Multi-dataset Time Series Anomaly Detection

MSBD 5002 Data Mining, Fall 2021

Kar Chun, CHAN
Student No: 20729353
Department of Computer Science and Engineering & Department of Mathematics
The Hong Kong University of Science and Technology
kcchanby@connect.ust.hk

Chun Ting Jeff, LAM
Student No: 12222973
Department of Computer Science and Engineering & Department of Mathematics
The Hong Kong University of Science and Technology
ctjlam@connect.ust.hk

Kai Kwong, LEUNG
Student No: 20727941
Department of Computer Science and Engineering & Department of Mathematics
The Hong Kong University of Science and Technology
kkleungai@connect.ust.hk

ABSTRACT

This project is based on the KDD Cup 2021, one of the most famous competitions in data mining area. Time series data is composed of a sequence of values over time, which exists in many applications, such as stock prices and websites. However, some values may deviate so much from other normal observations, and these unnormal values are called anomalies. Anomalous data can indicate critical incidents, such as technical incidents. Anomaly detection on time series data is to identify these abnormal data points, where machine learning technique is progressively being utilized to detect anomalies automatically.

In this paper, we will discuss 2 unsupervised data mining approach, 1. statistical and 2. Matrix profile to detect the anomaly location from 250 dataset. For each approach, we will discuss the data engineering in detail, the rationale and performance of the used model and finally, compare both approach and select the best one for the prediction model which give the result of the submission file.

In addition, we will also discuss the supervised anomaly detection on KPI dataset at the final part of the report.

CCS CONCEPTS

• Computing methodologies → Data Mining;

KEYWORDS

Data Mining, Time Series, Outlier Detection, Arima, TODS.

1. Problem and Data Definition

Given the time series data, the task is to discover the location of the anomaly. Specifically, KDD Cup 2021 provides many time series data. For each one time series data, the first section is training data and is completely free of anomalies, and the second section is test data set that contains exactly one anomaly. The files use a naming convention id name split-number.txt that provides a split between test and train. For example, there will be an anomaly from 35000 onwards in 001 UCR Anomaly 35000.txt.

Some algorithms may report the location of the anomaly with different formats, such as the begin, the center, and the end location of the anomaly. In this task, we choose center location, and the submission file should contain two columns, file ID and the location of anomaly.

2. Data Engineering in details

To provide a more reliable anomaly detection over the multi dataset, ARIMA and Matrix Profile model are shortlisted as 2 options to be studied for this project. Both data engineering methodologies will be discussed in the sections and a comparison will be performed so as to attain a much better result submission.

2.1 ARIMA

The name ARIMA is a short name abbreviated from the meaning of AR - Autoregression, I - Integrated and MA – Moving Average. Detailed description as shown below [5]:

AR - Shows a changing variable that regresses on its own prior or lagged values

I - Observes the difference between static data values and previous values. The goal is to achieve stationary data that is not subject to seasonality. That means the statistical properties of the data series, such as mean, variance and autocorrelation, are constant over time.

MA – Shows is the dependency between an observed value and a residual error from a moving average model applied to previous observations

The ARIMA model is only used for non-seasonal time series data for the prediction of a given time series based on its own past values.

2.1.1 Exploratory Data Analysis and Stationery Check

The time series data used in this project is a kind of univariate data which show the relationship between one quantitative variable, we called "Data Value" in section 2, and time, we called "Location" in section 2. The study of time series data can help us to learn the future behavior of Data Value (Out Sample) based on the historical Data Value (In Sample).

2.1.1.1 Noise, Level, Trend, Seasonality

To study the behavior of time series data, it is important for us to understand the components of time series data. The reason is that some components of time series data have consistent or recurrence that can be modelled while some components of time series data do not have consistent or recurrence that cannot be modelled. Normally, time series data consists of 4 components described as below [1]:

- Level: The average value in the series.
- Trend: The increasing or decreasing value in the series.
- Seasonality: The repeating short-term cycle in the series.
- Noise: The random variation in the series.

Among the 4 components described above, all Data Value consist of Level and Noise must while Seasonality and Trend may not always exist in. So, to perform an effective outlier detection from these 250 multiset time series data, we need to identify which components will be existed in each of the multi dataset before performing the modelling process. The reason is that our target is to identify the outlier from the unseen part of each dataset, by assumption in statistic modelling, trend and seasonality are not irregular in nature which need to subtract them out before modelling to avoid the trend effects.

For those datasets which do not consist of Seasonality and Trend, we called it stationery data. Oppositely, datasets contain Seasonality and Trend are called non-stationery data. There are many ways to discover the components of time series, below section will introduce the most effective way to perform the stationery check.

2.1.1.2 Stationery Check

As mentioned, there are many ways to discover the components of time series such as statistical test approach, options shown as below [2]:

- Augmented Dickey-Fuller test (ADF Test)
- PP test
- KPSS test
- CH test
- OCSB test

Or, plotting approach, options shown as below:

- Seasonal Decompose Plot
- Autocorrelation Plot (ACF)
- Partial Autocorrelation Plot (PACF)
- Seasonal Decompose Plot

Below is one of the examples which take a dataset to study its components by plotting approach.

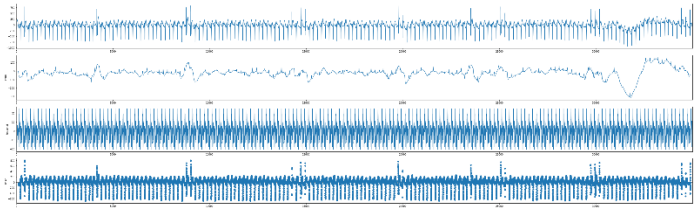

Figure 1: The Components decomposition of dataset 001_UCR_Anomaly_35000

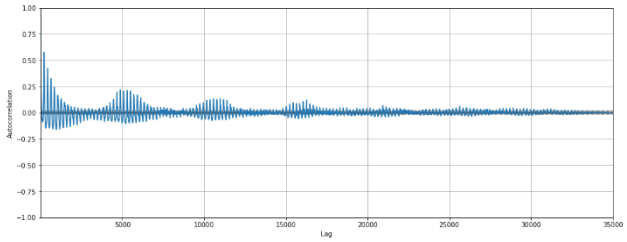

Figure 2: The Autocorrelation Plot of dataset 001_UCR_Anomaly_35000

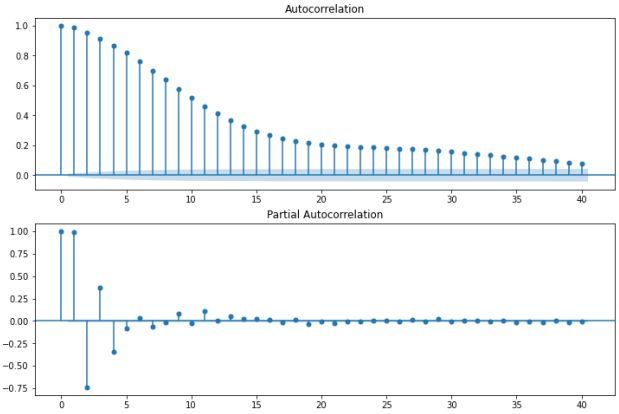

Figure 3: The ACF and PACF plot of dataset 001_UCR_Anomaly_35000

By comparing both statistical test approach and plotting approach, we decided to use statistics test approach. The reasons are that 1. Plotting approach use of eyeball test to identify the correlation between Data Value over time change, it is not always accurate. 2. The multi dataset consists of 250 datafiles which is not efficient to perform this test one by one manually.

In our data engineering approach, ADF test is used for the stationery check in this project. The reason is that ADF test is a kind of statistical significance test depends on the significance level, called p-value. By which, hypothesis testing involved with below setting:

- H_0: Null Hypothesis - p-value is less than 0.05 then the data is stationary

- H_1: Alternative Hypothesis - p-value is larger than 0.05 then the data is not stationary

ADF test involve high order of regressive process in the modelling process, below is the equation of how it works:

$$y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} .. + \phi_p \Delta Y_{t-p} + e_t$$

where

alpha: the coefficient of the first lag on Y
Y_t: the value of the time series at time t
$y_{(t-1)}$: lag 1 of time series
delta $Y_{(t-1)}$: first difference of the series at time (t-1)

To explain, because the null hypothesis assumes the α =1, which means there is regular components or unit root with the dataset, the p-value derived from the above equation should be less than the significance level (0.05) so as to reject the null hypothesis. And hence, proving that the dataset is stationary [3].

2.1.2 Transformation Time Series data from non-stationary to Stationary

There are many ways to transform the time series data from non-stationary to stationary, as shown below:

- Differencing

Differencing enables us to subtract a linear regular pattern from the time series data. Let's take an example, given the series Z(t), we create the new series:

$$Y(i) = Z(i) - Z(i-1)$$

The differenced data Y(i) will contain one less point than the original data Z(i). Depends on the data engineering needs, you can difference the data more than once [4].

- Log Transformation

Log transformation enables us to stabilize the variance of a series by using the log() function. Log() can transform the value in a manner that smaller value will get a less shrinking while larger value will get a larger shrinking. In this way, the variance of the whole time series dataset can be reduced in the same manner which could be transformed stationary.

For negative data, we can handle the case by adding a constant to bring data to positive before applying the log transformation. This constant can then be subtracted back from predicted result later.

- Seasonal Differencing

Seasonal Differencing enable us to remove these seasonal patterns from the time series data. In our case, we assign every 200 locations as a seasonal window and smooth the data by subtracting the rolling mean.

In our data engineering approach, Differencing and Log transformation are not used for transforming the multi datasets from non-stationary to stationary. The reason is

that approach cannot handle the seasonality well, which means the handle the transformation by a universal scale over the whole time series dataset. Oppositely, Seasonal Differencing by subtracting the rolling mean from the time series data can be much accurate, especially good for handling unseen data which it is not sure how the pattern or seasonality exists in them.

In our implementation, we first do the ADF test. If it is non-stationery (p-value < 0.05), we do the seasonal differencing. Else, no differencing process is required for the modelling.

```
# Assign window width for calculating
moving average
train = train.assign(seasonIndex=lambda
x: (x.index // 200))
test = test.assign(seasonIndex=lambda
x: (x.index // 200))

# Find Moving Average for each window
SeasonMean =
train.groupby(train['seasonIndex']).mea
n()
SeasonMean =
SeasonMean.rename(columns={'DataValue':
'Mean'})
SeasonMeanforTest =
test.groupby(test['seasonIndex']).mean(
)
SeasonMeanforTest =
SeasonMeanforTest.rename(columns={'Data
Value': 'Mean'})

# REMOVE SEASONALITY FROM TRAIN & TEST
BY SUBTRACTING EACH TIME SERIES DATA
FROM THE MOVING AVERAGE
train['SeasonMean'] =
train['seasonIndex']
train['SeasonMean'] =
train['SeasonMean'].map(dict(SeasonMean
)['Mean'])
train['SmoothedDataValue'] =
train['DataValue'] -
train['SeasonMean']

test['SeasonMean'] =
test['seasonIndex']
test['SeasonMean'] =
test['SeasonMean'].map(dict(SeasonMeanf
orTest)['Mean'])
test['SmoothedDataValue'] =
test['DataValue'] - test['SeasonMean']
```

Figure 4: The code segment of seasonal differencing used

2.1.3 Choosing Hyperparameter for ARIMA model

An ARIMA model has three major hyperparameters which shown below:

- AR, denoted as p, specify the number of lag observations or autoregressive terms in the model.
- I, denoted as d, specify the difference in the nonseasonal observations.
- MA, denoted as q, specify the size of the moving average window.

Again, there are many ways to identify the most suitable hyperparameters, as shown below:

- ACF and PACF plots
  Manually identify the p, d, q order from plotting result
- Grid Search
  Computationally identify the best P, D, Q order for building ARIMA model by compare AIC value contributed by each single search in training process

In our data engineering approach, since the task is to identify anomaly detect from 250 different datasets, grid search is much efficient and automatic way to handle the task. Grid search is a brute force algorithm that keep trying all the combinations of hyperparameter (P, D, Q). Every time a combination of (P, D, Q) is passed as the parameters to ARIMA model, an AIC value is returned after training and can be used as an estimator to determine the goodness of the set of hyperparameters, the lower is the AIC, the better is the hyperparameters for the specific dataset. The low AIC values suggest that these models nicely straddle the requirements of goodness-of-fit and parsimony [6].

But the disadvantage of using grid search to choose the hyperparameters is that it is highly computationally expensive and take very long time for the brute force process.

2.1.4 In Sample & Out Sample Prediction

In previous section, we discussed the identification of the best hyperparameters in the training phases. This

section we will discuss the details of performing prediction both In Sample (training data) and Out Sample (testing data).

We start the prediction by re-build the ARIMA model with the best P, D, Q order obtained before. All training data is passed to the ARIMA model with best P, D, Q order (In Sample Prediction). Then, we continue the prediction iteratively, each location per steps, to do the prediction over period of testing data (Out Sample Prediction).

For the implementation of In Sample Prediction, by using the API provided by statsmodel .tsa.statespace.sarimax, it is convenient to get the In Sample Prediction result by below API call. Below is code:

```
# TRAIN BEST UNIVARIATE MODEL WITH BEST P, D, Q order
mod = SARIMAX(train_uni,
order=best_order,
enforce_invertibility=False)
res = mod.fit(disp=False)

# Get Prediction by training data (In-Sample Prediction)
predict = res.get_prediction()
predicted_mean = predict.predicted_mean
s_predicted_mean =
predicted_mean.sort_values(ascending=False)[:50]  # Get the top maximum anomalies
```

Figure 4: The code segment of In Sample Prediction of ARIMA

For the implementation of Out Sample Prediction, we performed by getting last prediction from the training process and take this as a starting point to iteratively get the prediction of next location by using the function result.extend() and get_prediction(). Below is code:

```
point_forecast =
res.get_prediction(end=mod.nobs)
pred =
{point_forecast.predicted_mean.index[0]
: point_forecast.predicted_mean[0]}

for t, v in test_uni[:-1].iteritems():
    row = pd.Series(v, index=[t])
    res = res.extend(row)
    point_forecast =
res.get_prediction()
```

Figure 5: The code segment of Out Sample Prediction of ARIMA

2.1.5 Outlier Detection

In this project, there is no clear definition of Outlier Anomaly. So, there is no exact method to identify the anomaly. In our task, we studied 2 anomaly detection approach, one is choosing by estimating Mean Absolute Percentage Error (MAPE) which is a distance-based estimation for identifying anomaly, the second one is choosing by estimating the Z Score which is a statistical-based estimation for identifying anomaly with help of gaussian distribution. Below illustrate both approach:

- MAPE approach

This approach is simple, we identify the anomaly by determining the MAPE of each Data Value over the span of time series. The one has the largest MAPE is the anomaly. But it is not a good way for identifying anomaly in the project case since there may have so many data points in the dataset which possess the same MAPE approach. In this way, the anomaly is no longer an anomaly.

```
# Method 1: Identify the anomaly by taking the maximum MAPE loss - cannot identify if there is so many max MAPE loss value
max_anomaly_loss = max(mape_list)
anomaly_location =
mape_list.index(max_anomaly_loss)
```

Figure 6: The code segment of anomaly identification by MAPE approach

- Z-Score approach

Instead, we take the Z-Score approach which make use of gaussian distribution to identify the anomaly. Here we calculate the Z-score of each testing data by the formula as below:

Z score = [True Data Value – Mean(Predicted Value)] / Std(Predicted Value)

Z-scores can quantify the unlikeliness of an observation assume the multi dataset follow the normal distribution, it is the number of standard deviations above or below the mean that each data value falls on [7]. In this way, we can obtain the least occurrence of a data point over the dataset, which means the most deviated data point from

the mean of prediction will be identified in terms of gaussian distribution. Obviously, the highest Z-scores is the anomaly. Below illustrates the relationship of Normal Distribution of Probability Density over Z-score [8].

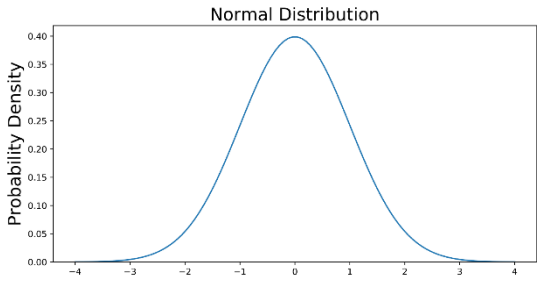

Figure 7: Normal Distribution of Probability Density over Z-score

But, using this approach should aware that determine the largest Z-score need to consider both the positive and negative side since we take the largest deviated from the mean, not the largest positive value of Z-score.

Below is implementation of Z-scores approach

```
# Method 2: Identify the anomaly by the
largest z-score
ons_z = (test_uni -
np.mean(np.array(ons_predicted_mean)))
/ np.std(np.array(ons_predicted_mean))
ons_z_max = max(ons_z.tolist())
ons_z_min = min(ons_z.tolist())  # new
add
if abs(ons_z_max -
np.mean(np.array(ons_predicted_mean)))
> abs(
        ons_z_min -
np.mean(np.array(ons_predicted_mean))):
# new add
    ons_anomaly_location =
ons_z.tolist().index(ons_z_max)
else:  # new add
    ons_anomaly_location =
ons_z.tolist().index(ons_z_min)  # new
add
f_ons_anomaly_location =
ons_anomaly_location + anomaly
print("Out-Sample Anomaly Location: ",
str(f_ons_anomaly_location))
```

Figure 8: The code segment of anomaly identification by Z-score approach

2.2 Matrix Profile

Matrix Profile is introduced in 2016, data structure for time series analysis developed by Eamonn Keogh at the University of California Riverside and Abdullah Mueen at the University of New Mexico.[9] Time series analysis mainly focused on two aspect, which are anomalies and trends. The distance profile is a vector of minimum Z-Normalized Euclidean Distances. The profile index contains the index of its first nearest neighbor. In other words, it is the location of its most similar sub-sequence. [10]. Python provides stumpy library for matrix profile computation, and therefore, stumpy will be used for computing the anomaly.

```
import stumpy
import numpy as np

if __name__ == "__main__":
    your_time_series = np.random.rand(10000)
    window_size = 50  # Approximately, how many data points might be found in a pattern

    matrix_profile = stumpy.stump(your_time_series, m=window_size)
```

Figure 9: The usage of stumpy library

2.2.1 Matrix Profile – Data Preprocessing

A sliding window m is applied. In the algorithm, In the algorithm, distance profile of the sliding window m using Z-Normalized Euclidean Distances will be computed. Therefore, the value m, the size of sliding window, is a deterministic factor for affecting the accuracy of the anomaly detection.

2.2.2 Matrix Profile – Window size m

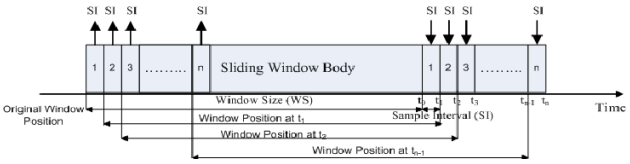

Figure 10: Sliding window algorithm

Usually, window size should be determined according to the heartbeat of the time series data, which is a loop of the data. However, in this project, the loop is not provided, and not any kinds of data are provided, but a list of arrays, and the length of array. Therefore, window size m become a difficult problem to determine how the sliding window size should be applied on each dataset. Therefore, a general approach is made, hopefully each set of data can be applicable to that particular window size.

The following are the 4 set of window slide is applied to find anomaly point

1. 640
2. 100
3. Aggregate approach
4. Human Determined Window Size

For window size 640, it is a default value in stumpy library. And therefore, this value is applied to check with anomaly point. For the generated result, please refer to file submission_20211028-640.csv

For window size 100, as the specification has an allowance of the anomaly point to be within a range of 100. Therefore, window size 100 is applied such that the anomaly time series data can be within that particular range. For the generated result, please refer to file submission_20211028-100.csv.

For the aggregate approach, as the matrix profile will compute the result for each time series window, therefore, by applying different window size, the aggregation of each matrix profile is sum up, and finally to look for the most anomaly point (which it the max distance) of all matrix profile being summed up. However, it is nearly impossible that the length of dataset of windows size being computed, as such like dataset 241, it has a large data. Therefore, an approach from AIST AI Research Center (AIRC) by Genta Yoshimura is taken [11]. By starting from a window size 40 to compute for matrix profile, and the next iteration will be a window size multiply by 1.1, until it reaches 767. For that 30 different window size, all score in matrix profile will be summed up, and therefore it will find the applicable anomaly point. For the generated result, please refer to submission_20211123-All.csv.

For the last approach, by the time of aggregate approach run, list of figures has been generated for each dataset. The following as dataset index 1 with different window size.

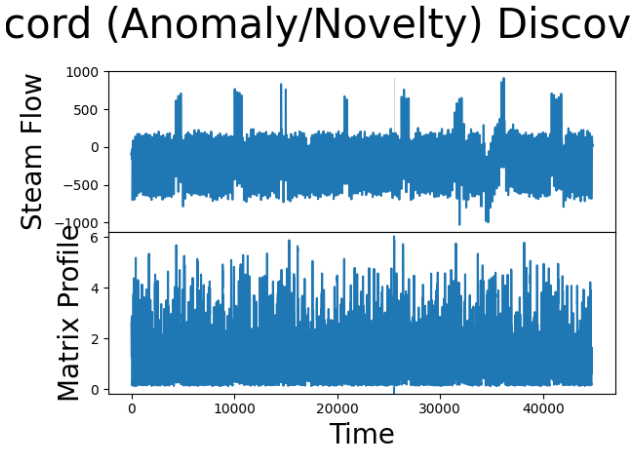

Figure 11: Index 1 with window size 40 matrix profile

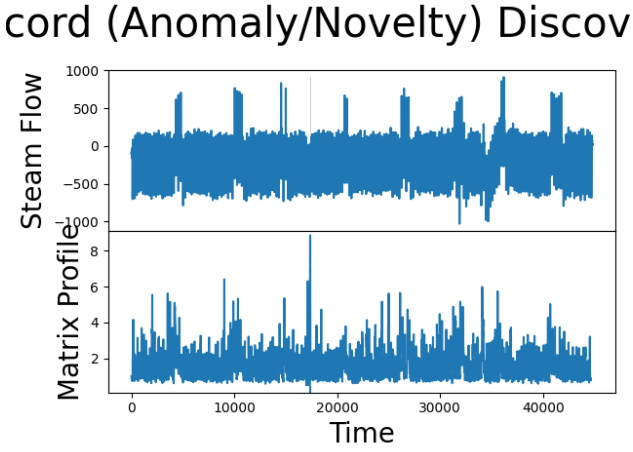

Figure 12: Index 1 with window size 127 Matrix Profile

Window size is determined by human observation, on how each heartbeat of dataset, and which anomaly position is significant. By having the human input result after reviewing all figure generated for window size in window_size.csv, we used it again as the input for Stumpy detector, which same as the first and second approach, where the window size will be retrieved from the csv itself. For the generated result, please refer to submission_user_selected.csv.

2.2.3 Matrix Profile – Anomaly Point Discovery

Matrix profile will only compute or indicate the starting anomaly index. Therefore, the index found in matrix profile will add by half of the window size, and therefore try to further approach the center of the anomaly time series.

$anomaly = max(mp[:,0]) + (m/2)$

m: Window size

mp: Matrix Profile

```
def detect(self,X, window_size=100):
    if (self.GPU == True):
        mp = stumpy.gpu_stump(X, m=window_size, device_id=self.all_gpu_devices)
    else:
        mp = stumpy.stump(X, window_size)

    discord_idx = np.argsort(mp[:, 0])[-1]
    return int(discord_idx+(window_size/2))
```

Figure 13: Code segment of detection and final computation

3. Model Performance

3.1 ARIMA

Two factors will be used to estimate the model performance, the first factor is the average of Mean Absolute Percentage Error (MAPE) over all the location of prediction [12], the second factor is by plotting based which plot the 50 maximum predicted data value and compare with the ground truth.

Let's take a dataset of 151_UCR_Anomaly_10000.txt as an example to illustrate the average MAPE of In Sample versus Out Sample. As shown in the below diagram, the yellow line shows the MAPE of each location at Out Sample location range while the blue line is the average MAPE of In Sample location range. The MAPE of each predicted data value at Out Sample range get converged and very close to the average MAPE of In Sample prediction. The average MAPE for Out Sample prediction is 0.591 while the average for In Sample prediction is 0.341.

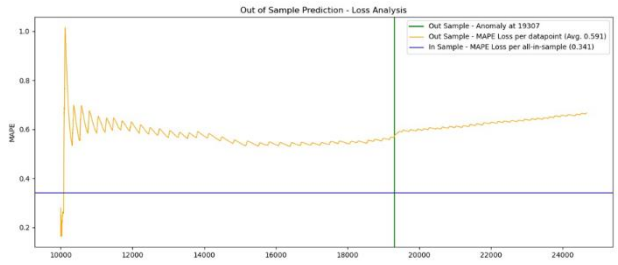

Figure 14: MAPE Comparison between In Sample and Out Sample from 151_UCR_Anomaly_10000.txt

Let's take the same dataset to analyze the prediction result by plotting the Max 50 Data Value Prediction over In Sample and Out Sample ground truth. In figure 10, we plot the prediction and anomaly location of In Sample. We can see that the black dot is the predicted result of max 50 data value, the yellow line is the ground truth and the green line is the anomaly location.

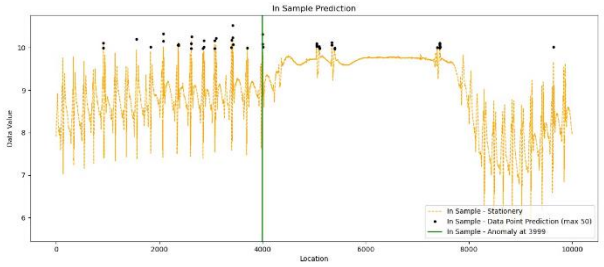

Figure 15: Plotting the Max 50 Data Value Prediction over In Sample (ground truth) from 151_UCR_Anomaly_10000

In figure 16, we plot the prediction and anomaly location of Out Sample. We can see that the yellow line is the ground truth of Out Sample, the green line is the anomaly location and the black dot is the max 50 predictions.

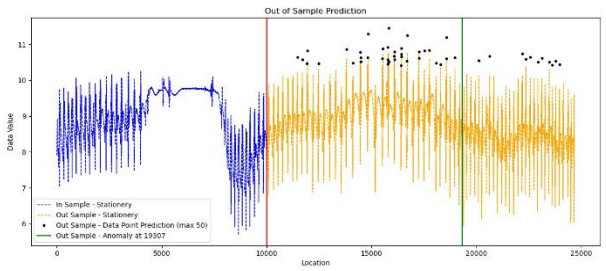

Figure 16: Plotting the Max 50 Data Value Prediction over Out Sample (ground truth) from 151_UCR_Anomaly_10000

Both illustrated the model performance is not bad. But, of course, it is only the one of the datasets, more analysis of different other datasets will be required to attain a reliable result.

4. Model Selection for Result Submission

After the above data engineering discussion, we concluded that Matrix Profile (Stumpy) will be used for result submission. The major reason is affected by "Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress" [13]. In that paper, there is a claim that this works we make a surprising claim the majority of the individual exemplars in these datasets suffer from one or more of four flaws. These flaws are triviality, unrealistic anomaly density, mislabeled ground truth and run to failure bias. Because of these four flaws, we believe that most published comparisons of anomaly detection algorithms may be unreliable. And the result that the paper claims this complex approach can be duplicated with a single line of code and a few minutes of effort. Therefore, also with our own comparison, we choose matrix as final method. Our reasons are as below:

4.1 Drawback of ARIMA

- Hyperparameter identification by Grid Search is expensive and difficult to determine the P, D, Q order for all the datasets. The reason is that different dataset has their specific behavior such as trend, seasonality etc, it is impossible include all possible combination P, D , Q order for grid search due to the hardware resources limitation.

- Seasonal differencing approach rely on a random guess of season window which used to determine the rolling mean. There is no effective method to identify the suitable seasonal window for different dataset. This

reveal that the ARIMA modelling may not be reliable in the case that the data is non-stationary.

4.2 Advantages of Matrix Profile

Matrix Profile can deduce the result in a straightforward and easy understandable way. The computation of the algorithm can be parallelized using GPU technology, such that efficiency can be easily achieved. And for most of the real-life cases, a ground truth does not exist, or clearly defined, as every single time can be an anomaly data. For different machine learning or complex model, usually the result can only be applicable for certain cases, or some kind of dataset. However, it is barely to involve all model in prediction, while matrix profile can provide one parameter, which is the window size, and easily adopt to different dataset.

5. Bonus Task – Supervised Learning

5.1 Problem and Data Definition

In this part of the paper, we are going to investigate the KPI dataset provided by the AIOps challenge. The AIOps challenge. The KPI dataset is collected from many real-world internet companies. The KPI dataset is a time-series dataset as it has a timestamp field for recording when the event occurs and a value field for that event.

The aim of the problem is to classify if there is any anomaly value in the KPI dataset. The dataset provides a label field as ground truth for indicating whether the corresponding record is an anomaly. As this is a supervised learning problem, we will try to use the XGboost classifier to classify the anomaly record from the KPI dataset.

5.2 Data Engineering in details

5.2.1 Model

The KPI dataset provided two files, one is for training purposes, and one is for testing purposes. The dataset had four fields, namely timestamp, value, label and KPI ID. As we are going to classify anomaly values from this time-series dataset, the timestamp and value are the two most important fields for our supervised learning model. The timestamp field the time-series feature to the value field. This gives the characteristics to data namely, observed, trend, seasonal and residual characteristics which enables us to create some statistical features from the historical data of the value field [14]. Those created statistical fields will be used as input features for the XGboost classifier.

5.2.2 Data pre-processing

Firstly, we use the numpy to read the training and testing dataset. The dataset was sorted by timestamp column to ensure the data is in ascending order of timestamp because of the time-series property. The value field column was extracted as a separate data frame. The data frame was normalized with a minmax scaler to ensure data are varied on comparable scales which enhance the efficiency of our classification. The below figures fig.15 and fig.16 showed the training and testing data before and after performing the normalization.

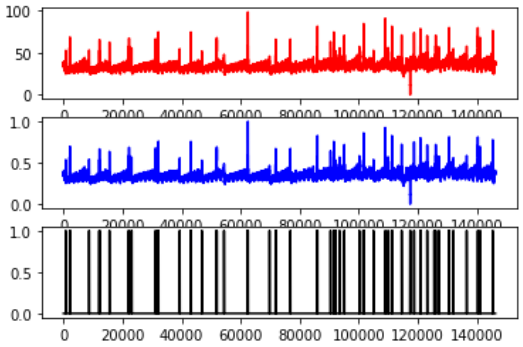

Fig 17. Training data before/after normalization

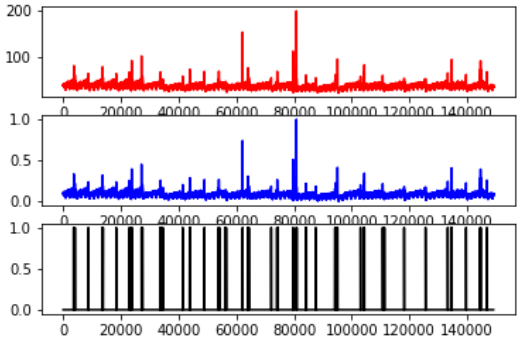

Fig 18. Testing data before/ after normalization.

5.2.3 Feature engineering

As there is only one value field useful from the provided dataset, we need to create more fields as input features for supervised learning. Based on the time-series nature of the data, several fields related with the statistical means and variances of different time windows are created as input features for supervised learning.

5.2.4 Building time series of different window sizes

The time-series data can be viewed as composed of a sequence of time-series data with different window sizes. Window size means how many values are within this time series. Each value in the series can be a member of

multiple sequences of other time-series at the same time. The number of windows and its sizes is unknown and dependent on the trend and seasonal effect of the data itself. In this paper, the following number of window sizes, namely 25, 50, 100, 200, 300, 400, 500, 600 are selected after several trial runs.

5.2.5 Differencing the time series

The differencing technique was used to transform the time series data into learnable features for supervised learning [14]. In this paper, we only calculate the difference between time series t and its previous time series t-1. The following differencing values of each different window sizes are calculated.

Name	formula
natural logarithm of the value	Log(value)
difference between value at t and t1	(value t) - (value t-1)
percentage difference between values at t and t -1	((value t) - (value t-1))/ ((value t-1) + 1e-10)
mean of values in window	np.mean(t)
variance of values in window	np.sum(np.square(t - np.mean(t))

5.2.6 XGBoost classifier

In this paper, the XGboost classifier was selected for the supervised classification of anomaly. The XGboost classifier is the hottest classifier in the market. It implements the gradient boosting decision tree algorithm. It is fast and accurate. [15]

5.3 Model Performance

5.3.1 Training and testing

Both the training and testing datasets are transformed with new features. Then the training dataset is fitted into the XGboost classifier with n_estimators=100 to train the classifier.

The accuracy of both training and testing, and the classification report is as below,

Train Accuracy: 0.9855
Test Accuracy: 0.9629

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	2829649
1	0.12	0.15	0.13	54398
accuracy			0.96	2884047
macro avg	0.55	0.56	0.56	2884047
weighted avg	0.97	0.96	0.97	2884047

The classification error and log loss graph are shown as fig 19 below.

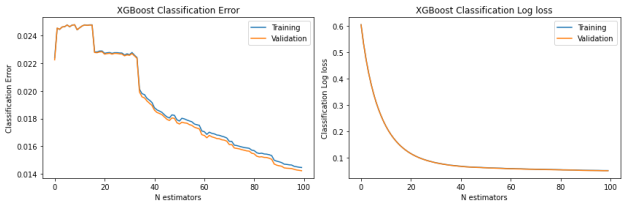

Fig. 19 Classification error and log loss graph

6. Conclusion

Anomaly means some random data points which do not have the same behaviour as the majority data. Obviously, there is no clear definition about anomaly such as distance deviation, error distribution deviation and etc. In our project, we studied two different unsupervised methodologies to detect anomaly, one is Stumpy which is a distance-based approach and another one is ARIMA, which is a statistical based approach.

For ARIMA, we detect the anomaly by we taking the Z-Score approach which make use of gaussian distribution to identify the anomaly from the data. In recent study, the performance by statistical approach is far better than several machine learning model [16]. But, this may not be a good approach for solving multiple datasets, such as 250 datasets in this project, due to its requirement of expensive computation resources. In this project, we takes over 15 days to process all the datasets and provide 250 anomaly location.

For matrix profile, we detect anomaly by calculating the distance. There is no ground truth needed and only parameter needed for the algorithm is sliding window size. A normal way for determining window size is by having the heartbeat of the time series data, however, the information is not provided in this project. Therefore, four ways (two constant approach, one aggregate

approach, and one human computer interaction approach) were made so as to find anomaly. The most difficult part was finding a good window size for detection.

7. Future works

From this paper, we learned that no there was no single machine learning model could solve all the anomaly detection problems due to the fact that each dataset has their own features and characteristics based on different data sources, such as intrusion detection, fraud detection, system health monitoring and etc.

Future works will be on trying to analyze the anomaly detection problems with a combination of different machine learning models. We will improve our machine learning and optimization model

ACKNOWLEDGMENTS

Thanks to Professor Chen Lei, Jackson and all Teaching Assistant of The Hong Kong University of Science and Technology for the support of this project.

REFERENCES

[1] Machine Learning Mastery 2017. How to Decompose Time Series Data into Trend and Seasonality from https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/

[2] Toward Data Science 2020. 7 Statistical Tests to validate and help to fit ARIMA model from https://towardsdatascience.com/7-statistical-tests-to-validate-and-help-to-fit-arima-model-33c5853e2e93

[3] Machine Learning Plus 2019. Augmented Dickey Fuller Test (ADF Test) – Must Read Guide from https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/

[4] Finance Train 2021. Transforming a Series to Stationery from https://financetrain.com/transforming-a-series-to-stationary

[5] Master's in Data Science 2021. What is Arima Modeling from https://www.mastersindatascience.org/learning/what-is-arima-modeling/

[6] CoolStatsBlog 2013. Using AIC to Test ARIMA Models from https://coolstatsblog.com/2013/08/14/using-aic-to-test-arima-models-2/

[7] Statistics By Jim 2021. 5 Ways to Find Outliers in Your Data from https://statisticsbyjim.com/basics/outliers/

[8] Toward Data Science 2018. How to Use and Create a Z-Table (Standard Normal Table) from https://towardsdatascience.com/how-to-use-and-create-a-z-table-standard-normal-table-240e21f36e53

[9] All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, Eamonn Keogh (2016). IEEE ICDM 2016.

[10] Marrs, T. (2020, February 28). Introduction to matrix profiles. Medium. Retrieved November 23, 2021, from https://towardsdatascience.com/introduction-to-matrix-profiles-5568f3375d90.

[11] 産総研人工知能研究センターairc. (2021, August 27). 機械学習研究チームの吉村玄太 特定集中専門研究員が、27th ACM SIGKDD conference on knowledge discovery and data mining（kdd 2021）コンペティション（kddcup）の multi-dataset time series Anomaly detection 部門で、5 位（全 176 参加チーム中）に入賞しました Retrieved November 23, 2021, from https://twitter.com/AIResearchAIST/status/1431115026568335361.

[12] Machine Learning Plus 2021. ARIMA Model – Complete Guide to Time Series Forecasting in Python from https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/

[13] Wu, R., & Keogh, E. (2021). Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. IEEE Transactions on Knowledge and Data Engineering

[14] Jason Brownlee, How to Remove Trends and Seasonality with a Difference Transform in Python, https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/

[15] XGBoost Documentation, https://xgboost.readthedocs.io/en/stable/

[16] Machine Learning Mastery 2018. Comparing Classical and Machine Learning Algorithms for Time Series Forecasting from https://machinelearningmastery.com/findings-comparing-classical-and-machine-learning-methods-for-time-series-forecasting/