# DETECTING COVID-19 OUTBREAK WITH ANOMALOUS TERM FREQUENCY

An Undergraduate Research Scholars Thesis

by

YILE CHEN

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Faculty Research Advisor: Dr. Xia Hu

May 2022

Major: Computer Engineering

# RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

# TABLE OF CONTENTS

# ABSTRACT

Detecting COVID-19 Outbreak with Anomalous Term Frequency

Yile Chen
Department of Computer Science and Engineering
Texas A&M University

Research Faculty Advisor: Dr. Xia Hu
Department of Computer Science and Engineering
Texas A&M University

Previously many studies have aimed at predicting the trend of a disease through time series forecasting using machine learning methods. However, data extracted from the real world is often noisy, which can pose numerous challenges for directly predicting the trend, and therefore leading to suboptimal prediction results. Furthermore, real-world data is usually very large, that is, having very long time periods. When it comes to data of such scale, trend forecasting becomes intractable even to state-of-the-art forecasting algorithms such as RNN-LSTM. In the past, not much research has been conducted in applying anomaly detection for disease outbreak detection, including the most recent COVID-19 pandemic. Consequently, in this research, we propose redefining the problem into outbreak detection, which aims to predict whether a future point is or is not a sign of a large scaled COVID-19 outbreak. Through simplifying a complex regression problem into a binary classification problem, the requirements of the learning model may be decreased and therefore the learning performance may be enhanced.

# ACKNOWLEDGMENTS

**Contributors**

I would like to personally thank my undergraduate research advisor, Dr. Xia Hu, for accepting me to his research team and giving me freedom to pursue this project while providing constant and valuable feedback.

Thanks to my research team for contributing to the package used for this research project, and primarily to my PhD mentors Kwei-Herng Lai and Daochen Zha for giving me a role in their excellent research, and letting me experiment with my ideas as well as helping me review my thesis. Also, thanks to Guanchu Wang for always being available and constantly providing meaningful guidance and valuable answers throughout the whole project.

I would like to thank my friends and colleagues, along with the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, to my father and mother, for faithfully supporting my education, and always believing in me and pushing me to improve. Thank you for being my inspiration and supporter.

**Funding Sources**

Undergraduate research was supported by the Undergraduate Research Scholars Program and the Engineering Honors Program at Texas A&M University.

This work was not supported by any funding.

# NOMENCLATURE

| | |
|---|---|
| AD | Anomaly Detection |
| OD | Outlier Detection |
| ML | Machine Learning |
| NN | Neural Network |
| SLP | Single-layer Perception |
| MLP | Multi-layer Perception |
| CNN | Convolution Neural Network |
| RNN | Recurrent Neural Network |
| TODS | Time Series Outlier Detection System |
| ARMA | Autoregressive Moving Average |
| ARIMA | Autoregressive Integrated Moving Average |
| RD | Reachability Distance |
| LRD | Local Reachability Density |
| iForest | Isolation Forest |
| KNN | $k$-Nearest Neighbor |
| MP | Matrix Profile |
| SVM | Support Vector Machine |
| API | Application Programming Interface |

# 1. INTRODUCTION

## 1.1 Motivation

Since the unexpected outbreak of COVID-19, the globe has been swept with rapidly growing infected cases, which has lead to medical supplies running short, healthcare systems getting overwhelmed, and fears of an economic downfall. It has become a common concern for mankind as the confirmed cases rise with astonishing speed, and on March 11, 2020 the World Health Organization (WHO) officially declared that the COVID-19 outbreak is a pandemic. According to reports from the WHO, there have been over 28 million people infected leading to over 900,000 deaths as of September 10, 2020. In the United States alone there have been more than 6.5 million confirmed cases at the time of this writing, higher than anywhere else in the world. Despite the inaccuracies associated with medical predictions regarding disease outbreaks, and the commonly underestimated uncertainties that exists [1], there are numerous benefits we gain from analyzing the data and forecasting its trend. In the event of this global health crisis, the ability to accurately forecast the future spread of the disease as well as analyzing the death and recovery rates will help us better understand the current situation, discover insights on the future development of the disease, and thus allowing us to make better preparations.

## 1.2 Pandemic Forecasting

When it comes to modeling and forecasting epidemics in the past, machine learning approaches have been widely adopted due to their capabilities to process significant amounts of relevant data and handle the underlying complexity of different epidemiological situations [2]. Models developed using machine learning aim to generate results for longer lead-times that have greater reliability in predictions and higher generalization abilities [3, 4, 5, 6, 7, 8]. Due to the complexity and non-linearity nature of the COVID-19 pandemic, along with its differences from other outbreaks in the past, the ability to produce adequate results using traditional models is to be determined [9, 10]. Moreover, the availability and reliability of data accessible is essential for the accuracy of traditional forecasting methods [11]. Furthermore, model uncertainty is significantly increased due to numerous variables that affect the disease's spread, varying containment methods and complex population-wide behavior in different geopolitical regions [12]. Despite the plethora of academic papers that investigate the forecasting of COVID-19, most approaches available in public repositories combine the SIR epidemiological model [8] or its extension SEIR with traditional machine learning regression [13, 14, 15]. In terms of methods, neural networks, along with forests and regression tree remain popular approaches [16, 17]. Others are limited to the basic methods of Bayesian Networks, classification and regression tree (CART), genetic programming, and Naïve Bayes [8].

## 1.3 Machine Learning and Neural Networks

Machine learning (ML) is a type of algorithm that allows software programs to predict outcomes more accurately without being explicitly programmed. The basis of ML is to develop algorithms that receive input data and apply math or statistical analysis to predict some output, while constantly updating its results as new data gets passed in. ML has many real-world applications and is widely used for outcome prediction, Natural Language Processing (NLP), image recognition and classification, as well as in many other aspects of science. Neural Networks (NN) are an essential part of ML algorithms that incorporate the concept of simulating the biological neural networks in the human brain to process information. One of the most renowned definitions of neural network states that a neural network is a computing system constructed of numerous simple and highly interconnected processing units, which process information through their dynamic state response from external inputs. The neuron is the basic computational unit in a NN, which is also referred to as a unit or node. Each node receives information either from external sources or from other nodes and computes an output. Neural networks consist of multiple cascading layers of these nodes, which are connected to others in previous and following layers. Each neuron takes in the output data from previous layers and applies a weighted function (usually a sum) to process the input data. Then it uses a non-linear function (usually referred to as an activation function) to scale the computation results and sends this modified version of the data to its succeeding layers.

The weights are constantly adjusted to improve its output and will eventually be able to produce correct results and proceed to reinforce its learning.

There are many classes of neural networks, but the two most common types are Feedforward Neural Networks (FNN) and Recurrent Neural Networks (RNN). In a FNN, connections between nodes do not form a cycle. The information in this network only flows from the input nodes through the hidden nodes and to the output node in a forward direction. RNN consists of connections that form a directed cycle, meaning data can propagate forward but also backward from later to earlier processing stages. In contrast with FNNs, RNNs can utilize their internal memory to process inputs of arbitrary sequence. This allows them to fulfill tasks such as speech recognition, unsegmented and connected handwriting recognition, along with other general sequence processes. There are generally three types of FNNs: single-layer perception (SLP), multi-layer perception (MLP), and convolution neural networks (CNN). SLP is the simplest FNN as it only contains a single output layer with no hidden layers. MLP consists of multiple layers of nodes that are interconnected in a feed-forward manner. CNNs are also like normal NNs but the node connection pattern is inspired by the visual cortex. Nodes respond to stimuli in a restricted space called the receptive field, and these fields overlap partially to cover the entire visual field. CNN is widely applied in image classification and recognition as well as recommender systems.

In the past, CNNs and RNNs have been widely adopted in machine learning problems, but neither of them are perfect, as they present certain issues. CNN has yet to improve due to low generalization ability, poor results in crowded-scenes, and lacking equivariance [18]. Moreover, due to the supervised learning mechanisms of CNN, large and annotated data must be available for proper learning, otherwise leading to underfitting. Furthermore, the performance of CNN is highly influenced by hyperparameter selection. Therefore, a crucial design issue that needs to be addressed is the careful selection of hyperparameters to achieve optimized results [19]. RNN also faces some drawbacks, which include a fixed number of hidden units, long training times due to large parallelism and sequential nature, and limited use of distant context [20, 21]

## 1.4 Anomaly Detection

Anomaly detection (AD), which is also known as outlier detection, refers to the problem of analyzing data and discovering patterns that do not conform to the expected behavior. These nonconforming patterns are generally referred to as anomalies, outliers, discordant observations, exceptions, or contaminants in different domains of application [22]. An anomaly is also considered as an outlier, as one of the most globally accepted outlier definitions [23] states that an outlier is a data object that deviates significantly from the rest of the objects, as if it was generated by a different mechanism. Anomalies and outliers are the two most frequently used terms in the field of anomaly detection and are sometimes interchangeable. Anomaly detection techniques have been proven to be efficient and widely adopted for applications such as credit card fraud detection, medical anomaly detection, industrial damage detection and network intrusions [24]. Different from time series forecasting, anomaly detection may not focus on predicting the future trend and values, or deal with minor noises that lead to variation in values, but rather aims to predict whether the upcoming value is drastically different from the regular distribution. Therefore, the learning ability of an anomaly detection model will be lower and easier to achieve reasonable results.

### *1.4.1 TODS: Time Series Outlier Detection System*

While ML models have proven promising in time series outlier detection tasks, human expertise is critical to build an effective pipeline for time series outlier detection. Since varying application scenarios result in unique characteristics in data, extensive trials are required to identify and implement the best fit processing modules and hyperparameters for a specific task, which dramatically hinders the application of time series outlier detection in the real world. To reduce human efforts of manually building pipeline for time series outlier detection, we developed TODS: Time Series Outlier Detection System [25], which provides an end-to-end solution for real-world time series outlier detection applications.

TODS is a highly modular open-source system for automated time series outlier detection that can allow end-users to conveniently deploy an outlier detection pipeline for a specific application scenario. The fundamental building block in TODS is a primitive, which is an implementation of a function along with hyperparameters. A pipeline is constructed of multiple primitives and is defined as a Directed Acyclic Graph (DAG). Pipelines typically consists of four primitives, which include data preprocessing, time series preprocessing, feature analysis, and detection algorithms. Based on different application scenarios, different pipelines can be constructed from these primitives. Figure 1.1 shows the overall structure of the pipeline.

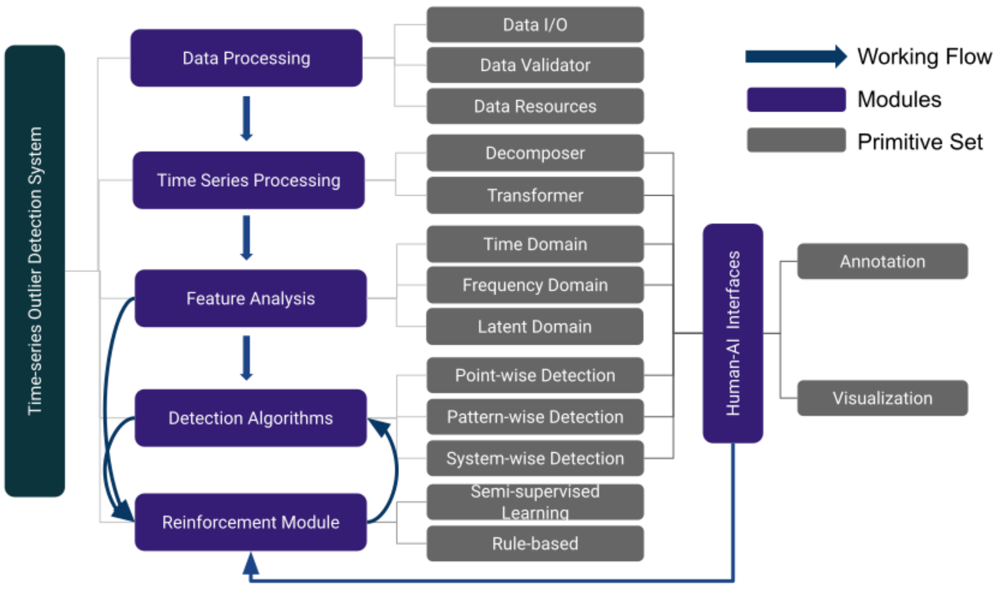


Figure 1.1: Overview of TODS Pipeline

Graphical User Interface (GUI) is provided to enable simple pipeline construction, where users can create and deploy pipeline through drag-and-drop actions. Moreover, a data-driven searcher is built in and can be utilized to explore suitable pipelines for a specified task automatically. In this research, TODS is fully utilized to construct pipelines that perform outlier detection on COVID-19 time series data.

## 1.5 Research Objectives

The methods introduced in section 1.2 are targeted towards time series forecasting, which aim to predict exact values such as infections and deaths. However, they require extensive research on modeling the trend, seasonality, as well as dealing with many nuances such as significant noise in the data. Here we provide a different prospective of understanding the data of COVID-19, by viewing the outbreaks (initial and consequent outbreaks) as anomalies and examining whether there will be more waves of outbreaks in the future using anomaly detection. In the past, there have been few applications of anomaly detection in disease outbreaks, such as early disease outbreak detection systems that monitors health care data for irregularities [26, 27] or enhancing surveillance systems to detect outbreaks rapidly [28]. However, past research focused on detecting an outbreak at its early stage, yet disease outbreaks in the past have exhibited characteristics such as multiple waves of outbreaks [29]. This characteristic has also been observed in the COVID-19 pandemic. The lack of research in detecting continuous waves of outbreaks that happen during a pandemic like COVID-19 leaves a gap in the application of anomaly detection, and its possibilities are yet to be explored. To overcome the drawbacks in existing models and bridge the gap between anomaly detection and modeling the outbreaks of COVID-19, we combine the flexibility of constructing anomaly detection pipelines using TODS with a data-driven machine learning approach. The objectives of the current study are as follows:

1. Explore the possibility of using anomaly detection to predict COVID-19 outbreaks by designing and deploying anomaly detection pipelines on multivariate time series data.

2. Using an automated searcher, generate and test all possible pipeline configurations in the defined search space to find the most suitable pipeline for COVID-19 outbreak detection.

3. Demonstrate that using the TODS anomaly detection package, it is possible to provide end-to-end solutions for building and deploying anomaly detection pipelines to address real-world problems.

# 2. ANOMALY DETECTION ALGORITHMS

Anomaly detection is a difficult task, especially since detecting anomalies can be challenging when they overlap with nominal clusters, yet these clusters need to be dense enough for a reliable model to be built. The contamination issue, referring to using an input dataset that is contaminated with anomalies, could make this task more difficult as the final model may be degraded by anomalies if the training algorithm lacks robustness [30]. Therefore, it is essential to recognize and understand distinctive techniques that are currently applied in anomaly detection. Many categories have been created in an attempt to classify anomaly detection algorithms by their characteristics, some of them include classification-based, clustering-based, nearest-neighbor-based, statistical-based, information theoretic-based, and spectral-based [31]. Yet there is still continuous debate on how to categorize these algorithms, so here a more general category set is discussed. It is commonly observed that anomaly detection methods can be derived from these three fields of computing: regression models, classical machine learning models, and deep learning models. In this section, several methods are described and analyzed from all three fields.

## 2.1 Regression Models

Anomaly detection using regression models has been extensively applied to time series data forecasting. The basic regression-based anomaly detection technique contains two steps. In the first step, the data is fitted with a regression model. In the second step, a residual for each test instance is used to determine the anomaly score at that instance. One representative of the regression method is called Autoregression (AR), which is a linear model that relies on past period values to predict current ones. Current period values are the sum of past outputs multiplied by some numeric factor. Autoregression is denoted as $AR(p)$, where $p$ is the order of the model and represents the number of lagged values that are included.

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + c + \epsilon_t$$

The coefficient $a_1$ is the numeric constant by which the lagged variable $X_{t-1}$ is multiplied by, and it can be interpreted as a part of the previous value that will remain in the future. These coefficients should always be between -1 and 1. $\epsilon_t$ is the residual and represents the difference between the prediction for time t and the correct value. Residuals are generally unpredictable values since an existing pattern would be captured by the other incumbents of the model. Anomalies that are present in the training data can influence the regression parameters and thus causing inaccurate results from the regression model. A robust technique to handle such anomalies while fitting regression models is applied in Autoregressive Integrated Moving Average (ARIMA) [32, 33]. The basic regression models have variants in which the technique can be used to detect anomalies in multivariate time-series data, such as one generated from an Autoregressive Moving Average (ARMA) model [34]. The technique applied was to transform the multivariate time series data to a univariate time series by combining the components of the multivariate time series linearly.

## 2.2 Classical Machine Learning Models

### *2.2.1 Local Outlier Factor: LOF*

Local outlier factor (LOF) is a well-known distance-based approach that studies the neighborhood of each data instance to identify outliers as described in [35]. For any given data instance, the LOF score is equal to the ratio of the average local density of the k-nearest neighbors of the instance with the local density of the data instance itself. To further understand the local outlier factor of a data instance, we first examine the reachability distance (RD) and local reachability density (LRD). Let $k$ be a natural number, the $reachability\ distance$ of object $p$ with respect to the object of $o$ is defined as

$$RD_k(p, o) = max(k\text{-}distance(o), distance(p, o))$$

It represents the maximum of the $k\text{-}distance$ of $p$ and the distance between $p$ and $o$. Note that the distance measure is problem-specific (Euclidean, Manhattan, etc.) If a point $o$ exists within the $k$ neighbors of $p$, the reachability distance would be the $k\text{-}distance$ of $p$, which is the distance between $p$ and its $k^{th}$ nearest neighbor. The $local\ reachability\ density\ (LRD)$ of $p$ is defined as

$$LRD_k(p) = \frac{1}{\sum_{o \in N_k(p)} \frac{RD_k(p,o)}{|N_k(p)|}}$$

Intuitively, the local reachability density of an object $p$ is the inverse of the average reachability distance of $p$ from its $k$ nearest neighbors. According to the LRD formula, the larger the average reachability distance (i.e., neighbors are far from the point), the smaller density of points will be present around it. This implies how far a point is from the nearest cluster of points. For an anomalous instance, its local density will be lower than that of its neighbors, while for a normal instance lying in a dense region, its local density will be similar to that of its neighbors [31]. Finally, the $local\ outlier\ factor\ (LOF)$ of $p$ is defined as:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} LRD_k(o)}{|N_k(p)|} \times \frac{1}{LRD_k(p)}$$

Here, the local reachability density of each point is compared with the average and those of its $k$ nearest neighbors. LOF is the ratio of the average LRD of the $k$ neighbors of $p$ to the LRD of $p$. It is easy to see that the higher the local reachability densities of $p$'s $k$ nearest neighbors, and the lower $p$'s local reachability distance, and the higher the LOF value. Intuitively, if the point is an outlier, the LRD of the point would be less than the average LRD of its $k$ nearest neighbors, resulting in a high LOF value. On the other hand, if the point is not an outlier and lies in a dense region, the ratio of the average LRD of its $k$ nearest neighbors is roughly equal to the LRD of that point (since the density of the point and its neighbors' are approximately equal). Then the LOF value will be nearly equal to 1.

### *2.2.2 Isolation Forest (iForest)*

Most common anomaly detection techniques are based on constructing a profile on what is normal data, and anomalies are those instances that do not properly conform to the defined normal profile. However, Isolation Forest is a type of isolation algorithm that identifies anomalies by separating outliers from the rest of the data points, it exists as an unsupervised machine learning algorithm. The concept of Isolation Forest was brought up by Liu [36] and uses random forests to compute an isolation score for each data instance. It takes advantage of two quantitative properties of anomalies: 1. They have attribute values that vary drastically from those of normal instances. 2. They are the minority consisting of only a few instances.

Isolation Forest works on the principle of the decision tree algorithm and recursion. To isolate the outliers, the algorithm recursively generates partitions on the datasets by randomly selecting a feature from the given set of features and then randomly selecting a split value between the minimum and maximum values of the selected feature. Arguably, this random partitioning of features will produce smaller paths in trees for anomalous data points compared to the normal data points in the dataset, since they need fewer random partitions to be isolated. Once the path distance is calculated, it is averaged and normalized to calculate the anomaly score, which is used to determine if a point is an anomaly. In the paper, the author states that this algorithm achieves low linear run time complexity and small memory requirement by exploiting the method of subsampling and demonstrates outlier detection performance significantly better than LOF on real-world datasets.

### *2.2.3 $k$-Nearest Neighbors (KNN)*

The basic nearest neighbor technique for anomaly detection is based on the following definition: the anomaly score of a data instance is defined as its distance to its $k^{th}$ nearest neighbor in a given data set [31]. Detection methods using this technique are simple to implement and require no prior assumptions of the data distribution model. They are suitable for either type 1 or type 2 anomaly detection. However, since they rely on calculating the distances between all data records, they suffer from exponential computational growth, as the computational complexity is directly proportional to both the number of data records $n$ and data dimensionality $m$. Therefore, k-nearest neighbor and relevant methods that have $O(n^2m)$ time complexity are not practical for datasets with high dimensionality unless their time complexity can be improved. Despite the various implementations of the k-nearest neighbor (kNN) algorithm for anomaly detection, they all calculate the nearest neighbors of a data record using some suitable distance calculation formula such as the

Euclidean distance or Mahalanobis distance. Euclidean distance is defined by the equation below, which is simply the vector distance.

$$d(x_i, y_i) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

The Mahalanobis distance is given by the equation below, which is the distance of an observation $\vec{x} = (x_1, x_2, x_3, ..., x_N)^T$ from a set of observations with mean $\vec{\mu} = (\mu_1, \mu_2, \mu_3, ..., \mu_N)^T$ and covariance matrix $C$.

$$d_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T C^{-1} (\vec{x} - \vec{\mu})}$$

The Mahalanobis distance is computationally expensive to calculate for large datasets of high dimension compared to the Euclidean distance since it requires passing through the entire data set in order to identify its attribute correlations.

An optimized KNN algorithm was introduced by Ramaswamy [37] to produce a ranked list of potential outliers. A point $p$ is anomalous if no more than $n - 1$ other points in the dataset have a higher distance to $k^{th}$ neighbor $D^k$, where $k$ is a user-specified value. This method utilizes a partition-based technique, which first finds the clusters of the data instances and computes the lower and upper bounds of its distance from its $k^{th}$ nearest neighbor for every instance in each partition. Then this information is used to identify and prune the partitions that are not possible to contain the top $k$ anomalies. Anomalies that are in the unpruned partitions are then computed in the final phase [31].

#### *2.2.4 Matrix Profile (MP)*

Similarity join is a method to find trends and anomalies within time series data. It is essentially comparing subsequences of the time series against itself by computing the distance between each pair of subsequences. The Matrix Profile (MP) was first introduced in [38], and has advantages such as fast, domain agnostic, requires only a single parameter, and can provide an exact solution. In general, the Matrix Profile is a vector that stores the (z-normalized) Euclidean distance between any subsequence within a time series and its nearest neighbor. The Matrix Profile utilizes a sliding window approach, given a window size $w$, the algorithm:

1. Computes the Euclidean distance of windowed sub-sequences against the whole time series.
2. Ignores trivial matches by setting an exclusion zone.
3. Updates the distance profile with the minimal values.
4. Sets the first nearest-neighbor index.

Through the whole process the distance calculations occur $n - m + 1$ times, where $n$ is the length of the times series and $m$ is the window size. The resulting vector of pairwise Euclidean distances is also known as a distance profile. An exclusion zone is required to prevent trivial matches since the sub-sequences are chosen from the time series itself, which is simply half of the window size before and after the current window index. If the subsequence repeats itself in the data, then there will be at least one perfect match thus the minimum Euclidean distance will be zero (or close to zero due to existing noise). In contrast, if the subsequence contains outlier data causing it to be more unique, there would not be a close match and the Euclidean distance would be large. Therefore, high values in the Matrix Profile indicate anomalous data or uncommon patterns, and low values signify repeatable motifs and could provide valuable insights into the times series. Note that a motif is a repeated pattern and discord is an anomaly. Utilizing the Matrix Profile, it is trivial to find the top-K number of motifs or discords.

## 2.3 Deep Learning Models

### *2.3.1 Autoencoder*

Autoencoders are neural network-based models that utilize unsupervised learning techniques to discover the underlying correlations among data and represent data in a smaller dimension. Specifically, it's a neural network architecture where a bottleneck is imposed in the network that forces an encoded knowledge representation of the original input using an encoder network.

Then a decoder network decodes the encoding to recreate the input. If the input features were relatively independent of one another, it would be difficult to compress and reconstruct this sequence. However, if the data exhibits some structure such as correlations between input features, this structure can be learned and leveraged when the input is passed through the bottleneck of the network. An Autoencoder has the following components:

1. Encoder: Takes in the input and produces a lower-dimensional encoding.
2. Bottleneck (Code or Embedding): The lower-dimensional hidden layer which produces the encoding. The bottleneck layer consist of a fewer number of nodes and the number of nodes in this layer also determines the dimension of the encoding.
3. Decoder: Takes the encoding and recreates the input.

The encoder and decoder are represented as:

$$\phi : \mathcal{X} \rightarrow \mathcal{F}$$

$$\psi : \mathcal{F} \rightarrow \mathcal{X}$$

$$\phi, \psi = \underset{\phi,\psi}{\arg\min} \, |X - (\phi \circ \psi)X|^2$$

The encoder function, denoted by $\phi$, maps the original data $\mathcal{X}$, to a latent space $\mathcal{F}$, which is present at the bottleneck. The decoder function, denoted by $\psi$, maps the latent space $\mathcal{F}$ at the bottleneck to the output. The output is the same as the input function in this case. Therefore, it is trying to recreate the original input after some generalized non-linear compression. The encoding network can be represented by a standard neural network function passed through an activation function $\sigma$, with $z$, $W$, $b$ being the latent dimension, weights and bias respectively.

$$z = \sigma(Wx + b)$$

Similarly, the decoding network can be represented in the same fashion, but using different weights, bias, and activation functions.

$$x^{'} = \sigma^{'}(W^{'}z + b^{'})$$

Then loss function can then be written in terms of these network functions, and we use this loss function to train the neural network through the standard backpropagation procedure.

$$\mathcal{L}(x, x^{'}) = |x - x^{'}|^2 = |x - \sigma^{'}(W^{'}(\sigma(W^{'}x + b^{'})) + b^{'})|^2$$

The autoencoder aims to select the encoder and decoder functions in such a way that it requires minimal information to encode the input such that it be can be regenerated on the other side.

Many distance-based anomaly detection techniques (e.g. KNN) suffer the curse of dimensionality since they compute distances of all the data points in the whole feature space. However, Autoencoders can be applied to anomaly detection to avoid the complex high dimension space, mainly through the method of dimensionality reduction [39]. The Autoencoder will encode data into a subspace and decode back the features while normalizing the data. For normal data, the input will be similar to the output as the network will learn the features well. For anomalies, the input and output will be significantly different, therefore signaling the existence of anomalous data.

# 3. METHODOLOGY

Anomaly detection generally follows a specific set of procedures. After constructing the dataset, the first step in the anomaly detection pipeline is time series processing. Time series processing can be understood as filtering and cleaning the data so that it can be successfully inputted into the algorithm for detection. This step usually includes getting rid of missing data points, ensuring the data is continuous and smoothing the data. The next step is feature extraction, which will extract features that can be used by the detection algorithms. Finally, the last stage is applying the detection algorithms that perform the actual detection on input data.

## 3.1 Time Series Processing

Time series is a sequence of data collected at regular time intervals that are evenly spaced and ordered, which leads to a potential for correlation between variables and different timestamps. Time series data may require preprocessing when being modeled with machine learning algorithms. For example, some algorithms prefer standardized and normalized data. Moreover, various differencing operations can be applied to remove seasonal and trend structures from the time series to simplify the prediction problem. Given a time series dataset, there are four transforms that are popularly applied before fitting them into machine learning models for training and prediction. They are power transform, difference transform, standardization, and normalization. The TODS package utilized in this research has various time series processing primitives that fall in these categories.

## 3.2 Feature Extraction

Time series features can be categorized into three main categories: time domain feature, frequency domain feature, and latent factor feature. Here are some features that are included in the TODS package and utilized in this research: statistical features from statsmodels [40], Spectral Residual Transform, Fast Fourier Transform (FFT), Discrete Cosine Transform, Hodrick–Prescott (HP) Filter, Truncated Singular Value Decomposition (SVD), Temporal Regularized Matrix Factorization (TRMF). Generally, some specific features can improve the overall performance of anomaly detection algorithms on a dataset. To find such features, we perform a brute force method of evaluating the performance of all possible combinations on one feature analysis algorithm and one anomaly detection algorithm within our defined search space.

## 3.3 Outbreak Detection

In this section we aim to demonstrate how effective anomaly detection is for detecting COVID-19 outbreaks. Moreover, we intend to achieve the following:

1. Establish baseline results using non-anomaly detection methods, such as some supervised classification and some unsupervised clustering algorithms.

2. Explore the possibilities of whether tweets data can improve the anomaly detection results.

3. Compare the results of anomaly detection methods to the baseline and validate if anomaly detection can be utilized for COVID-19 outbreak detection.

### *3.3.1 Baseline Models*

Four commonly used machine learning classification algorithms were chosen to produce the baseline results that will be used in comparison with the performance of anomaly detection methods. Two of them are supervised methods, which are Logistic Regression and Support Vector Machine (SVM). The other two are unsupervised clustering methods, Spectral Clustering and K-Means Clustering. All four are directly used from the scikit-learn package [41] with default parameters. For clustering methods, the number of clusters was set to 2 since we treat it as a binary classification problem.

### *3.3.2 Anomaly Detection Models*

We fit our dataset into 12 different anomaly detection models to compare with the baseline results. The tested models fall under the following categories.

1. Distance Based Models

Models under this category score the samples based on a distance metric. Intrinsically they are unsupervised models and do not require training. Within this category, we apply Local Outlier Factor (LOF) [35], Connectivity-based Outlier Factor (COF) [42], Cluster-based Local Outlier Factor (CBLOF) [43], Subspace Outlier Detection (SOD) [44], and $k$-Nearest Neighbor (KNN) [37].

2. One-Class Classification Models:

   This category of models is trained to learn a decision function to identify normal samples. Then test data is fitted into the trained classifier to generate anomaly scores for each sample based on the similarity to the training set. The One-class Support Vector Machine (OCSVM) [45] algorithm is applied for this experiment.

3. Deep Learning Models:

   Deep learning models have proven successful for anomaly detection tasks. Here we utilized the Autoencoder (AE) [39], Variational Autoencoder (VAE) [46], and DeepLog [47].

4. Isolation Based Models:

   For this type of model, we use Isolation Forest [36, 48]. It is based on the fact that anomalous data points are few and different. By randomly selecting a feature then randomly selecting a split value between the minimum and maximum values, it isolates observations to determine which ones are anomalies.

5. Other Models:

   There are two more models used for anomaly detection that do not fall into the above categories. The first one is Histogram-based Outlier Score (HBOS) [49]. HBOS assumes that the features are independent and builds histograms to calculate the degree of anomalies. For multivariate anomaly detection, a histogram is computed for every single feature, which is scored individually and combined in the end. The second model is the Lightweight On-line Detector of Anomalies (LODA) [50]. LODA is an ensemble system of weak detectors that forms a strong anomaly detector. It can be useful to process a large number of samples in real-time or when the detector needs to be updated on-line.

One final variable to specify in the anomaly detection pipeline is the contamination ratio. The contamination ratio is the number of anomalies over the total number of data points. It is a threshold for the anomaly detection algorithms to determine how many points should be considered anomalous based on their anomaly score.

### 3.4 Automated Pipeline Searcher

TODS provides data driven searchers to automatically construct and run pipelines from a defined search space. The search space contains the primitives selected for each module of the pipeline as discussed earlier. In this study, we utilize the brute force searcher which generates all possible combinations of the selected modules in the search space.

# 4. EXPERIMENT

The primary goal of this study is to the explore the extent to whether an anomaly detection approach can be applied to identify and predict outbreaks from the COVID-19 case dataset along with Twitter data. This research follows the general procedure of performing anomaly detection as mentioned above. Initially, COVID-19 case data and Twitter data were collected and stored in their respective files. In the next phase the Twitter data was cleaned and geo-filtered, then combined with the COVID-19 case data to form a multivariate dataset. Next, in the feature extraction and anomaly detection component, the performance of different combinations of augmented features and detection algorithms was investigated. Sudden changes in the daily time-series of both COVID-19 cases and tweets in the USA were examined, and the performance of the anomaly detection pipeline was evaluated in terms of three commonly used metrics.

## 4.1 Dataset Overview

High-quality datasets are essential for generating good detection results. In this research, we examine numerous datasets related to COVID-19, through which we extract and formulate some datasets that can be of great use in the detection pipeline. Current datasets available for COVID-19 research exhibit several inefficiencies, such as high variance in testing rate and testing capabilities between countries, publicly available data on infection rates contain deficiencies, and inconsistencies in reporting such as under-reporting [14]. In particular, there is a lack of understanding regarding the underlying factors that impact the reliability, availability, and accuracy of reported cases on different scales, as well as quantifiable criteria for how the spread of the virus is impacted by quarantine and social distancing efforts. To overcome these challenges that exist in current datasets, we combine two datasets that are widely used in COVID-19 research: time series data provided by the COVID Tracking Project at the Atlantic [51] along with the Coronavirus (COVID-19) Geo-Tagged Tweets Dataset [52] to construct two new multivariate time series datasets for our anomaly detection pipeline.

The time series data provided by the COVID Tracking Project at the Atlantic is updated daily at a US state and national level along with hospitalization, testing, and recovery information provided in a CSV file. It includes 20 features as follows: date, death, death increase, hash (a hash for this record), hospitalized cumulative, hospitalized currently, hospitalized increase, in ICU cumulative, in ICU currently, negative (negative PCR tests), negative increase, on ventilator cumulative, on ventilator currently, pending (pending tests), positive (cases), positive increase, recovered, states, total test results, and total test results increase. Each row of the dataset represents a date, starting from January 13, 2020, and is continuously updated every day until March 7, 2021.

Another dataset – COVID-19 Tweets Dataset is used in conjunction with the COVID-19 cases dataset to provide more degrees of information, which may enhance the performance of anomaly detection algorithms. In the past, social media platforms such as Twitter and Facebook have become an active source of information during a crisis as they tend to break the news much faster than official news outlets and emergency response agencies [53, 54]. As users continuously share information and incite conversations regarding the crisis issue, these social platforms accumulate a significant amount of socially generated data. Amongst these platforms, Twitter, a microblogging platform, has provided researchers with an Application Programming Interface (API) that is convenient for accessing the data (tweets) on its platform, which has become the primary source of information for researchers to gain better insights about a crisis event.

In the original data, the Twitter streaming API was used for accessing tweets from the real-time Twitter feed from March 20, 2020, up to April 9, 2021, the time of this writing. Four keywords, “corona”, “#corona”, “coronavirus”, and “#coronavirus” were used for filtering the Twitter stream up to April 17, 2020. As the pandemic evolved, many new keywords emerged, and the number of keywords has been evolving continuously to 46 words at the time of the writing. Overall, the data collection process has been continuous during the provided period with a few exceptions mentioned in [52]. This continuous data collection process ensures that the temporal patterns in the tweets can be representative of the user’s online discussion regarding the pandemic. Since our research focuses on data in the USA, we chose the geo-tagged tweets dataset that was parsed from the original one. However, one drawback is that a significantly smaller number of tweets are geo-tagged: out of 310 million tweets, 141k tweets (0.045%) were found with geolocation [52]. Complying with Twitter's content redistribution policy, this dataset only provided the tweet IDs in the CSV file for each date. We obtained the original tweets by hydrating these IDs, and this process is further explained in the next section.

## 4.2 Dataset Preprocessing

This study focuses only on the USA's data for analysis and prediction of the COVID-19 outbreak. Therefore, we chose only the national level dataset from the COVID-19 time series datasets provided by the Atlantic. While the original dataset provided many columns of information, only four were chosen for this study: death, death increase, positive, and positive increase. We used the data entries between March 20, 2020, and February 23, 2021, inclusively. Since the original dataset had no missing entries, no more preprocessing was performed on it.

To generate the actual tweets from the geo-tagged tweets dataset, we used the twarc python library [55] to hydrate the tweet IDs. This generated a JSON file for each date. As the Geo-tagged tweets dataset contained not only USA tweets, but also ones from worldwide, we had to write a script to parse through the JSON data files and filter out tweets only from the USA. Then a script was written to count the number of tweets in the dataset for each day, and the result was retrieved for all the dataset files. In total, the collected dataset contained nearly 28 million tweets from March 20, 2020, to February 23, 2021, with the number of tweets ranging from 6 thousand to 295 thousand daily. Now that we have the number of COVID-19-related tweets in the USA every day, we joined the results with the numeric cases dataset to form the COVID-19 and Tweets dataset. Since the tweets only started on March 20, 2020, we selected the COVID-19 dataset to begin at that date as well to have a uniform time series length.

One last step is to manually label the timestamps in the dataset to indicate anomaly and normal events, this serves as the ground truth when evaluating the detection algorithms. Due to the difficulty of knowing the exact dates of outbreaks and the lack of official reports claiming any specific dates as outbreaks, we had to define the anomalies in the dataset based on its trend and values. Here we define an anomaly point as the sudden spike of an upwards trend in the positive increase cases since we consider an outbreak to be caused by the sudden increase of positive COVID-19 cases. By adding a column "ground truth" in the dataset, we label the anomaly points with the number 1 and the normal points with the number 0. Finally, we have two datasets that are ready to be fitted into the models: the COVID-19 dataset and the COVID-19 and Tweets dataset, both are mostly the same except that the COVID-19 and Tweets dataset contains an additional feature that is the number of COVID-19 related tweets each day. Figure 4.1 shows five features in the dataset, with the red dot indicating the anomaly.

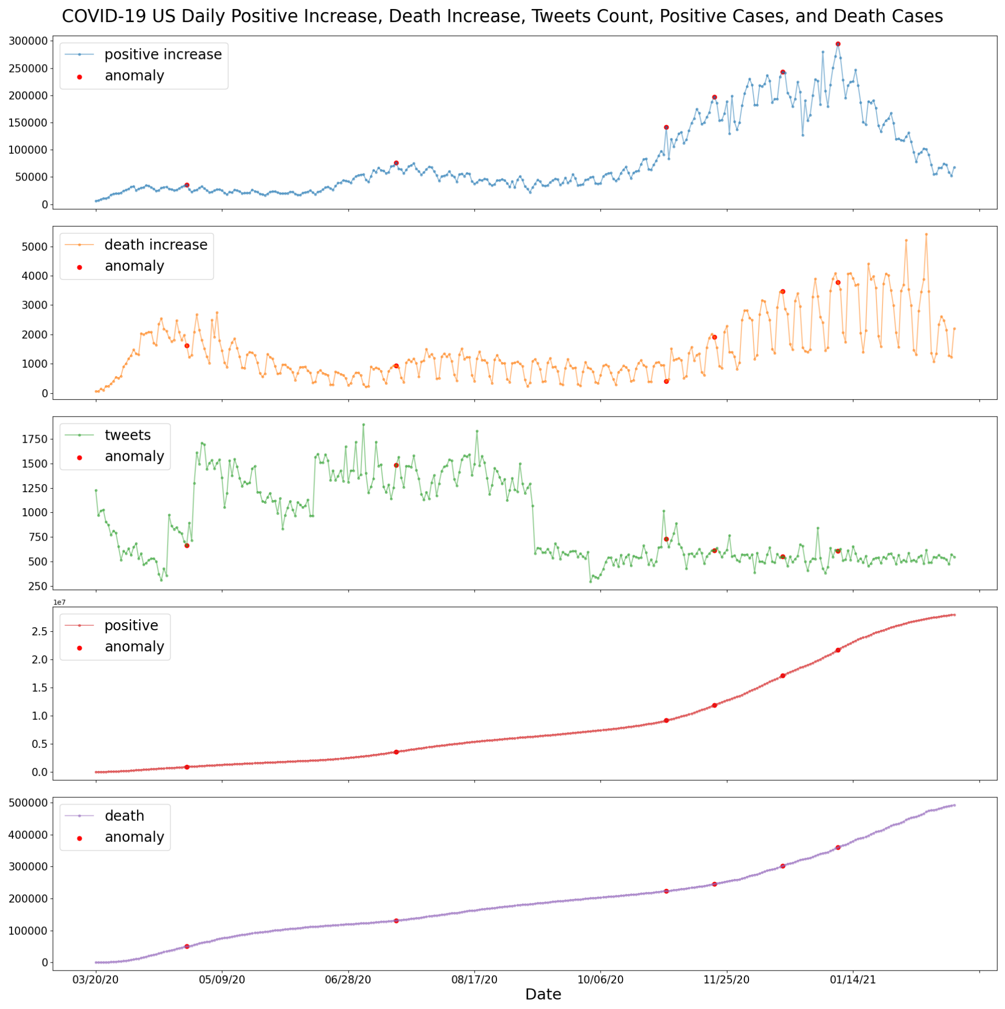


Figure 4.1: Visualization of daily positive increase, death increase, tweets count, positive cases, and death cases in the US

Table 4.1 shows the statistics of the two datasets that are collected for this study. Note that points are the number of days or timestamps in the time series, and dimensions are the number of features in the time series. The two datasets have the same statistics except that the COVID-19 and Tweets dataset has one more dimension which is the daily tweets count.

Table 4.1: Statistics of the datasets

| Dataset | Points | Dimensions | Anomalies | Anomaly % |
|---|---|---|---|---|
| COVID-19 Dataset | 341 | 4 | 6 | 1.76 |
| COVID-19 and Tweets Dataset | 341 | 5 | 6 | 1.76 |

## 4.3 Pipeline Setup

Here we define the search space for the automated searcher, which are the primitives in each module of the anomaly detection pipeline.

After careful examination, a seven-day cycle is observed in our dataset, specifically for death increase, positive case increase, and total death and positive cases. In fact, this seven-day cycle is rigorously tested and proved in this research [56]. To remove such weekly cycle in the time series, a few approaches can be considered as described in [56], such as simple differencing $(Y_t - Y_{t-7})$, seven-day moving average, harmonic function (series of sine/cosine functions), or using dummy variables control for day-to-day variation. For the scope of this research, we only applied the moving average smoothing to our dataset with a sliding window of 7 days. Thus, the time series processing search space only includes moving average smoothing.

For feature analysis, 25 features that are included in TODS packaged are added to the search space. They are statistical mean, statistical median, statistical g-mean, statistical abs-sum, statistical h-mean, statistical maximum, statistical minimum, statistical mean-abs, statistical mean-abs-temporal-derivative, statistical mean-temporal-derivative, statistical median-abs-deviation, statistical kurtosis, statistical skew, statistical std, statistical var, statistical variation, statistical vec-sum, statistical willison-amplitude, statistical zero-crossing, Spectral Residual Transform, Fast Fourier

Transform, Discrete Cosine Transform, HP filter, Truncated SVD, and TRMF. All of them used the default hyperparameters.

A total of 12 anomaly detection algorithms introduced earlier were added to the TODS pipeline search space. The hyperparameters were not dynamically tuned and used their default values. Finally, we define the contamination ratios for the search space. For our original datasets, there are 341 total points with 6 anomalies and 335 normal points, therefore the contamination ratio is $\frac{6}{341} = 0.017595$. For the moving average smoothed datasets, there are 335 total points with 6 anomalies and 329 normal points, therefore the contamination ratio is $\frac{6}{335} = 0.01791$. Consequently, we round the two numbers and for both situations, the default contamination score is set to 0.018. Moreover, to examine whether a slightly lower or larger contamination ratio would affect the selection of anomalous points, four more contamination ratios are tested in the pipeline: 0.015, 0.02, 0.022, 0.025. After calculation and rounding, they corresponding to 5, 6, 7, and 8 points for the datasets.

The four baseline models specified in the methodology section are directly used from the scikit-learn package [41] with default parameters. For clustering methods, the number of clusters was set to 2 since we treat it as a binary classification problem.

## 4.4 Evaluation Metrics

For a binary classification problem as well as an anomaly detection problem, the prediction results are binary labels, typically 0 for negative (normal) and 1 for positive (anomaly). There are four cases for prediction results: For the negative class, if the prediction is negative, it is referred to as a true negative (TN) and if positive, it is a false positive (FP). For a positive class, if the prediction is positive, it is called a true positive (TP) and if negative, it is a false negative. These four classes are used to calculate precision, recall, and F1-score, which are the three most used metrics to evaluate the performance of an anomaly detection model.

### *4.4.1 Precision*

Precision, also known as the positive predictive value, is a measure of the number of correct positives the model claims compared to the total number of positives it claims. It aims to answer the question of what proportion of positive predictions was correct. The formula for precision is shown below.

$$Precision = \frac{TP}{TP + FP}$$

### *4.4.2 Recall*

Recall, also called true positive rate, represents the number of positives the model claims compared to the actual number of positives there are throughout the data. It answers the question of what proportion of actual positives was identified correctly. Below is the formula for recall.

$$Recall = \frac{TP}{TP + FN}$$

### *4.4.3 F1-score*

Finally, the F1-score is the weighted average of the recall and precision.

$$F1\text{-}\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

### *4.4.4 Ranged-based Anomaly Labels*

Classical anomaly detection methods are principally concerned with point-based anomalies, which are anomalies that occur at a single point in time. However, due to the limited amount of data and sparse anomaly points in the dataset, it is not ideal nor reasonable to aim at detecting the exact anomaly points. In the case of the COVID-19 data in this research, there is no significant meaning to only aim at predicting the exact points of an outbreak, since the outbreaks do not happen at some exact date, but rather happen gradually and reach some peak value. To account for this situation, an extra step is added to process the predicted label before it is used for calculating the precision, recall, and F1-score.

We consider a ranged-base anomaly approach by considering a seven-day window before an actual outbreak label. Note that an outbreak corresponds to a true label (1), while normal corresponds to a false label (0). For any true label in the prediction array that falls within the seven-day window of an actual true label, we count that as a true positive by setting the index of that actual true label in the predicted array to be true and change that predicted true label to be false. In other words, if a predicted true label is within a seven-day window before an actual true label, we consider that it correctly predicted the actual true label. This range-based label processing is used for all the algorithms in the experiments.

## 4.5 Experiment Procedure and Settings

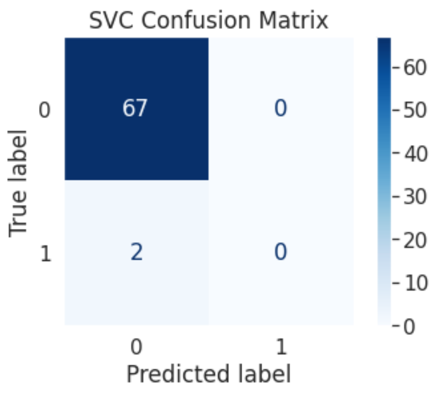
In the first round of experiments, we run the COVID-19 dataset on the four baseline models to generate the baseline results. Then, we run the TODS pipeline and only define the search space to include detection algorithms and the contamination ratios. This is to set the baseline results for the TODS pipeline. For the third round, we add the times series processing module containing the moving average smoothing to the same search space as before. For the last round, the feature extraction module with 25 features is added to the search space.

All the code and models used in this study are implemented using Python 3.6 programming language. The experiments were conducted on a remote Linux server with Ubuntu 16.04 as the operating system.

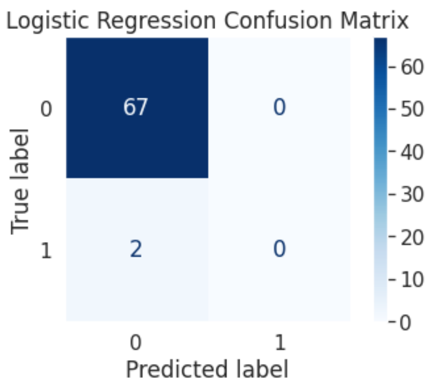
# 5. RESULTS

## 5.1 Baseline Results

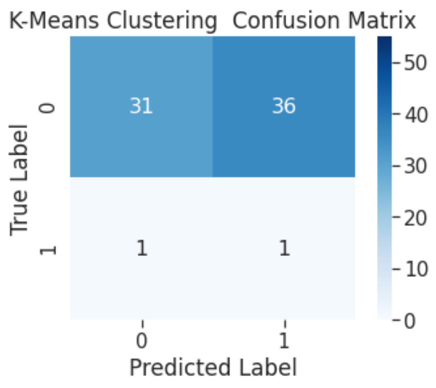
To collect the baseline results, four models were implemented on the datasets. The two supervised classification models are Support Vector Classifiers (SVC) and Logistic Regression. The two unsupervised clustering models are K-Means Clustering and Spectral Clustering. Precision, recall, and F1-score for the anomaly class are considered in the evaluation of the algorithms' performances. To obtain these metrics, the confusion matrix is computed (Figure 5.1) for each algorithm. More detailed information regarding the confusion matrix concept and its metrics can be found in [57]. Table 5.1 shows the results for the selected algorithms.



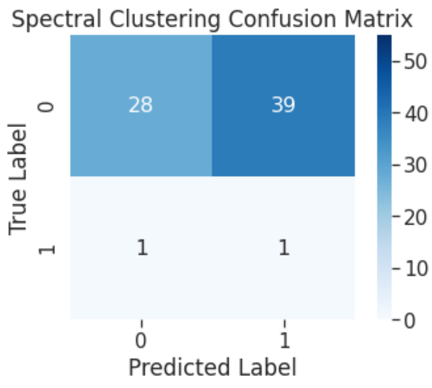
(a)



(b)



(c)



(d)

Figure 5.1: (a) SVC (b) Logistic Regression (c) K-Means Clustering (d) Spectral Clustering

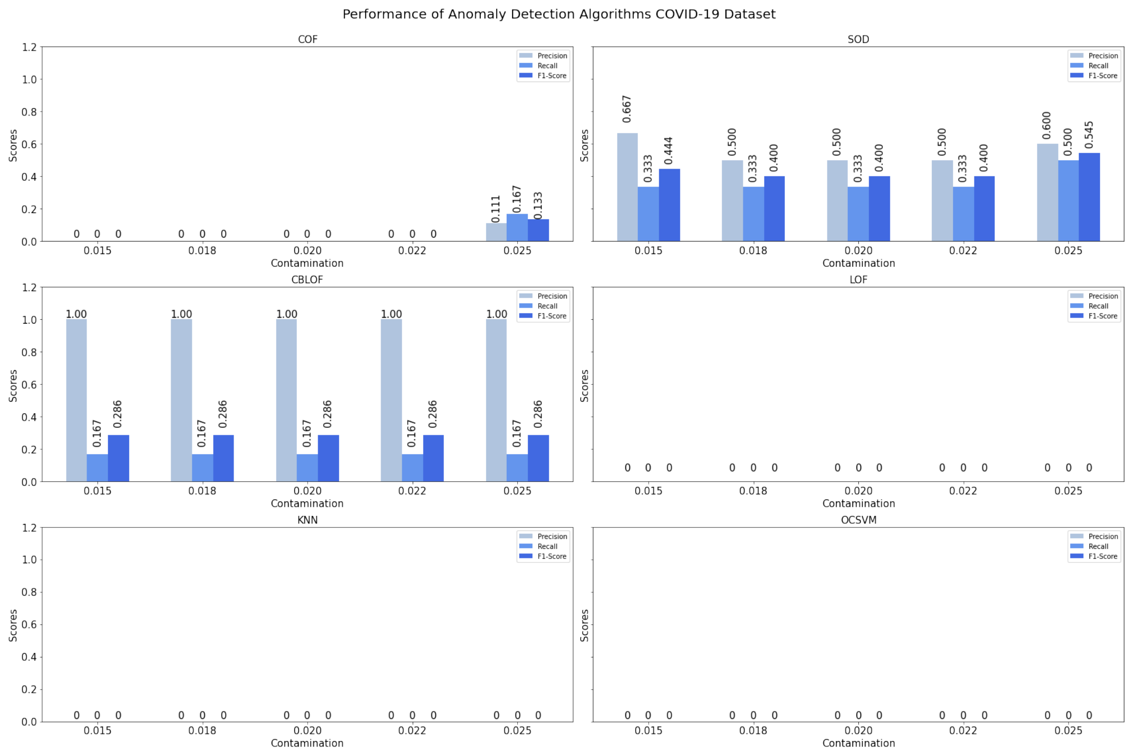
Table 5.1: Comparison of baseline models' performance

| | **SVC** | **Logistic Regression** | **K-Means Clustering** | **Spectral Clustering** |
|---|---|---|---|---|
| Precision | 0.00 | 0.00 | **0.03** | **0.03** |
| Recall | 0.00 | 0.00 | **0.50** | **0.50** |
| F1-score | 0.00 | 0.00 | **0.05** | **0.05** |

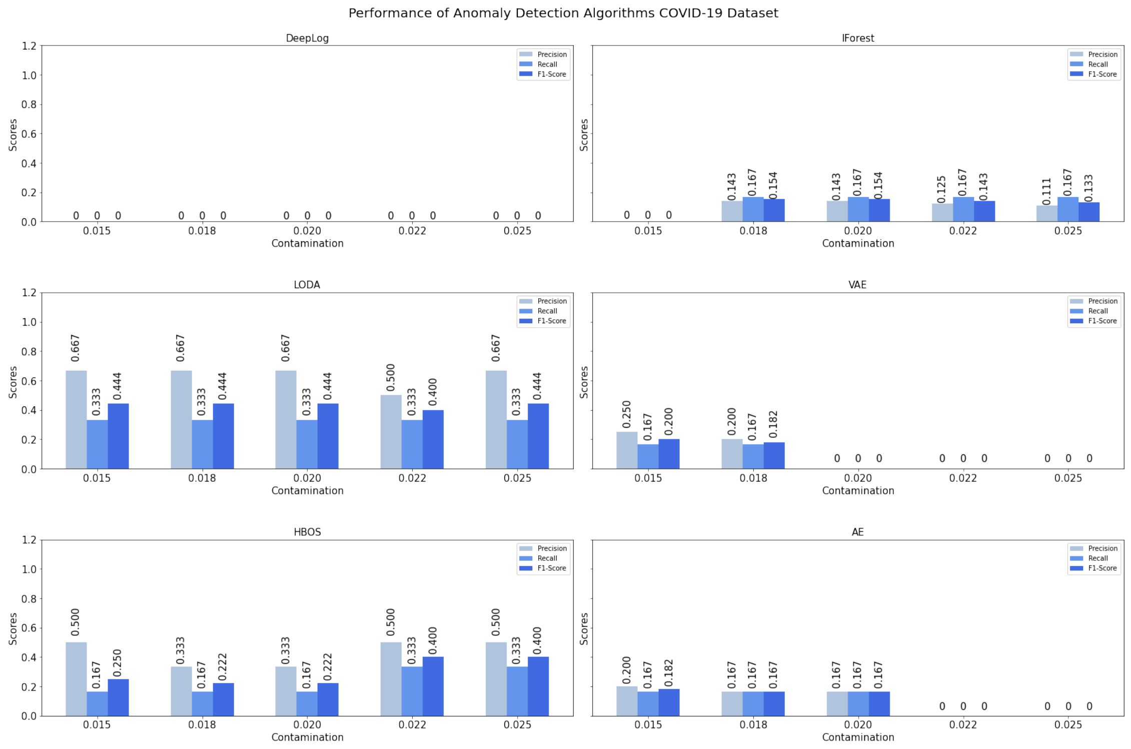
Since there are much less incidents of anomalies compared to normal data points, with less than 2% of the whole data that are anomalies, the datasets used for training and testing suffer from imbalance targeted data. This characteristic can be captured from the results above, as for the two supervised classification algorithms SVC and Logistic Regression, no anomaly points were classified nor correctly classified. This results in the precision, recall, and F1-score to be 0. For the unsupervised clustering algorithms, they yielded the same results for all three metrics. From the confusion matrix, we know that both have one true positive prediction, and one false negative. This is correctly reflected in the recall, since out of the two actual positive labels, only one is correctly predicted, thus resulting in a recall of 0.5. Precision is quite low for both cases, implying that although there may be many positive predictions, only a small proportion is correct, which is also represented in the confusion matrix. The F1-score is also quite low which is caused by the low precision score. Although the amount of true negative and false negative predictions is slightly different for the two models, the results for precision, recall and F1-score remained the same due to rounding.

## 5.2 Anomaly Detection Results

We examined the possibilities of applying anomaly detection algorithms to detect COVID-19 outbreak. The anomaly detection algorithms were applied to both the COVID-19 dataset and COVID-19 and Tweets dataset, then the results were compared. An overview of the results can be found in Figure 5.2 and Figure 5.3, with Figure 5.2 showing the results on the COVID-19 dataset, and Figure 5.3 shows the results for the COVID-19 and Tweets dataset.

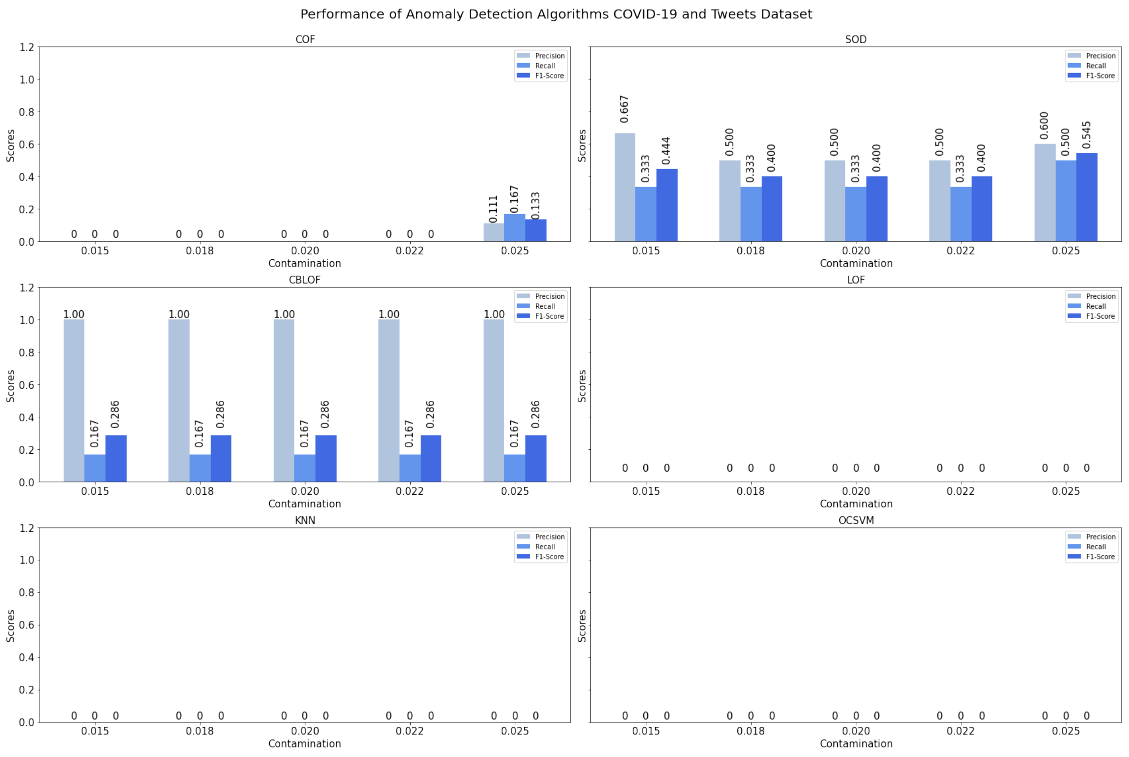


(a)

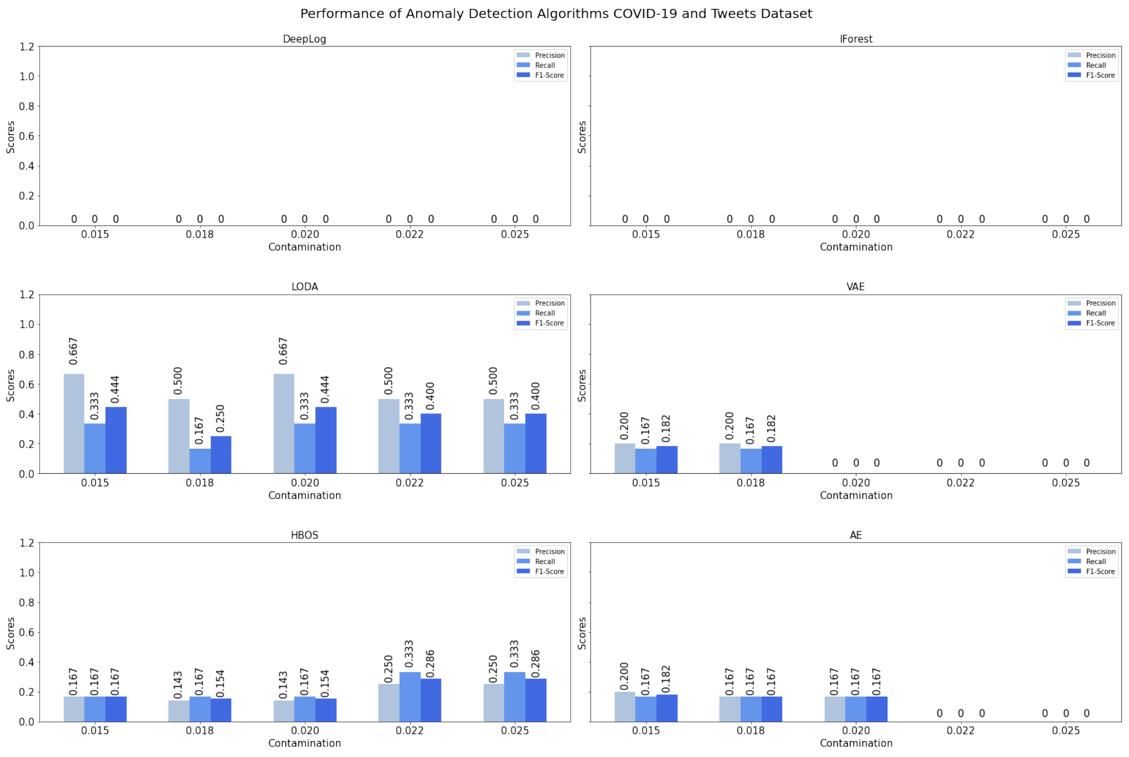


(b)

Figure 5.2: (a) Results for COF, SOD, CBLOF, LOF, KNN, and OCSVM on COVID-19 Dataset (b) Results for DeepLog, iForest, LODA, VAE, HBOS, and AE on COVID-19 Dataset



(a)



(b)

Figure 5.3: (a) Results for COF, SOD, CBLOF, LOF, KNN, and OCSVM on COVID-19 and Tweets Dataset (b) Results for DeepLog, iForest, LODA, VAE, HBOS, and AE on COVID-19 and Tweets Dataset

Each figure has two plots with 6 subplots each, totaling 12 different anomaly detection algorithms. For each algorithm, results for precision, recall, and F1-score are independently gathered for the five different contamination ratios specified earlier. By examining the plots, the first thing to notice that there are many blank ones. This implies that the detection algorithm did not predict any outbreaks in the data. For both datasets, the LOF, KNN, OCSVM, and DeepLog algorithm behaved poorly as they did not predict any of the outbreaks. For the COVID-19 and Tweets dataset, there is one more algorithm that has zero for all three metrics and it is iForest. Some algorithms performed the same on both datasets, meaning they had the same scores for precision, recall, and F1-score. These algorithms are COF, SOD, CBLOF, and AE. The VAE performance was mostly the same for both datasets with only a higher precision of 0.05 when contamination is 0.015 for the COVID-19 dataset.

### *5.2.1 Evaluating the Effect of Tweets Data on Outbreak Detection*

Table 5.2: Comparison of anomaly detection models on two datasets

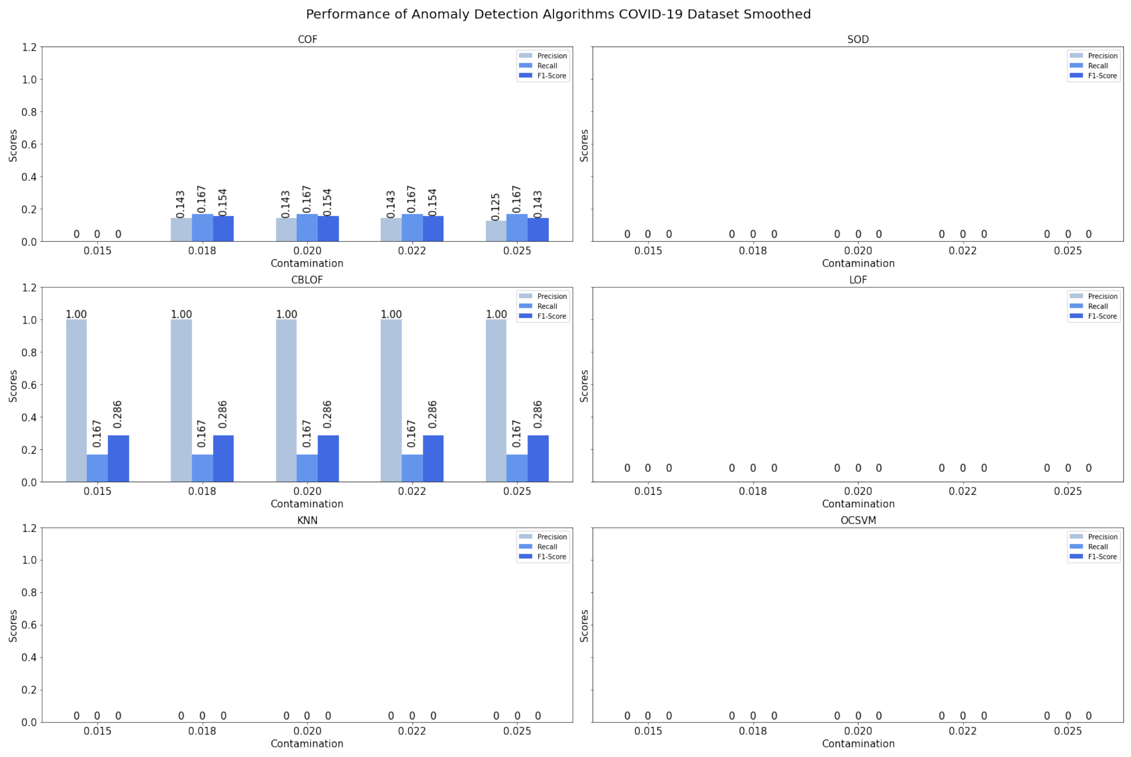
| | | COVID-19 Dataset | | | COVID-19 and Tweets Dataset | | |
|---|---|---|---|---|---|---|---|
| Contamination | Metric | iForest | HBOS | LODA | iForest | HBOS | LODA |
| 0.015 | Precision | 0 | **0.5** | 0.667 | 0 | 0.167 | 0.667 |
| | Recall | 0 | 0.167 | 0.333 | 0 | 0.167 | 0.333 |
| | F1-score | 0 | **0.25** | 0.444 | 0 | 0.167 | 0.444 |
| 0.018 | Precision | 0.143 | **0.333** | **0.667** | 0 | 0.143 | 0.5 |
| | Recall | 0.167 | 0.167 | **0.333** | 0 | 0.167 | 0.167 |
| | F1-score | 0.154 | **0.222** | **0.444** | 0 | 0.154 | 0.25 |
| 0.020 | Precision | 0.143 | **0.333** | 0.667 | 0 | 0.143 | 0.667 |
| | Recall | 0.167 | 0.167 | 0.333 | 0 | 0.167 | 0.333 |
| | F1-score | 0.154 | **0.222** | 0.444 | 0 | 0.154 | 0.444 |
| 0.022 | Precision | 0.125 | **0.5** | 0.5 | 0 | 0.25 | 0.5 |
| | Recall | 0.167 | 0.333 | 0.333 | 0 | 0.333 | 0.333 |
| | F1-score | 0.143 | **0.4** | 0.4 | 0 | 0.286 | 0.4 |
| 0.025 | Precision | 0.111 | **0.5** | **0.667** | 0 | 0.25 | 0.5 |
| | Recall | 0.167 | 0.333 | 0.333 | 0 | 0.333 | 0.333 |
| | F1-score | 0.133 | **0.4** | **0.444** | 0 | 0.286 | 0.4 |

One interesting discovery was that the rest of the algorithms performed better on the COVID-19 dataset rather than on the COVID-19 and Tweets dataset. They are iForest, HBOS, and LODA. Table 5.2 shows the direct comparison between the results of these algorithms on the two datasets. iForest performed better on the COVID-19 dataset compared to the COVID-19 and Tweets dataset across all contamination ratios except for 0.015. For HBOS, it is interesting to see that its precision and F1-score on the COVID-19 dataset were better than the COVID-19 and COVID-19 dataset across all ranges, but the recall stayed the same. This implies that although it had the same correct positive predictions on both datasets, it had fewer False Positives when predicting the COVID-19 dataset, thus resulting in the same recall but better precision. LODA's performance on the COVID-19 dataset is only slightly better for contamination ratios 0.018 and 0.025.

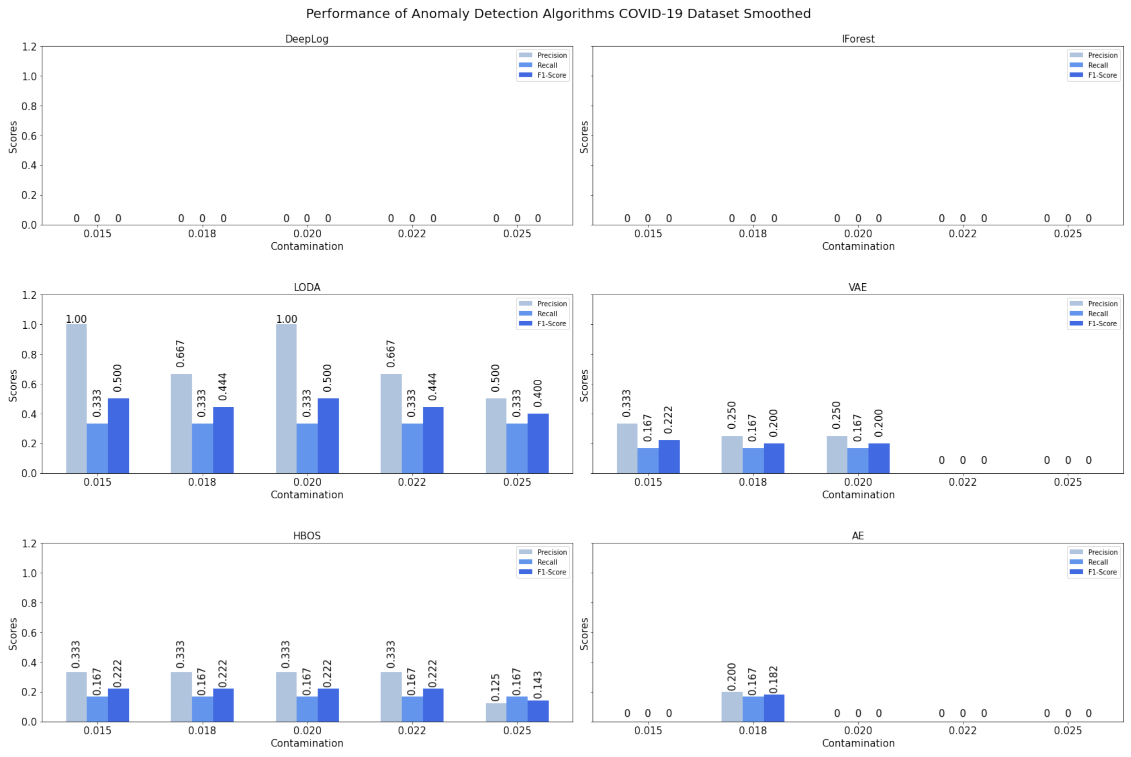
From these analyses, it may be reasonable to bring up the assumption that using the amount of COVID-19 related tweets each day as an additional feature did not result in better performance of the anomaly detection algorithms. From the visualized data, it can be concluded that although there were peaks and sudden changes of trend in the tweets data, they did not align well with the peaks of the daily positive increase or death increase. Moreover, the anomaly labels were chosen from the peaks in positive increase and did not align well with the probable anomalies of the twitter data. All these factors may have introduced more noise to the data, eventually causing the results to be worse than the dataset without twitter data.

### *5.2.2 Evaluating the Effect of Smoothed Data and Other Features on Outbreak Detection*

Two more experiments were conducted, one was applying a seven-day moving average smoothing on the dataset, another was adding a feature extraction module before the detection algorithm in the TODS pipeline to extract an additional feature from the original dataset. The motivation for applying a moving average smoothing was that an approximate seven-day cycle was observed in the dataset, and we wanted to explore whether smoothing the data could remove some noise and possibly improve the performance of the algorithms. First, we examine the performance of the anomaly detection algorithms on the smoothed data. Figure 5.4 shows the results.



(a)



(b)

Figure 5.4: (a) Results for COF, SOD, CBLOF, LOF, KNN, and OCSVM on COVID-19 Dataset Smoothed (b) Results for DeepLog, iForest, LODA, VAE, HBOS, and AE on COVID-19 Dataset Smoothed

Varying results were observed from the plots, as some algorithms had better performance while others had worse. COF was able to get better performance on the smoothed dataset since it achieved better scores for contamination ratios 0.018, 0.02, and 0.022, which previously did not predict any outbreaks correctly. LODA achieved better precision and recall for contamination ratios 0.015 and 0.02 on the smoothed data. VAE had slightly better performance for contamination ratio 0.02. Regarding algorithms that performed worse on smoothed data, SOD and iForest had the most drastic changes as they performed above average on the original data but failed to correctly predict any outbreaks across all contamination ratios on the smoothed data. HBOS and AE both had slightly lower precision and recall on the smoothed data, and CBLOF displayed the same performance on both datasets.

These results show that smoothing the time series resulted in varying performances across different anomaly detection algorithms. One thing to consider is that although smoothing may remove some noise in the data, it could also cause the anomaly points to be less significant compared to before smoothing. While some anomaly detection algorithms may suffer from the loss of features due to smoothing, others were able to outperform compared to using raw data. Therefore, we can learn that whether moving average smoothing can improve a model's performance on a time series dataset is a case-by-case situation, and it requires trial and error on different models and datasets to achieve desirable results.

The motivation for applying feature extraction to the dataset is that by generating more features, we can increase the dataset dimension, which may lead to improved performances of the anomaly detection algorithms. While we also tested the performance of the anomaly detection algorithms after applying different feature extraction methods, the results were mainly unchanged, with only slight improvements for few specific anomaly detection algorithms and specific contamination ratios. Since there were not any substantial discoveries, the results of these experiment trials are not included in this study. However, what can be learned is that the feature extraction methods that were tested may not have extracted the most meaningful features from the time series data that could improve the performance of the detection algorithms. There are still many features to explore, and it may be pursued in future studies.

## 5.3 Comparing Anomaly Detection Models to Baseline Models

Finally, we compare the anomaly detection model results with the baseline model results. Two of the best performing baseline models are selected, which are K-Means Clustering and Spectral Clustering. For anomaly detection models the four best were chosen, they are SOD, LODA, HBOS, CBLOF. The results are concluded in Table 5.3.

Table 5.3: Comparison of anomaly detection models to baseline models

| | K-Means Clustering | Spectral Clustering | SOD | LODA | HBOS | CBLOF |
|---|---|---|---|---|---|---|
| Precision | 0.500 | 0.500 | **0.667** | **0.667** | 0.500 | 1.000 |
| Recall | 0.030 | 0.030 | **0.333** | **0.333** | 0.333 | 0.167 |
| F1-score | 0.050 | 0.050 | **0.444** | **0.444** | 0.400 | 0.286 |

The four chosen anomaly detection models, which are the best out of all the tested anomaly detection models, outperform all compared baseline models in outbreak detection for the COVID-19 dataset. They had higher or equal precision, and much higher recall and F1-score compared to the baseline models. This implies that not only were the anomaly detection models able to predict fewer False Positives, they were also capable of predicting more True Positives. When comparing between the four anomaly detection models, SOD and LODA were the best, with equal performance metrics that slightly above HBOS. Although CBLOF had a precision of 1, it suffers more from low recall. It can be observed that anomaly detection models have the capability to detect abnormal upswing trends and outbreaks in the COVID-19 pandemic data. This justifies our assumption that using anomaly detection could be an appropriate technique in detecting and predicting the outbreaks in COVID-19 waves, as well as for discovering new patterns in the dataset.

# 6. CONCLUSION

## 6.1 Summary

This study evaluated the applicability of anomaly detection models for predicting COVID-19 outbreaks. The models indicated promising results in terms of predicting outbreaks in the time series data without requiring the various assumptions of traditional epidemiological models. Anomaly detection models, as an alternative to traditional machine learning models and epidemiological models, show potential in predicting COVID-19 outbreak. Although there have been many studies that model COVID-19 from its epidemiological dynamics, few focus on tackling the problem of outbreak detection through anomaly detection methods. We used COVID-19 data along with COVID-19 related Twitter data representing the entire United States and evaluated the performance of the 12 classic anomaly detection models together with 4 baseline classification and clustering models. Various anomaly detection models outperformed the baseline models as well as other anomaly detection models in both precision and recall scores. Our contribution can be summarized as follows:

1. We proposed using an anomaly detection pipeline, which consists of data processing to feature extraction and anomaly detection, to attempt solving the unsupervised anomaly detection problem of prediction COVID-19 outbreak with multivariate time series data.

2. Using an automated searcher, we were able to define our search space and allow the searcher to generate all possible pipeline configurations, as well as train the model and generate the results automatically. This allowed us to explore the results of different anomaly detection models, feature extraction algorithms and preprocessing methods to find the most suitable pipeline for our use.

3. This study demonstrates the capability of the constructed anomaly detection pipeline to detect anomalous patterns and outbreaks in pandemic time series data. Our TODS package

is aimed to provide an end-to-end solution for building and deploying anomaly detection pipelines in real-world scenarios. Not only can it construct the most robust pipeline for detection outbreaks during the COVID-19 pandemic, but it may also be applied to many different settings.

## 6.2 Future Work

Moving forward, there are many paths to pursue that build off this study as well as address its limitations. The main limitation of the study is the selection of modules for each pipeline. For the scope of this study, we limited the time series preprocessing to use only moving average smoothing and limited the feature extraction to use only one feature extraction method. This choice, although provided sufficient results to serve our experiments for this study, has significantly limited the configuration of pipelines generated by the searcher. Further experimentation with different pipeline configurations could provide more in-depth insights on which combination of preprocessing methods, extracted features, and detection algorithms will yield the best results.

Another limitation is the selection of hyperparameters for each module of the pipeline. Although some reasonable results were obtained through default hyperparameters, continued exploration and experimentation with optimization-based techniques to select the model's hyperparameters could lead to enhanced performance. Developing and testing new searchers such as a Bayesian rule-based searcher could also provide new insights on searching for the most optimal hyperparameters and models for a specific task.

Moreover, other avenues of exploration could include experimenting with different anomaly detection approaches. For this study, only point-wise anomalies were considered, yet there also exist pattern-wise and system-wise anomalies. Building upon this study, it is also possible to define sequences of COVID-19 time series data as anomalous and perform range-based anomaly detection. Finally, we can explore the method of ensemble and investigate the performance of ensemble anomaly detection algorithms.

# REFERENCES

[1] S. Makridakis, A. Wakefield, R. Kirkham, *et al.*, "Predicting medical risks and appreciating uncertainty," *Foresight: The International Journal of Applied Forecasting*, no. 52, pp. 28–35, 2019.

[2] S. Ghamizi, R. Rwemalika, M. Cordy, L. Veiber, T. F. Bissyandé, M. Papadakis, J. Klein, and Y. Le Traon, "Data-driven simulation and optimization for covid-19 exit strategies," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3434–3442, 2020.

[3] R. M. Burke, M. P. Shah, M. E. Wikswo, L. Barclay, A. Kambhampati, Z. Marsh, J. L. Cannon, U. D. Parashar, J. Vinjé, and A. J. Hall, "The norovirus epidemiologic triad: predictors of severe outcomes in us norovirus outbreaks, 2009–2016," *The Journal of infectious diseases*, vol. 219, no. 9, pp. 1364–1372, 2019.

[4] C. J. Carlson, E. Dougherty, M. Boots, W. Getz, and S. J. Ryan, "Consensus and conflict among ecological forecasts of zika virus outbreaks in the united states," *Scientific reports*, vol. 8, no. 1, pp. 1–15, 2018.

[5] E. F. Kleiven, J.-A. Henden, R. A. Ims, and N. G. Yoccoz, "Seasonal difference in temporal transferability of an ecological model: near-term predictions of lemming outbreak abundances," *Scientific reports*, vol. 8, no. 1, pp. 1–6, 2018.

[6] N. A. Rivers-Moore and T. R. Hill, "A predictive management tool for blackfly outbreaks on the orange river, south africa," *River Research and Applications*, vol. 34, no. 9, pp. 1197–1207, 2018.

[7] R. Yin, V. H. Tran, X. Zhou, J. Zheng, and C. K. Kwoh, "Predicting antigenic variants of h1n1 influenza virus based on epidemics and pandemics using a stacking model," *PloS one*, vol. 13, no. 12, p. e0207777, 2018.

[8] S. F. Ardabili, A. Mosavi, P. Ghamisi, F. Ferdinand, A. R. Varkonyi-Koczy, U. Reuter, T. Rabczuk, and P. M. Atkinson, "Covid-19 outbreak prediction with machine learning," *Algorithms*, vol. 13, no. 10, p. 249, 2020.

[9] D. Ivanov, "Predicting the impacts of epidemic outbreaks on global supply chains: a simulation-based analysis of the covid-19/sars-cov2 case," *Transp. Res. E https://doi. org/10.1016/j. tre*, 2020.

[10] I. S. Koolhof, K. B. Gibney, S. Bettiol, M. Charleston, A. Wiethoelter, A.-L. Arnold, P. T. Campbell, P. J. Neville, P. Aung, T. Shiga, *et al.*, "The forecasting of dynamical ross river virus outbreaks: Victoria, australia," *Epidemics*, vol. 30, p. 100377, 2020.

[11] F. Petropoulos and S. Makridakis, "Forecasting the novel coronavirus covid-19," *PloS one*, vol. 15, no. 3, p. e0231236, 2020.

[12] A. Darwish, Y. Rahhal, and A. Jafar, "A comparative study on predicting influenza outbreaks using different feature spaces: application of influenza-like illness data from early warning alert and response system in syria," *BMC research notes*, vol. 13, no. 1, pp. 1–8, 2020.

[13] G. Pandey, P. Chaudhary, R. Gupta, and S. Pal, "Seir and regression model based covid-19 outbreak predictions in india," *arXiv preprint arXiv:2004.00958*, 2020.

[14] N. Soures, D. Chambers, Z. Carmichael, A. Daram, D. P. Shah, K. Clark, L. Potter, and D. Kudithipudi, "Sirnet: understanding social distancing measures with hybrid neural network model for covid-19 infectious spread," *arXiv preprint arXiv:2004.10376*, 2020.

[15] Z. Yang, Z. Zeng, K. Wang, S.-S. Wong, W. Liang, M. Zanin, P. Liu, X. Cao, Z. Gao, Z. Mai, *et al.*, "Modified seir and ai prediction of the epidemics trend of covid-19 in china under public health interventions," *Journal of thoracic disease*, vol. 12, no. 3, p. 165, 2020.

[16] O. T. Muurlink, P. Stephenson, M. Z. Islam, and A. W. Taylor-Robinson, "Long-term predictors of dengue outbreaks in bangladesh: A data mining approach," *Infectious Disease Modelling*, vol. 3, pp. 322–330, 2018.

[17] F. Koike and N. Morimoto, "Supervised forecasting of the range expansion of novel non-indigenous organisms: Alien pest organisms and the 2009 h1n1 flu pandemic," *Global Ecology and Biogeography*, vol. 27, no. 8, pp. 991–1000, 2018.

[18] Z. Li, W. Yang, S. Peng, and F. Liu, "A survey of convolutional neural networks: analysis, applications, and prospects," *arXiv preprint arXiv:2004.02806*, 2020.

[19] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[20] W. De Mulder, S. Bethard, and M.-F. Moens, "A survey on the application of recurrent neural networks to statistical language modeling," *Computer Speech & Language*, vol. 30, no. 1, pp. 61–98, 2015.

[21] K. Smagulova and A. P. James, “A survey on lstm memristive neural network architectures and applications,” *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019.

[22] V. Chandola, “Outlier detection-a survey varun chandola, arindam banerjee, and vipin kumar,” 2007.

[23] D. M. Hawkins, *Identification of outliers*, vol. 11. Springer, 1980.

[24] V. Hodge and J. Austin, “A survey of outlier detection methodologies,” *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[25] K.-H. Lai, D. Zha, G. Wang, J. Xu, Y. Zhao, D. Kumar, Y. Chen, P. Zumkhawaka, M. Wan, D. Martinez, *et al.*, “Tods: An automated time series outlier detection system,” *arXiv preprint arXiv:2009.09822*, 2020.

[26] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, “Rule-based anomaly pattern detection for detecting disease outbreaks,” in *AAAI/IAAI*, pp. 217–223, 2002.

[27] W.-K. Wong, A. W. Moore, G. F. Cooper, and M. M. Wagner, “Bayesian network anomaly pattern detection for disease outbreaks,” in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 808–815, 2003.

[28] D. L. Buckeridge, H. Burkom, M. Campbell, W. R. Hogan, A. W. Moore, *et al.*, “Algorithms for rapid outbreak detection: a research synthesis,” *Journal of biomedical informatics*, vol. 38, no. 2, pp. 99–113, 2005.

[29] S. Merler, P. Poletti, M. Ajelli, B. Caprile, and P. Manfredi, “Coinfection can trigger multiple pandemic waves,” *Journal of theoretical biology*, vol. 254, no. 2, pp. 499–507, 2008.

[30] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, “A comparative evaluation of outlier detection algorithms: Experiments and analyses,” *Pattern Recognition*, vol. 74, pp. 406–421, 2018.

[31] V. Chandola, “Anomaly detection: A survey varun chandola, arindam banerjee, and vipin kumar,” 2007.

[32] A. M. Bianco, M. Garcia Ben, E. Martinez, and V. J. Yohai, “Outlier detection in regression models with arima errors using robust estimates,” *Journal of Forecasting*, vol. 20, no. 8, pp. 565–579, 2001.

[33] D. Chen, X. Shao, B. Hu, and Q. Su, "Simultaneous wavelength selection and outlier detection in multivariate regression of near-infrared spectra," *Analytical Sciences*, vol. 21, no. 2, pp. 161–166, 2005.

[34] P. Galeano, D. Peña, and R. S. Tsay, "Outlier detection in multivariate time series via projection pursuit," 2004.

[35] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

[36] F. T. Liu, K. M. Ting, and Z. hua Zhou, "Isolation forest," in *In ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. IEEE Computer Society*, pp. 413–422.

[37] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 427–438, 2000.

[38] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets," in *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 1317–1322, Ieee, 2016.

[39] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pp. 4–11, 2014.

[40] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

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

[42] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 535–548, Springer, 2002.

[43] Z. He, X. Xu, and S. Deng, “Discovering cluster-based local outliers,” *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[44] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, “Outlier detection in axis-parallel subspaces of high dimensional data,” in *Pacific-asia conference on knowledge discovery and data mining*, pp. 831–838, Springer, 2009.

[45] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[46] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[47] M. Du, F. Li, G. Zheng, and V. Srikumar, “Deeplog: Anomaly detection and diagnosis from system logs through deep learning,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, 2017.

[48] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation-based anomaly detection,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–39, 2012.

[49] M. Goldstein and A. Dengel, “Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm,” *KI-2012: Poster and Demo Track*, pp. 59–63, 2012.

[50] T. Pevnỳ, “Loda: Lightweight on-line detector of anomalies,” *Machine Learning*, vol. 102, no. 2, pp. 275–304, 2016.

[51] Atlantic, “The covid tracking project data.” 2020, `https://covidtracking.com/` Accessed Apr. 9, 2021. [Online].

[52] R. Lamsal, “Coronavirus (covid-19) tweets dataset,” *IEEE Dataport*, vol. 10, 2020.

[53] M. Imran, C. Castillo, F. Diaz, and S. Vieweg, “Processing social media messages in mass emergency: A survey,” *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, pp. 1–38, 2015.

[54] M. Imran, F. Ofli, D. Caragea, and A. Torralba, “Using ai and social media multimodal content for disaster response and management: Opportunities, challenges, and future directions,” 2020.

[55] DocNow, "Docnow/twarc." 2020, `https://github.com/DocNow/twarc` Accessed Apr. 9, 2021. [Online].

[56] G. Wang, Z. Gu, X. Li, S. Yu, M. Kim, Y. Wang, L. Gao, and L. Wang, "Comparing and integrating us covid-19 data from multiple sources with anomaly detection and repairing," *arXiv e-prints*, pp. arXiv–2006, 2020.

[57] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.