DeepFGSS: Anomalous Pattern Detection using Deep Learning

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Akash Kulkarni

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Edward McFowland III

May, 2019

Acknowledgements

Firstly, I would like to thank my adviser, Prof. Edward McFowland III for his guidance, mentorship and bringing out the researcher in me.

I would also like to thank my thesis committee members, Prof. Vipin Kumar and Prof. Erika Helgeson for providing their valuable feedback on my thesis.

Finally, I would like to thank my parents for their constant love, my friends for their unwavering support and encouragement and my brother, Vishal, for pushing me to pursue masters abroad.

Dedication

To my mom, dad and best buddy, Anurag Kakati.

Abstract

Anomaly detection refers to finding observations which do not conform to expected behavior. It is widely applied in many domains such as image processing, fraud detection, intrusion detection, medical health, etc. However, most of the anomaly detection techniques focus on detecting a single anomalous instance. Such techniques fail when there is only a slight difference between the anomalous instance and a non-anomalous instance. Various collective anomaly detection techniques (based on clustering, deep learning, etc) have been developed that determine whether a group of records form an anomaly even though they are only slightly anomalous instances. However, they do not provide any information about the attributes that make the group anomalous. In other words, they are focussed only on detecting records that are collectively anomalous and are not able to detect anomalous patterns in general. FGSS [45] is a scalable anomalous pattern detection technique that searches over both records and attributes. However, FGSS has several limitations preventing it from functioning on continuous, unstructured and high dimensional data such as images, etc. We propose a general framework called DeepFGSS, which uses Autoencoder, enabling it to operate on any kind of data. We evaluate its performance using four experiments on both structured and unstructured data to determine its accuracy of detecting anomalies and efficiency of distinguishing between datasets containing anomalies and ones that do not.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Any data instance is considered to be an anomaly if its characteristics deviate from what is considered normal, expected or standard. Such deviations can be outliers, or an exception, or a novel instance, or some contaminant or simply noise. Hence, the purpose of detecting anomalies can be outlier detection, noise removal, or sometimes detecting interesting patterns. Generally, anomaly detection refers to the problem of finding any pattern in data that does not conform to expected behavior. Anomaly detection is significantly used in applications such as banking systems, health care, cyber-security, military surveillance for enemy activities and other safety-critical systems.

Usually, anomalies are recognized as harmful deviations. For example, as mentioned previously, detecting fraudulent transaction in bank statements, detecting a tumor in MRI, detecting unusual network traffic and so on. However, anomalies do not necessarily have to be bad either. For example, for an advertisement company getting Rate Of Interest exceedingly high for this month compared to the gain from previous months is a good sign.

Irrespective of whether an anomaly is harmful or not, techniques used for detecting such aberrations are the same. Over time, there have been various techniques developed which work on different principles and assumptions. Most of the techniques developed so far focuss on detecting point anomalies. In the next section, we will explain what are point anomalies along with techniques to detect them followed by limitations of such techniques.

1.1 Point Anomaly Detection

Point anomaly is an anomaly when a single data instance is anomalous. Chandola, Banerjee, and Kumar [11] provided an exhaustive categorization of anomaly detection techniques and their assumptions and principles.

Classification based anomaly detection techniques are supervised. First, a model is trained on the training dataset to classify a data instance. Then, the model classifies the data instance into anomalous or normal class. Examples of such machine learning techniques are logistic regression, SVM, etc.

Nearest neighbor based anomaly detection techniques, first, calculate the neighborhood of the data instance based on some distance metric (for eg, Euclidean distance for continuous attributes, matching the coefficient for categorical attributes, etc) and then, label data instances falling in the dense neighborhood as normal and those far from their nearest neighbor as anomalies. Examples of such techniques can be K-Nearest Neighbor (KNN), or intelligent data structures such as k-d trees, etc.

Anomaly detection techniques based on clustering work on different principles:- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)[28], Shared Nearest Neighbor (SNN) [27], WaveClustering [58]- classifies those data instances as anomalies which do not belong to any cluster; Self Organizing Map (SOM) [65], K-means clustering, Expectation Maximization labels those data instances as anomalies that are far from the centroid of their nearest cluster; Cluster-Based Local Outlier Factor(CBLOF) [35], CDtrees [61] labels those data instances as anomalous which belong to a cluster which is either small or sparse.

Statistical anomaly detection techniques learn a model which can generate the data seen in the training dataset and then statistical inference test is applied to determine if a given test instance is an anomaly or not based on whether it occurs in a high probabilistic regions or low. These can be parametric such as Gaussian-based model like Z-score or extreme value analysis [59], or non-parametric techniques (such as histogram based outlier score[32], Generative Adversarial Networks [33], etc).

1.2 Limitations

These anomaly detection methods focus on the identification of a single anomalous data record. For example, detecting a fraudulent transaction in financial data, detecting a system malfunction while analyzing system log files, detecting bots that generate fake reviews on social media platform, detecting energy leakage in smart houses or detecting excessive power utilization in data centers, etc. Some of the anomaly detection techniques devised are also domain specific, making them hard to adapt to a different problem.

However, in the real world, sometimes it's hard to detect anomalous behavior where each individual data instance can be slightly anomalous making it difficult for anomaly detection techniques to be effective, mentioned in the previous section, to detect. For example, in the case of detecting fraudulent transactions in financial data, an intelligent fraudster will attempt to disguise their activity so that it closely resembles legitimate transactions making each fraudulent transaction only slightly anomalous. This makes the situation worse to discern the malign activity. Similarly, an intelligent hacker might also try to mimic normal network activity while sniffing data. In the case of customs officials responsible for detecting smuggled items, they must decide which shipments to inspect. An intelligent smuggler would intelligently sneak in contrabands by making similar illegal shipments - let's say, with same declared contents on the contraband, shipping through the same company name, to the same port.

In such cases, classification techniques will not be able to learn a clear decision boundary between normal and anomalous instances. Nearest neighbor based techniques also fail to detect such anomalies as the records are only slightly anomalous and hence might fall into a denser region with other normal instances. Clustering based techniques might also fail as anomalous instances which almost resemble normal instances might not be farther from the centroid of the cluster or anomalous instances might themselves be part of a larger cluster. Traditional statistical techniques will also not be able to distinguish slightly anomalous instances because they might not necessarily fall in a region with low probability density.

One way to detect such anomalous behaviors is by searching for groups of these similar data instances each of which is only slightly anomalous. Such anomalies are

called collective anomalies whereas all the previous anomaly detection techniques aim for detecting point anomalies (where a single data point in a dataset is sufficiently anomalous)

Chapter 2

Collective Anomaly Detection

Collective anomaly is defined as a collection of related data instances that are anomalous compared to the entire data set but individual data instances may not be anomalies themselves. For example, in a UDP flooding attack in the network domain, the attacker sends multiple concurrent UDP packets which causes the host application to throttle and reply with an ICMP destination unreachable packet. This event overloads the host machine and renders it inaccessible. A single UDP packet sent on a port is not an anomalous event. However, when their frequency exceeds a certain level, it becomes collectively anomalous. Figure 2.2 illustrates a famous example [11] of a human electrocardiogram output with a small region being highlighted in red which denotes a collective anomaly. Note that the low value of ECG itself is not an indication of abnormal behavior.

2.1 Background

The survey by Chandola, Banerjee, and Kumar [11] provides an exhaustive survey of various collective anomaly detection techniques developed until 2009. Collective anomalies majorly have two kinds of relations: sequential and spatial. Some techniques [9] convert the sequences into a finite set of features and run a point anomaly detection algorithm. One challenge in such methods is that the sequences can be of variable length. One of the techniques to overcome that challenge is box-modeling [10] which maps each term in the sequence to one of the boxes based on its value and use those

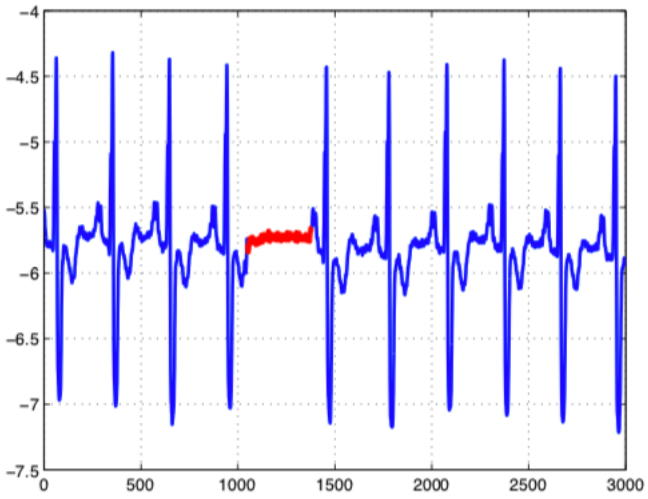

Figure 2.1: Collective anomaly corresponding to an Atrial Premature Contraction in an human ECG output

boxes as a set of features to use any of the point anomaly detection techniques. Recently, deep learning based techniques have also been developed to detect anomalous sequences with variable length. One such technique utilizes Long Short Term Memory RNNs [7] which predicts several time steps ahead of the input. If the prediction error is greater than some threshold for a given number of time steps, it indicates that the most recent sequence is a collective anomaly. Several clustering based techniques [8] have also been developed with similarity metric such as longest common subsequence, etc. to detect collective anomalies as a cluster. NKICAD algorithm [3] is another clustering based algorithm which first runs x-means algorithm [49] and then the features of the data are aggregated to form a new set of features and each cluster is re-clustered. The cluster with the minimum variance is returned as a collective anomaly. Information theoretic based co-clustering technique [2] has also been developed which return the largest row cluster as the candidate collective anomalies. Hurst Parameter based Collective Anomaly Detection (HCADET) [1] is a method which first runs x-means clustering on the entire dataset and then sorts the clusters based on their H [14] value and returns top-η clusters as a set of collective anomaly candidates.

Most of these techniques work in an unsupervised fashion with no need for labeled

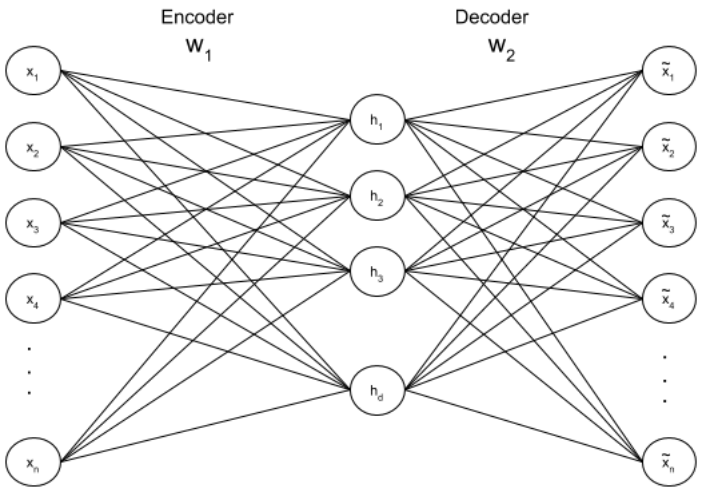

Figure 2.2: Collective anomaly corresponding to an Atrial Premature Contraction in an human ECG output

data. However, these techniques do not provide any insights about the attributes which are anomalous with respect to normal data.

In the next section, we will explain three techniques which can detect both anomalous records and attributes (called as anomalous patterns) along with their limitations.

2.2 Detection of Anomalous Patterns

2.2.1 Problem Statement

Anomalous Pattern Detection is slightly different from anomaly detection. Instead of identifying individual anomalies, the aim is to identify a set of records and attributes that together form an anomalous pattern. The underlying assumption of anomalous pattern detection is that the majority of data generated belongs to the same distribution representing normal behavior of the system and the groups of records that are unexpected given the typical data distribution are called anomalous patterns.

McFowland et al. [45] defines the anomalous pattern detection problem as *"detecting groups of anomalous records and also categorizing their features with the intention of understanding the anomalous process that generated these groups"*. The premise on

which anomalous pattern detection techniques work is that most records in the data are normal which follow an expected behavior and are generated by a single background process. A very small amount of data contains anomalous records that possess characteristics different from the expected behavior as they are generated by an alternate process. Most techniques are able to detect such anomalous records due to their significant difference in behavior. However, these techniques fail when the anomalous process is only slightly different than the background (normal) process. Anomalous pattern detection techniques rely on the assumption that records generated by the anomalous process are also self-similar just like normal records are expected to have similar characteristics among themselves. For example, in the case of network intrusion, the same task might be repeated a number of times to gain unauthorized access to a system. In health monitoring, a disease outbreak can lead to a large number of disease cases with almost identical features being reported. Anomalous pattern detection techniques rely on this assumption and try to look for collective anomalies by analyzing groups of records together.

We explain three major anomalous pattern detection techniques in this section. All three techniques extend the idea of Bayesian Network Anomaly Detector [20] where they first learn the distribution under null hypothesis H_0 using a Bayesian Network and then compute the likelihood of all records under H_0. Then, records with low likelihood values are reported as anomalies. As explained before, this is one of the point anomaly detection techniques and fails to detect records if they are only slightly anomalous under H_0. This limitation exists because a Bayesian Network can only learn the joint probability distribution of the entire data and none of the characteristics embedded in the group structure of the records. The three techniques that we will explain in this section utilize the Bayesian Network to detect groups of records and a subset of attributes that makes the group a collective anomaly.

2.2.2 Anomaly Pattern Detection (APD)

Anomaly Pattern Detection (APD)[21] technique utilizes Bayesian Network to detect anomalous subsets instead of anomalous records. It works in four steps. Step 1, by learning the distribution through Bayesian Network, it computes the likelihood of each record. Step 2, given a fixed anomaly score threshold,it makes all records exceeding it

as candidate anomalies. Step 3, it evaluates a set of candidate rules, each consisting of a conjunction of attribute values. For example, in KDD Cup Network Intrusion dataset, one possible rule could be *"protocol_type=tcp AND service=http"*. For each possible rule Q, it computes the number of records matching the rule Q. Formally, for any rule Q constituting of attributes $\{A_m = v \text{ AND } A_n = u\}$, records R_i and R_j can be in the group if and only if $A_{im} = B_{jm} = v$ and $A_{in} = B_{jn} = u$. Step 4, it then checks if the observed number of anomalies in the test set is significantly larger than the expected number of anomalies using Fisher's Exact Test. If the observed count is large then APD signals that some records in that subset might be anomalous and it also provides the set of attributes from the rule that was selected.

However, APD has several limitations. First, it considers grouping those records with matching attribute values per the selected rule. Such a strict constraint lowers the probability of detecting a subset of anomalous records with non-matching attributes according to the selected rule. Second, APD scores each record based on its individual anomalous behavior which limits its capability to detect records when they are collectively compared to normal records. Third, it can not detect patterns that are affected over more than 2 attributes as it considers rules comprising of only two attributes. Considering all possible combinations of attributes is computationally impractical. Hence, it might not be suited for high dimensional data. Four, it does not leverage self-similarity of anomalous records and hence records in the subset might not be generated by the same anomalous process. Hence, it is difficult to tell if the subset that is detected is truly an anomalous pattern or simply noise. Last, attributes detected by APD are the ones corresponding to the selected rule and hence might not correspond to the truly anomalous subset of attributes. Hence, APD doesn't provide any information about attributes that are affected by the anomalous process.

2.2.3 Anomalous Group Detection (AGD)

Anomalous Group Detection is another technique that detects a group of records in a categorical dataset such that they have similar characteristics. AGD overcomes some limitations of APD. Unlike APD, AGD makes sure that records detected are not noise or generated by a different anomalous process. Secondly, it also allows records to be

detected even when their attribute values do not match. Also, it enhances it's detection power by utilizing self-similarity among the attributes generated by the anomalous process.

First, AGD learns the probability distribution of the data under H_0 using Bayesian Network. Second, it identifies all the candidate subsets, based on a greedy search heuristic. It forms each subset iteratively based on a search heuristic which works as follows:
For each $R_i \in Data_{test}$, initialize $S \leftarrow R_i$, and then, add any $R_j \in Data_{test} - Data_S$ such that $F(S \cup R_j) > F(S)$ where $F(S)$ is likelihood ratio statistic defined as follows:

$$F(S) = \frac{P(Data_S|H_1(S))}{P(Data_S|H_0)} = \frac{\prod_{i \in S} P(R_i|H_1(S))}{\prod_{i \in S} P(R_i|H_0)} \tag{2.1}$$

where $H_1(S)$ represents an alternative hypothesis learned from the records in S. Third, for each subset S, compute the score as the likelihood ratio statistic of that subset, $F(S)$. Finally, the subset with the highest score is obtained as the most anomalous group. This scoring metric gives a higher score to anomalous records, as well as setting a constraint of similarity between records in a group. If the records in S are similar to each other, then the alternate hypothesis, $H_1(S)$ will be able to model them tightly. This will result in a high value of the likelihood $P(Data_S|H_1(S))$, thus increasing the score $F(S)$. Also, records that do not belong to the distribution under H_0 will have a low value of the likelihood $P(Data_S|H_0)$, again increasing the group score $F(S)$.

Still, AGD addresses only some of APD's limitations. AGD has several limitations too. To generate all possible subsets of records in a computationally feasible manner, it relies on the greedy search heuristic to reduce the search space. Because of this, the algorithm may fail to identify the most anomalous subset of the data. Furthermore, AGD does not provide any insights into the attributes that make the group of records anomalous.

2.2.4 FGSS using Bayesian Network

Fast Generalized Subset Scan (FGSS) [45] is the method which detects anomalous pattern by searching over subsets of records and attributes both. FGSS overcomes the limitations of both APD and AGD. It detects self-similar anomalous records and also provides information about the set of attributes that make the group anomalous. Unlike

APD which depends on the pre-defined rules or AGD which follows a greedy approach which limits their capability to find the most optimal pattern globally, FGSS guarantees the detection of the subset which is the most anomalous in the entire dataset. It assumes that all attributes are categorical variables.

Formally, let a set of records be denoted as $R_1...R_N$ and set of attributes as $A_1...A_M$ and for each record R_i, v_{ij} is the value of each attribute A_j. Any subset S is defined as $S{=}R \times A$, where $R \subseteq \{R_1...R_N\}$ and $A \subseteq \{A_1...A_M\}$. FGSS finds the most anomalous subset as

$$S^* = R^* \times A^* = \mathsf{argmax}_S \; F(S) \tag{2.2}$$

where R^* and A^* denotes the most anomalous set of records and attributes respectively and the score function $F(S)$ defines the anomalousness of the subset.

First, a Bayesian Network learns the distribution of data under H_0 that no anomalies are present. Second, for each value v_{ij} of the attribute A_j of record R_i, FGSS computes its likelihood l_{ij} given H_0. l_{ij} represents the conditional probability of the observed value v_{ij} given all its parent attribute values for record R_i under the null hypothesis H_0. Third, for each l_{ij} it computes an empirical p-value range p_{ij} (we explain this in more detail in further section) which serves as a measure of how uncommon it is to see a likelihood as low as l_{ij} under H_0. Fourth, it then begins searching for the subset S^* such that $F(S^*)$ is maximum in the following way; it first selects a random subset of attributes A' and it then looks for the subset of records R' that maximizes $F(R' \times A')$; then it fixes the set of records as R' and then it finds next set of attributes A'' which maximizes $F(R' \times A'')$. It keeps repeating this alternating process until it finds R^* and A^* and returns $S^*{=}R^* \times A^*$ as the most anomalous subset.

FGSS is currently the state of the art technique in terms of scalability, flexibility, and accuracy. FGSS uses the Higher Criticism (HC) statistic [25] and the Berk-Jones (BJ) statistic [6] as $F(S)$. Also, McFowland et al. [45] proved that these statistics follow LTSS property [48] which enables FGSS to scan both records and attributes by evaluating a linear number of subsets, making it scalable to datasets containing large number of records. We plan to explain FGSS in more detail in further sections.

Despite being such a powerful technique, FGSS framework has few limitations because it uses Bayesian Network to model the distribution of the data under the null. We first explain the theory of Bayesian Network and then understand how it limits the

capability of FGSS to detect collective anomalies.

Chapter 3

Motivation

3.1 Preliminary Theory

To understand the problems in the state of the art technique in collective anomaly detection i.e. FGSS, we need to understand what Bayesian Networks are and how they work. Then, we will go through its limitations and understand how it limits the effectiveness of FGSS framework. Then, we will explain how to overcome those limitations using deep neural networks. One such network is Autoencoder which we will embed in FGSS framework that is explained in more detail in the next section.

3.1.1 Bayesian Network

Probabilistic Graphical Model (PGM)[41] is a probabilistic model in the form of a graph structure to represent the conditional dependence between random variables. Each node in the graph represents a random variable, where the edges between the nodes represent probabilistic dependencies among the corresponding random variables.

Probabilistic graphical models with the directed acyclic graph (DAG) as its model structure are called as Bayesian Network [5]. A DAG is determined by vertices (or nodes) V and a set of directed edges E between those vertices. Each $A_i \in V$ represents a random variable (or an attribute) and an edge $e_{ij} \in E$ directed from a random variable A_i to A_j indicates a causal relation where the value of A_j is dependent of the value of A_i. We call A_i as the parent of A_j and A_j as the child of A_i. A Bayesian Network must satisfy two conditions. First, there should be no cyclic dependency in

the DAG structure, or in other words, no node should be the ancestor of itself. Second, each node is conditionally independent of other nodes given its markov blanket, or in other words, a node is only dependent on its parents, its children and other parents of its children. These two conditions enable the Bayesian Network to learn the joint distribution of variables in a computationally efficient way. This is achieved because of a significant reduction in the number of parameters (initially 2^M) required to represent the joint probability distribution. Apart from deciding the structure of the Bayesian Network, one needs need to compute the conditional probability distribution (CPD) at each node that depends only on its parents. In case of discrete random variables, CPD for each node can be written in the form of a table with each row representing a particular combination of its parent values based on which its value is determined. These individual CPD tables can be used to compute the joint probability distribution of the occurrence of more than one variable together.

Formal Definition

Formally, Bayesian Network [29] can be defined as follows: A Bayesian Network is defined by the pair $< G, \Theta >$, where $G =< V, E >$ represents a directed acyclic graph consisting of a set of vertices V and edges E. Each node $A_i \in V$ represents a random variable annotated by its conditional probability table, and each edge $e_{ij} \in E$ represents a causal connection from node A_i to node A_j. Bayesian Network represents the joint probability distribution over the set of all random variables V determined by a set of parameters Θ. Let $P(A_i)$ denote the set of all parents of A_i. The conditional probability table of A_i is determined by $\theta_{A_i} \in \Theta$ as follows:
For each value $a_i \in A_i$ and a particular instantiation of parents of A_i as $\pi_i \in P(X_i)$, θ_{A_i} contains a parameters $\theta_{a_i|\pi_i}$ defined as

$$\theta_{a_i|\pi_i} = p(a_i|\pi_i) \tag{3.1}$$

Once we compute $\theta_{a_i|\pi_i}$ $\forall a_i \in A_i$ for $i \in (1, N)$, a Bayesian Network specifies the joint probability distribution $\forall A_i \in V$ as follows:

$$p(A_1, A_2, ..., A_N) = \prod_{i=1}^{N} p(A_i|P(A_i)) \tag{3.2}$$

Figure 3.1 illustrates a simple example of Bayesian Network which depicts the probability of the software run to complete, denoted by C. This is only dependent on whether the software is run at a high speed or low speed, denoted by S. The speed of the software is further dependent on whether there is a hardware issue, represented by H, or if the WiFi speed is slow, represented by W. Also, there is high chance that if the wifi is slow, another website which is rendering some content also might be affected, denoted by Web.

The joint probability distribution of all variables can be calculated as follows:

$$P(H, W, S, Web, C) = P(H).P(W|H).P(S|H, W).P(Web|H, W, S, C) \\ .P(C|H, W, S, Web) \tag{3.3}$$

The total number of parameters required to compute this is $2^5 - 1 = 31$. However, based on this structure, we can observe some of the properties of Bayesian Network. H and W do not depend on any other variables. However, when S is given, they both fall in each other's markov blanket and are conditionally dependent. Also, given a value for W, S and W are conditionally independent as they are outside markov blanket of each other. Similarly, C is independent of all other variables given S. Because of such properties, Bayesian Network provides a compact factorization of the equation 3.3 as follows:

$$P(H, W, S, Web, C) = P(H).P(W).P(S|H, W).P(Web|W).P(C|S) \tag{3.4}$$

Now, we can observe from the equation 3.4 that we need 1 parameter each for $P(H)$ and $P(W)$, 2 parameters each for $P(Web|W)$ and $P(C|S)$ and 4 parameters for $P(S|H, W)$. Hence, the BN form reduces the number of the model parameters, which belong to a multinomial distribution in this case, from 31 to 10 (1+1+2+2+4) parameters. This reduction not only makes the learning computationally faster but it also makes the inference efficient by being robust to bias-variance trade-off.

Compared to learning the parameters of Bayesian Network, structure learning is a lot more difficult task. Learning a good topology of the Bayesian Network might require some prior information such as known causal relationships between attributes or other expert knowledge. With no prior knowledge, learning the best DAG structure is considered to be an NP-Complete [12] problem because the number of DAG structures

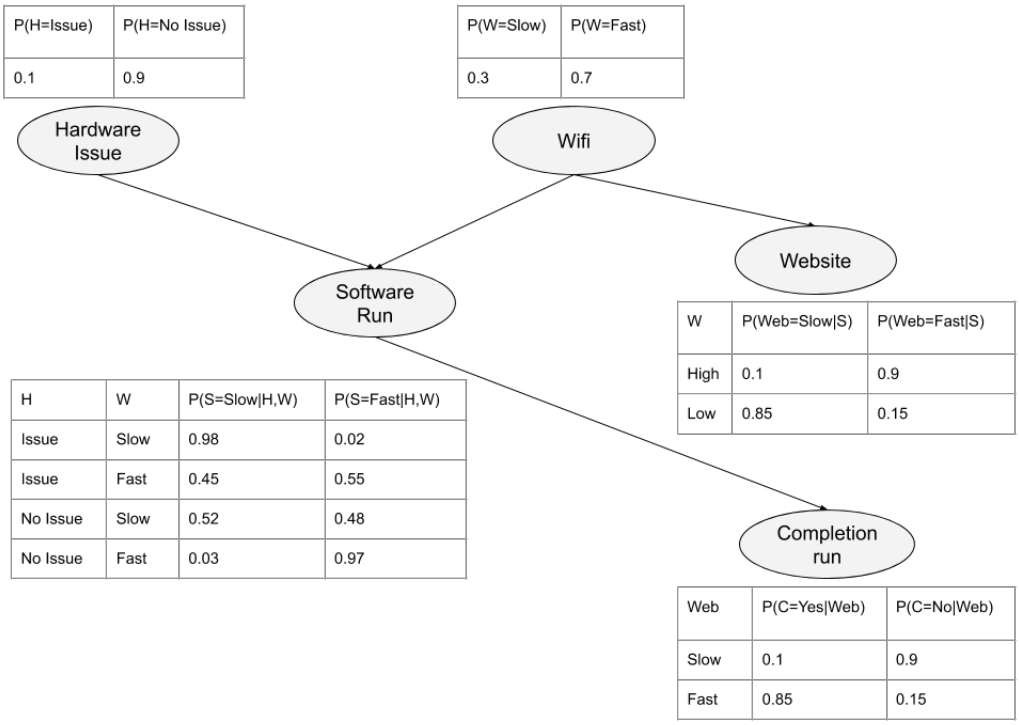

Figure 3.1: Illustration of software run completion using Bayesian Network

possible is exponential in terms of the number of random variables. Table 3.1 mentions the four scenarios for bayesian learning. Except for the first case, all other cases are computationally intractable [5]. For the second case, one can use EM to find a locally optimal maximum-likelihood estimate of the parameters. Third and fourth case requires searching through a model space in a greedy fashion which we will not be focusing on.

Case	BN structure	Observability	Proposed learning method
1	Known	Full	Maximum-likelihood estimation
2	Known	Partial	EM (or gradient ascent), MCMC
3	Unknown	Full	Search through model space
4	Unknown	Partial	EM + search through model space

Table 3.1: Four cases for BN learning problems

3.1.2 Autoencoder

Ian Goodfellow et al.[39] explains Deep Learning as *"a subset of Machine Learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones"*. Deep learning techniques[57] learn the mapping of input space to some output space incrementally through its hidden layer architecture, defining low-level features (such as words in text, peripheral edges of an object in image) and then higher level features (such as sentences in text, specific entities in the image). Each hidden unit of a trained deep learning network represents one aspect of the whole and together they help represent the mapping from input space to final output space.

One of the deep learning networks used for the task of representation learning in an unsupervised manner is Autoencoders [53]. Autoencoders are the deep learning models which aim to understand the representation of the data by trying to reconstruct it. Figure 3.2 shows the simplest auto-encoder which has a multi-layer perceptron (MLP) - like structure with one hidden layer. The difference between an MLP and an autoencoder is that the output layer in the autoencoder has the same cardinality of its input layer whilst the cardinality of the output layer in an MLP is the number of classes the perceptron should be capable of classifying. The motivation for this architecture is that hidden layer represents latent features [4] which, in other words, are the representation of the input in smaller latent space. Hence, an autoencoder plays an important role in dimensionality reduction: after the training phase, the output layer is usually thrown away and the encoder network is used to build a new dataset of samples with lower dimensions.

Formal Definition

Formally, for a k^{th} input vector x_k, w_1 and b_1 represent the weights and bias of encoder network E, and w_2 and b_2 represent the weights and bias of the decoder network D. A simple encoder could be represented as a network such that $D(E(x)) = x$. Encoder network E learns a stochastic mapping from input vector x to hidden vector

h as $p_{encoder}(h|x)$. Similarly, decoder network's task is to learn the reverse stochastic mapping as $p_{decoder}(\tilde{x}|h)$ where $\tilde{x}$ represents generated input. Just like any deep learning network, autoencoders are also trained using gradient descent technique where gradients are getting updated using backpropagation. The feed-forward computation in autoencoder can be mathematically written as follows:

$$h_1 = a(w_1 x + b_1)$$

$$h_2 = a(w_2 h_1 + b_2)$$

.

.

.

$$\tilde{x} = a(w_n h_{n-1} + b_n)$$

where a is an activation function. The loss function for an autoencoder can be defined as follows:

$$L(x, D(E(x)) = \frac{1}{2}\sum_k (\tilde{x_k} - x_k)^2 \tag{3.5}$$

where, hidden layer h for an autoencoder with single hidden layer (as shown in figure 3.2) is computed as

$$E(x) = h = a(x_k.w_1 + b_1) \tag{3.6}$$

where a represents some activation function, and output layer $\tilde{x}$ can be generated as

$$D(h) = \hat{x} = a(h.w_2 + b_2) \tag{3.7}$$

Here, $\tilde{x_k}$ is the k^{th} generated input and x_k is the k^{th} real input and L represents total squared reconstruction error.

3.2 Addressing limitations

Bayesian Network is a suitable machine learning model in many applications for various reasons [64]. One, Bayesian Network can give good generalization error even with fewer

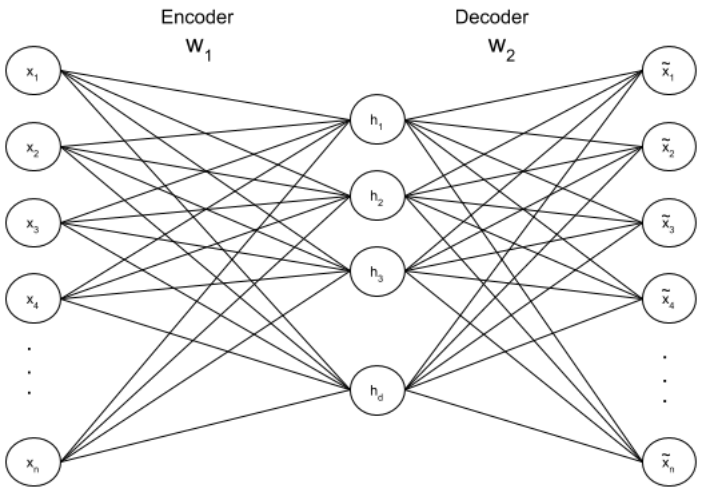

Figure 3.2: Autoencoder with one hidden layer illustrating its two components: encoder and decoder

data. Two, Bayesian Network work even when the data is missing [52] which makes it a popular model for data imputation or interpolation [43] [23] as it encodes all variables. Three, Bayesian Network is very powerful for it can also learn the best structure [44] to represent the conditional dependence among the features. This improves the explainability aspect of the model. Four, because it learns the dependence of variables on each other, it can also be used for causal inference urther helping in understanding the problem domain better. In fact, one can compute consequences given a cause and vice versa. This is done by calculating the probability of distributions of child nodes given parent nodes and vice versa. Five, Bayesian Network can be quick in answering queries once the model learns the structure and parameters as it contains the conditional probability distribution for every feature per every possible combination of its parent features.

However, Bayesian Network has several limitations [18], especially in the domain of collective anomaly detection when the anomalies are subtly anomalous when considered individually. We divide these limitations into 4 categories and explain each category of limitation and how they can be overcome by autoencoders.

First, Bayesian Networks work best with discrete variables. However, most of the real world data is continuous or a combination of discrete and continuous variables. Many techniques have been developed to deal with continuous variables. One such type

is discretization of continuous variables into intervals [30] [47] (i.e. changing continuous data into multinomial data). However, discretization might lead to some loss of information and Bayesian Network might end up learning approximate characteristics of the true distribution model. This might make the Bayesian Network lose its statistical capability to detect subtle anomalies. Another type of approach is using naive bayesian structure [66], and then learn the distribution in a non-parametric way. Some techniques deal with continuous variables directly by assuming a certain distribution for every node in the network [37] and then learning its parameters. Real world data is too complex and such probabilistic assumptions might not often be true leading to loosely learned probability distribution. These techniques do not learn the true characteristics of the data [15] because of various assumptions which might be significant in detecting anomalies especially if they are subtly anomalous. On the other hand, autoencoder is a non-parametric deep learning model that doesn't assume anything about the data and can directly work with continuous data. Hence, it ends up learning the true characteristics of the data with more confidence.

Second, in order for Bayesian Network to be effective, the data must have some structure in it in terms of features being well defined. However, most real world data is unstructured. Furthermore, Bayesian Network needs to be pre-fed with extracted meaningful features [51] [60]. On the other hand, autoencoder can directly work with raw unstructured data. For example, for detecting heart irregularities through its sound, one needs to compute features such as audio signal type, strength, duration of the sound, frequency, etc. to come up with a structure in order to use Bayesian Network whereas audio signals can be directly fed to an autoencoder which can learn the probability distribution of the sound. For unstructured data, learning latent variables becomes extremely important in order to learn a better approximation of the true distribution of the data. However, it is difficult to learn latent variables using Bayesian Networks. Even though there are many techniques [16] [54] which can make Bayesian Networks learn latent variables, one needs to specify information such as the number of hidden variables, cardinality of each hidden node, etc. Identifying all possible combination of random variables is exponential [26] and hence, providing an optimal number of variables is challenging. Autoencoders, on the other end automatically learn latent variables representing significant features in hidden layers. This is convenient when the

underlying anomalous process is only subtly anomalous and can only be distinguished in latent space. Also, Bayesian Networks are not effective in learning high dimensional unstructured data such as images, genomic data, etc. As the dimension increases, the efficacy of Bayesian Network in learning conditional dependence among features decreases. Several techniques that use Bayesian Network to work with images, audio, etc., require extraction of meaningful features [50]. On the other, Autoencoder can work with such high dimensional data as it learns significant features given a large number of features and then anomalies can be detected based only on the small set of significant latent features learned.

Third, the power of Bayesian Network depends heavily on the structure learned. However, defining and learning the structure which represents the true data is a difficult task [13]. In order to provide good priors to the network, expert knowledge [42] is needed for modeling Bayesian Network. For instance, initial order of variables that are needed to be learned, subjective probability of a given variable to be the parent of another variable [55] for the nodes keeping the order of nodes specified under consideration, etc. Such knowledge is hard to be specified by an expert. Even though there are techniques developed for obtaining such prior knowledge based on statistical property of the data, it still relies on approximate assumptions. Such assumptions might give away crucial information making it hard to identify slightly anomalous records. On the other hand, autoencoder doesn't assume anything in prior and instead starts with a randomly initialized weights and other defined hyperparameters and then tries to fit the data to that structure by updating weights, learning the statistical distribution that closely matches with the true distribution. Also, Bayesian Networks fail to learn the cyclic dependency among variables (for example, A depends B and B depends on A) and hence, can not learn the correlation between variables or combination of variables. In order for Bayesian Networks to learn the conditional dependence among features, all features must obey local markov property [5]. In fact, the structure learning problem itself is an N-P Complete problem. Though some techniques exist to relax the time complexity, they either follow a greedy approach based on some heuristics [56] [63] or restrict the space of models. In both cases, it might miss the best structure possible and learn an approximate model with high bias. Autoencoders, being a variant of Artificial Neural Network (ANN) which acts as a universal approximator [38], can learn

any structure without any restrictions like the absence of cyclic dependency, etc. This enables autoencoder to capture all dependencies and correlations in the data which might then make the detection of subtle collective anomalies efficient.

Four, there are several issues in training a Bayesian Network. One such issue is that learning the structure and parameters together is an NP-Complete problem [12]. This is because the objective function of the Bayesian Network is not a convex function. Although techniques [34] have been developed to make the objective function convex, it makes too many assumptions and as a result, the final learned model could actually be worse compared to the one obtained without the convex function. On the other hand, the objective function of autoencoders is based on the squared reconstruction error which is convex (refer equation 3.5). Also, in order to achieve good accuracy, Bayesian Network must learn all conditional dependencies among all variables. This might increase the model's complexity drastically. One can say, as the accuracy increases, complexity could also increase. However, in case of autoencoder, one can fix the model's complexity by various hyper-parameters like activation functions, number of hidden layers, number of hidden units in each layer, etc. And using gradient descent, it learns the best weights to fit the probability distribution and achieve desirable accuracy. From the qualitative perspective, one cannot justify how close the learned model is from true distribution because during the training process of Bayesian Network, the directions of dependence among variables might change multiple times and stop at a sub-optimal model. It can not be predicted if continuing the training process to next step gives a better model or worse. On the other hand, autoencoder's training process involves finding a good local minima in a convex objective function using gradient descent and hence gives an idea of how far the given model is from the true model based on how soon the training process is halted.

There are many other limitations of Bayesian Network such as a Bayesian Network cannot be applied in an online setting. Generally, a block of data is given to the learning algorithm to learn the structure and its parameters. On the other hand, autoencoder being a deep learning network can be trained by providing one sample at a time using stochastic gradient descent with batch size being 1 and hence, can be used in an online setting. However, we will not be focussing on such limitations in this thesis.

Chapter 4

Proposed Solution

We employ a deep autoencoder network to learn the probability distribution of the data. More specifically, we use autoencoder to learn the hidden significant features in the normal data records under the H_0. Using these hidden features, autoencoder then reconstructs the input and provides reconstruction error for that particular input. We then convert these reconstruction errors into empirical p-value ranges and then run FGSS on them to get a subset S such that $F(S)$ is maximum.

4.1 Property of reconstruction error

An autoencoder learns the significant features hidden in the latent space and reconstructs the input from those features with some errors. These reconstruction errors are independent and follow a normal distribution even though attributes are correlated. For one dimensional input, the magnitude of the reconstruction error tells how anomalous that input is. For multivariate input, one way to find out the anomalousness of the input instance is the summation of reconstruction errors across all attributes for that input. However, we focus on individual attribute's reconstruction error because our approach not only detects anomalous records but also provides information of the attributes whose reconstruction errors are unexpectedly high under H_0.

First, we split the dataset containing only normal records into two sets, $Data_{model}$ and $Data_{exp}$. We train the autoencoder on $Data_{model}$ to learn the probability distribution of the normal process that generated those records. Then, we generate the

reconstruction errors by feeding the $Data_{exp}$ to autoencoder. Let x_{ij} and $\tilde{x_{ij}}$ represent the j^{th} attribute of i^{ith} record for the original input $x_i \in Data_{exp}$ and reconstructed input $\tilde{x_i}$ respectively, where $j \in [1, m]$ for m attributes, reconstruction vector r_i can be represented as follows:

$$r_i =< r_{i1}, r_{i2}, ..., r_{ij}, ..., r_{im} >= (x_i - \tilde{x_i})^2 \tag{4.1}$$

where,

$$x_i =< x_{i1}, x_{i2}, ..., x_{ij}, ..., x_{im} > \tag{4.2}$$

$$\tilde{x_i} =< \tilde{x}_{i1}, \tilde{x}_{i2}, ..., \tilde{x}_{ij}, ..., \tilde{x}_{im} > \tag{4.3}$$

Here, r_{ij} indicates how likely is it to observe an attribute value for attribute j for given record x_i. Higher the reconstruction error r_{ij}, the more anomalous that attribute value is for the record.

4.2 p-value ranges

While performing the hypothesis test in statistics, the p-value helps to determine the significance of the results. A p-value is a measure of the evidence against the null hypothesis. Lower p-value (typically ≤ 0.05) indicates a stronger evidence against the null hypothesis, hence we reject the null hypothesis and accept the alternative hypothesis.

In the case of anomaly detection, a null hypothesis states that there are no anomalies present in the data and an alternative hypothesis states that the data contains anomalies. Typically, a lesser p-value for an instance indicates that the instance is an anomaly. For example, in Bayesian network [45], lesser p-value indicates lesser likelihood of that data instance under H_0. However, in autoencoder, anomalous data instances tend to have higher reconstruction error under H_0. Thus, we define p-value metric such that higher reconstruction errors correspond to lower p-values.

In order to compute the p-value, we first split the data containing no anomalies into $Data_{model}$ and $Data_{exp}$, train the autoencoder on $Data_{model}$ and calculate the reconstruction errors on $Data_{exp}$. These reconstruction errors represent the distribution of the errors of each attribute under H_0, represented by r_{ij}. Now, we take the test data $Data_{Test}$ which contains some anomalies and calculate reconstruction errors using the

trained autoencoder under H_0. Let r_{ij}^t represent the reconstruction error of j^{th} attribute of i^{th} test record.

Traditionally, an empirical p-value for j^{th} attribute of i^{th} test record can be computed as

$$p_{ij} = p(r_{ij}^t) = \frac{1}{n} \sum_{k=1}^{N_{train}} I(r_{kj} \geq r_{ij}^t) \tag{4.4}$$

where N_{train} denotes the total number of samples in $Data_{exp}$. One can note that attributes which are anomalous will have lower p_{ij} because of corresponding reconstruction error r_{ij}^t being larger. Apart from the test records having some true distribution, the reconstruction error r_j of an attribute j also has some distribution of reconstruction errors under H_0. Since we know that $Data_{exp}$ contains no anomalies in it, an empirical cumulative distribution function can be calculated using equation 4.4. If $Data_{test}$ has no anomalies present in it, then r_{ij}^t is from the same distribution as r_{ij}. In this case, p_{ij} asymptotically follows a uniform distribution(0,1) for each $j \in [1, m]$. If $Data_{test}$ has few records which are slightly anomalous, then p_{ij} will not follow uniform distribution(0,1) under H_0. Similarly, the attribute which has slightly anomalous values for some records are also not distributed as Uniform(0,1). This helps in getting insights about the attribute values which are slightly anomalous under H_0.

McFowland et al. [45] suggested to use empirical p-value ranges instead of p-values to avoid the bias towards larger p-values when there is a chance of having tied likelihood values in the case of Bayesian networks, in our case, tied reconstruction error values. For example, consider a column j with all identical values. It will result in $p_{ij} = 1$ for each record hence making it significant at some α. However, having all r_{ij}^t with identical values and similar to r_{ij} should not result in significant p-values. The occurrence of this scenario is really low when the attributes are continuous. However, for discrete attributes (for eg. x_{ij}=0 for some x_i), the autoencoder might reconstruct the exact same value for that attribute, x_{ij}^t resulting in identical p values. This leads to ties in reconstruction errors. Since most of the dataset might contain a mixture of continuous and discrete attributes, it is safer to use empirical p-value ranges compared to p-values.

To compute p-value range for the reconstruction error of j^{th} attribute of i^{th} record,

we compute two terms

$$N_{beat}(r_{ij}^t) = \sum_{k=1}^{N_{train}} I(r_{ij}^t < r_{kj}) \tag{4.5}$$

$$N_{tie}(r_{ij}^t) = \sum_{k=1}^{N_{train}} I(r_{ij}^t = r_{kj}) \tag{4.6}$$

Then, p-value range is a range bounded by two terms p_{min} and p_{max} i.e. $[p_{min}(p_{ij}), p_{max}(p_{ij})]$ where,

$$p_{min}(p_{ij}) = \frac{N_{beat}(r_{ij}^t)}{N_{train}+1} \tag{4.7}$$

$$p_{max}(p_{ij}) = \frac{N_{beat}(r_{ij}^t) + N_{tie}(r_{ij}^t) + 1}{N_{train}+1} \tag{4.8}$$

Once we have empirical p-value ranges for each r_{ij}^t, we can draw an empirical p-value from this range and if $Data_{test}$ and $Data_{exp}$ were from same distribution i.e. no anomalies were present in $Data_{test}$, then the drawn empirical p-values will be asymptotically distributed as Uniform(0,1). With reference to the example of identical column values mentioned before, this approach would lead to $N_{beat}(r_{ij}^t) = 0$ and $N_{tie}(r_{ij}^t) = 1$. Hence, this will result into an empirical p-value range of (0,1), and randomly drawing a p-value from this range would result into Uniform(0,1) which verifies our intuition.

4.3 Testing Hypothesis

To carry out statistical hypothesis testing, we need to find a significance level α which is the probability of rejecting the null hypothesis when it is true. Typically, to detect whether a uni-variate data point is drawn from a given distribution, we first compute the p-value of that data point under the null hypothesis that the data point belongs to the given distribution. Then, at a particular significance level α, we reject the null hypothesis if p-value $< \alpha$ or p-value $> 1 - \alpha$. We define the significance of a given p-value as

$$p_{sig} = I(p < \alpha \; or \; p > 1 - \alpha) \tag{4.9}$$

In our approach, in order to determine whether a given subset is generated from an anomalous process, we first calculate p-value range for that subset. Then, we determine

the proportion of p-value range significant at α. In other words, the probability of a value drawn from [$p_{min}(p_{ij})$,$p_{max}(p_{ij})$] is $< \alpha$ or $> (1 - \alpha)$. In case of an autoencoder, anomalous records or attributes tend to have higher reconstruction error under H_0 leading to lower p-values. In other words, for the anomalous set of records, attributes which are anomalous have their corresponding $p_{ij} < \alpha$. Hence, we determine the significance of th j^{th} attribute of i^{th} record as follows:

$$p_{sig}(p_{ij}) = \begin{cases} 0 & \text{if } p_{min}(p_{ij}) > \alpha \\ 1 & \text{if } p_{max}(p_{ij}) < \alpha \\ \frac{\alpha - p_{min}(p_{ij})}{p_{max}(p_{ij}) - p_{min}(p_{ij})} & \text{otherwise} \end{cases}$$

Then, significance of the subset $S = R^S \times A^S$ can be determined by adding all the significance values in the subset as follows:

$$p_{sig}(S) = \sum_{i \in R^S} \sum_{j \in A^S} p_{sig}(p_{ij}) \tag{4.10}$$

If we were considering a uni-variate data point such as $p_{sig}(p_{ij})$, we reject H_0 if $p_{sig}(p_{ij}) < \alpha$. However, in order to carry out hypothesis testing over the entire subset S, we first compute the expected count of $p_{sig}(p_{ij}) \in p_{sig}(S)$ significant at α as

$$p_{exp}(S) = \alpha \times N_S \text{ , where } N_S = \sum_{i \in R^S} \sum_{j \in A^S} 1 \tag{4.11}$$

Informally, $p_{sig}(S)$ denotes the observed number of p-value ranges in a subset S that are significant at α. Each p-value range is determined by the proportion of the range significant at α. On the other hand, $p_{exp}(S)$ denotes the expected number of p-value ranges in a subset S that should be significant at α. Based on these two terms, we reject null hypothesis H_0, if $p_{sig}(S) > p_{exp}(S)$, otherwise we assume that the subset contains normal records.

4.4 Metric to determine atypicality of subets

If the autoencoder is perfectly trained (i.e. it has learned the true typical characteristics of the normal records), it would be able to reconstruct the records with 0 reconstruction error. However, for the anomalous records, the attributes which diverge more from

normal behavior tend to have higher reconstruction errors. We can note that the atypicality of the attribute value is measured by how far it's reconstruction error is from the true expected value i.e. 0. Similarly, in order to determine the atypicality of the subset, we employ a non-parametric scan statistic called as the Berk Jones (BJ) statistic [6]. The BJ statistic for a given S gives us a score which indicates the divergence of the subset from the normal behavior. This divergence depends on the number of significant p-value ranges in the subset S. Formally, we employ the BJ statistic for a given S as follows:

$$\tau_S = \Phi_{BJ}(p_{sig}(S), N_S, \alpha) = N_S \times K(\frac{p_{sig}}{N_S}, \alpha) \tag{4.12}$$

where K measures the Kullback-Liebler divergence between the observed proportion of significant p-values at α i.e. $\frac{p_{sig}}{N_S}$ and expected proportion of p-values i.e. α.

Once we compute all the candidate subsets, the algorithm should calculate $\tau_S \; \forall \; S$, and then returns the most anomalous subset S^* such that τ_{S^*} is maximum. However, computing all possible candidate subsets is exponential in N, and is infeasible for even moderately sized datasets.

Algorithm 1 Deep FGSS

1. Given Training dataset D_{train}, Testing dataset D_{test}
2. Split D_{train} into $Data_{model}$ and $Data_{exp}$
3. Train Autoencoder on D_{model}
4. Compute reconstruction error matrix $E_{exp} = \{r_{ij}\}$
5. Compute reconstruction error matrix $E_{test} = \{r^t_{ij}\}$
6. Compute p-value matrix $P = \{p_{ij}\}$
7. For each R_i, form search group $SG_i = \{R_j\}$
8. For each SG_g where $g \in (1, G)$ and G denotes total number of search groups
   (a) Repeat for β number of iterations
      i. Initialize $A \leftarrow$ random set of $\{A_j\}$
      ii. Repeat until convergence:
         A. Set $R = \max_{R'} F(R^{'} \times A)$
         B. Set $A = \max_{A'} F(R \times A^{'})$
      iii. Set $S_g = R^* \times A^*$
9. Output $S^* = \max_{S_g} F(S_g)$

4.5 Efficient Search Technique

In order to resolve the issue of exponential time complexity of computing all possible subsets, most of the anomalous pattern detection techniques put some restrictions in the search space or follow a greedy approach to find the best set of subsets. Hence, these techniques might miss the most anomalous subset present in the dataset. Also, these techniques do not scale well with the size of the dataset, making it infeasible for larger datasets.

Daniel B. Neill [48] devised LTSS property which enables searching only N of the 2^N possible subsets of records and still provides us the guarantee of finding the most anomalous subset. The LTSS property states that *If we have a scoring function $F(S)$ to maximize over all subsets of records, and a ranking function $G(R_i)$ which assigns a rank $\Gamma_i \in (1, N)$ to R_i, the subset S^* with the maximum score $F(S^*)$ consists of the top-k ranked records.* In our case we have

$$F(S) = \tau_S \tag{4.13}$$

$$G(R_i) = \sum_{j \in (1,M)} p_{sig}(p_{ij}) \tag{4.14}$$

The intuition behind the above ranking function is that the anomalous records tend to have larger reconstruction error r_{ij} for some of the attributes, and inturn a larger reconstruction error r_i for the entire record. This results in a larger number of significant p-value ranges at level α for that record. This leads to a higher $G(R_i)$ value for the more anomalous record than the record which is slightly lesser anomalous. The records which are not generated by any anomalous process will have very fewer or no significant p-value ranges leading to lower $G(R_i)$ values. If we sort all the records in descending order based on their corresponding $G(R_i)$ values, we can form the subset $S^{(k)} = R^{(k)} \times A$ in a smarter way by including only the set of top-k ranked records $R^{(k)} = \{R_i\} \; \forall \; \Gamma_i \in (1, k)$ such that $F(S^{(k)}) > F(S^{(k+1)})$. This happens when the $(k+1)^{th}$ ranked record has zero or very less number of significant p-value ranges and is likely a normal record. Hence, intuitively all the subsequently lower ranked records with $\Gamma_i \geq k + 1$ are also likely to be generated by the normal process.

This way we can find the most anomalous subset of records given all the attributes. In order to identify the most anomalous subset of columns for a given set of records, we can perform the same optimization step for columns. For the given set of records, we can rank all the columns according to their corresponding values of $G(A_j)$ and incorporate only the top-k attributes in the subset $S^{(k^{'})} = R \times A^{(k^{'})}$ where $A^{(k^{'})} = \{A_j\} \; \forall \; \Gamma_j \in (1, k)$ such that $F(S^{(k^{'})}) > F(S^{(k^{'}+1)})$. Note that, here Γ_j indicates the rank of the attribute instead of record.

Formally, we define the above two optimization [36] steps for some α, total number of records N and total number of columns M as follows:

Step 1: For the set of columns A_j for $j \in (1, M)$, rank all the records R_i for $i \in (1, N)$ using the ranking function as

$$G(R_i) = \sum_{j \in (1,M)} p_{sig}(p_{ij}) \tag{4.15}$$

and then find $R^{(k)} = \{R_l : \Gamma_l \in (1, k)\}$ such that $F(R^{(k)} \times A) > F(R^{(k+1)} \times A)$

Step 2: For the set of rows R_i for $i \in (1, N)$, rank all the columns A_j for $j \in (1, M)$ using the ranking function as

$$G(A_j) = \sum_{i \in (1,N)} p_{sig}(p_{ij}) \tag{4.16}$$

and then find $A^{(k)} = \{A_l : \Gamma_l \in (1, k)\}$ such that $F(R \times A^{(k)}) > F(R \times A^{(k+1)})$

Once we find $R^{(k)}$ in the Step 1, we set $R = R^{(k)}$. Step 2 gives $A^{(k)}$ for the current set of records R. We then set $A = A^{(k)}$. We repeat the above two steps alternatively and at the end of every iteration we keep getting a better set of records and columns which maximizes $F(S)$. We stop this alternative process until we find the most optimal subset $S^* = R^* \times A^*$ such that $F(S^*)$ is maximum. Intuitively, we treat the entire data as the subset and keep shortening the subset by removing a set of records and columns. The criteria for the removal is that the difference between the observed number and expected number of significant p-value ranges should increase. This repeats until it converges i.e. when the difference is maximum. At this point, we obtain the most anomalous subset in the entire dataset. Note that we always start the alternating optimization process with a random set of columns or records. Hence, we repeat this convergence process for different random initialization and return the subset with a maximum score among all such iterations.

The underlying assumption is that all normal records are generated by a single background process and we learn the characteristics of this process using autoencoder under H_0. Also, there is a separate anomalous process that generates anomalous records whose characteristics are different compared to the normal process. We can take advantage of such a disparity by inducing similarity constraint to group similar records. We then run the above optimization process within each group. This will return not only the most optimal subset but also ensures that all the detected records have self-similar characteristics. Another advantage of enforcing a similarity constraint is that in the presence

of different types of anomalies, it will ensure that the detected subset contains all the records that belong to a single anomaly type. Different types of anomalies are generated by their corresponding anomalous processes which differ slightly from each other. To compute the similarity between two records, we can use metrics like Euclidean distance, Correlation-coefficient, Cosine similarity, etc. Formally, a search group for a record R_i of radius r can be defined as $\{R_j : d(R_i, R_j) \leq r\}$ where d measures the similarity between two data records. We can choose r depending on how strictly we want to enforce the similarity constraint. Also, one should be careful in choosing the metric as it depends on the input type. For instance, choosing Euclidean distance as the distance metric for images might not be suitable. A similarity metric should give a high value while comparing two normal records and a low value if we compare a normal record with an anomalous record.

4.6 DeepFGSS Algorithm

DeepFGSS algorithm is an extension of the original FGSS algorithm [45] developed by McFowland et al. which uses deep networks like autoencoders to enable FGSS to work with high dimensional unstructured data with both discrete and continuous attributes. Note that DeepFGSS works not only with autoencoder but any deep neural network architecture for learning the probability distribution under the null hypothesis.

We illustrate the main steps of DeepFGSS in algorithm 1. We first split the training data that contains no anomalies into two parts: $Data_{model}$ and $Data_{exp}$ and train autoencoder on $Data_{model}$ to learn the probability distribution of normal records under the null hypothesis H_0 that no anomalies are present. We then calculate reconstruction error of $Data_{exp}$ as E_{exp} which represents the expected distribution of reconstruction errors of data under H_0. We also calculate reconstruction errors of the $Data_{test}$ as E_{test} and then, compute p-value ranges for the reconstruction errors of each record's attribute value $r^t_{ij} \in E_{test}$ by comparing it with the reconstruction errors $r_{ij} \in E_{exp}$ $\forall i \in (1, N)$ of the same column j. We then randomly draw a p-value $p_{ij} \in$ Uniform$[p_{min}(p_{ij}), p_{max}(p_{ij})]$ for all the attributes for each record. Optionally, we can enforce similarity constraints by forming search groups for each R_i in $Data_{test}$ using some distance metric d and radius r. Then, we can run the two alternating optimization steps within each

search group and identify the most anomalous subset in that search group as S_g. The algorithm then returns the subset S_g with maximum BJ statistic score $F(S_g)$ as S^*.

Note that, unlike FGSS, there is an additional requirement in DeepFGSS to store the distribution of the normal data E_{exp} along with the trained model which represents the expected probability distribution of the normal data. Autoencoder maps the raw input space to the probability distribution space of reconstruction errors.

There are various parameters that can be manually tuned in DeepFGSS. In order to choose the significance level α, one can consider all the unique reconstruction error values in the test dataset within some α_{max} and run the optimization steps for each of the unique $\alpha \leq \alpha_{max}$. The value of the α_{max} can be set depending on the disparity between anomalous and normal records. Anomalies which are easily detectable lie on the extreme corners when you plot the distribution of the normal records. However, anomalies which are slightly anomalous, when considered individually, tend to share many characteristics with the normal records and hence do not necessarily lie on the extreme ends of the distribution. Also, one can choose any distance metric d depending on the data and radius r depending on how small/strict the neighborhood should be. Apart from these, if we have some prior expert knowledge of the data, we can add tweaks such as the minimum or the maximum number of records a neighborhood can have. We can also choose other non-parametric scan statistics such as the Higher Criticism (HC) [25] statistics which can determine the similarity between two subsets. Along with these parameters, the deep learning model itself can be treated as a parameter where one can choose models such as Bidirectional Generative Adversarial Network [24] , Long Short Term Memory [7] for time series data, etc. We do not employ any other model in this thesis but the similar idea can be extended to other deep learning models and use FGSS on top of it.

4.7 Computational Complexity

The most critical challenge in anomalous pattern detection is the infeasibility of evaluating all the possible subsets of the dataset. Even for a moderately sized dataset consisting of N records and M attributes, there are a total number of $2^N \times 2^M$ possible subsets.

FGSS takes advantage of LTSS property to reduce the problem of evaluating all subsets from $O(2^N \times 2^M)$ to $O(N)$. To compute ranks for each record, we need to sum over the p_{sig} of all the attributes in $O(M)$ complexity for each record leading to $O(NM)$ for all records. After computing the ranks for all the records, sorting them takes $O(N\log N)$. The worse case time complexity for evaluating top-k ranked subsets is $O(N)$. Similarly, for ranking each column, computing p_{sig} for all records takes $O(MN)$ and sorting them takes $O(M\log M)$. Hence, evaluating top-k ranked subsets of columns takes the worst case time complexity of $O(M)$. Each single optimization run comprises of maximizing over both records and columns that takes time complexity of

$$\begin{aligned} O(NM) + O(N\log N) + O(N) + O(MN) + O(M\log M) + O(M) = \\ O(NM + N\log N + M\log M) \end{aligned} \tag{4.17}$$

Considering γ is the average number of times the optimization ran before converging for each β number of iterations, and u is number of unique alpha values $\leq \alpha_{max}$, the total time complexity becomes

$$u \times \beta \times \gamma \times O(NM + N\log N + M\log M) = O(u\beta\gamma(NM + N\log N + M\log M)) \tag{4.18}$$

On enforcing similarity constraints, FGSS runs within each neighborhood. The maximum number of neighborhoods possible is N with k being the maximum number of records a neighborhood can be comprised of. In case of constrained FGSS, total time complexity across all neighborhood becomes

$$O(u\beta\gamma N(kM + k\log k + M\log M)) \tag{4.19}$$

Note that the above time complexity is still less than $O(2^N \times 2^M)$. Also, one can enforce stricter constraints on the number of neighborhoods, minimum/maximum neighborhood size (k) and number of unique alpha values (u) to improve the running time with a reduced chance of finding the most optimal subset.

Chapter 5

Evaluations

In this section, we explain the performance of DeepFGSS on two datasets: network intrusion and handwritten digits. We evaluate the performance in two ways. One, we measure the efficiency of how well DeepFGSS can distinguish between datasets containing and not containing anomalies. We plot power curve and ROC curve for evaluating this measure. The other performance metric determines how well DeepFGSS can detect anomalies in the dataset. We plot jaccard curve and precision-recall curve for determining the detection accuracy. We compute these metrics for varying dataset sizes, namely 100, 250, 500, 750 and 1000 with varying proportion of anomalies, namely 0%, 3%, 6%, 9%, 12% and 15%.

5.1 Experiments

5.1.1 Power curve

Power of an algorithm is defined as the probability of rejecting the null hypothesis when it is false. In our case, null hypothesis H_0 represents no anomalies are present in the dataset and alternative hypothesis H_1 represents the dataset contains some anomalies. Mathematically, power can be defined as follows:

$$\text{power} = p(\text{reject } H_0 \,|\, H_1 \text{ is true}) \tag{5.1}$$

Power of DeepFGSS depends on three primary factors: significance criterion, size of the dataset and the proportion of anomalous records. We study the power of DeepFGSS

at a significance criterion, $\alpha_p = 0.05$, for dataset size N varying as 100, 250, 500, 750 and 1000 with proportion of anomalies *prop* increasing as 0%, 3%, 6%, 9%, 12% and 15%. To calculate the power at a given N and *prop*, we first randomly draw 100 datasets D_{normal} consisting of N number of records with no anomalies in it and 10 datasets D_{test} of N number of records with *prop* % anomalies in it and calculate the dataset score for each of them. We define the dataset score as the score of the most anomalous subset found in that dataset i.e.

$$\tau_{D^i} = \tau_{S^* \subseteq D^i} \tag{5.2}$$

where S^* is the most anomalous subset in i^{th} drawn dataset D^i. We then compute the power of DeepFGSS for D_{test} at a given dataset size N and proportion of anomalies *prop* as follows:

$$O^i_{beats} = \sum_{j=1}^{100} I(\tau_{D^i_{test}} > \tau_{D^j_{normal}})$$

$$E_{beats} = 100 \times (1 - \alpha_p)$$

$$power(D_{test}) = \frac{\sum_{i=1}^{10} I(O^i_{beats} > E_{beats})}{10}$$

Here, O_{beats} is the observed number of times the score of the anomalous dataset is greater than scores of normal datasets drawn whereas E_{beats} is the expected number of times the score of the anomalous dataset must be greater than the scores of normal datasets drawn.

Informally, power of the DeepFGSS tells what should be the minimum proportion of anomalies in order for the algorithm to perform well for a given dataset size at a given significance criterion. For example, a power of 0.7 means that 7/10 times DeepFGSS is able to detect anomalous subsets (which makes the score of the dataset higher) in a given sized dataset and given proportion of anomalies.

5.1.2 ROC curve

In this experiment, we treat the task of distinguishing normal and anomalous datasets as a binary classification problem and plot Receiver Operating Characteristic (ROC)

curve. In our case,

$$\text{True Positive Rate}(TPR) = \text{Proportion of anomalous datasets detected as anomalous}$$
$$\text{False Positive Rate}(FPR) = \text{Proportion of normal datasets detected as anomalous}$$

The ROC curve signifies the the ability of DeepFGSS algorithm to differentiate between anomalous and normal datasets as its discriminating dataset score's threshold is varied. Area under the ROC curve determines the probability of ranking anomalous datasets higher than the normal datasets. In other words, we will obtain a higher area under the curve when DeepFGSS assigns high scores to anomalous datasets compared to normal datasets. Also, it is worth noting the fact that the power curve tells about the power of DeepFGSS for a given dataset size and proportion of anomalies at a fixed significance criterion α_p. However, ROC curve tells about the efficiency of DeepFGSS for a given dataset size and proportion of anomalies as we vary α_p because α_p is equivalent to the false positive rate.

To plot ROC curve, we randomly draw 5 sample datasets from the normal datasets under H_0 and 5 sample datasets of size N containing *prop* % anomalies in it. We then compute scores for all 10 datasets and compute TPR and FPR at varying threshold values and then plot ROC curve and compute the area under that curve.

5.1.3 Jaccard curve

In order to evaluate how well DeepFGSS is identifying the anomalies as a subset, we evaluated the accuracy within the detected subset using the Jaccard index. Jaccard index (also called as the Jaccard similarity coefficient) is simply a ratio of the size of the intersection to the size of the union. Jaccard index, in other words, is a measure of the similarity between two sample sets.

We use Jaccard index to evaluate the similarity between how the predicted set of anomalous records and the actual set of anomalous records as we increase the proportion of anomalies in the dataset. To compute Jaccard index, we define two sets T^a and T^p of size N as follows:

$$T_i^a = \begin{cases} 1, & \text{if } R_i^t \text{ is anomaly} \\ 0, & \text{otherwise} \end{cases}$$

$$T_i^p = \begin{cases} 1, & \text{if } R_i^t \in S^* \\ 0, & \text{otherwise} \end{cases}$$

and, then compute Jaccard index as follows:

$$J = \frac{\sum_i^N I(T_i^a = 1 \text{ and } T_i^p = 1)}{\sum_i^N I(T_i^a = 1 \text{ or } T_i^p = 1)} \tag{5.3}$$

We compute J for a given dataset size N with *prop* increasing as 0%, 3%, 6%, 9%, 12% and 15%. We generate 5 jaccard curve plots each for N = 100, 250, 500, 750 and 1000.

5.1.4 Precision-Recall curve

Another way to assess the quality of the subset detected by DeepFGSS is by evaluating how well it ranks more anomalous subsets over less anomalous subsets. To evaluate this, we obtain 5 most anomalous subsets and assign each record a score that is a summation of two terms: the score of the subset that it belongs to (otherwise 0) and its reconstruction error. Formally, we can write the score of the record R_i as:

$$L(R_i) = \tau_S + \sum_{j=1}^{M} r_{ij} \quad \text{where } R_i \in S \tag{5.4}$$

The idea being all subsets are first ranked based on the subset scores τ_S and then all records within the subset S are ranked based on their individual record's reconstruction r_i. This scoring metric allows records to have higher score either because they are similar to other anomalous records or are individually anomalous. Also, it helps in detecting For $R_i \notin \{S_1, S_2, S_3, S_4, S_5\}$, where S_i is i^{th} most anomalous subset, $L(R_i) = 0$. We then compute precision and recall at various thresholds and plot the precision-recall curve for N = 100, 250, 500, 750 and 1000, and *prop* = 0%, 3%, 6%, 9%, 12% and 15%. We then report the area under precision-recall curve. Higher area indicates that DeepFGSS is able to rank more anomalous subsets higher than less anomalous subsets within a dataset. Within the ranked subset, DeepFGSS is ranking more anomalous records higher than the less anomalous records.

5.2 Datasets

5.2.1 Network Intrusion

We use KDD Dataset [17] which is a famous benchmark dataset in the field of Network Intrusion Detection. This was developed as part of The Third International Knowledge Discovery and Data Mining Tools Competition to build a system which can detect bad connections (called as intrusions or attacks) in a simulated military network environment. KDD training dataset contains approximately 4,000,000 single connection records. Each connection record consists of 41 features and a label. These 41 features include basic characteristics of an individual TCP connection like protocol type, duration, total bytes transferred etc; domain specific features like total number of failed login attempts, total number of operations to create file etc or; traffic features computed in a two second time window like count of the connections to the same host or same service or to different hosts etc. There is a total of 7 discrete and 34 continuous features. FGSS requires all the continuous features to be discretized into 5 levels losing crucial characteristics of 34 out of 41 features. On the other hand, DeepFGSS directly works on continuous and discrete variables. We have selected two of the common attacks for our evaluation: Apache2 and Neptune.

We use autoencoder with 3 hidden layers. The first and third hidden layers contain 10 units and the second layer contains 5 units. We used ReLU activation function and RMSProp optimizer to train the model at a learning rate of 0.001, batch size of 50 and 20000 epochs. We also used Euclidean distance to enforce similarity constraints with a mean of the Euclidean distance between any two instances as the radius.

Figure 5.1 shows the power curves for different dataset sizes showing how the average power of DeepFGSS changes with an increase in the proportion of anomalies. DeepFGSS is able to distinguish datasets perfectly at 12% anomalies or greater for Apache2 and 6% or greater for Neptune. Also, the steady increase in power is also proportional to the size of the dataset. It is clearer in the case of Apache2. When the proportion of anomaly is 9%, power is 0.2 for dataset size of 100 records and 250 records, then power grows to 0.25 for dataset size of 500 records, 0.3 for dataset size of 750 records and power reaches 0.47 when the dataset contains 1000 records. For Neptune, at 6% there is a slight increase in the power from 0.8 reaching 1.0 when the dataset contains at least 500

records. At lower proportions of anomalies, the tendency of DeepFGSS to differentiate between anomalous and normal datasets is very low. This is because DeepFGSS is focussed on detecting the subset with maximum score τ_S and this score can be higher for a larger subset even though it contains only normal records with each individual record adding to τ_S. However, for larger proportions of anomalies, the smaller subsets formed by anomalous records are sufficiently large to be detected as the most anomalous subset leading to a higher power value. Minor errors in the power values (for example at smaller proportions of anomalies when N=100) can be due to the random sampling of the datasets. We can conclude from figure 5.1 that it is easier to differentiate datasets containing Neptune attacks than Apache2 attacks.

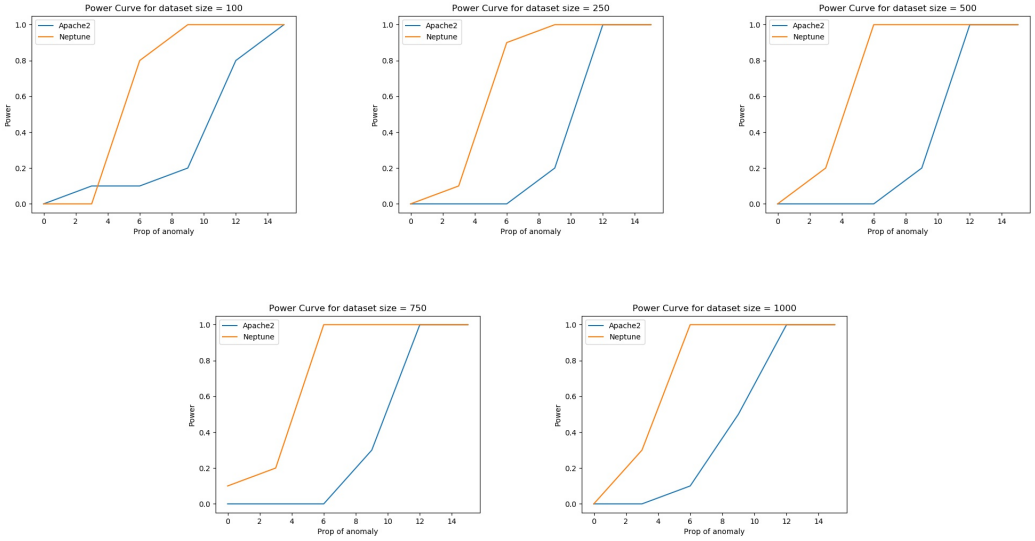

Figure 5.1: Power curves for KDD dataset for two types of anomalies: Apache2 and Neptune

Figure 5.2.1 shows a heatmap of the area under the ROC curve for KDD dataset for both the attack types. We observe similar trends through ROC curves that the ability of DeepFGSS to rank anomalous datasets higher than normal datasets increases with the increase in the proportion of anomalies. DeepFGSS can perfectly rank all the anomalous datasets higher than all the normal datasets when the proportion of anomalies is greater

than 12% in the case of Apache2 and 6% in the case of Neptune. Also, it is interesting to note that for a fixed proportion of anomalies, the area under ROC curve increases gradually with the increase in the dataset size.

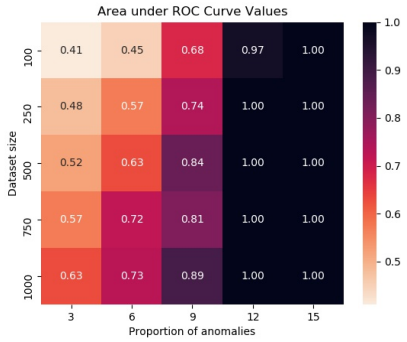

Figure 5.2: Area under ROC for Apache2

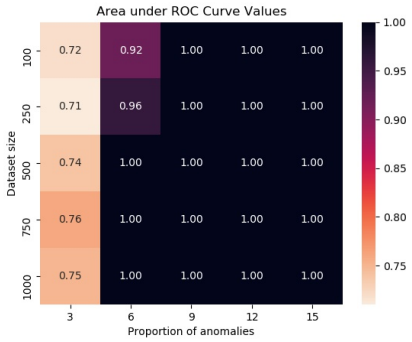

Figure 5.3: Area under ROC for Neptune

Figure 5.4 shows the jaccard curve for different dataset sizes for both anomaly types. As the proportion of anomalies increases, the average Jaccard index also increases. We observe that the Jaccard index reaches around 0.8 at 12% anomalies in the case of Apache2 and at 6% in the case of Neptune. The maximum value of Jaccard index is observed to be around 0.85 and 0.95 for Apache2 and Neptune respectively. From our experiments, we identified that the reason for the steady increase in the Jaccard index is the increase in precision without much drop in the recall.

Figure 5.5 and 5.6 shows the precision-recall plots for the dataset size of 1000 records for both Apache2 and Neptune, respectively. We can observe that precision starts from low value for both Apache2 and Neptune at some cases. This indicates that sometimes the highest ranked subset might contain mostly normal records. However, it is being ranked high because of its size being large making the cumulative score of the entire subset high. After this initial low precision, we see a sudden jump in the precision value. This provides some confidence that the subsequently ranked subsets are the anomalous ones containing records most of which are anomalies. Precision-recall plots for all other dataset sizes can be found in Appendix A and Appendix B.

Figure 5.2.1 shows a heatmap of the area under precision-recall curve for KDD dataset for both anomaly types. It can be clearly observed that the area under precision-recall curve increases as the proportion of anomalies also increases. This is more evident

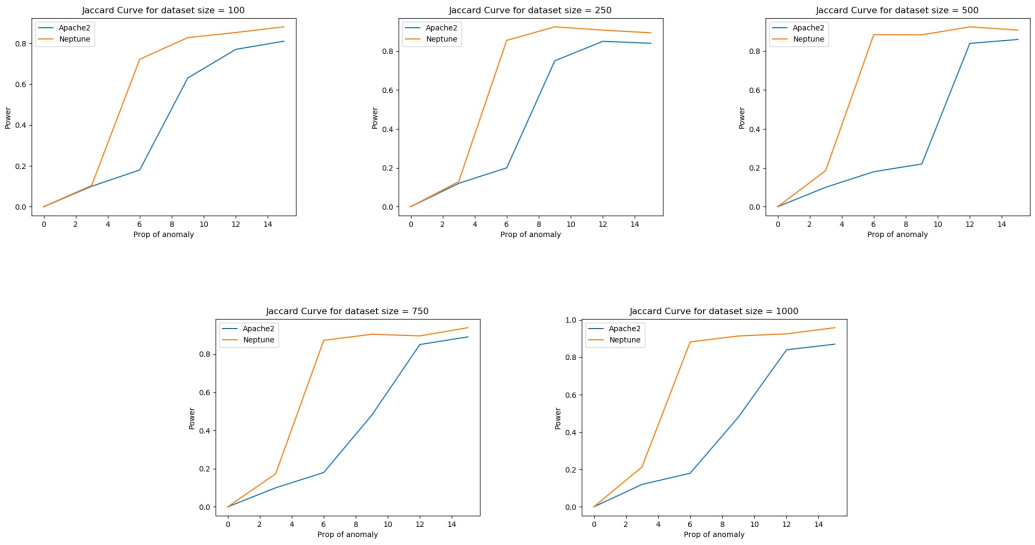

Figure 5.4: Jaccard curves for KDD dataset for two types of anomalies: Apache2 and Neptune

in the case of Apache2 attack type. DeepFGSS is able to rank the subsets and records within the subset best when the proportion of anomalies is greater than 12% for Apache2 and 6% for Neptune. Also, since the area under precision-recall curve is not 1, it means that some of the records in the detected subset are normal records but most of the records are anomalous.

5.2.2 Handwritten Digits

The MNIST database is a famous benchmark dataset for testing machine learning performance on images. It consists of 60,000 training and 100,000 test samples of handwritten digits. Each record in the dataset is flattened out vector of pixel intensities of a 28×28 image. Each digit is labeled as one of the digits from 0-9 which tells the number written in the image. For the anomaly detection dataset, we treat all the images of 0 digit as the training set under the H_0 and all the images of digit 7 as anomalies. In order to create test dataset, we inject some of the images from class 7 into the images of class 0 and then run DeepFGSS on it.

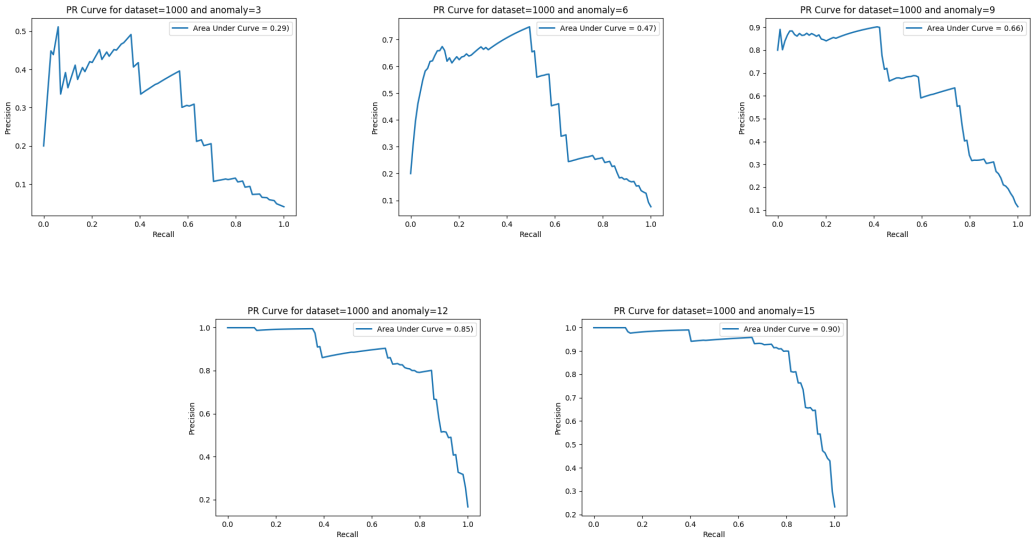

Figure 5.5: Precision-Recall curves for KDD dataset with N=1000 and Apache2 anomaly

We learned the distribution of the data belonging to the class 0 using the autoencoder with the same architecture as we used for KDD dataset and learning rate of 0.001. We use sigmoid activation function for learning the distribution of pixels after scaling all the pixel values to 0-1 scale. We used a smaller batch size of 5 and 1000 epochs. We did not enforce similarity constraints for this dataset.

We can see similar trends of increase in the power with the increase in the proportion of anomalies (refer figure 5.9). DeepFGSS is able to differentiate between normal datasets and anomalous datasets perfectly when 9% of the dataset is comprised of anomalies at a size of 100 records and for larger datasets at least 6% anomalies. Also, when the proportion of anomalies is 3%, the power grows steadily from 0.3 at dataset size of 100 records to 0.8 when the number of records reaches 1000. This can be explained by a simple intuition that 3% of 1000 is greater than 3% of 100 leading to a greater possibility of finding a larger dense subset comprising of a relatively large number of anomalies.

From figure 5.10, we can observe that DeepFGSS is able to rank all the anomalous datasets higher than the dataset containing no anomalies perfectly when there is at

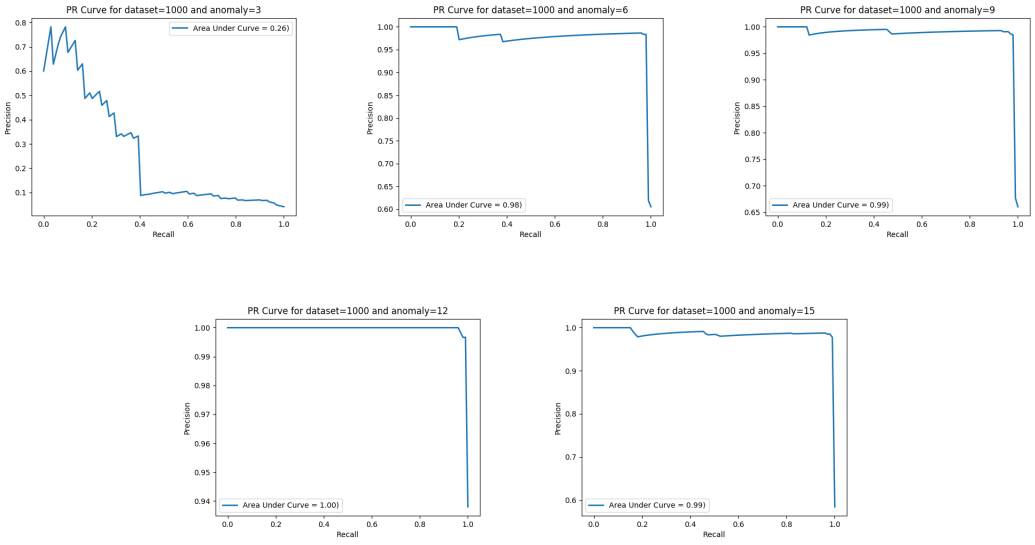

Figure 5.6: Precision-Recall curves for KDD dataset with N=1000 and Neptune anomaly

least 6% anomalies in it. For larger datasets, DeepFGSS can easily differentiate between normal and anomalous datasets evident by higher AUC values. Also, at 3% anomalies, we see an increase in AUC with the increase in the size of the dataset.

We can infer similar results from the jaccard curve plots for MNIST (refer figure 5.11). Also, the increase in the Jaccard index slightly becomes steady above 9% proportion of anomalies in the dataset. It can be noted that the highest Jaccard index score obtained among all dataset sizes is around 0.6 compared to a Jaccard index of 0.8 in case of KDD dataset. This is because of not enforcing similarity constraints leading to slightly lesser precision.

From the precision-recall plots for dataset size of 1000 records shown in figure 5.12, we can infer similar trend as we have seen in the case of KDD dataset. Refer Appendix C for precision-recall plots for other dataset sizes. Also, the heatmap shown in figure 5.13 shows a higher area under the precision-recall curve for all dataset sizes when the proportion of anomalies is at least 9%. It can be noted that the area under precision-recall curve here is slightly lesser compared to KDD dataset. This can be improved by inducing similar constraints which can increase the precision of DeepFGSS by obtaining

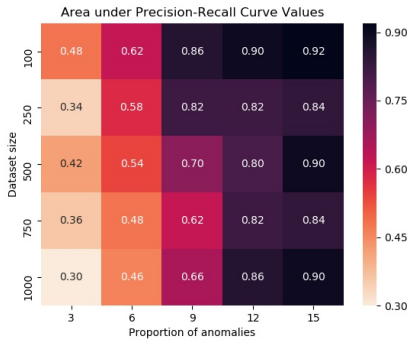

Figure 5.7: Area under Precision-Recall curve for Apache2

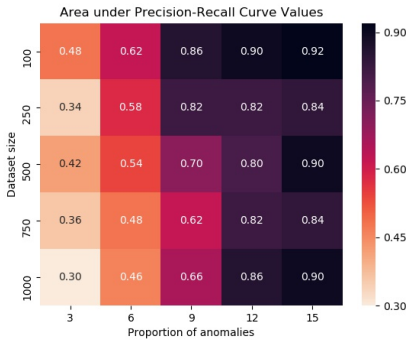

Figure 5.8: Area under Precision-Recall curve for Neptune

subsets in a smaller neighborhood.

From all the experiments, we can infer that the performance of DeepFGSS increases with an increase in both dataset size and proportion of anomaly. Also, DeepFGSS is able to perform sufficiently well at 12% anomalies for Apache2 and 6% anomalies for Neptune in KDD dataset and 9% for images of digit 7 in MNIST dataset. Note that FGSS is not able to work with MNIST dataset as it is a high dimensional and unstructured data whereas DeepFGSS is able to predict anomalies in such a dataset with sufficient accuracy.

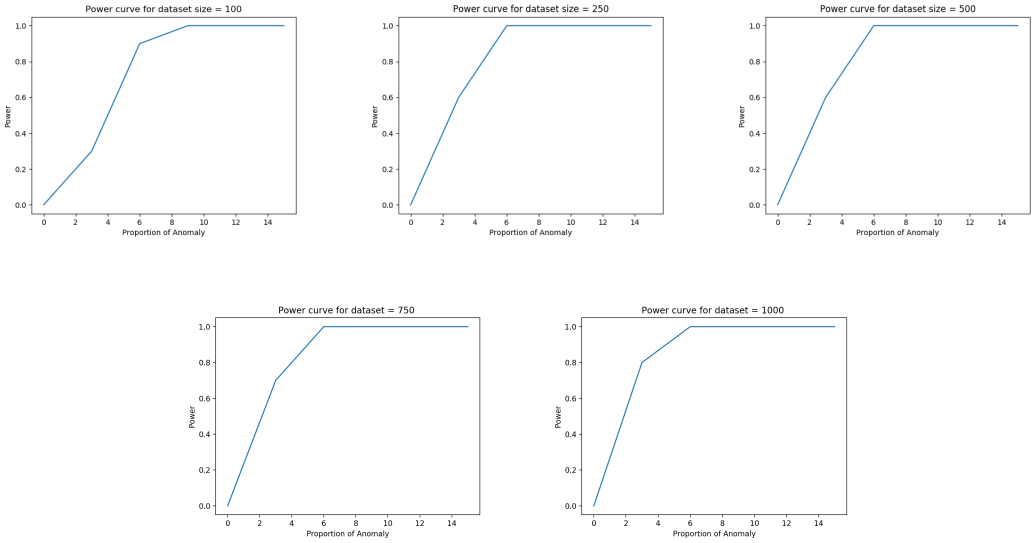

Figure 5.9: Power curves for MNIST dataset

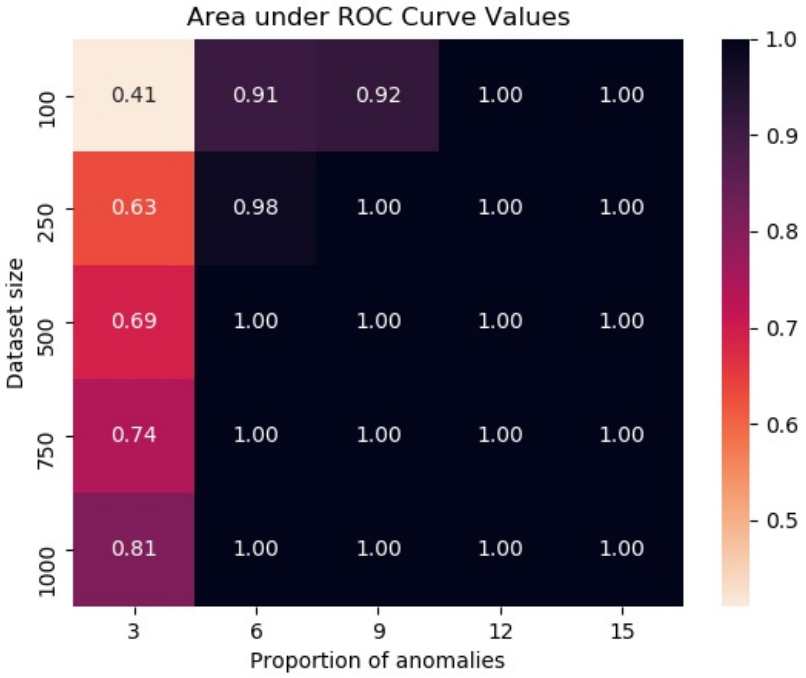

Figure 5.10: Area under ROC curve for MNIST dataset with different dataset size and proportion of anomalies

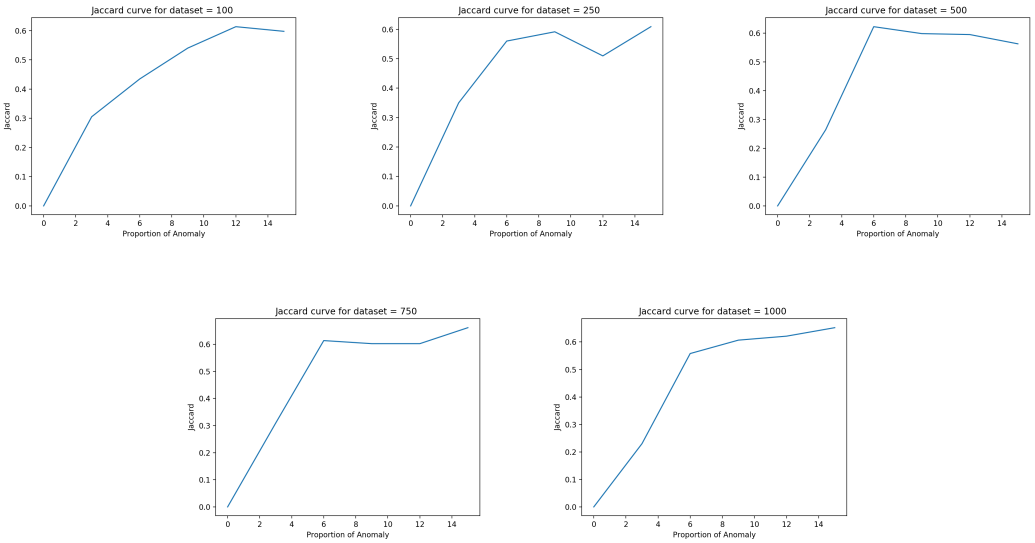

Figure 5.11: Jaccard curves for MNIST dataset

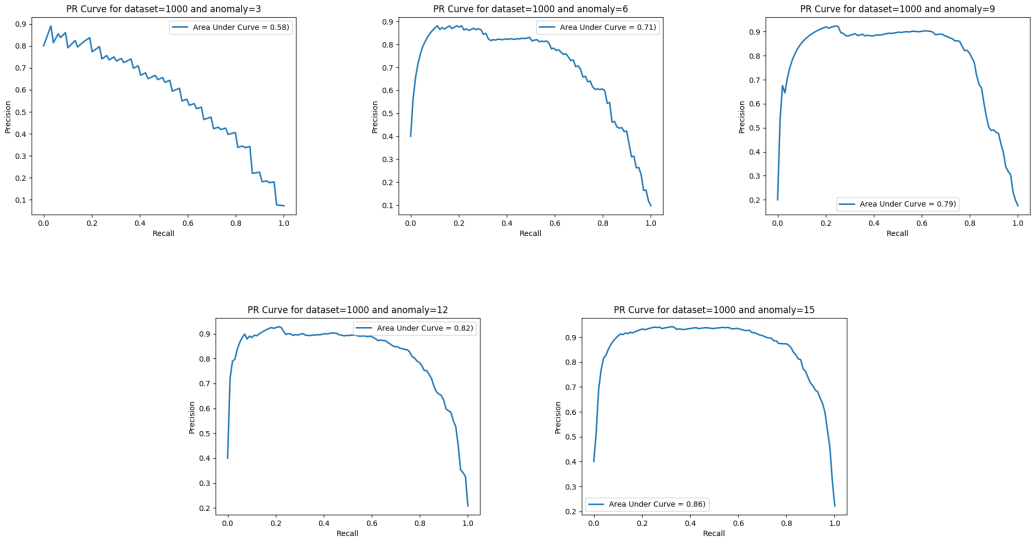

Figure 5.12: Precision-Recall curves for MNIST dataset for N=1000

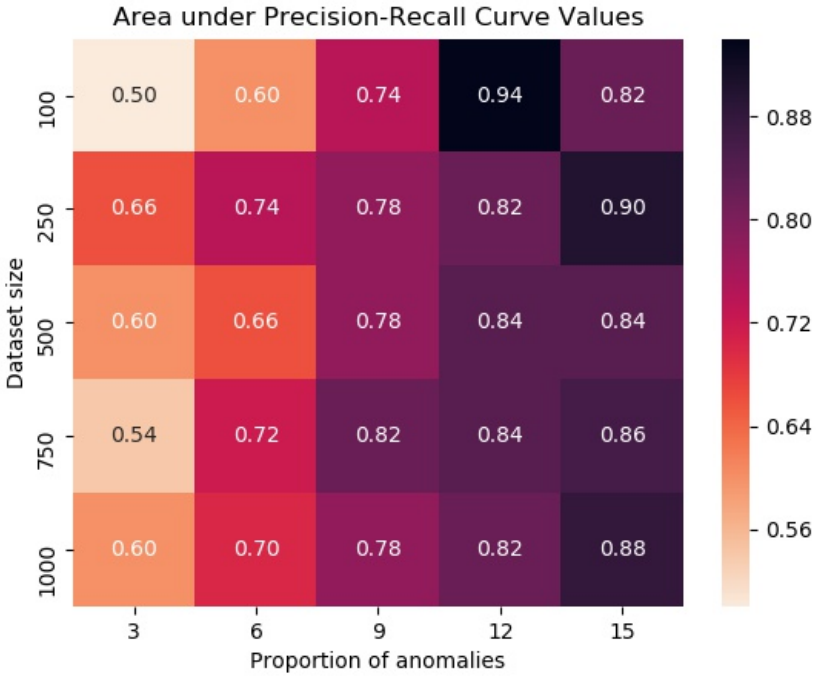

Figure 5.13: Area under Precision-Recall curve for MNIST dataset with different dataset size and proportion of anomalies

Chapter 6

Conclusion and Discussion

In this thesis, we focus on the anomalous pattern detection technique. We first explain various point anomaly detection techniques along with its limitations. We then mention some of the recent collective anomaly detection techniques based on clustering and deep learning that overcome the limitations of point anomaly detection techniques. However, they fail in detecting anomalous patterns in the data as they focus only on a group of records that collectively form an anomaly. APD, AGD, and FGSS are the three major anomalous pattern detection techniques that provide a subset of records and attributes which together do not conform to the expected behavior. We then propose a deep learning based framework, called DeepFGSS, which extends the capabilities of FGSS to work with continuous, unstructured and high dimensional data along with more advantages. We evaluated it on both structured and unstructured datasets. We show that DeepFGSS is able to distinguish normal datasets from the anomalous datasets at smaller proportions of anomalies using power curves and ROC curves. We then demonstrated that DeepFGSS is able to detect anomalous patterns with high accuracy using precision-recall curves and Jaccard curves.

Currently, we have only evaluated the accuracy on the detection of anomalous records. New approaches/metrics need to formed for evaluating the predicted set of columns. Also, we have used the Berk Jones scan statistic to score subsets. We can use various other scan statistics such as the Higher Criticism [25] etc and see if they enhance the detection capability of DeepFGSS. We have used Autoencoder to compute reconstruction errors which are then converted into empirical p-value ranges. We can

use other deep learning networks such as Variational Autoencoders [40], Bi-directional Generative Adversarial Networks [24] which learns the mapping of input data onto some normal distribution. The values from the normal distribution corresponding to input records can then be used to convert into empirical p-value ranges. Also, DeepFGSS is currently being evaluated over datasets containing anomalies of a single type. Because it has the capability of enforcing similarity constraints, it would be interesting to observe its performance to detect subsets corresponding to single anomaly type when multiple types of anomalies are present. Finally, we can also evaluate the efficacy of DeepFGSS in a multiple model setting where M_0 will learn the distribution of data under the null hypothesis that no anomalies are present and multiple M_is each learning the distribution of i^{th} anomaly type. The idea will be to detect anomalies which have not been detected before by any of M_is.

References

[1] Mohiuddin Ahmed. "Collective anomaly detection techniques for network traffic analysis". In: *Annals of Data Science* (2018), pp. 1–16.

[2] Mohiuddin Ahmed and Abdun Naser Mahmood. "Network traffic pattern analysis using improved information theoretic co-clustering based collective anomaly detection". In: *International conference on security and privacy in communication networks*. Springer. 2014, pp. 204–219.

[3] Mohiuddin Ahmed and Abdun Naser Mahmood. "Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection". In: *Annals of Data Science* 2.1 (2015), pp. 111–130.

[4] Dana H Ballard. "Modular Learning in Neural Networks." In: *AAAI*. 1987, pp. 279–284.

[5] Irad Ben-Gal. "Bayesian networks". In: *Encyclopedia of statistics in quality and reliability* 1 (2008).

[6] Robert H Berk and Douglas H Jones. "Goodness-of-fit test statistics that dominate the Kolmogorov statistics". In: *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 47.1 (1979), pp. 47–59.

[7] Loïc Bontemps, James McDermott, Nhien-An Le-Khac, et al. "Collective anomaly detection based on long short-term memory recurrent neural networks". In: *International Conference on Future Data and Security Engineering*. Springer. 2016, pp. 141–152.

[8] Suratna Budalakoti et al. "Anomaly detection in large sets of high-dimensional symbol sequences". In: (2006).

[9] T Caudell and D Newman. "An adaptive resonance architecture to define normality and detect novelties in time series and databases". In: *IEEE World Congress on Neural Networks, Portland, Oregon*. 1993, pp. 166–176.

[10] Philip K Chan and Matthew V Mahoney. "Modeling multiple time series for anomaly detection". In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE. 2005, 8–pp.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[12] David Maxwell Chickering. "Learning Bayesian networks is NP-complete". In: *Learning from data*. Springer, 1996, pp. 121–130.

[13] David Maxwell Chickering, David Heckerman, and Christopher Meek. "Large-sample learning of Bayesian networks is NP-hard". In: *Journal of Machine Learning Research* 5.Oct (2004), pp. 1287–1330.

[14] Richard G Clegg. "A practical guide to measuring the Hurst parameter". In: *arXiv preprint math/0610756* (2006).

[15] Barry R. Cobb, Rafael Rumí, and Antonio Salmerón. "Bayesian Network Models with Discrete and Continuous Variables". In: 2007.

[16] HerskovitsE CooperG. "A Bayesian method for the induction of probabilistic networks from data". In: *Machine Learning* 9.4 (1992), pp. 309–347.

[17] KDD Cup. *Available on: http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html*. 2007.

[18] Rónán Daly, Qiang Shen, and Stuart Aitken. "Learning Bayesian networks: approaches and issues". In: *The knowledge engineering review* 26.2 (2011), pp. 99–157.

[19] Kaustav Das. *Detecting patterns of anomalies*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA MACHINE LEARNING DEPT, 2009.

[20] Kaustav Das and Jeff Schneider. "Detecting anomalous records in categorical datasets". In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2007, pp. 220–229.

[21] Kaustav Das, Jeff Schneider, and Daniel B Neill. "Anomaly pattern detection in categorical datasets". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2008, pp. 169–176.

[22] Atabak Dehban et al. "Denoising auto-encoders for learning of objects and tools affordances in continuous space". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 4866–4871.

[23] Marco Di Zio et al. "Bayesian networks for imputation". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 167.2 (2004), pp. 309–322.

[24] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial feature learning". In: *arXiv preprint arXiv:1605.09782* (2016).

[25] David Donoho, Jiashun Jin, et al. "Higher criticism for detecting sparse heterogeneous mixtures". In: *The Annals of Statistics* 32.3 (2004), pp. 962–994.

[26] Frederick Eberhardt, Clark Glymour, and Richard Scheines. "On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables". In: *arXiv preprint arXiv:1207.1389* (2012).

[27] Levent Ertöz, Michael Steinbach, and Vipin Kumar. "Finding topics in collections of documents: A shared nearest neighbor approach". In: *Clustering and Information Retrieval*. Springer, 2004, pp. 83–103.

[28] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[29] Nir Friedman, Iftach Nachman, and Dana Peér. "Learning bayesian network structure from massive datasets: the sparse candidate algorithm". In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1999, pp. 206–215.

[30] Nir Friedman, Moises Goldszmidt, et al. "Discretizing continuous attributes while learning Bayesian networks". In: *ICML*. 1996, pp. 157–165.

[31] Jonas Gehring et al. "Extracting deep bottleneck features using stacked auto-encoders". In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 3377–3381.

[32] Markus Goldstein and Andreas Dengel. "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm". In: (2012).

[33] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[34] Yuhong Guo and Dale Schuurmans. "Convex structure learning for Bayesian networks: Polynomial feature selection and approximate ordering". In: *arXiv preprint arXiv:1206.6832* (2012).

[35] Zengyou He, Xiaofei Xu, and Shengchun Deng. "Discovering cluster-based local outliers". In: *Pattern Recognition Letters* 24.9-10 (2003), pp. 1641–1650.

[36] Yu-Chi Ho, R_S Sreenivas, and P Vakili. "Ordinal optimization of DEDS". In: *Discrete event dynamic systems* 2.1 (1992), pp. 61–88.

[37] Reimar Hofmann and Volker Tresp. "Discovering structure in continuous variables using Bayesian networks". In: *Advances in neural information processing systems*. 1996, pp. 500–506.

[38] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[39] Yoshua Bengio Ian Goodfellow and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[40] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[41] Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[42] Wai Lam and Fahiem Bacchus. "Using causal information and local measures to learn Bayesian networks". In: *Proceedings of the Ninth international conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1993, pp. 243–250.

[43] Jau-Huei Lin and Peter J Haug. "Exploiting missing clinical data in Bayesian network modeling for predicting medical problems". In: *Journal of biomedical informatics* 41.1 (2008), pp. 1–14.

[44] Dimitris Margaritis. *Learning Bayesian network model structure from data*. Tech. rep. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2003.

[45] Edward McFowland, Skyler Speakman, and Daniel B Neill. "Fast generalized subset scan for anomalous pattern detection". In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 1533–1561.

[46] Yisroel Mirsky et al. "Kitsune: an ensemble of autoencoders for online network intrusion detection". In: *arXiv preprint arXiv:1802.09089* (2018).

[47] Stefano Monti and Gregory F Cooper. "A multivariate discretization method for learning Bayesian networks from mixed data". In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1998, pp. 404–413.

[48] Daniel B Neill. "Fast and flexible outbreak detection by linear-time subset scanning". In: *Advances in Disease Surveillance* 5 (2008), p. 48.

[49] Dan Pelleg, Andrew W Moore, et al. "X-means: extending k-means with efficient estimation of the number of clusters." In: *Icml*. Vol. 1. 2000, pp. 727–734.

[50] Maleeha Qazi et al. "Automated Heart Wall Motion Abnormality Detection from Ultrasound Images Using Bayesian Networks." In: *IJCAI*. Vol. 7. 2007, pp. 519–525.

[51] Quratulain N Rajput and Sajjad Haider. "Use of Bayesian network in information extraction from unstructured data sources". In: *International Journal of Information Technology* 5.4 (2009), pp. 207–213.

[52] Carsten Riggelsen and Ad Feelders. "Learning Bayesian Network Models from Incomplete Data using Importance Sampling." In: *AISTATS*. Citeseer. 2005.

[53] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[54] Manon J Sanscartier and Eric Neufeld. "Identifying Hidden Variables from Context-Specific Independencies." In: *FLAIRS Conference*. 2007, pp. 472–478.

[55] Adamo L de Santana et al. “Strategies for improving the modeling and interpretability of Bayesian networks”. In: *Data & Knowledge Engineering* 63.1 (2007), pp. 91–107.

[56] Mauro Scanagatta et al. “Learning Bayesian networks with thousands of variables”. In: *Advances in neural information processing systems*. 2015, pp. 1864–1872.

[57] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.

[58] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. “Wavecluster: A multi-resolution clustering approach for very large spatial databases”. In: *VLDB*. Vol. 98. 1998, pp. 428–439.

[59] Alban Siffer et al. “Anomaly detection in streams with extreme value theory”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017, pp. 1067–1075.

[60] Rajiv Subrahmanyam and Hector Aguilar-Macias. *Extracting information from unstructured data and mapping the information to a structured schema using the naïve bayesian probability model*. US Patent 8,577,829. Nov. 2013.

[61] Huanliang Sun et al. “CD-trees: An efficient index structure for outlier detection”. In: *International Conference on Web-Age Information Management*. Springer. 2004, pp. 600–609.

[62] Cheng Tan et al. “Netbouncer: active device and link failure localization in data center networks”. In: *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association. 2019, pp. 599–613.

[63] Ioannis Tsamardinos et al. “Scaling-up bayesian network learning to thousands of variables using local learning techniques”. In: *Vanderbilt University DSL TR-03-02* (2003).

[64] Laura Uusitalo. “Advantages and challenges of Bayesian networks in environmental modelling”. In: *Ecological modelling* 203.3-4 (2007), pp. 312–318.

[65] Juha Vesanto, Esa Alhoniemi, et al. “Clustering of the self-organizing map”. In: *IEEE Transactions on neural networks* 11.3 (2000), pp. 586–600.

[66] Qiong Wang et al. “Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy”. In: *Appl. Environ. Microbiol.* 73.16 (2007), pp. 5261–5267.

Appendix A

Precision-Recall Plots for KDD Dataset: Apache2

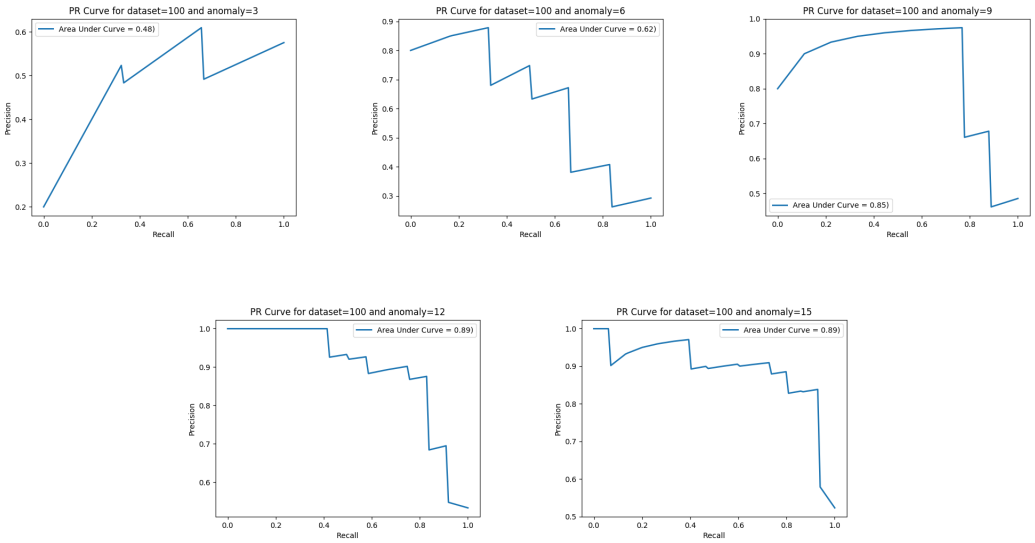

Figure A.1: Precision-Recall curves for KDD dataset with N=100 and Apache2 anomaly

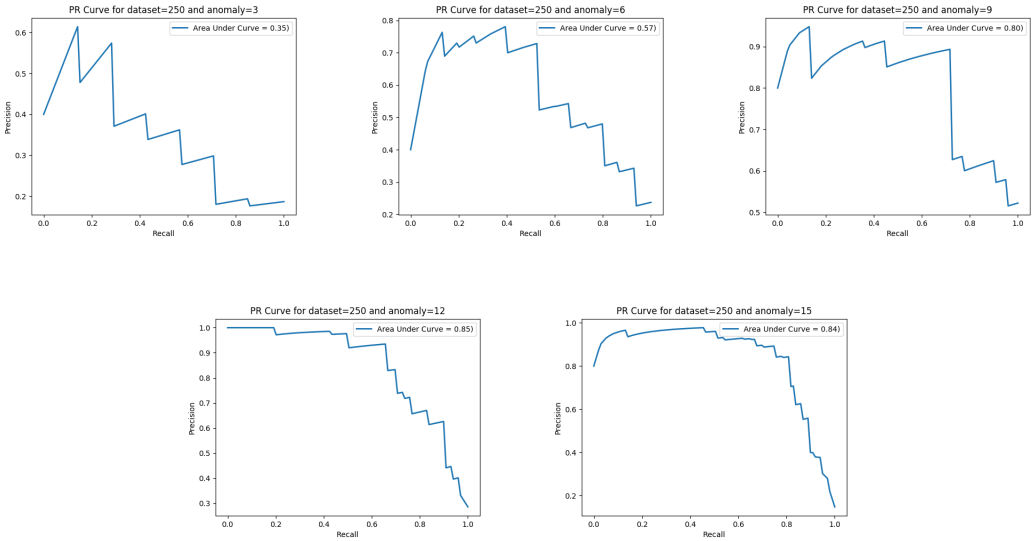

Figure A.2: Precision-Recall curves for KDD dataset with N=250 and Apache2 anomaly

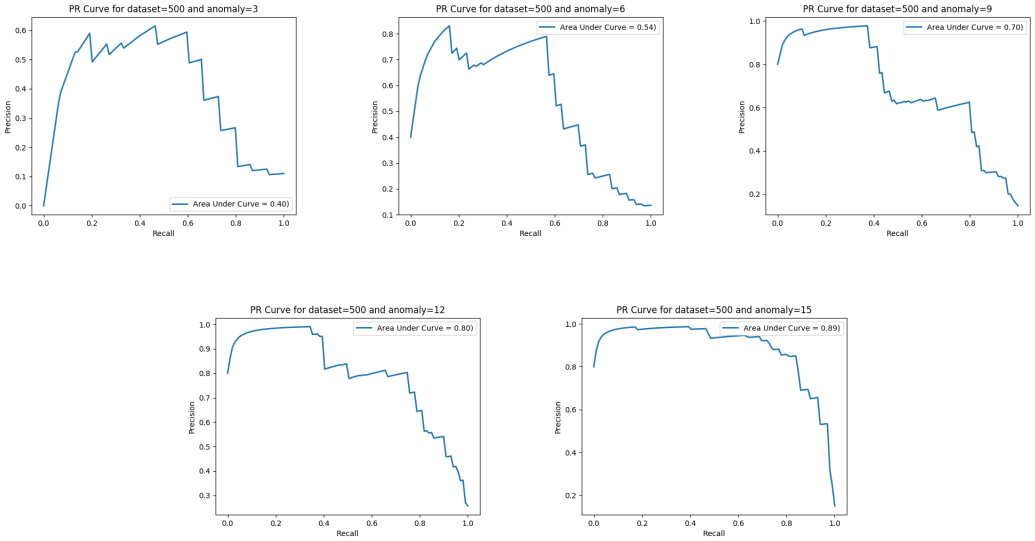

Figure A.3: Precision-Recall curves for KDD dataset with N=500 and Apache2 anomaly

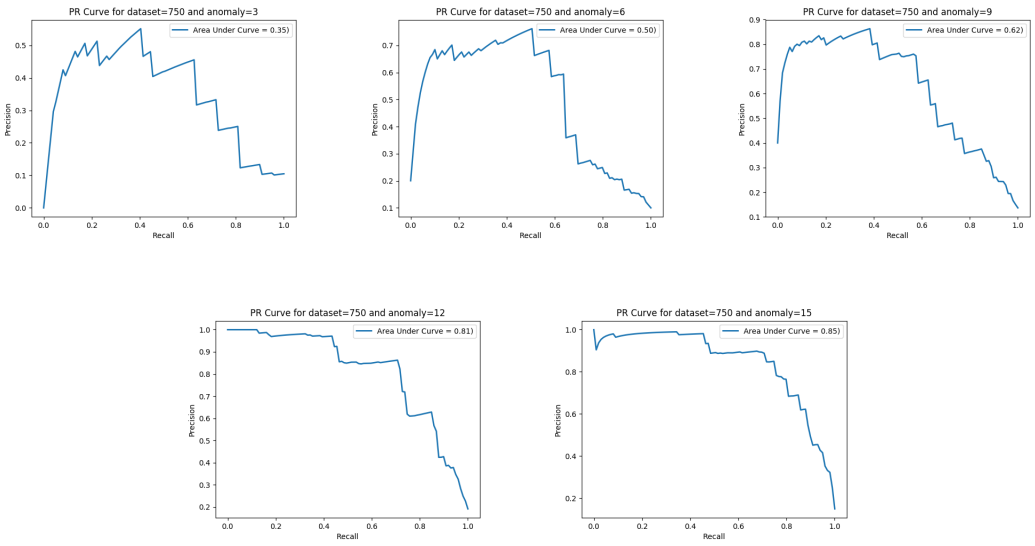

Figure A.4: Precision-Recall curves for KDD dataset with N=750 and Apache2 anomaly

Appendix B

Precision-Recall Plots for KDD Dataset: Neptune

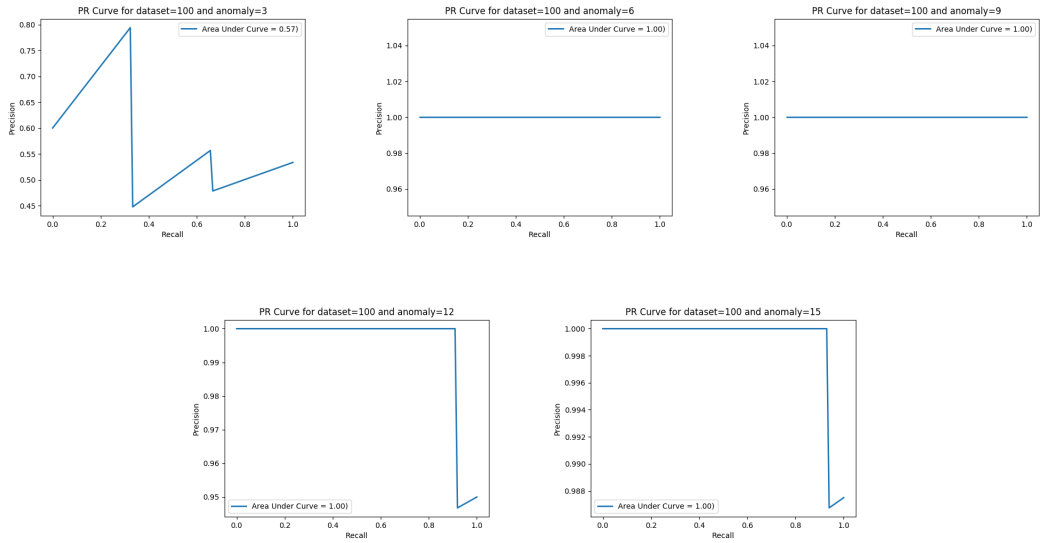

Figure B.1: Precision-Recall curves for KDD dataset with N=100 and Neptune anomaly

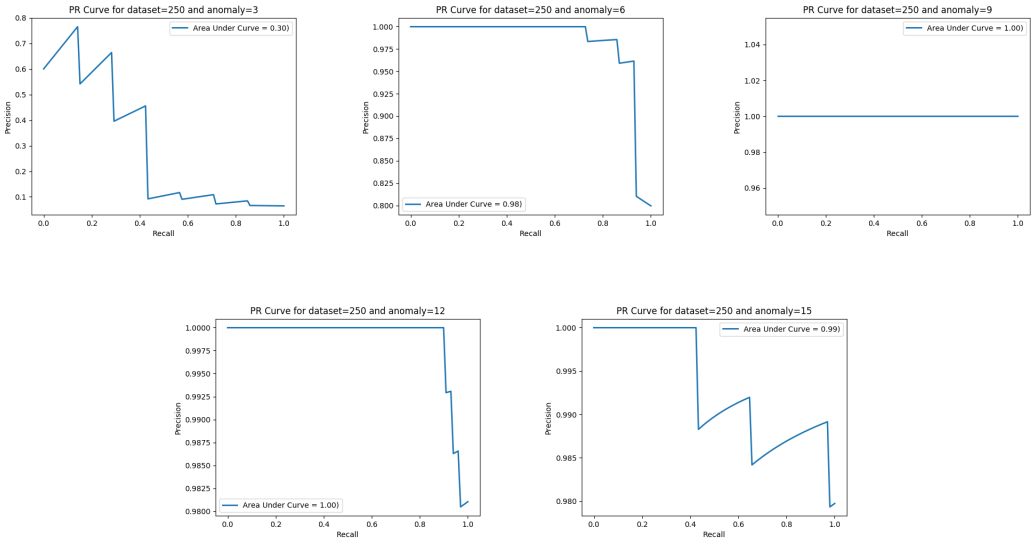

Figure B.2: Precision-Recall curves for KDD dataset with N=250 and Neptune anomaly

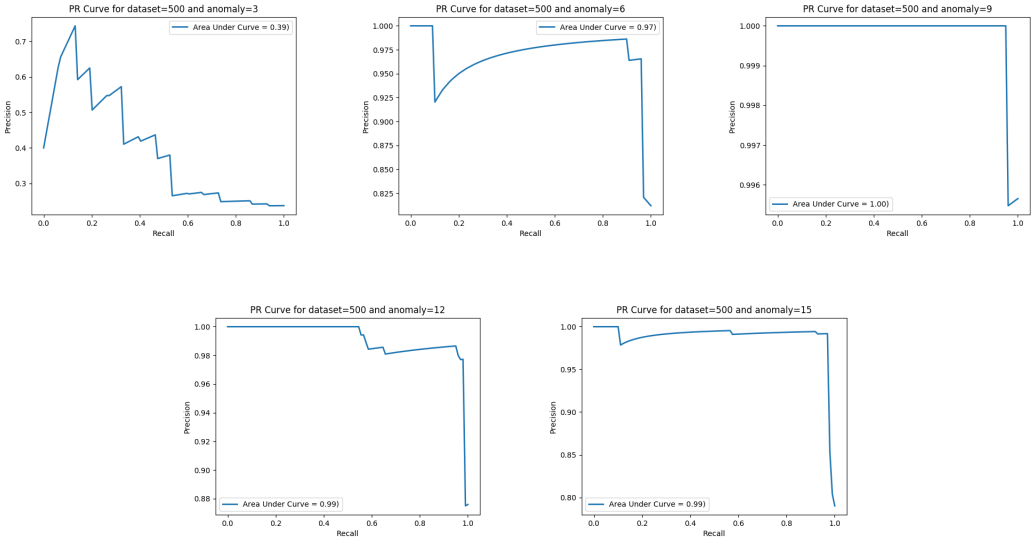

Figure B.3: Precision-Recall curves for KDD dataset with N=500 and Neptune anomaly

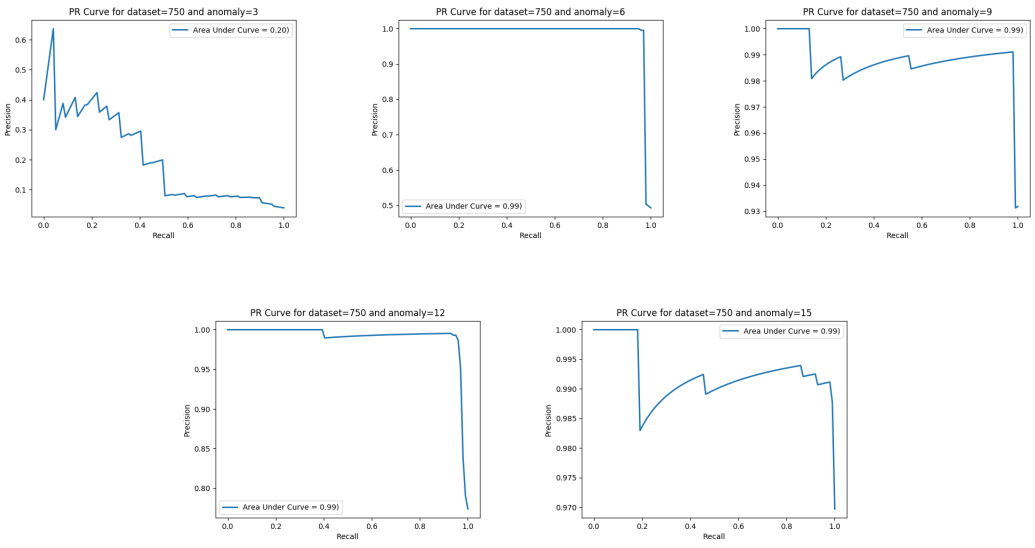

Figure B.4: Precision-Recall curves for KDD dataset with N=750 and Neptune anomaly

Appendix C

Precision-Recall Plots for MNIST Dataset

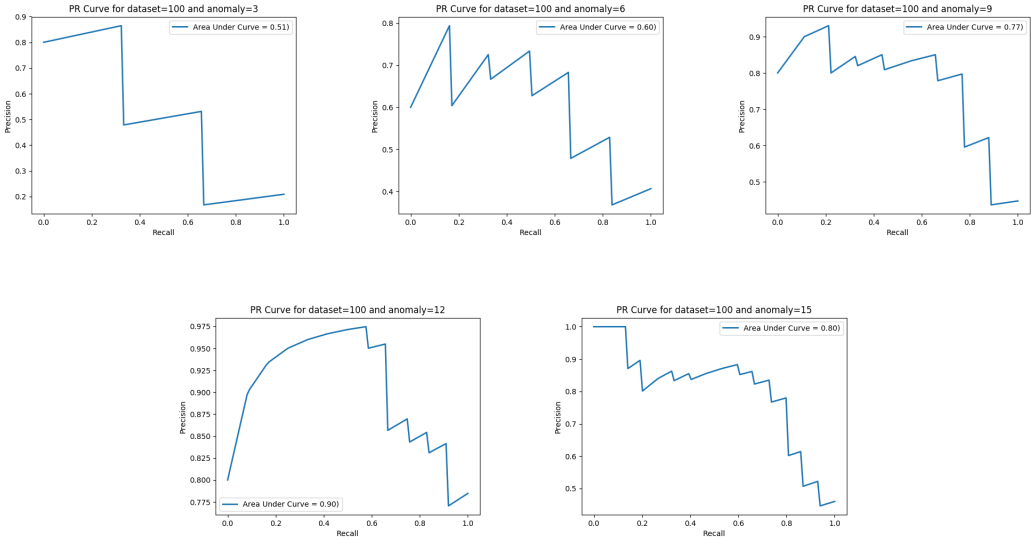

Figure C.1: Precision-Recall curves for MNIST dataset for N=100

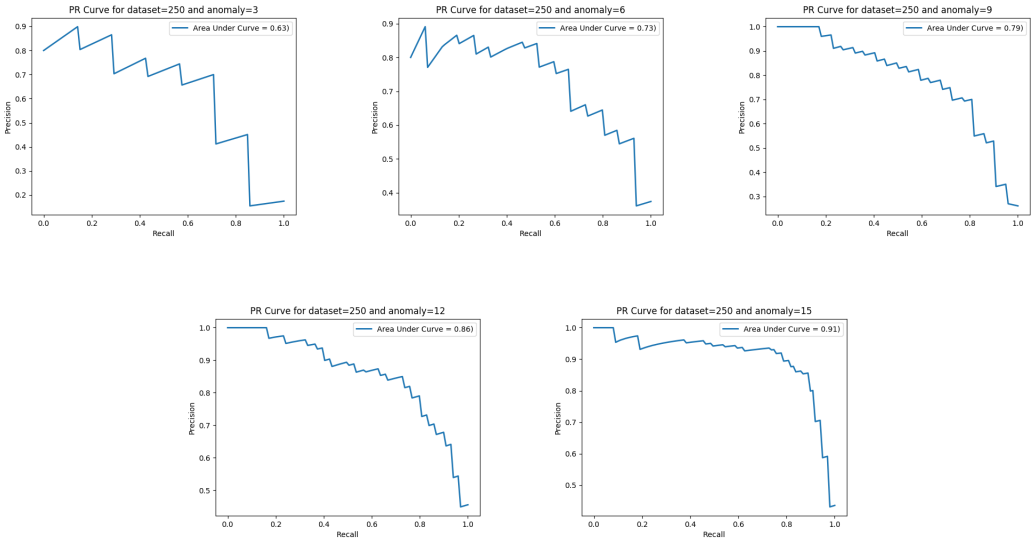

Figure C.2: Precision-Recall curves for MNIST dataset for N=250

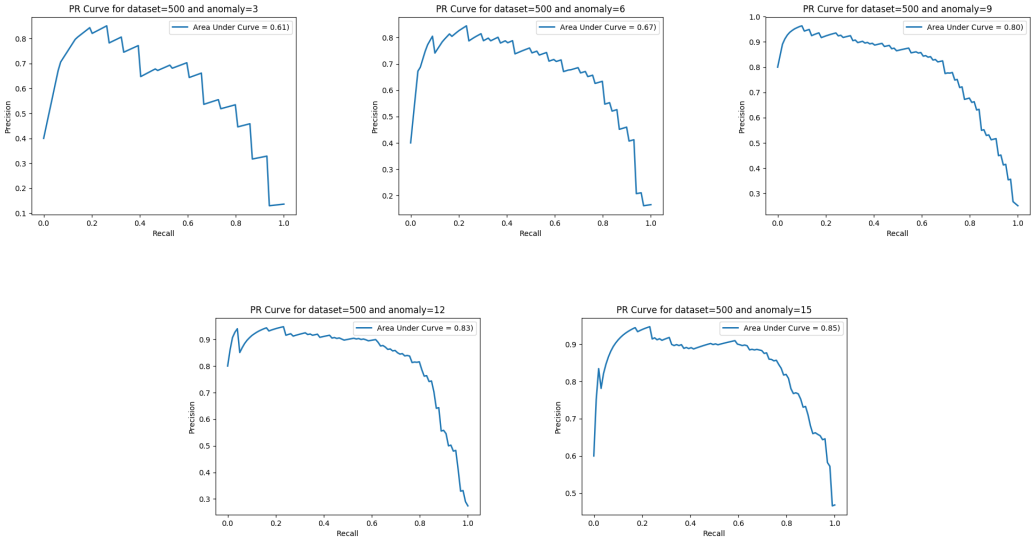

Figure C.3: Precision-Recall curves for MNIST dataset for N=500

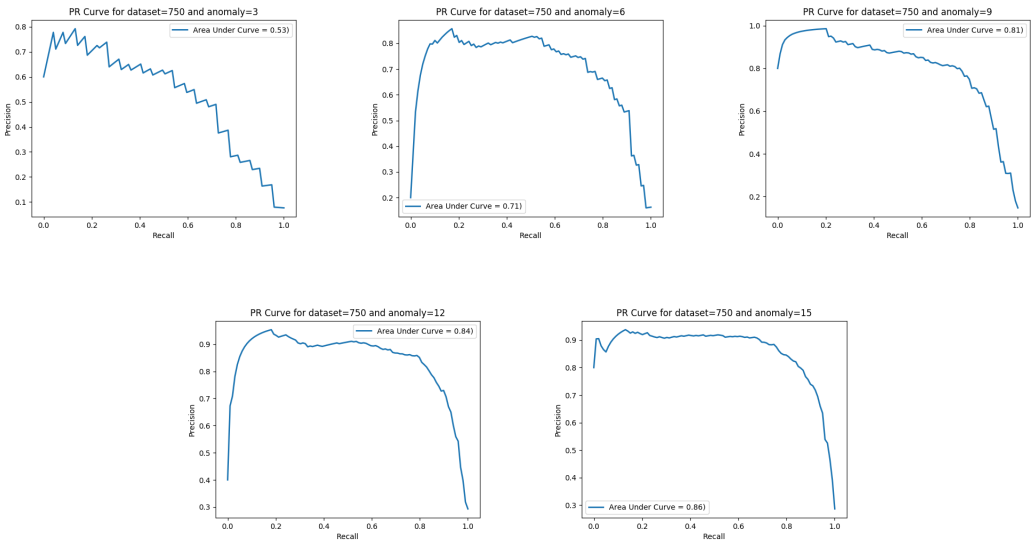

Figure C.4: Precision-Recall curves for MNIST dataset for N=750