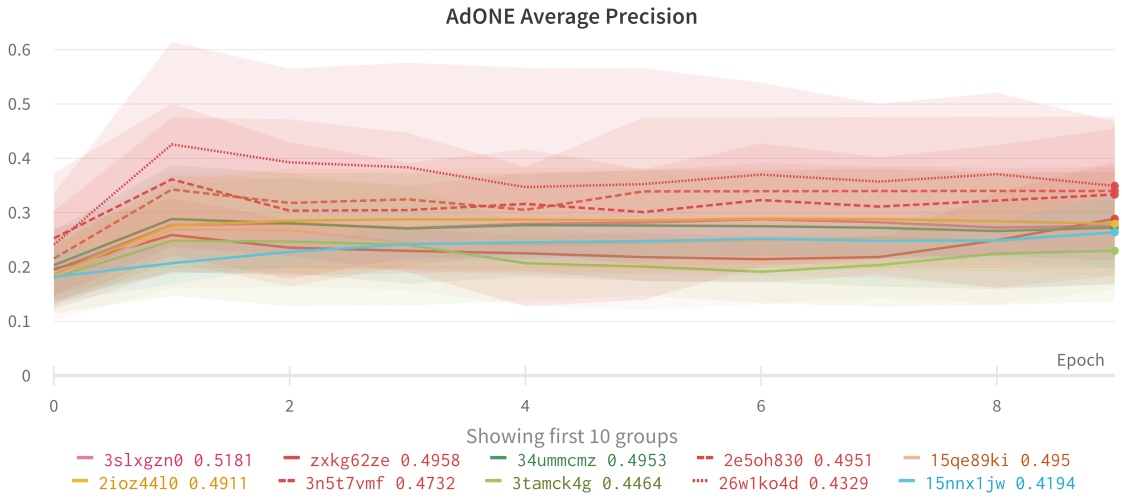


Figure 9.4: *Average Precision scores for top 10 best performing AdONE models*

AdONE Grid-Search Results Precision @ 50							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Prec.@50*
31k8o5kc	0.3	10	0.001	-1	0.01	4	20.50 ± 15.18 (56.0)
3slxgzn0	0.3	10	0.1	-1	0.01	4	16.86 ± 12.59 (44.0)
34ummcmz	0	10	0.1	-1	1e-5	4	15.25 ± 11.76 (42.0)
15nnx1jw	0.3	10	0.05	15	0.01	4	14.25 ± 12.02 (42.0)
1zgj0cud	0.3	14	0.1	-1	1e-5	4	12.00 ± 5.66 (16.0)

Table 9.30: *Precision @ 50 scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Recall @ 50							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Rec.@50*
31k8o5kc	0.3	10	0.001	-1	0.01	4	1.32 ± 1.11 (3.83)
3slxgzn0	0.3	10	0.1	-1	0.01	4	1.07 ± 0.92 (3.01)
34ummcmz	0	10	0.1	-1	1e-5	4	0.98 ± 0.84 (2.87)
15nnx1jw	0.3	10	0.05	15	0.01	4	0.91 ± 0.85 (2.87)
1zgj0cud	0.3	14	0.1	-1	1e-5	4	0.73 ± 0.59 (1.14)

Table 9.31: *Recall @ 50 scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Precision @ 100							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Prec.@100*
2e5oh830	**0.3**	**10**	**0.1**	**15**	**1e-5**	**4**	**27.67 ± 20.31 (51.0)**
zxkg62ze	0	10	0.1	30	1e-5	4	21.50 ± 15.22 (51.0)
3slxgzn0	0.3	10	0.1	-1	0.01	4	19.00 ± 15.68 (53.0)
15qe89ki	0	14	0.1	15	1e-5	4	17.14 ± 15.87 (51.0)
lzgj0cud	0.3	14	0.1	-1	1e-5	4	13.00 ± 9.90 (20.0)

Table 9.32: *Precision @ 100 scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Recall @ 100							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Rec.@100*
2e5oh830	**0.3**	**10**	**0.1**	**15**	**1e-5**	**4**	**3.58 ± 2.94 (6.98)**
zxkg62ze	0	10	0.1	30	1e-5	4	2.93 ± 2.12 (6.98)
3slxgzn0	0.3	10	0.1	-1	0.01	4	2.44 ± 2.28 (7.25)
15qe89ki	0	14	0.1	15	1e-5	4	2.17 ± 2.29 (6.98)
lzgj0cud	0.3	14	0.1	-1	1e-5	4	1.67 ± 1.69 (2.86)

Table 9.33: *Recall @ 100 scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Precision @ num. anomalies							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	Prec.@n. anom.
2e5oh830	**0.3**	**10**	**0.1**	**15**	**1e-5**	**4**	**35.06 ± 14.95** (51.71)
zxkg62ze	0	10	0.1	30	1e-5	4	30.53 ± 10.84 (52.12)
34ummcmz	0	10	0.1	-1	1e-5	4	28.02 ± 10.90 (52.26)
3slxgzn0	0.3	10	0.1	-1	0.01	4	27.81 ± 11.84 (54.17)
15qe89ki	0	14	0.1	15	1e-5	4	26.92 ± 11.46 (52.26)

Table 9.34: *Precision @ n. anomalies scores for top 5 best performing AdONE models*

9.6 Gridsearch Results for 3-Day Graphs

9.6.1 MLPAE

MLPAE Grid-Search Results ROC-AUC						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*ROC-AUC*
vcb34wm3	0.3	10	0.1	0.01	6	99.63 ± 0.01 (99.74)
u3ia2w3h	**0**	**8**	**0.001**	**0.01**	**4**	97.58 ± 2.86 (99.79)
2wuctwwh	0	8	0.001	1e-5	4	97.46 ± 2.42 (99.79)
3f6oazg8	0	10	0.001	1e-5	6	97.18 ± 2.76 (99.79)
bpg9bu9e	0	14	0.001	1e-5	4	95.30 ± 3.71 (99.85)

Table 9.35: *ROC-AUC scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Average Precision						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Avr. Prec.*
u3ia2w3h	**0**	**8**	**0.001**	**0.01**	**4**	**59.78 ± 16.87 (87.62)**
3f6oazg8	0	10	0.001	1e-5	6	52.93 ± 22.74 (88.08)
2wuctwwh	0	8	0.001	1e-5	4	50.01 ± 24.04 (88.08)
2ecbujay	0	14	0.001	1e-5	6	47.23 ± 27.81 (85.26)
bpg9bu9e	0	14	0.001	1e-5	4	37.82 ± 27.90 (90.64)

Table 9.36: *Average Precision scores for top 5 best performing MLPAE models*

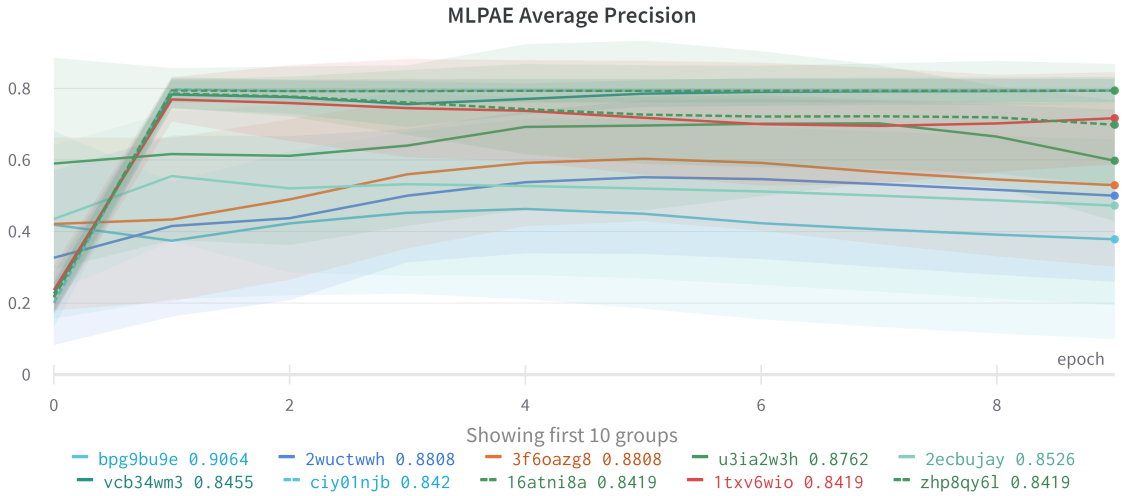


Figure 9.5: *Average Precision scores for top 10 best performing MLPAE models*

MLPAE Grid-Search Results Precision @ 50						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Prec. @ 50*
2zj8k87a	0.3	8	0.001	0.01	4	65.75 ± 23.75 (96)
348fnf24	0.3	10	0.001	1e-5	6	61.5 ± 20.39 (88)
u3ia2w3h	**0**	**8**	**0.001**	**0.01**	**4**	**53.75 ± 27.41 (88)**
3tsi8yqq	0.3	8	0.001	1e-5	4	51.75 ± 26.97 (88)
bpg9bu9e	0	14	0.001	1e-5	4	27.00 ± 28.45 (88)

Table 9.37: *Precision @ 50 scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Recall @ 50						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Rec. @ 50*
2zj8k87a	0.3	8	0.001	0.01	4	7.23 ± 3.16 (11.62)
37a6krmd	0.3	14	0.001	0.01	6	5.69 ± 3.21 (9.93)
3tsi8yqq	0.3	8	0.001	1e-5	4	5.65 ± 3.34 (9.8)
3hw5b6oq	0.3	14	0.001	1e-5	6	5.52 ± 3.45 (9.95)
3f6oazg8	0	10	0.001	1e-5	6	5.51 ± 3.50 (10.17)

Table 9.38: *Recall @ 50 scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Precision @ 100						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Prec. @ 100*
215eriv1	0.3	14	0.001	0.01	4	68.13 ± 13.08 (88)
2zj8k87a	0.3	8	0.001	0.01	4	64.88 ± 21.01 (97)
3f6oazg8	0	10	0.001	1e-5	6	53.75 ± 28.61 (91)
2wuctwwh	0	8	0.001	1e-5	4	42.62 ± 31.60 (91)
bpg9bu9e	0	14	0.001	1e-5	4	31.00 ± 29.03 (88)

Table 9.39: *Precision @ 100 scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Recall @ 100						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Rec. @ 100*
2tdamf51	0	8	0.001	0.01	4	14.35 ± 5.55 (20.49)
2zj8k87a	0.3	8	0.001	0.01	4	14.27 ± 5.83 (23.49)
37a6krmd	0.3	14	0.001	0.01	6	13.01 ± 4.96 (20.34)
3f6oazg8	0	10	0.001	1e-5	6	12.24 ± 7.40 (22.03)
2ecbujay	0	14	0.001	1e-5	6	9.44 ± 7.20 (20.34)

Table 9.40: *Recall @ 100 scores for top 5 best performing MLPAE models*

MLPAE Grid-Search Results Precision @ num. anomalies						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim.*	*lr*	*weight decay*	*num. layers*	*Prec.@n. anom.*
fy9tbmph	0.3	10	0.1	0.01	6	83.06 ± 3.55 (87.65)
3ch41a3r	0.3	8	0.001	0.01	6	67.62 ± 16.99 (88.35)
u3ia2w3h	**0**	**8**	**0.001**	**0.01**	**4**	$\mathbf{63.63 \pm 15.07}$ **(89.31)**
2wuctwwh	0	8	0.001	1e-5	4	52.35 ± 24.65 (87.97)
bpg9bu9e	0	14	0.001	1e-5	4	36.57 ± 32.55 (92.34)

Table 9.41: *Precision @ n. anomalies scores for top 5 best performing MLPAE models*

9.6.2 GCNAE

GCNAE Grid-Search Results ROC-AUC						
Model	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*ROC-AUC*
mi999b0v	**0.3**	**10**	**0.05**	**0.01**	**15**	**99.65 ± 0.08 (99.74)**
1y35x822	0	10	0.05	0.01	30	99.65 ± 0.08 (99.74)
38ct2ndq	0	10	0.05	0.01	15	99.65 ± 0.08 (99.74)
f8a9i8ug	0	14	0.05	0.01	30	99.65 ± 0.08 (99.74)
o03u5b9x	0.3	10	0.001	0.01	15	99.59 ± 0.15 (99.78)

Table 9.42: *ROC-AUC scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Average Precision						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Avr. Prec.*
mi999b0v	**0.3**	**10**	**0.05**	**0.01**	**15**	**79.90 ± 3.22 (84.54)**
38ct2ndq	0	10	0.05	0.01	15	79.89 ± 3.23 (84.54)
4izvliey	0.3	10	0.05	0.01	All	79.89 ± 3.21 (84.52)
f8a9i8ug	0	14	0.05	0.01	30	79.88 ± 3.24 (84.53)
3qzwa7y5	0.3	10	0.1	0.01	15	79.85 ± 3.23 (84.51)

Table 9.43: *Average Precision scores for top 5 best performing GCNAE models*

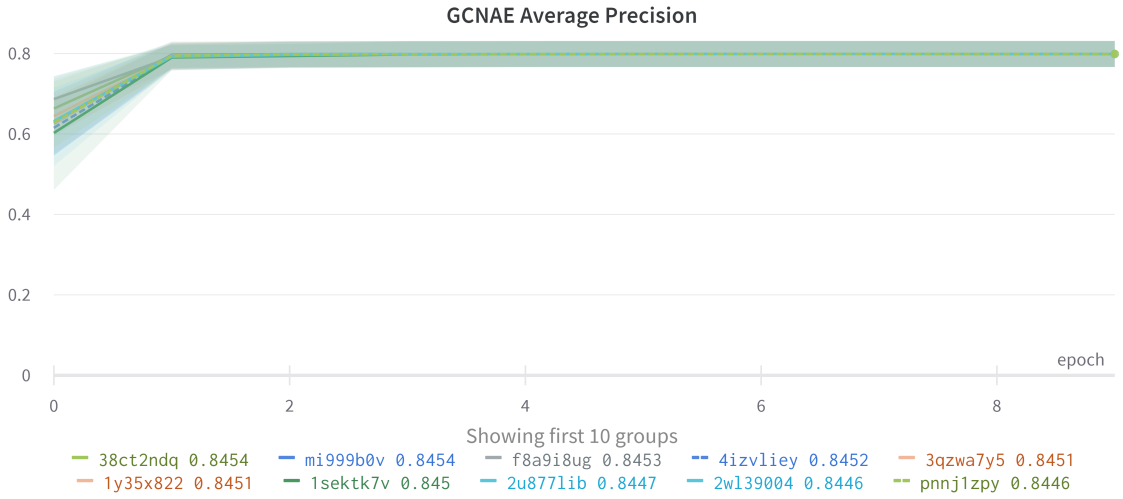


Figure 9.6: *Average Precision scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Precision @ 50						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Prec. @ 50*
1dlj96t6	0.3	14	0.05	0.01	15	67.00 ± 13.56 (80)
1ozq6vkk	0.3	14	0.1	0.01	30	66.75 ± 13.60 (80)
1752ircr	0.3	14	0.1	0.01	15	66.50 ± 12.82 (80)
czo2188r	0	14	0.001	0.01	15	64.00 ± 14.38 (82)
10e5j7do	0.3	10	0.001	0.01	30	62.25 ± 14.83 (80)

Table 9.44: *Precision @ 50 scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Recall @ 50						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Rec. @ 50*
1nbf3c7s	0	10	0.05	0.01	30	7.40 ± 2.19 (9.44)
1s9b29e1	0.3	10	0.1	0.01	All	7.36 ± 2.18 (9.44)
12rlmuqs	0	14	0.1	0.01	30	7.36 ± 2.17 (9.44)
1l4qsaou	0.3	10	0.1	1e-5	30	7.08 ± 2.67 (9.44)
1dcax052	0.3	14	0.1	1e-5	-1	6.14 ± 2.94 (9.44)

Table 9.45: *Recall @ 50 scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Precision @ 100						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Prec. @ 100*
1qtvhyry	0	10	0.1	0.01	15	76.00 ± 8.49 (85)
1752ircr	0.3	14	0.1	0.01	15	75.87 ± 8.15 (85)
31ojwjnk	0	10	0.1	0.01	All	75.75 ± 8.31 (85)
1piozbop	0	14	0.1	0.01	30	75.75 ± 8.41 (85)
2a7z1133	0.3	10	0.1	1e-5	15	72.88 ± 10.00 (85)

Table 9.46: *Precision @ 100 scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Recall @ 100						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Rec. @ 100*
1s9b29e1	0.3	10	0.1	0.01	All	16.62 ± 4.01 (19.85)
12rlmuqs	0	14	0.1	0.01	30	16.60 ± 4.05 (19.85)
1nbf3c7s	0	10	0.05	0.01	30	16.55 ± 4.01 (19.85)
1l4qsaou	0.3	10	0.1	1e-5	30	16.10 ± 5.28 (19.85)
1dcax052	0.3	14	0.1	1e-5	All	14.60 ± 5.05 (19.85)

Table 9.47: *Recall @ 100 scores for top 5 best performing GCNAE models*

GCNAE Grid-Search Results Precision @ num. anomalies						
Model ID	*Hyperparameters*					*Metric*
	dropout	*hidden dim*	*lr*	*weight decay*	*num. neighbors*	*Prec.@n. anom.*
3oryulop	0	10	0.1	0.01	All	83.52 ± 3.31 (87.90)
1bd71ks5	0.3	14	0.1	0.01	15	83.46 ± 3.20 (87.65)
11n8ow8v	0.3	10	0.1	0.01	15	83.45 ± 3.25 (87.65)
ltx3ei6a	0	14	0.1	0.01	30	83.40 ± 3.35 (87.90)
o03u5b9x	0.3	10	0.001	0.01	15	82.81 ± 4.15 (89.83)

Table 9.48: *Precision/Recall @ number of anomalies scores for top 5 best performing GCNAE models*

9.6.3 DOMINANT

DOMINANT Grid-Search Results ROC-AUC							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*hidden. dim*	*lr*	*num. neighbors*	*weight decay*	*ROC-AUC*
18nx5bjb	**0.8**	**0.3**	**10**	**0.1**	**-1**	**1e-5**	$\mathbf{98.45 \pm 0.16}$ **(98.6)**
kiikbidv	0.8	0.3	10	0.05	-1	1e-5	98.41 ± 0.14 (98.5)
1kj8oew3	0.8	0.3	10	0.05	15	1e-5	98.31 ± 0.17 (98.5)
32xiprzd	0.8	0	10	0.1	-1	1e-5	98.27 ± 0.29 (98.5)
fzaqw1z2	0.8	0.3	10	0.1	15	1e-5	98.26 ± 0.15 (98.5)

Table 9.49: *ROC-AUC scores for top 5 best performing DOMINANT models*

DOMINANT Grid-Search Results Average Precision							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*hidden. dim*	*lr*	*num. neighbors*	*weight decay*	Avr. Prec
18nx5bjb	**0.8**	**0.3**	**10**	**0.1**	**-1**	**1e-5**	$\mathbf{54.56 \pm 2.00}$ **(56.3)**
24l00z71	0.8	0.3	10	0.05	-1	1e-5	53.70 ± 1.98 (55.0)
1qq3wahm	0.8	0.3	10	0.05	15	1e-5	52.42 ± 2.70 (54.5)
32xiprzd	0.8	0	10	0.1	-1	1e-5	50.92 ± 5.55 (55.4)
1mwvtkyy	0.8	0	10	0.05	-1	1e-5	47.94 ± 5.28 (55.1)

Table 9.50: *Average Precision scores for top 5 best performing DOMINANT models*

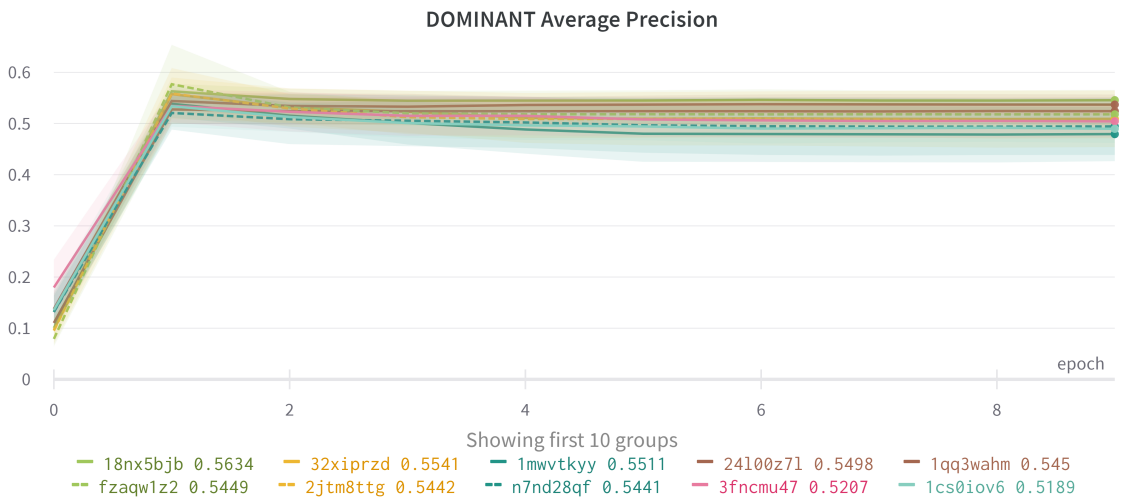


Figure 9.7: *Average Precision scores for top 5 best performing DOMINANT models*

DOMINANT Grid-Search Results Precision @ 50							
Model	*Hyperparameters*						*Metric*
	α	dropout	hidden_dim	lr	num. neighbors	w_decay	Prec.@50
18nx5bjb	**0.8**	**0.3**	**10**	**0.1**	**-1**	**1e-5**	**48.80 ± 1.79 (52.0)**
24l00z71	0.8	0.3	10	0.05	-1	1e-5	48.00 ± 1.63 (50.0)
1kj8oew3	0.8	0.3	10	0.05	15	1e-5	46.00 ± 8.87 (50.0)
29pbxi92	0.8	0	10	0.05	15	1e-5	42.00 ± 11.25 (50.0)
3a39wz24	0.8	0	10	0.05	-1	1e-5	33.71 ± 14.58 (54.0)

Table 9.51: *Precision @ 50 scores for top 5 best performing DOMINANT models*

DOMINANT Grid-Search Results Recall @ 50							
Model	*Hyperparameters*						*Metric*
	α	dropout	hidden_dim	lr	num. neighbors	w_decay	Rec.@50
18nx5bjb	**0.8**	**0.3**	**10**	**0.1**	**-1**	**1e-5**	**5.57 ± 0.58 (6.4)**
1qq3wahm	0.8	0.3	10	0.05	15	1e-5	5.03 ± 1.52 (5.9)
2jtm8ttg	0.8	0	10	0.1	15	1e-5	4.79 ± 1.30 (5.9)
32xiprzd	0.8	0	10	0.1	-1	1e-5	4.25 ± 1.83 (5.9)
3a39wz24	0.8	0	10	0.05	-1	1e-5	3.75 ± 1.79 (6.0)

Table 9.52: *Recall @ 50 scores for top 5 best performing DOMINANT models*

DOMINANT Grid-Search Results Precision @ 100							
Model	*Hyperparameters*						*Metric*
	α	dropout	hidden_dim	lr	num. neighbors	w_decay	Prec.@100
18nx5bjb	**0.8**	**0.3**	**10**	**0.1**	**-1**	**1e-5**	**58.80 ± 3.90 (63.0)**
1qq3wahm	0.8	0.3	10	0.05	15	1e-5	55.14 ± 8.42 (64.0)
2jtm8ttg	0.8	0	10	0.1	15	1e-5	53.00 ± 8.17 (63.0)
32xiprzd	0.8	0	10	0.1	-1	1e-5	49.29 ± 13.44 (63.0)
3a39wz24	0.8	0	10	0.05	-1	1e-5	42.43 ± 15.20 (62.0)

Table 9.53: *Precision @ 100 scores for top 5 best performing DOMINANT models*

DOMINANT Grid-Search Results Recall @ 100							
Model	*Hyperparameters*						*Metric*
	α	dropout	hidden_dim	lr	num. neighbors	w_decay	Rec.@100
18nx5bjb	0.8	0.3	10	0.1	-1	1e-5	13.41 ± 1.55 (15.56)
1qq3wahm	0.8	0.3	10	0.05	15	1e-5	11.98 ± 3.43 (15.80)
2jtm8ttg	0.8	0	10	0.1	15	1e-5	11.44 ± 3.24 (15.56)
n7nd28qf	0.8	0	10	0.05	15	1e-5	10.91 ± 3.80 (15.31)
32xiprzd	0.8	0	10	0.1	-1	1e-5	10.80 ± 4.00 (15.56)

Table 9.54: *Recall @ 100 scores for top 5 best performing DOMINANT models*

DOMINANT Grid-Search Results Precision @ num. anomalies							
Model	*Hyperparameters*						*Metric*
	α	dropout	hidden_dim	lr	num. neighbors	w_decay	Prec.@n. anom.
7x0w5a6a	0.8	0.3	10	0.1	-1	1e-5	58.00 ± 3.48 (60.7)
24l00z7l	0.8	0.3	10	0.05	-1	1e-5	57.69 ± 3.03 (60.0)
1kj8oew3	0.8	0.3	10	0.05	15	1e-5	55.58 ± 3.32 (59.3)
2rodmn5v	0.8	0	10	0.1	-1	1e-5	54.81 ± 5.46 (60.0)
1mwvtkyy	0.8	0	10	0.05	-1	1e-5	52.34 ± 4.83 (58.5)

Table 9.55: *Precision @ n. anomalies scores for top 5 best performing DOMINANT models*

9.6.4 AnomalyDAE

AnomalyDAE Grid-Search Results ROC-AUC							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*ROC-AUC*
2t5rb7gy	0.8	0	10	0.05	30	0.01	98.15 ± 1.33 (99.3)
1uk3wj5d	0.8	0	14	0.05	30	0.01	98.15 ± 1.44 (99.6)
3dcdcnep	0.8	0	10	0.05	-1	0.01	97.91 ± 1.26 (99.5)
1lg0qxgv	0.8	0	14	0.05	-1	0.01	97.87 ± 1.50 (99.4)
2nc1dl1t	0.8	0	10	0.1	30	0.01	97.07 ± 2.08 (99.6)

Table 9.56: *ROC-AUC scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Average Precision							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Avr. Prec.*
226id742	**0.8**	**0**	**10**	**0.1**	**30**	**1e-5**	**79.54 ± 4.75 (85.27)**
3p9hh5y3	0.8	0	10	0.05	30	1e-5	78.01 ± 7.82 (85.57)
914a24dq	0.8	0.3	14	0.1	30	1e-5	77.80 ± 7.53 (85.20)
h862r07r	0.8	0.3	14	0.05	30	1e-5	74.02 ± 16.64 (85.79)
21s4b62o	0.8	0	14	0.1	30	1e-5	71.86 ± 24.12 (86.87)

Table 9.57: *Average Precision scores for top 5 best performing AnomalyDAE models*

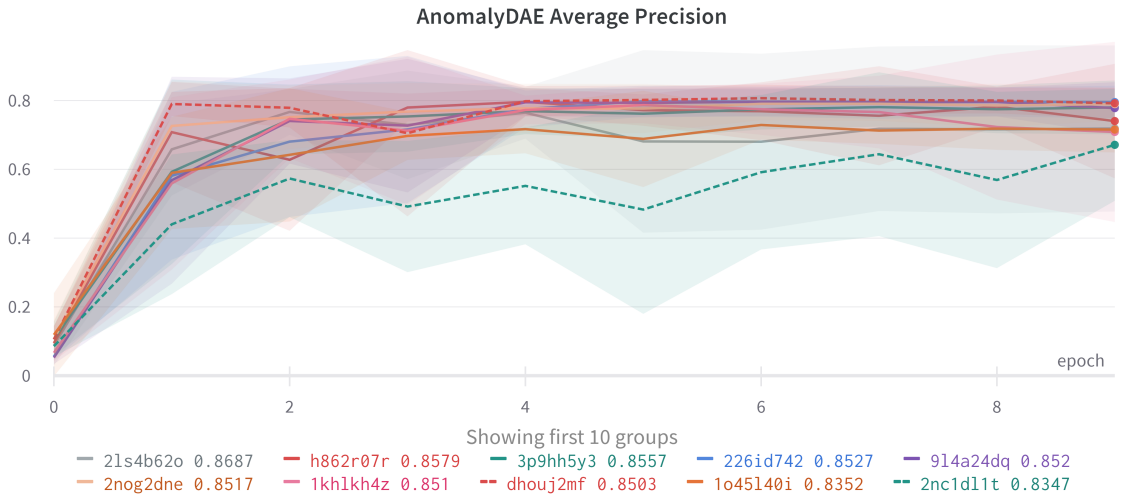


Figure 9.8: *Average Precision scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Precision @ 50							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Prec. @ 50*
38l0h0ca	0.2	0.3	10	0.1	30	0.01	80.00 ± 10.71 (94)
18ozr06u	NoNe	0.3	10	0.1	30	0.01	77.50 ± 12.59 (94)
1762gia1	0.2	0.3	14	0.1	30	0.01	71.25 ± 29.76 (94)
3jbc824b	0.2	0.3	10	0.05	30	1e-5	68.75 ± 30.01 (94)
1tfpm1k6	0.8	0.3	14	0.1	30	0.01	46.25 ± 35.80 (94)

Table 9.58: *Precision @ 50 scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Recall @ 50							
Model	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Rec. @ 50*
1dpki191	0.2	0	14	0.05	30	1e-5	8.79 ± 1.72 (11.1)
2xvftwvx	0.2	0.3	14	0.05	-1	1e-5	7.86 ± 2.34 (11.1)
2mv56hgg	0.2	0	14	0.1	-1	1e-5	7.83 ± 2.36 (11.1)
1dhsuvo4	0.2	0.3	10	0.1	-1	1e-5	7.75 ± 2.51 (11.1)
1tfpm1k6	0.8	0.3	14	0.1	30	0.01	5.31 ± 4.18 (11.9)

Table 9.59: *Recall @ 50 scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Precision @ 100							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Prec. @ 100*
2nog2dne	0.8	0	14	0.05	30	1e-5	87.38 ± 4.07 (92)
3b7kjcsx	0.8	0	14	0.05	-1	1e-5	87.12 ± 5.74 (92)
3parzgpq	0.8	0	10	0.1	30	1e-5	87.00 ± 4.63 (92)
7eu8hrtj	0.8	0.3	14	0.05	-1	0.01	77.43 ± 18.48 (92)
21s4b62o	0.8	0	14	0.1	30	1e-5	76.62 ± 29.65 (93)

Table 9.60: *Precision@100 scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Recall @ 100							
Model	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Rec. @ 100*
25j7845s	0.8	0	14	0.05	-1	1e-5	19.01 ± 4.13 (22.0)
11ugni9e	0.8	0	14	0.05	30	1e-5	18.94 ± 3.66 (22.0)
361h7mm7	0.8	0	10	0.1	30	1e-5	18.91 ± 3.92 (22.2)
3bhx9d6f	0.8	0.3	14	0.05	-1	1e-5	18.80 ± 3.72 (22.0)
144aln2i	0.8	0	10	0.05	30	1e-5	18.52 ± 4.05 (22.0)

Table 9.61: *Recall @ 100 scores for top 5 best performing AnomalyDAE models*

AnomalyDAE Grid-Search Results Precision @ num. anomalies							
Model ID	*Hyperparameters*						*Metric*
	α	*dropout*	*embed. dim*	*lr*	*num. neighbors*	*weight decay*	*Prec.@n. anom.*
361h7mm7	0.8	0	10	0.1	30	1e-5	80.96 ± 7.22 (89.9)
155aln2i	0.8	0	10	0.05	30	1e-5	77.89 ± 10.69 (88.1)
3jkkmfh8	0.8	0.3	14	0.05	30	1e-5	75.34 ± 18.27 (87.7)
2wxwfbx4	0.8	0.3	10	0.1	30	1e-5	73.47 ± 28.66 (87.7)
2tax9oqh	0.8	0	14	0.1	30	1e-5	71.29 ± 28.84 (88.4)

Table 9.62: *Precision @ n. anomalies scores for top 5 best performing AnomalyDAE models*

9.6.5 AdONE

AdONE Grid-Search Results ROC-AUC							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*ROC-AUC*
3jk2wo93	0.3	14	0.1	-1	0.01	4	96.47 ± 1.46 (98.65)
tnoed4sc	0.3	10	0.001	15	0.01	4	96.29 ± 2.27 (98.93)
1w0ei5pj	0	10	0.1	-1	0.01	4	95.79 ± 2.62 (98.69)
34deewwq	0	10	0.001	-1	1e-5	4	95.08 ± 2.79 (98.74)
11isub1x	0	10	0.001	30	0.01	4	94.59 ± 2.62 (98.92)

Table 9.63: *ROC-AUC scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Avr. Precision							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Avr. Prec*
1byfbkoy	**0**	**14**	**0.1**	**-1**	**1e-5**	**4**	**42.16 ± 11.35 (59.2)**
tnoed4sc	0.3	10	0.001	15	0.01	4	40.41 ± 13.36 (69.8)
1w0ei5pj	0	10	0.1	-1	0.01	4	36.32 ± 14.40 (57.5)
34deewwq	0	10	0.001	-1	1e-5	4	32.40 ± 14.31 (58.2)
11isub1x	0	10	0.001	30	0.01	4	29.03 ± 13.41 (60.5)

Table 9.64: *Average Precision scores for top 5 best performing AdONE models*

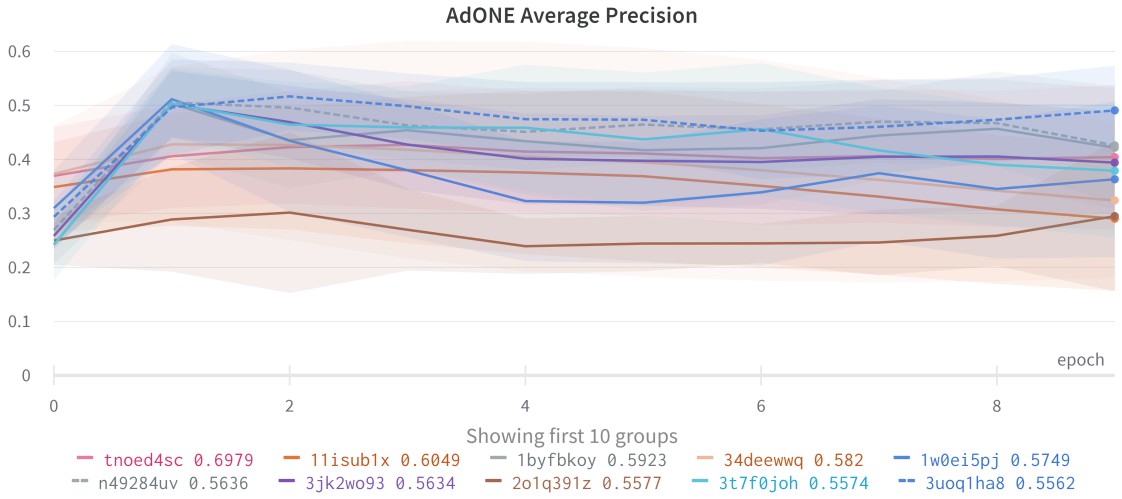


Figure 9.9: *Average Precision scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Precision @ 50							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Prec. @ 50*
tnoed4sc	0.3	10	0.001	15	0.01	4	38.50 ± 13.68 (66.0)
ulkcli2c	0.3	10	0.1	-1	1e-5	4	38.29 ± 14.53 (60.0)
1byfbkoy	**0**	**14**	**0.1**	**-1**	**1e-5**	**4**	$\mathbf{37.25 \pm 21.62}$ **(84.0)**
3t7f0joh	0	14	0.1	30	1e-5	4	33.00 ± 18.58 (52.0)
1gys7tee	0	10	0.1	15	0.01	4	25.00 ± 18.24 (58.0)

Table 9.65: *Precision @ 50 scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Recall @ 50							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Rec.@50*
ulkcli2c	0.3	10	0.1	-1	1e-5	4	4.41 ± 1.81 (7.1)
tnoed4sc	0.3	10	0.001	15	0.01	4	4.28 ± 1.92 (7.8)
1byfbkoy	**0**	**14**	**0.1**	**-1**	**1e-5**	**4**	$\mathbf{4.07 \pm 2.48}$ **(9.4)**
3t7f0joh	0	14	0.1	30	1e-5	4	3.32 ± 1.98 (5.8)
1gys7tee	0	10	0.1	15	0.01	4	2.85 ± 2.28 (7.0)

Table 9.66: *Recall @ 50 scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Precision @ 100							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Prec.@100*
y461s0qn	0.3	10	0.1	30	1e-5	4	46.14 ± 13.45 (61.0)
tnoed4sc	0.3	10	0.001	15	0.01	4	42.12 ± 17.18 (78.0)
ulkcli2c	0.3	10	0.1	-1	1e-5	4	42.00 ± 13.86 (63.0)
1byfbkoy	**0**	**14**	**0.1**	**-1**	**1e-5**	**4**	$\mathbf{41.88 \pm 18.16}$ **(76.0)**
1w0ei5pj	0	10	0.1	-1	0.01	4	33.75 ± 16.60 (61.0)

Table 9.67: *Precision @ 100 scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Recall @ 100							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	*Rec.@100*
2rid1qo5	0.3	14	0.1	30	1e-5	4	9.79 ± 3.89 (14.0)
ulkcli2c	0.3	10	0.1	-1	1e-5	4	9.64 ± 3.42 (14.9)
hkxtpf16	0.3	10	0.001	30	1e-5	4	9.37 ± 3.64 (14.8)
tnoed4sc	0.3	10	0.001	15	0.01	4	9.14 ± 4.43 (18.5)
1byfbkoy	0	14	0.1	-1	1e-5	4	9.02 ± 4.29 (16.9)

Table 9.68: *Recall @ 100 scores for top 5 best performing AdONE models*

AdONE Grid-Search Results Precision @ num. anomalies							
Model ID	*Hyperparameters*						*Metric*
	dropout	*hidden dim.*	*lr*	*num. neighbors*	*weight decay*	*num. layers*	Prec.@n. anom.
tnoed4sc	0.3	10	0.001	15	0.01	4	44.33 ± 13.2 (71.8)
8bpfgvu7	0	14	0.1	30	1e-5	4	42.48 ± 13.0 (58.4)
1w0ei5pj	0	10	0.1	-1	0.01	4	39.34 ± 15.2 (60.7)
34deewwq	0	10	0.001	-1	1e-5	4	34.68 ± 15.1 (60.9)
11isub1x	0	10	0.001	30	0.01	4	30.97 ± 13.9 (63.9)

Table 9.69: *Precision @ n. anomalies scores for top 5 best performing AdONE models*

9.7 Test Results Feature Distributions

Feature Distribution Num. Samples		
	True Pos	*True Neg*
Random	9	-
MLPAE	31	44
GCNAE	34	41
AnomalyDAE	39	36
AdONE	38	37

Table 9.70: *The amount of true positive and false positive samples the feature distributions are based upon*

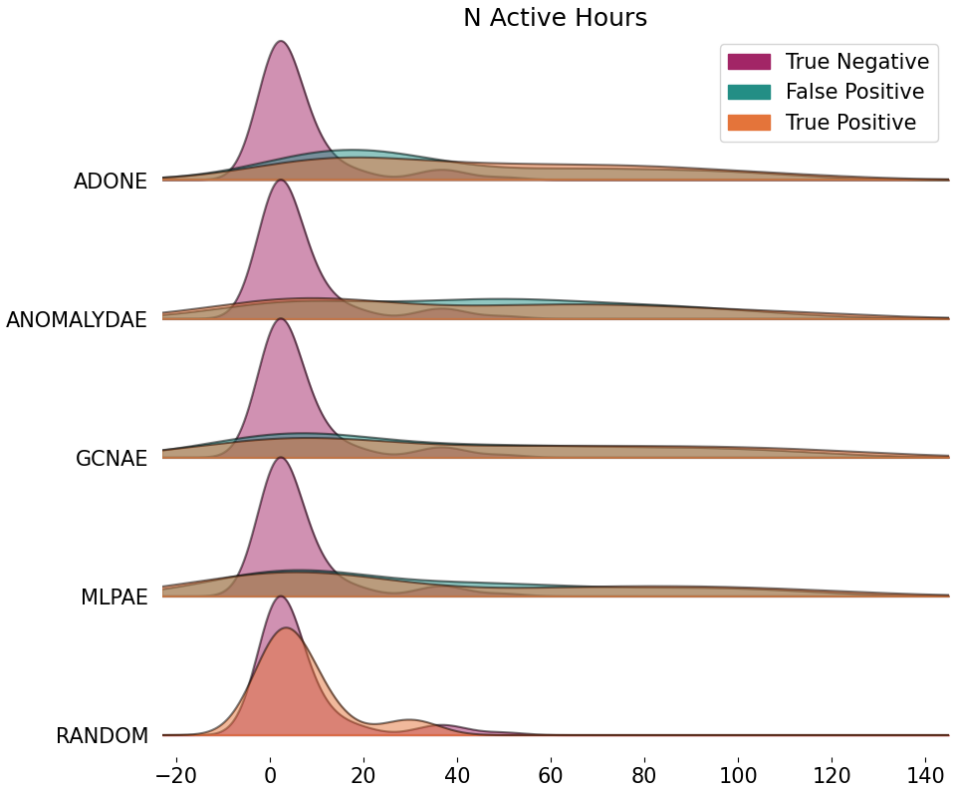


Figure 9.10: *Fitted active hours density distributions for the true positives and false positives in the top 25 of each model*

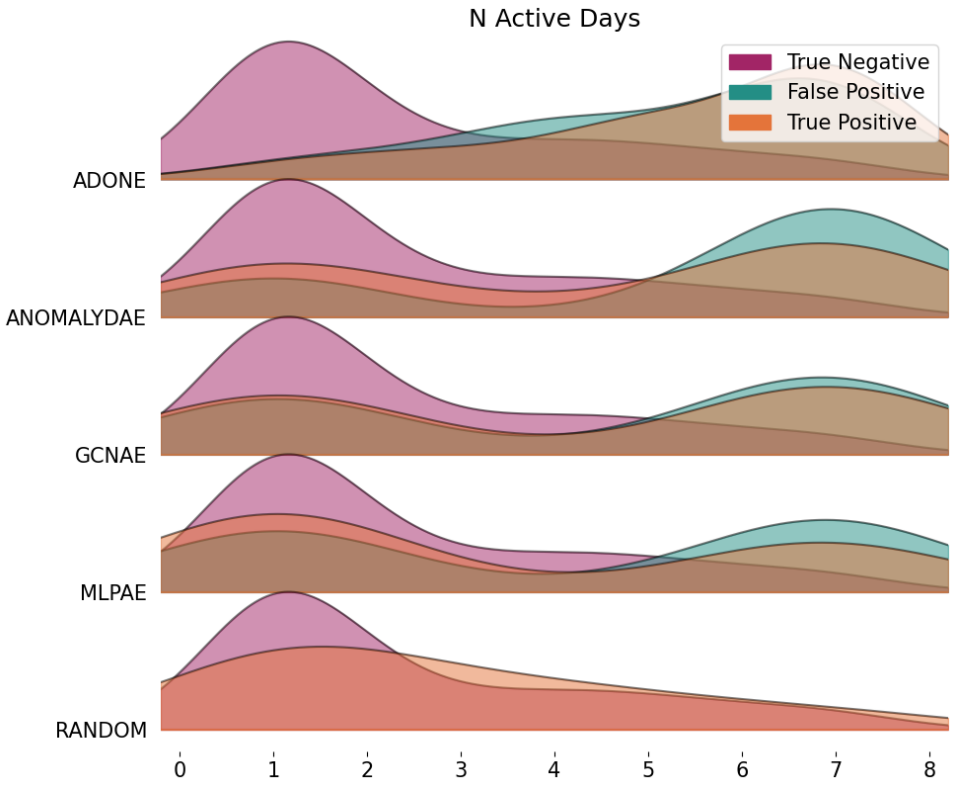


Figure 9.11: *Fitted active days density distributions for the true positives and false positives in the top 25 of each model*

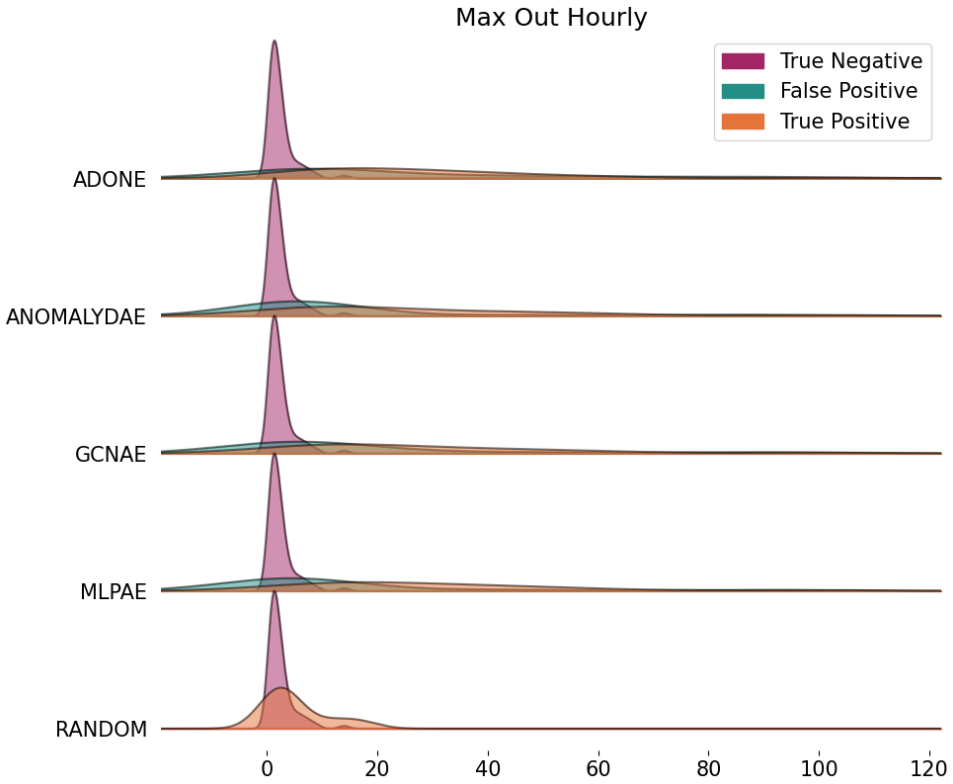


Figure 9.12: *Fitted maximum out hourly density distributions for the true positives and false positives in the top 25 of each model*

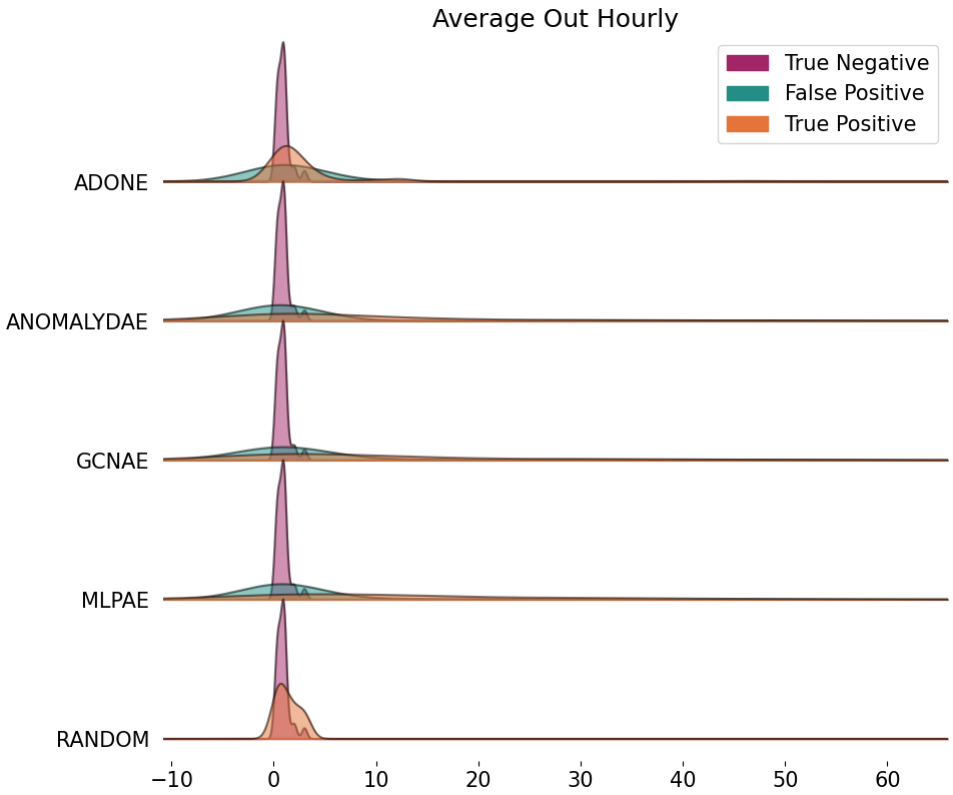


Figure 9.13: *Fitted average out hourly density distributions for the true positives and false positives in the top 25 of each model*

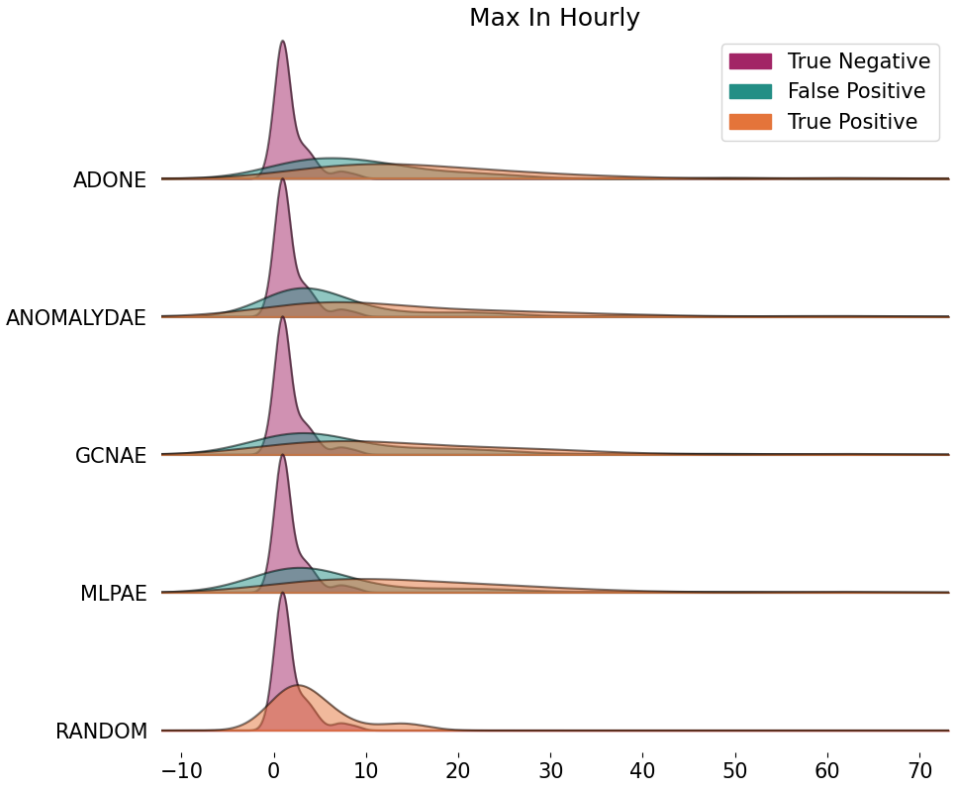


Figure 9.14: *Fitted maximum in hourly density distributions for the true positives and false positives in the top 25 of each model*

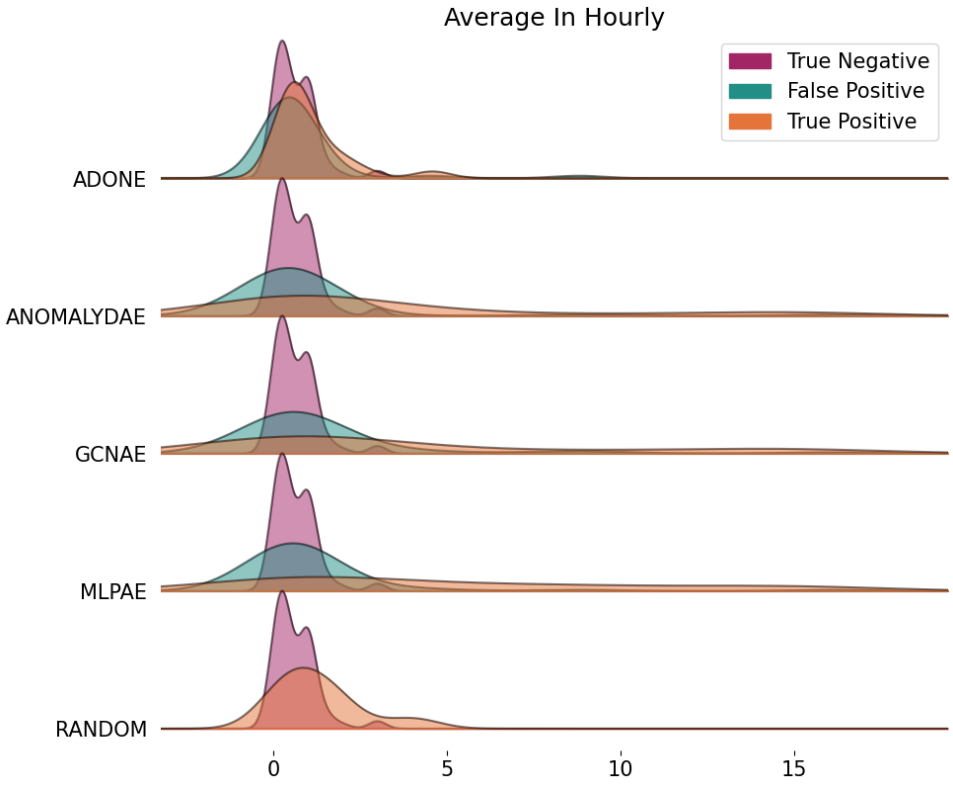


Figure 9.15: *Fitted average in hourly density distributions for the true positives and false positives in the top 25 of each model*

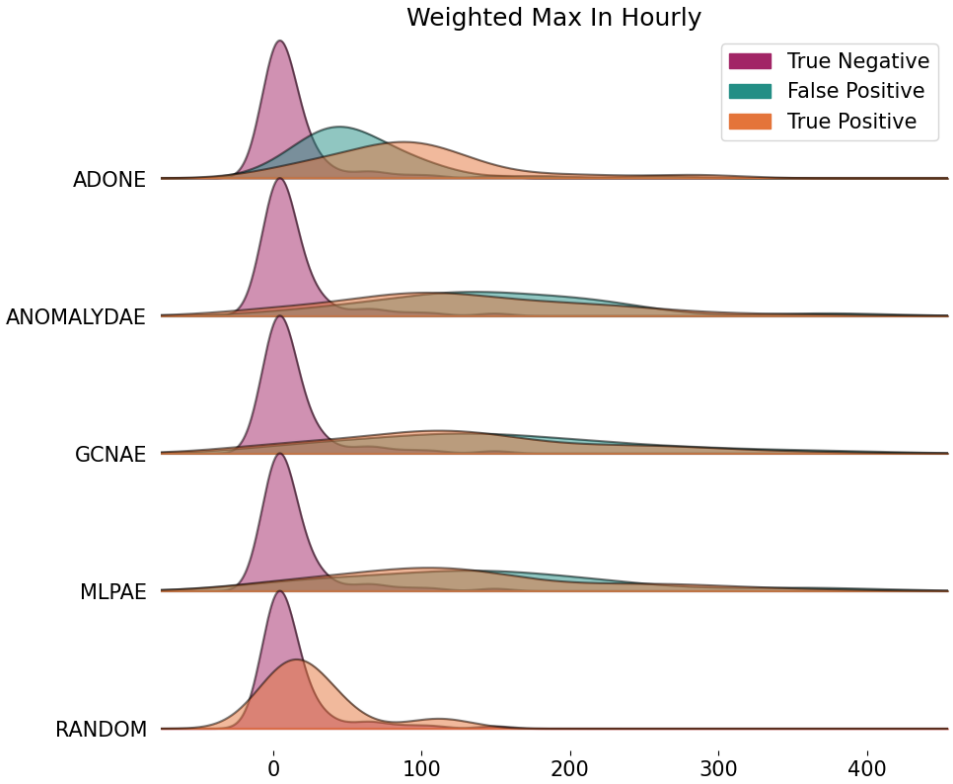


Figure 9.16: *Fitted weighted maximum out hourly density distributions for the true positives and false positives in the top 25 of each model*

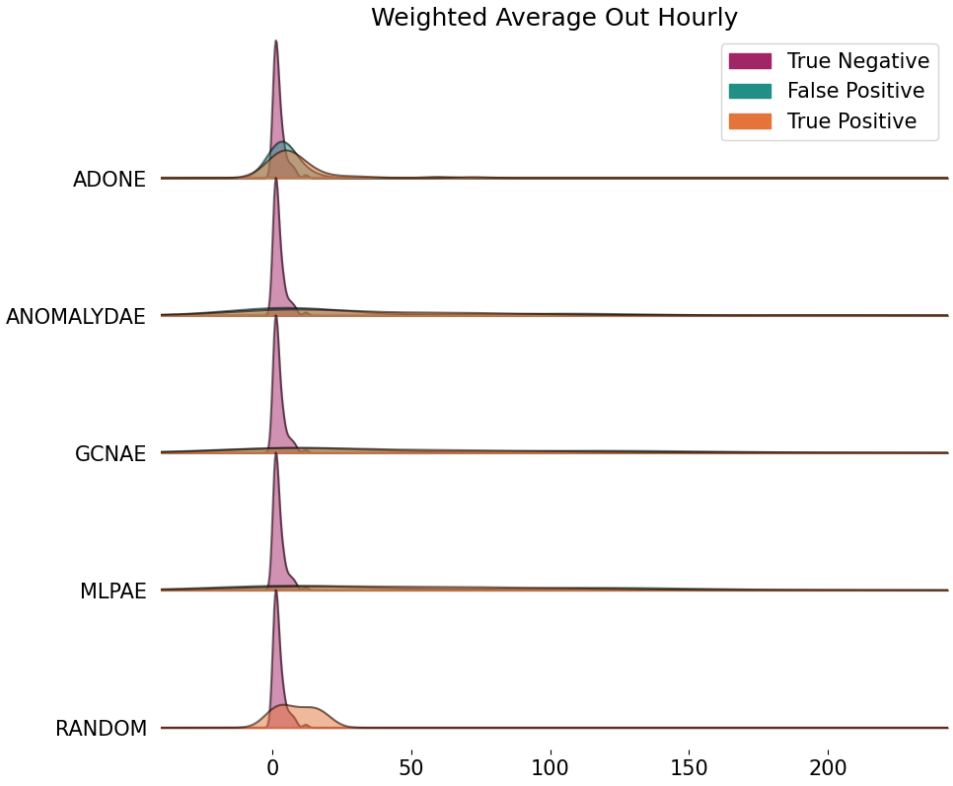


Figure 9.17: *Fitted weighted average out hourly density distributions for the true positives and false positives in the top 25 of each model*

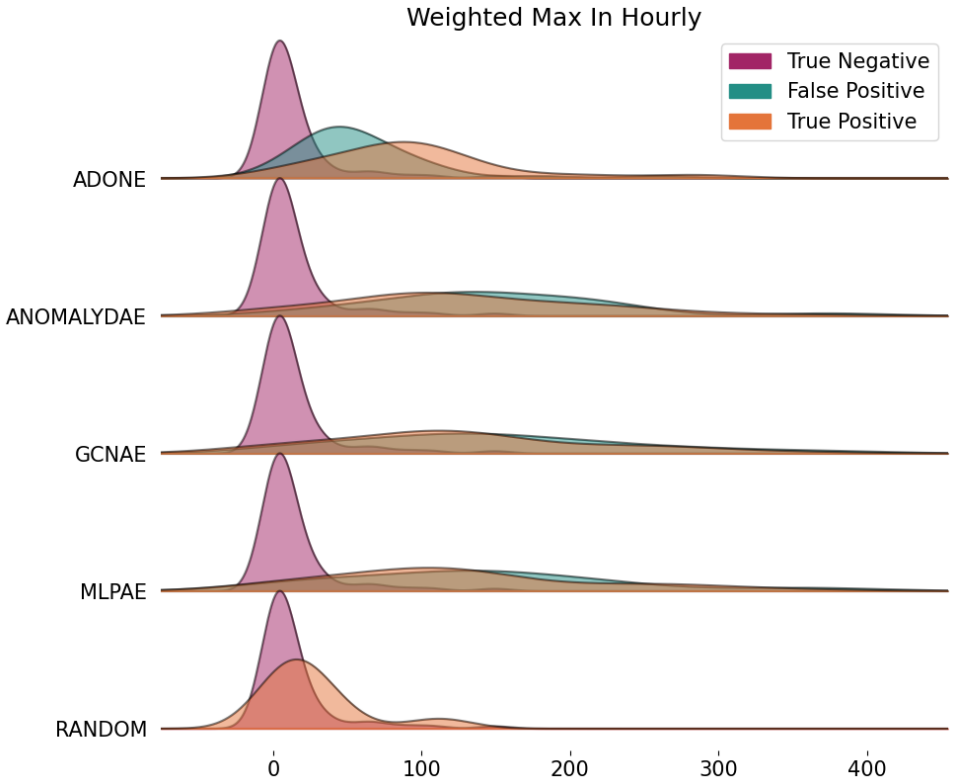


Figure 9.18: *Fitted weighted maximum in hourly density distributions for the true positives and false positives in the top 25 of each model*

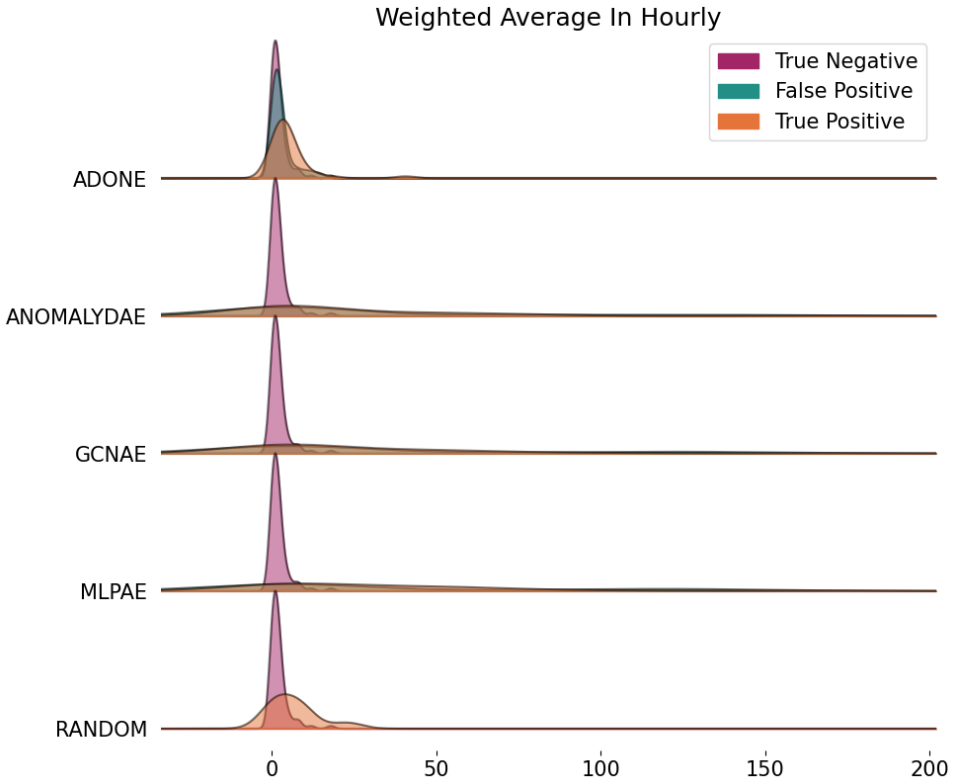


Figure 9.19: *Fitted weighted average in hourly density distributions for the true positives and false positives in the top 25 of each model*

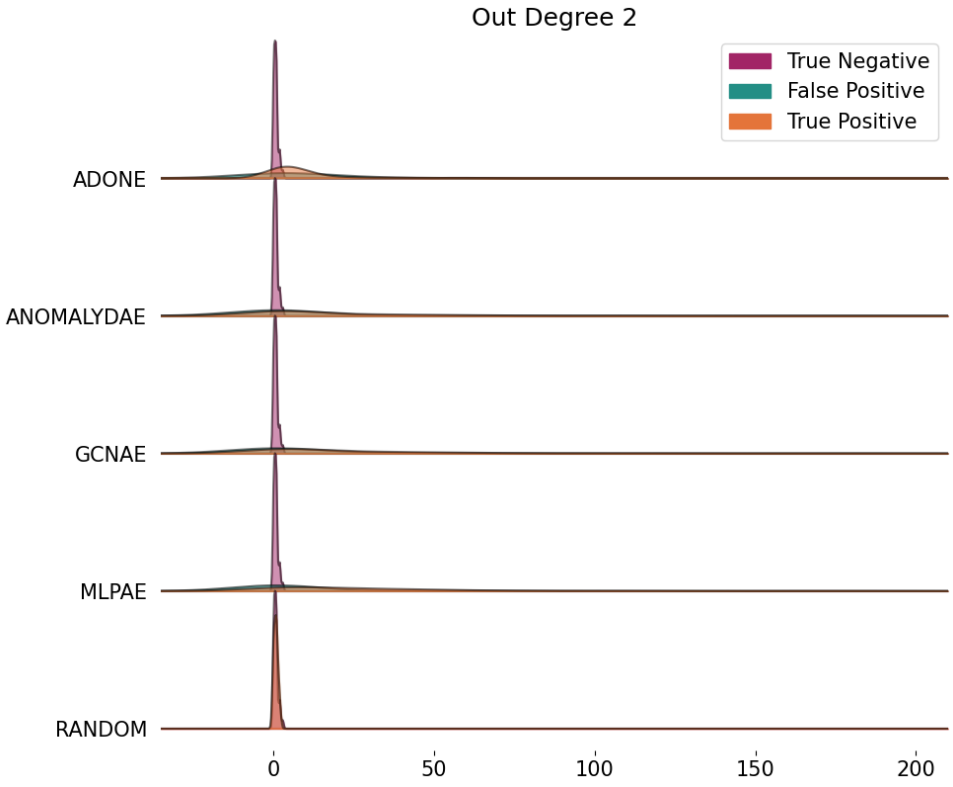


Figure 9.20: *Fitted Out Degree 2 density distributions for the true positives and false positives in the top 25 of each model*

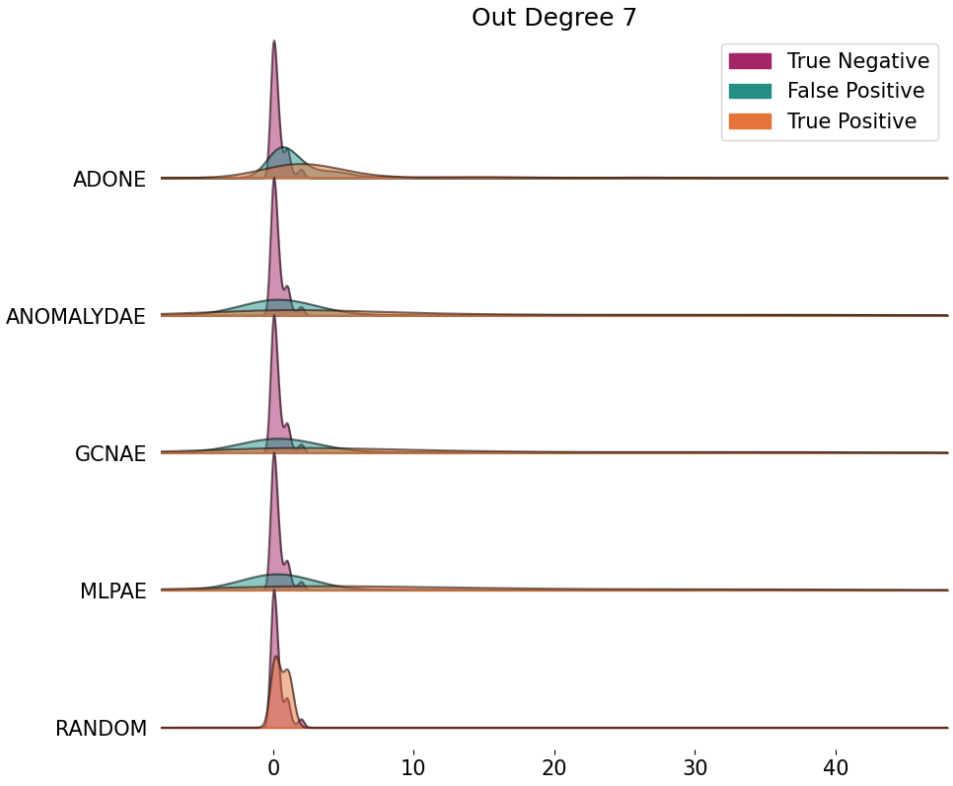


Figure 9.21: *Fitted Out Degree 7 density distributions for the true positives and false positives in the top 25 of each model*

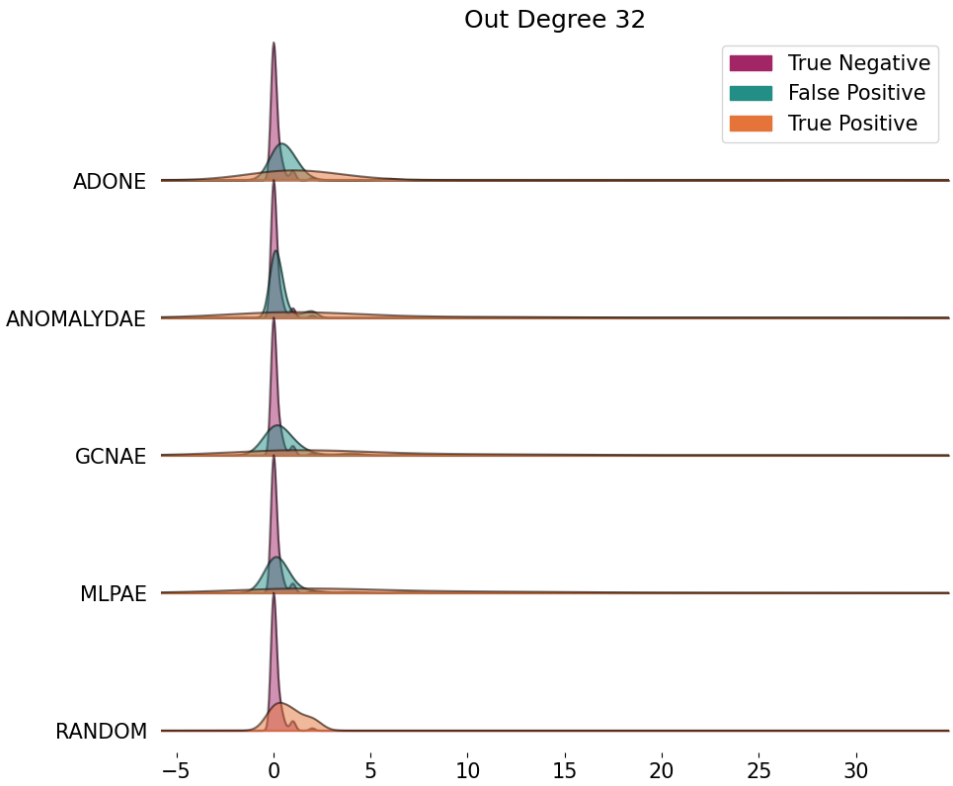


Figure 9.22: *Fitted Out Degree 32 density distributions for the true positives and false positives in the top 25 of each model*

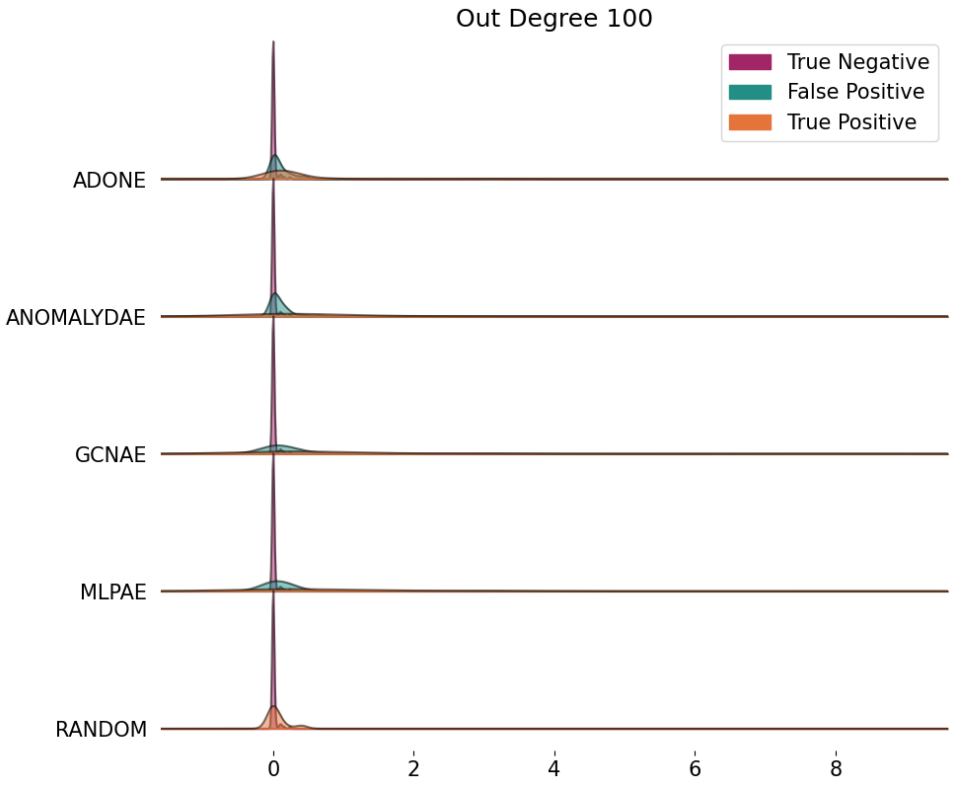


Figure 9.23: *Fitted Out Degree 100 density distributions for the true positives and false positives in the top 25 of each model*

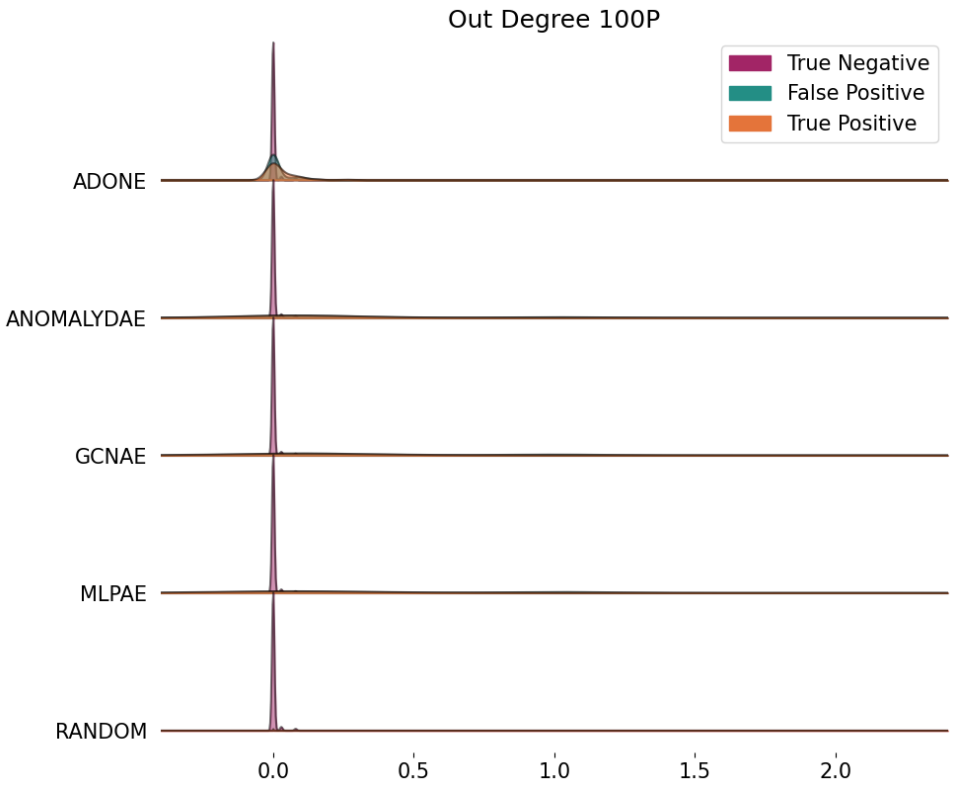


Figure 9.24: *Fitted Out Degree 100 Plus density distributions for the true positives and false positives in the top 25 of each model*

9.8 Validation Results Results Feature Distributions

Feature Distribution Num. Samples		
	True Pos	*True Neg*
MLPAE	39	19
GCNAE	38	20
AnomalyDAE	28	38
AdONE	36	7

Table 9.71: *The amount of true positive and false positive samples the feature distributions are based upon*

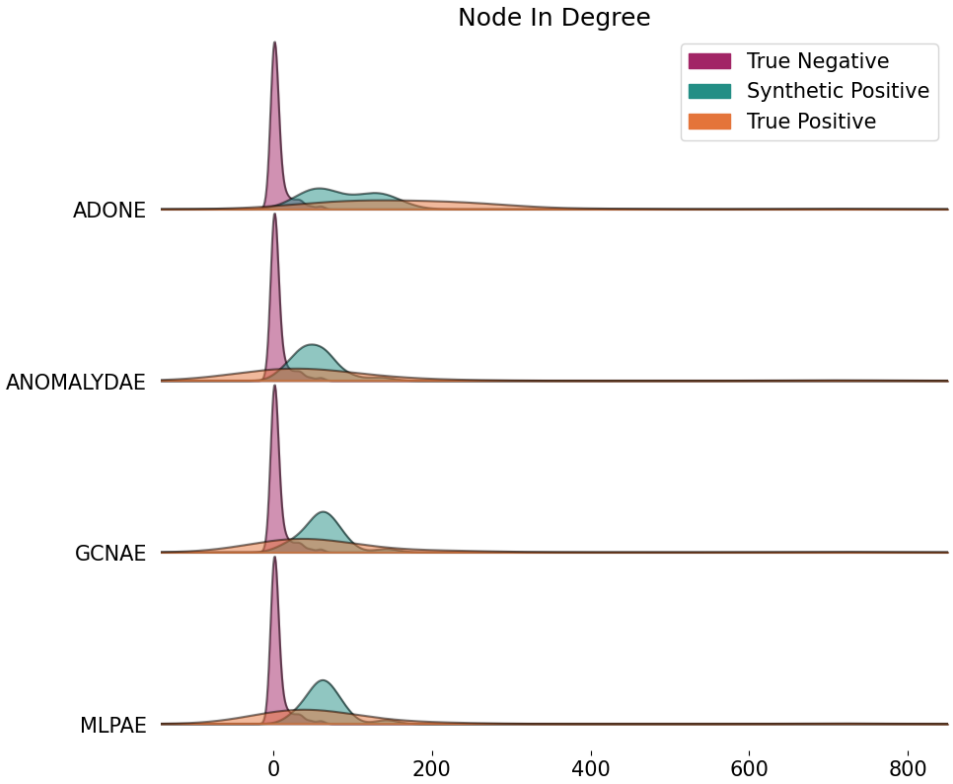


Figure 9.25: *Fitted Node In Degree density distributions for the true positives and synthetic positives in the top 25 of each model*

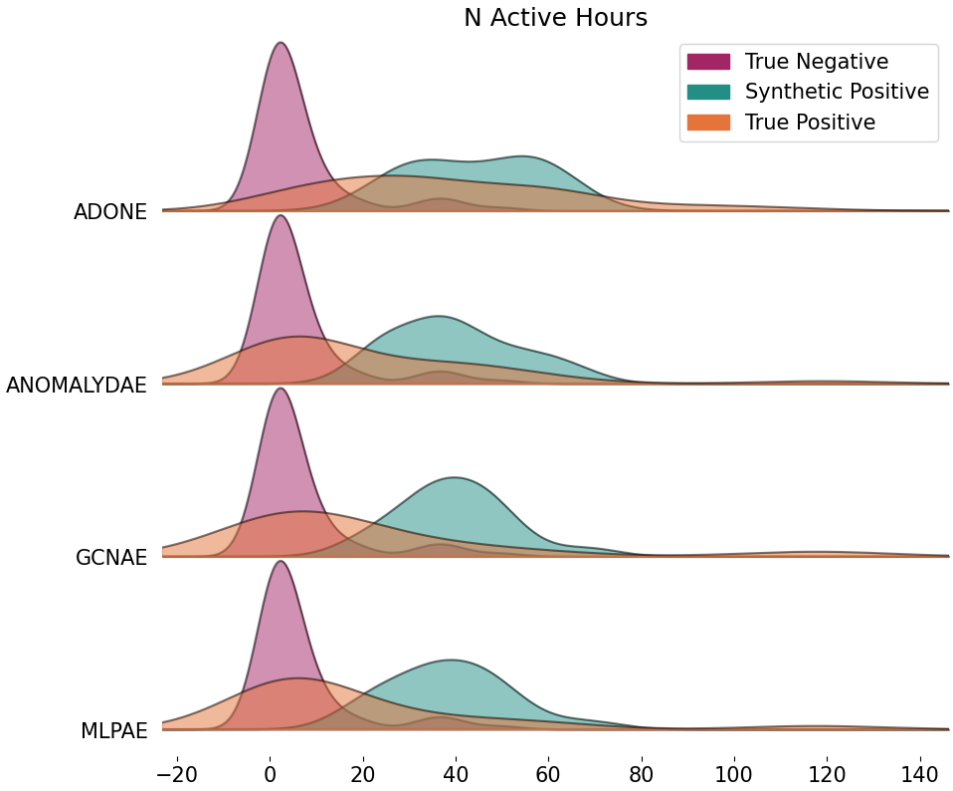


Figure 9.26: *Fitted N. Active Hours density distributions for the true positives and synthetic positives in the top 25 of each model*

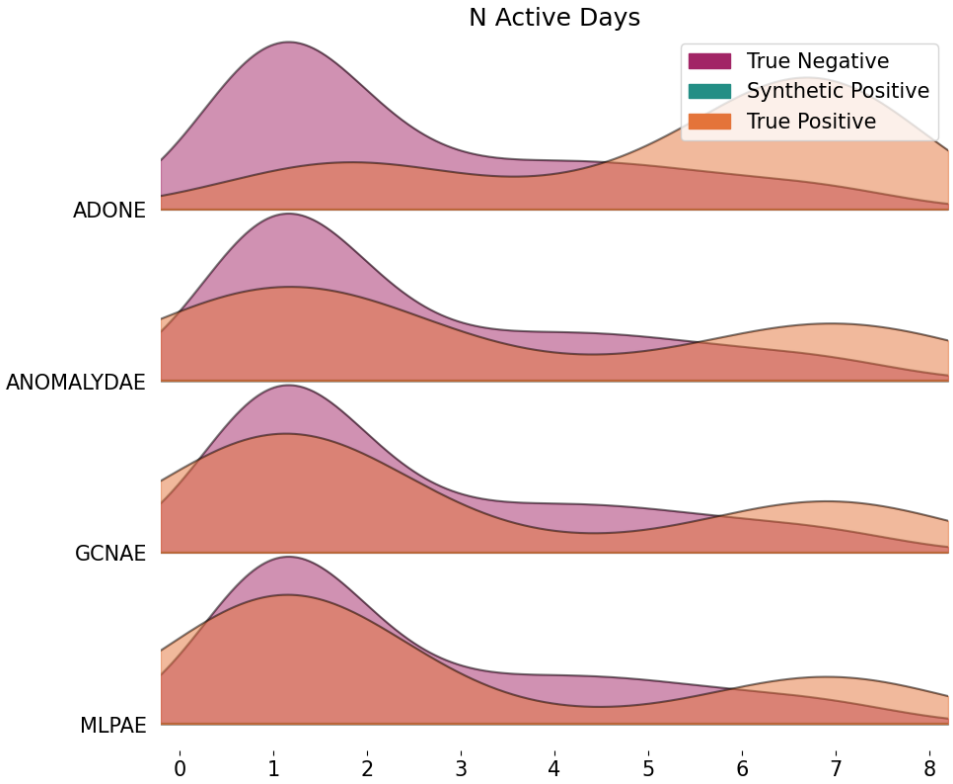


Figure 9.27: *Fitted N. Active Days density distributions for the true positives and synthetic positives in the top 25 of each model*

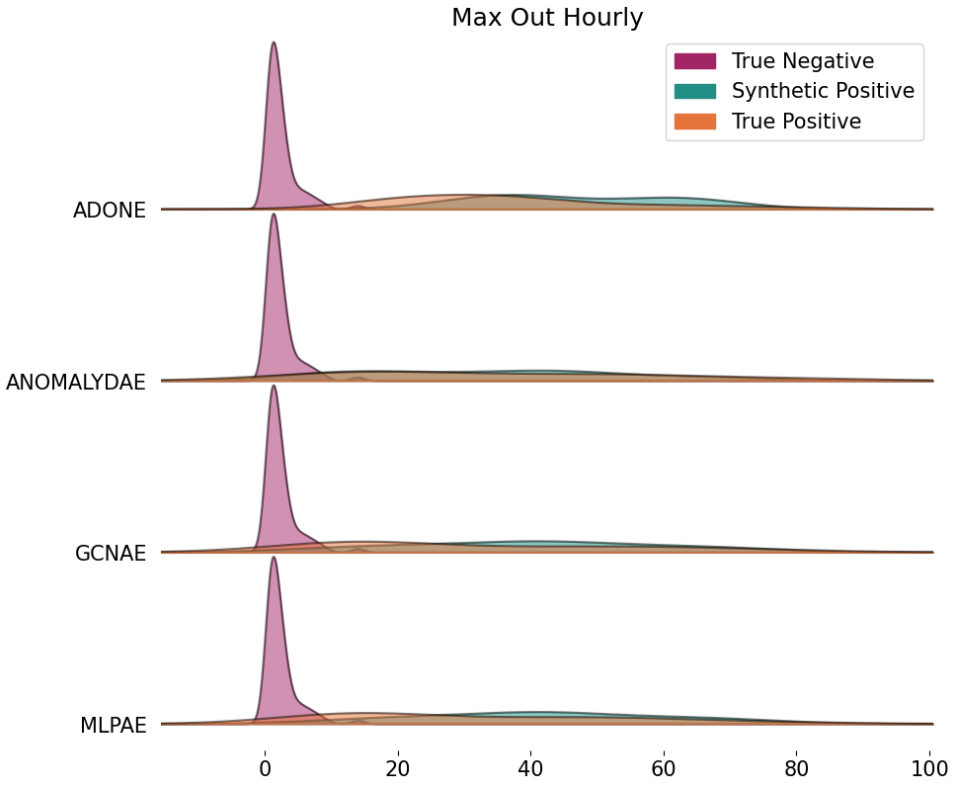


Figure 9.28: *Fitted Max Out Hourly density distributions for the true positives and synthetic positives in the top 25 of each model*

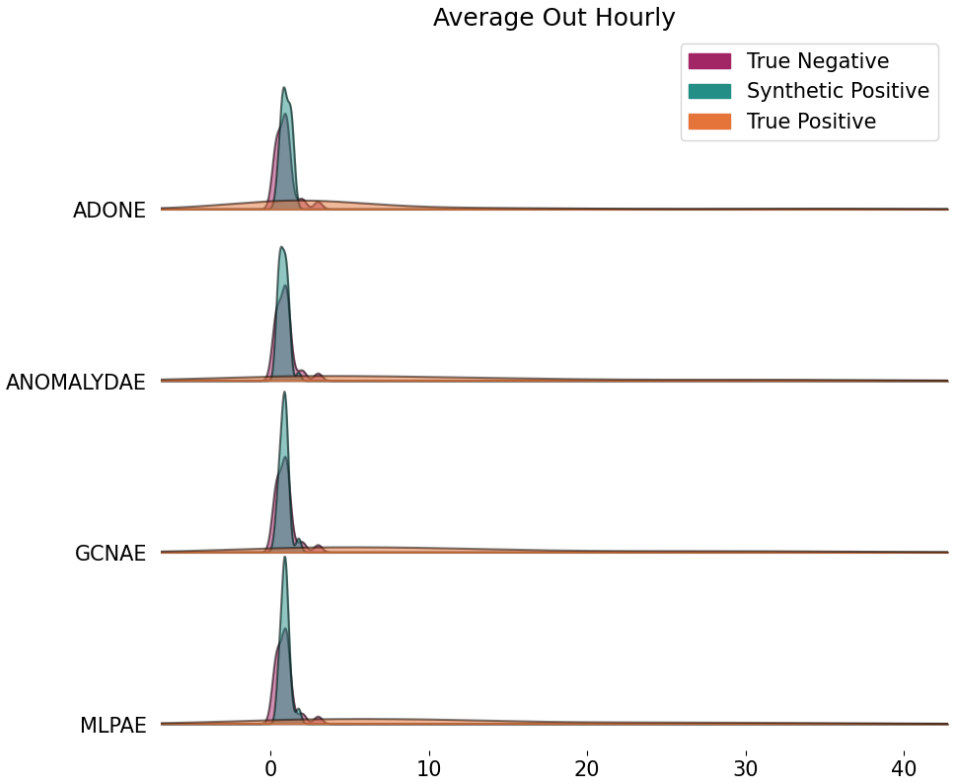


Figure 9.29: *Fitted Average Out Hourly density distributions for the true positives and synthetic positives in the top 25 of each model*$_{syn}$

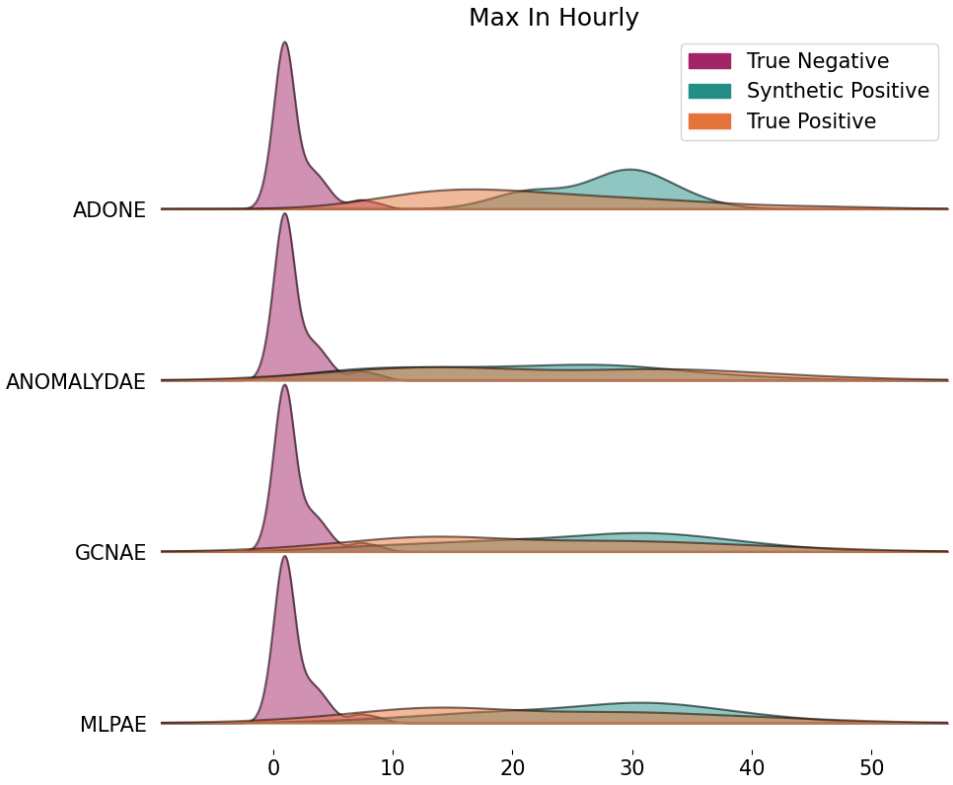


Figure 9.30: *Fitted Max In Hourly density distributions for the true positives and synthetic positives in the top 25 of each model*

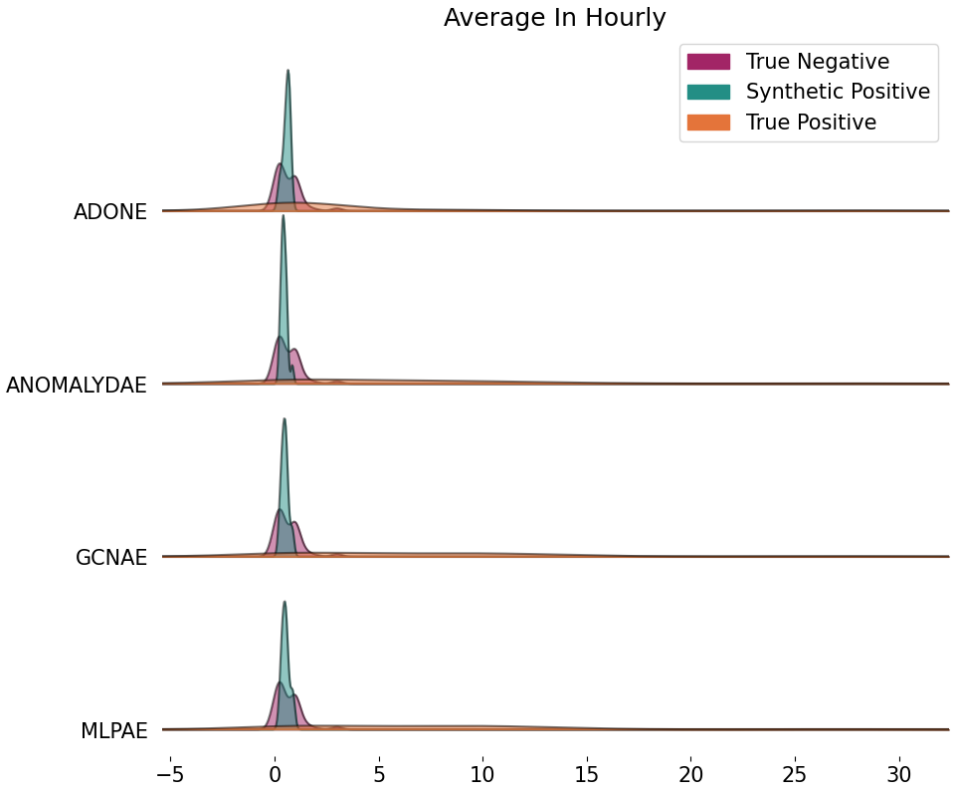


Figure 9.31: *Fitted Average In Hourly density distributions for the true positives and synthetic positives in the top 25 of each model*

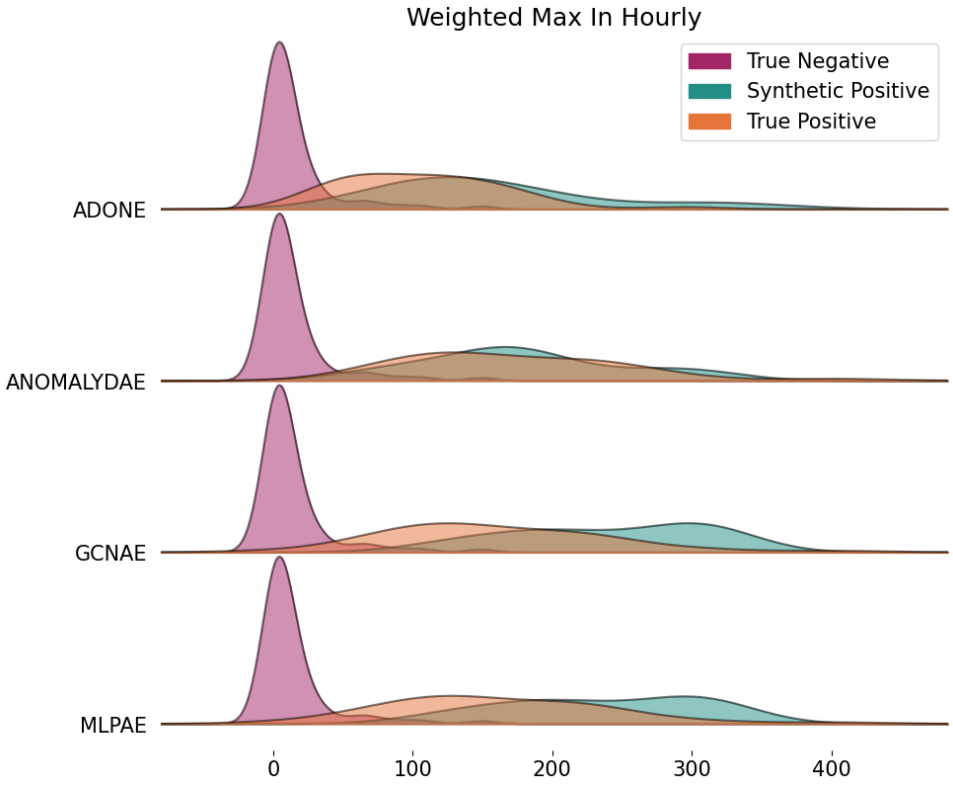


Figure 9.32: *Fitted Weighted Max In Hourly density distributions for the true positives and synthetic positives in the top 25 of each model*

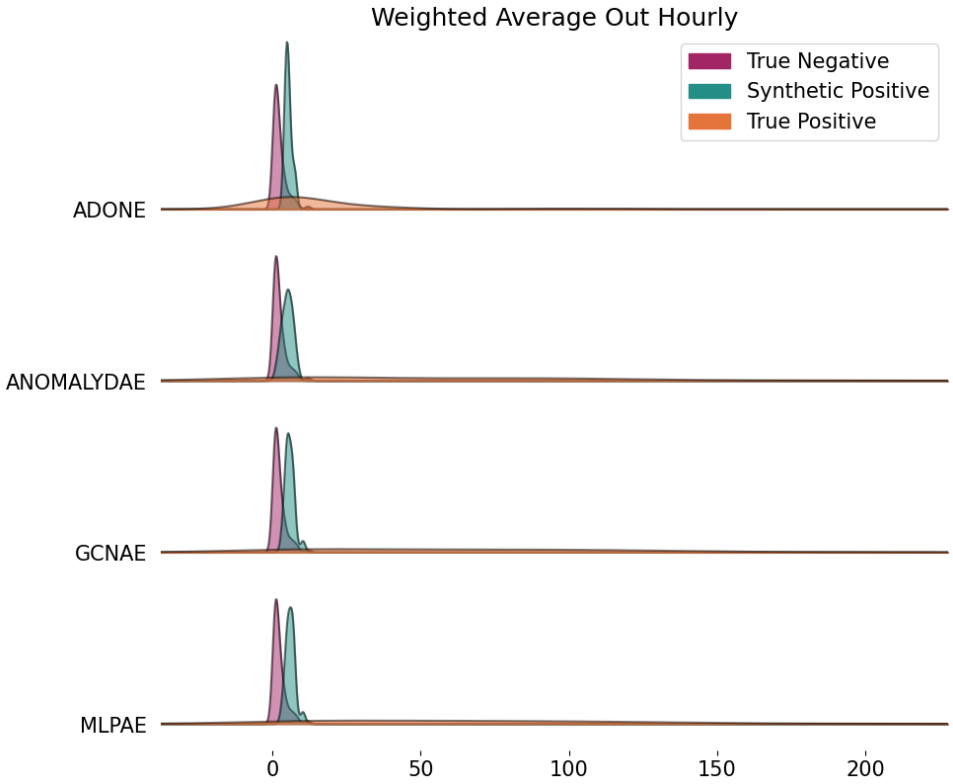


Figure 9.33: *Fitted Weighted Average Out Hourly density distributions for the true positives and synthetic positives in the top 25 of each model*

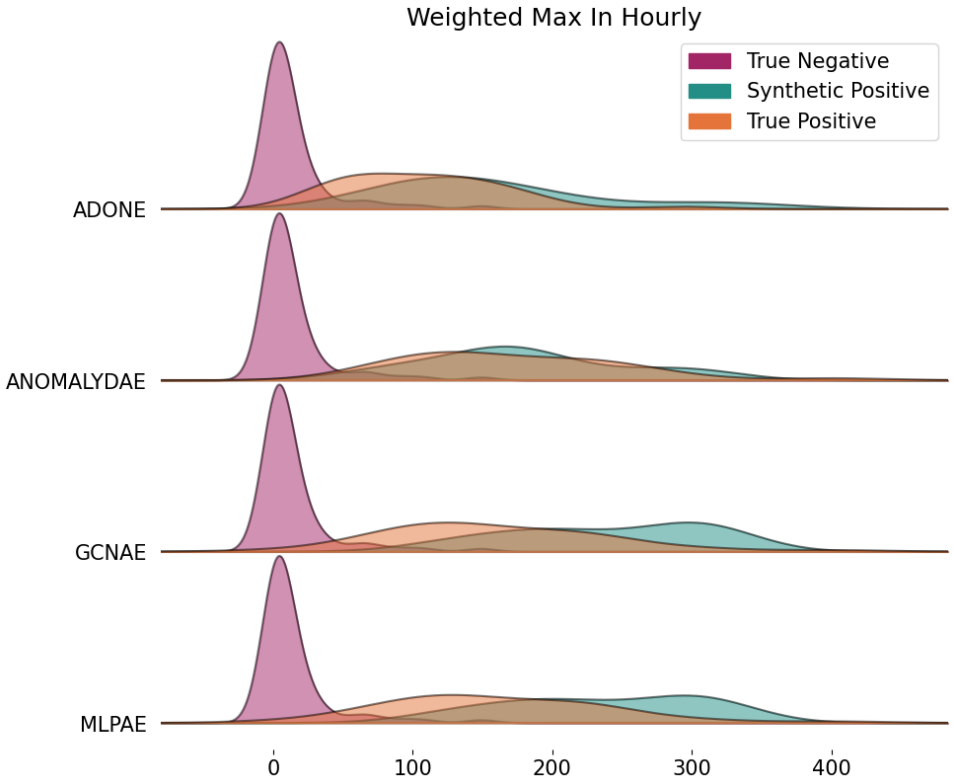


Figure 9.34: *Fitted Weighted Max In Hourly density distributions for the true positives and synthetic positives in the top 25 of each model*

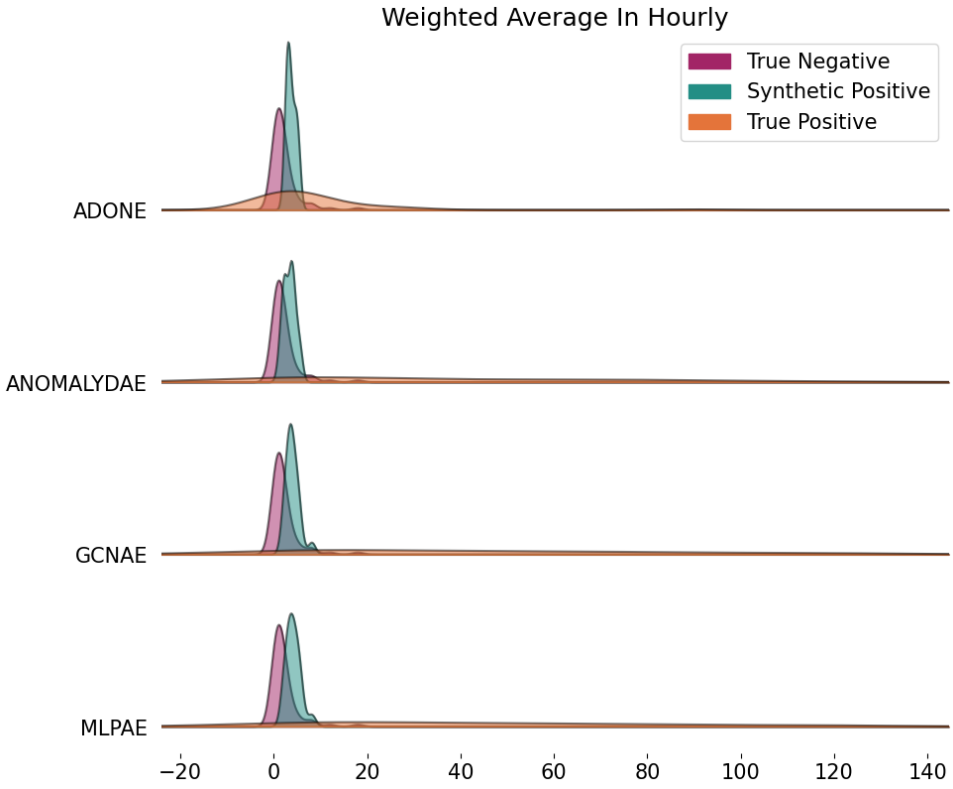


Figure 9.35: *Fitted Weighted Average In Hourly density distributions for the true positives and synthetic positives in the top 25 of each model*

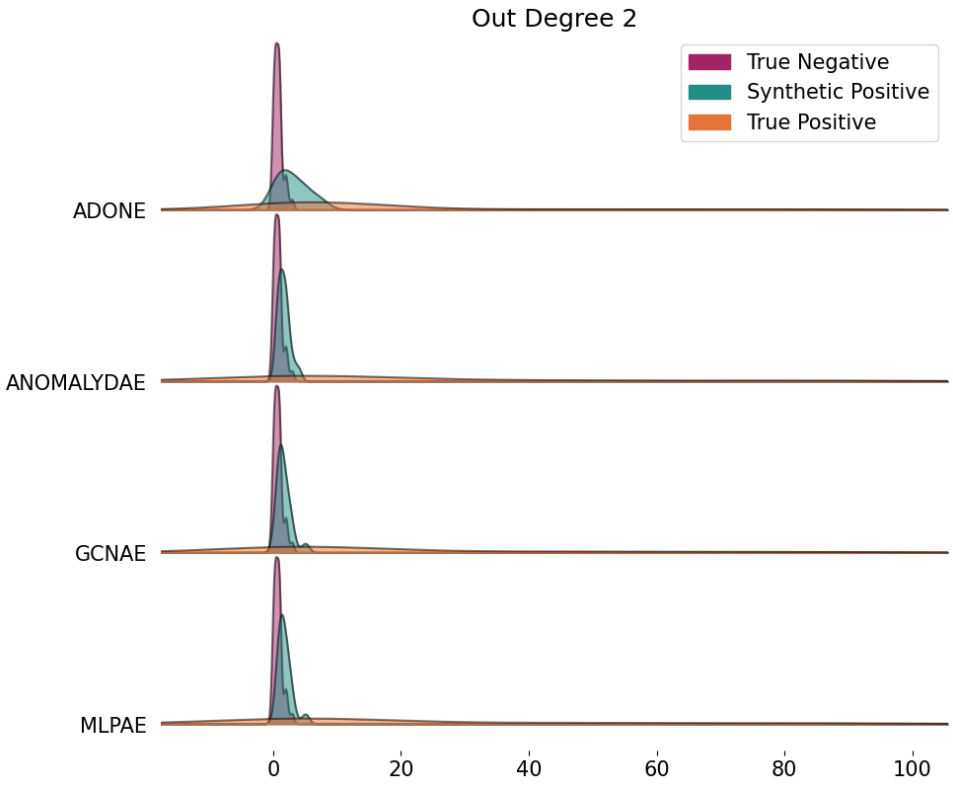


Figure 9.36: *Fitted Out Degree 2 density distributions for the true positives and synthetic positives in the top 25 of each model*

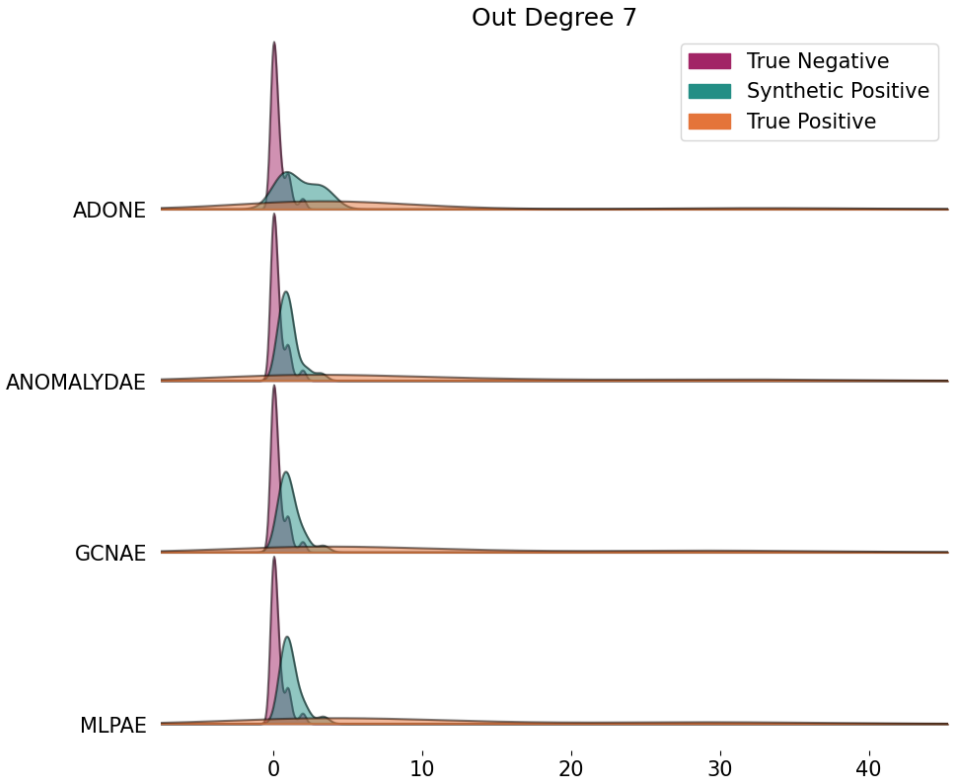


Figure 9.37: *Fitted Out Degree 7 density distributions for the true positives and synthetic positives in the top 25 of each model*

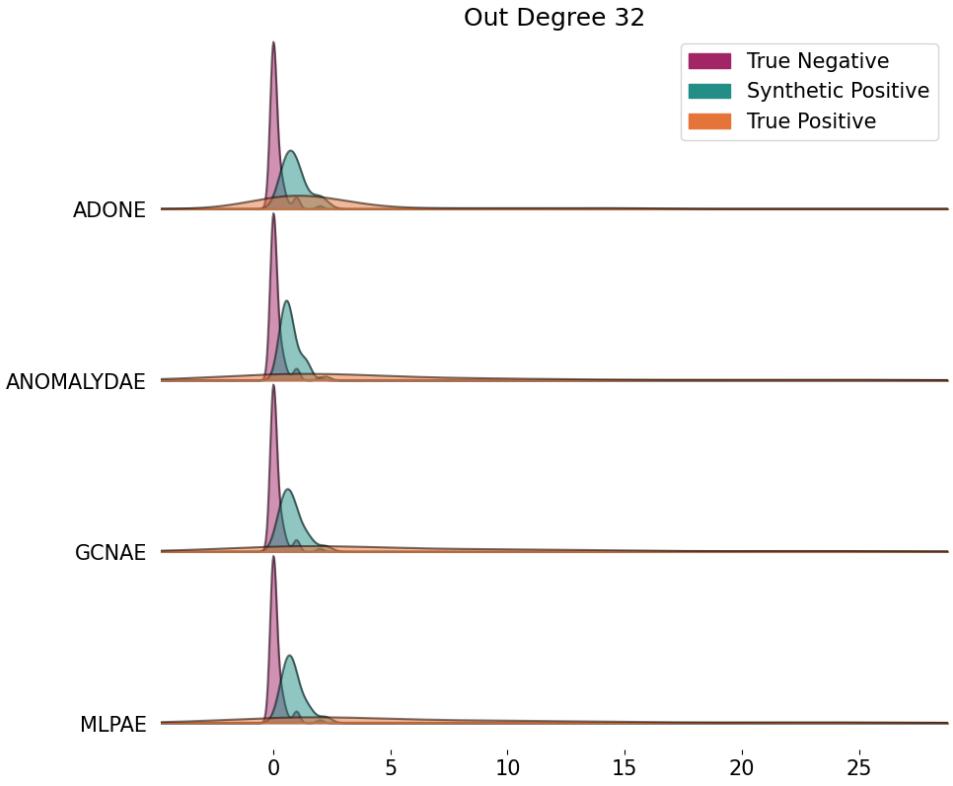


Figure 9.38: *Fitted Out Degree 32 density distributions for the true positives and synthetic positives in the top 25 of each model*

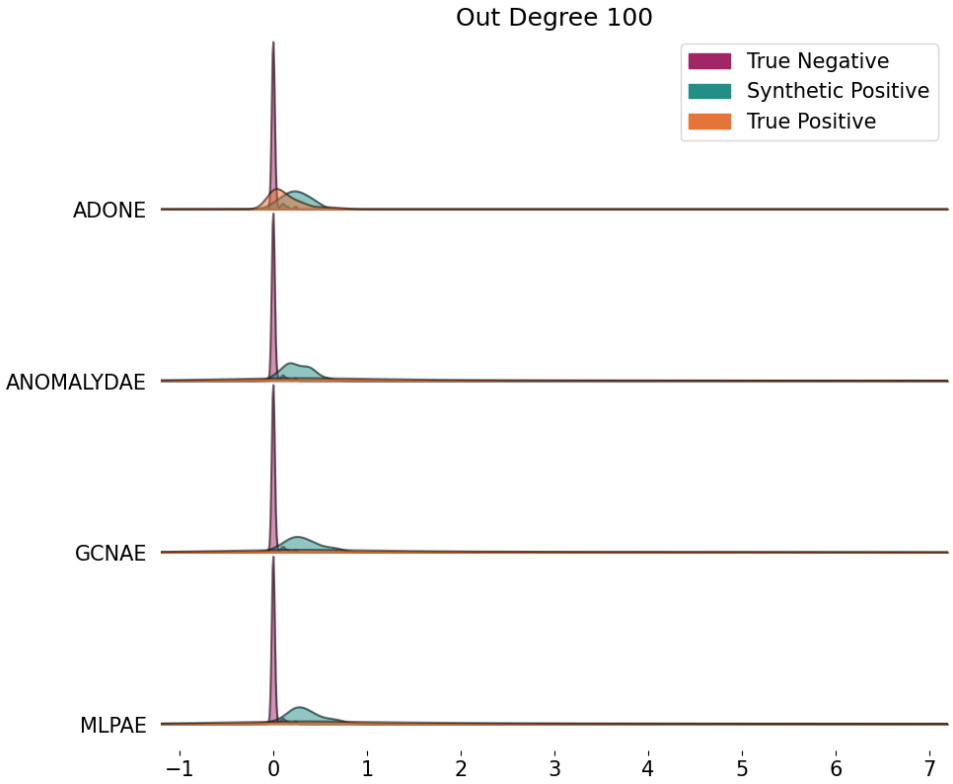


Figure 9.39: *Fitted Out Degree 100 density distributions for the true positives and synthetic positives in the top 25 of each model*

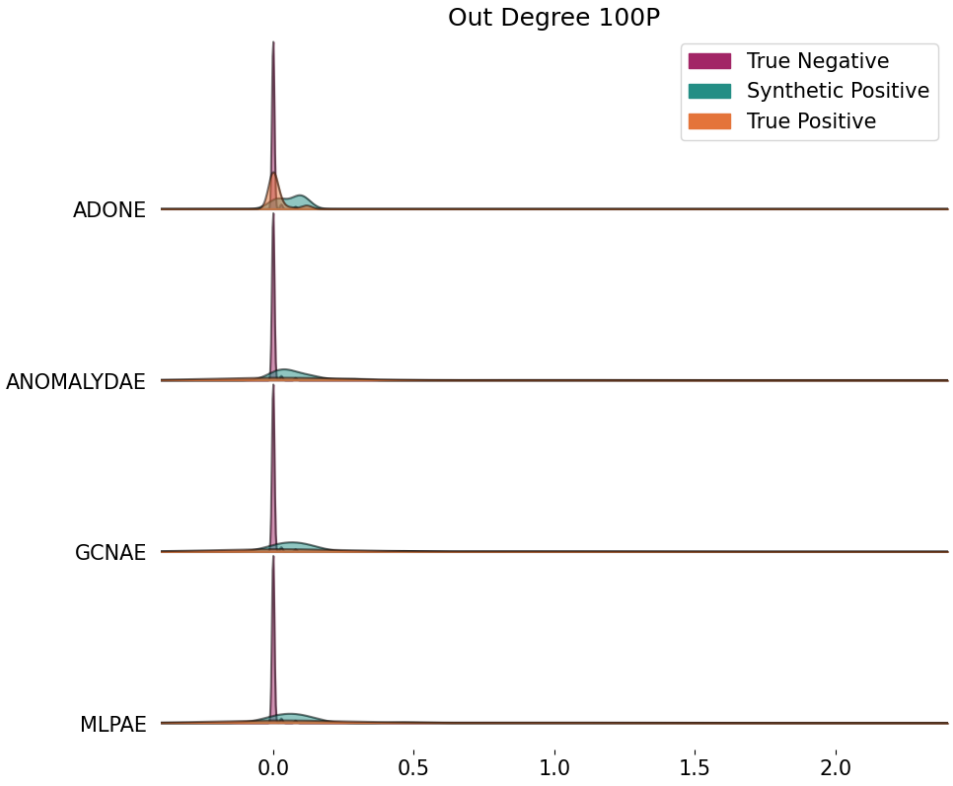


Figure 9.40: *Fitted Out Degree 100 Plus density distributions for the true positives and synthetic positives in the top 25 of each model*

9.9 Test Results Manual Evaluation

9.9.1 Data4 Manual Evaluation

Rank	Assigned Label	Explanation
1	0	-
2	0	Roleplay, not sexual
3	0	-
4	0	-
5	0	-
6	0	-
7	0	-
8	0	-
9	0	-
10	0	-
11	0	-
12	1	Asks for Snapchat. Flirty talk.
13	0	-
14	0	-
15	1	Gives out Snapchat when asked (one time)
16	0	-
17	0	-
18	0	-
19	0	-
20	0	-
21	0	-
22	0	-
23	1	Roleplay one time
24	0	-
25	0	-

Table 9.72: *Random Sampling Model Manual Evaluation for Data4*

Rank	Assigned Label	Explanation
1	0	Spamming
2	1	Sexting with minors "turns them into 5yos"
3	0	Spamming
4	1	One sexting conversation
5	1	Asks for Snapchat (to trade pics), lies about age
6	1	Wants to trade nudes on Snapchat
7	1	Wants to do 'freaky roleplay' with everyone
8	0	-
9	1	Asks for Snapchat, lies about age
10	0	Spamming
11	1	Asks for Snapchat (to trade pics), lies about age
12	0	Talks about kissing but nothing else
13	0	Talks about kissing but nothing else
14	0	-
15	1	Sexting
16	1	Already labelled in pre-modelling section
17	0	-
18	1	Sexting
19	1	Talks about sex and dicks
20	0	-
21	0	-
22	0	Spamming
23	1	Asks for Snapchat and lies about age
24	0	-
25	1	Sexting with multiple people in that week

Table 9.73: *MLPAE Manual Evaluation for Data4*

Rank	Assigned Label	Explanation
1	1	Sexting with multiple people in that week
2	0	Spamming
3	0	Spamming
4	1	Sexting with minors "turns them into 5yos"
5	1	One sexting conversation
6	1	Asks for Snapchat (to trade pics), lies about age
7	1	Sexting with multiple people in that week
8	1	Wants to trade nudes on Snapchat
9	1	Wants to do freaky rp with everyone
10	0	-
11	1	Asks for Snapchat, lies about age
12	0	Spamming
13	1	Asks for Snapchat (to trade pics) and lies about age
14	0	Talks about kissing but nothing else
15	0	Talks about kissing but nothing else
16	0	Talks about kissing but nothing else
17	1	One sexting conversation
18	0	-
19	1	Sexting
20	0	-
21	1	Asks for Snapchat
22	1	Talks about sex and dicks
23	0	Spamming
24	1	Sexting
25	0	-

Table 9.74: *GCNAE Manual Evaluation for Data4*

Rank	Assigned Label	Explanation
1	0	Spamming
2	0	Spamming
3	1	One sexting conversation
4	1	Already labelled, serial sexter
5	1	Sexting
6	0	-
7	1	Asks for Snapchat (to trade pics), lies about age
8	1	Sexting
9	1	Sexting with minors "turns them into 5yos"
10	0	Mild roleplay
11	0	writes to multiple people telling the same long story
12	1	asks for Snapchat (to trade pics), lies about age
13	1	-
14	1	sexting with multiple people in that week
15	1	wants to trade nudes on Snapchat
16	0	very long conv with friend (almost spam)
17	1	sexting with multiple people in that week
18	1	asks for nudes on Snapchat, lies about age
19	0	talks about kissing but nothing else
20	0	-
21	0	-
22	1	sexting with multiple people (including dog roleplay)
23	0	-
24	0	sexual talk but not really?
25	1	sexting

Table 9.75: *AnomalyDAE Manual Evaluation for Data4*

Rank	Assigned Label	Explanation
1	1	Asks for mail, phone number from many people
2	0	But spams people asksing for "greet"
3	1	Sexting with multiple people in that week
4	1	One sexting conversation
5	1	Already labelled, rp with multiple people, seems young
6	1	Asks for naughty photos many times on Snapchat
7	1	Already labelled, not much sexting but shares phone
8	1	Already labelled, serial sexter
9	1	Wants Snapchat and to send pictures, flirty talk
10	1	Demon roleplay, with blood etc. but not sex expliticly
11	0	Writes to multiple people telling the same long story
12	1	Asks for Snapchat and lies about age to match the other user
13	0	Spamming
14	0	Spamming
15	1	Asks everybody for nudes on Snapchat, lies about age
16	1	Serial sexter
17	0	Data Error/glitch two conversations where hes the only sender
18	0	-
19	1	Sells items with multiple people, they exchage payment info
20	0	Spamming/sells
21	1	Asks for Snapchat, says hes 17 and wants to date 13yo
22	1	Asks everybody for nudes on Snapchat, lies about age
23	0	-
24	0	Asks people to like their stuff
25	0	Spam

Table 9.76: *AdONE Manual Evaluation for Data4*

9.10 Sexually Charged Obscured Words

Obscured word	Original word
tvzakes	takes
bxxvbsx	boobs
psvvxyv	pussy
ouxxtvx	out
moxvns	moans
drvvesvx	dress
djjjijjjcjjjk	dick
rjjujjjbs	rubs
injjjsijjjde	inside
panxxxtees	panties
mo@ns	moans
stjjjrxxojjjkzzze	stroke
tijjjghxxxt	tight
fjjjujjjcjjjk	fuck
t11ghtens	tightens
vfxxxx	fuck
mo@nvvingv	moaning
xxcxvlxvxixvxt	clit
lxegsvcx	legs
wvaistvcx	waist
pjjjuzzzszzzsjjjy	pussy
lxviccinv	licking
dxveevperv	deeper

Table 9.77: *Example of how words are obscured in sexting conversations*

Technical

University of

Denmark

Building 324

2800 Kgs. Lyngby

Tlf. 4525 1700

www.compute.dtu.dk