IL2233-Project Report
Time-Series Prediction and Anomaly Detection

Conglei Xiang

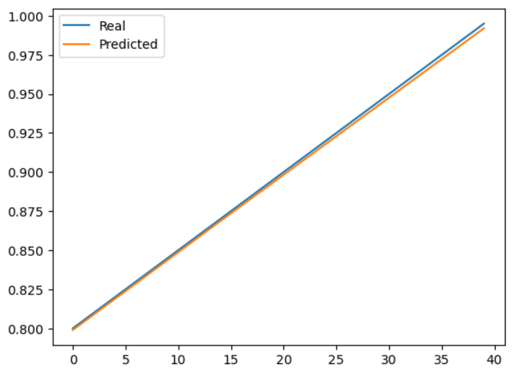
July 6, 2023

1 Task 1. Time-series prediction with neural networks

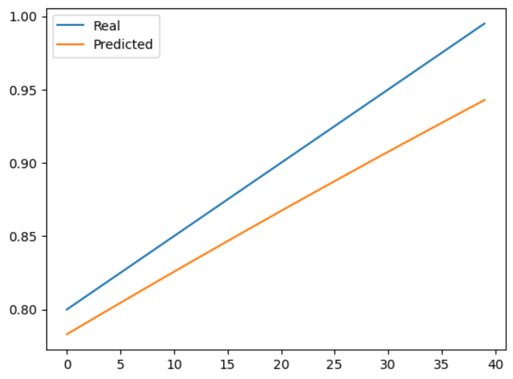
1.1 Prediction with synthetic series using MLP, RNN, and LSTM Dataset and Task

1. An equal-difference series starting from 0, ending to 1 (excluding 1), with a length of 200 points (step = 0.005).
   Design an MLP for one-step prediction. The output vector has a size of 1. Let the input vector be a size of 4.

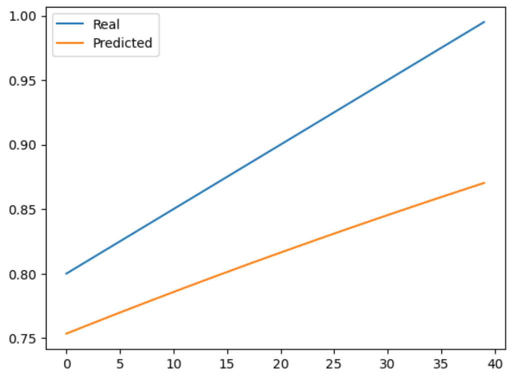
   The activation function sigmoid and tanh doesn't fits well in this section when only change the activation function, the result are shown in figure1:
   'Relu' and epochs=50, max absolute error is 0.0842



(a) Relu



(b) sigmoid



(c) tanh

Figure 1: Different activation function

'Relu' and epochs=100, max absolute error is 3.006e-3
'Relu' and epochs=300, max absolute error is 5.41e-5

Choose epochs=100 because of it has a balance between good accuracy and short time.

Hyper	Loss function	Optimizer	Layer	Activation function
	MSE	Adam	2	Relu + Linear

Table 1: Hyper parameters used in this section

2. An equal-difference series starting from 0, ending to 1, with a length of 200 points (step = 0.005), plus white noise i.e., random variable with zero mean and 1 variance. You may need to control the amplitude of the noise series in order to control the signal-noise ratio.
   Design an MLP for one-step prediction. The output vector has a size of 1. Let the input vector be a size of 4.

   No matter how much I increase the epochs and number of units, the MSE are always similar.And when epochs=1000, the output seems overfit.

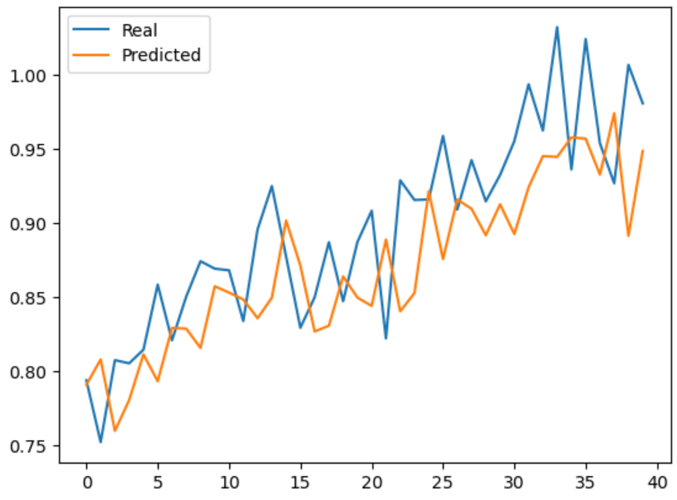


Figure 2: Prediction results when $\sigma = 0.001$, $epochs = 100$

Hyper	Loss function	Optimizer	Layer	Activation function
	MSE	Adam	2	Relu + Linear

Table 2: Hyper parameters used in this section

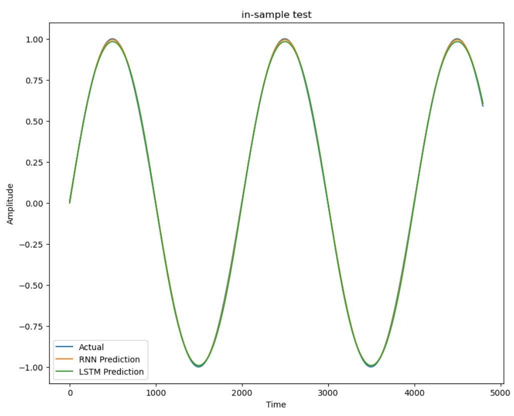
3. A deterministic series sampled from a sinusoidal wave with period 20 seconds, with a sample rate of 100 Hz. Generate sufficient samples (at least 3 periods of data) as needed to achieve good performance, e.g. MSE (mean squared error) below 0.5. Design an RNN and a LSTM for two-step prediction. The output vector has a size of 2. Set the input vector size by yourself.

   The hyper parameters used in RNN and LSTM are all number of units=64, optimizer='adam', loss='mse'. The epochs is chosen to be 1 because it can achieve good accuracy in this situation(Figure2). The MSE values are presented in table3.

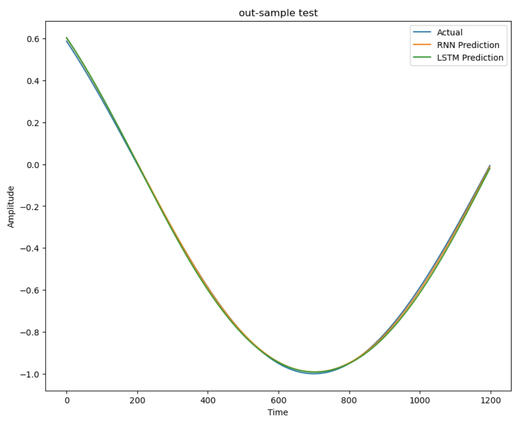
RNN performs a little better than LSTM

LSTM	RNN
$2.13e-4$	$8.09e-5$

Table 3: MSE of RNN and LSTM



(a) In-sample prediction



(b) Out-sample prediction

Figure 3: Prediction result when epochs=100 and variance=0.001

4. A stochastic series sampled from a sinusoidal wave with period 20 seconds, with a sample rate of 100 Hz, plus random white noise i.e., random variable with zero mean and 1 variance. Control the amplitude of the noise with a fractional number, e.g. 0.1.
   Design an RNN and a LSTM for two-step prediction. The output vector has a size of 2. Set the input vector size by yourself.

Hyper	Loss function	Optimizer	Layer
	MSE	Adam	2

Table 4: Hyper parameters used in this section

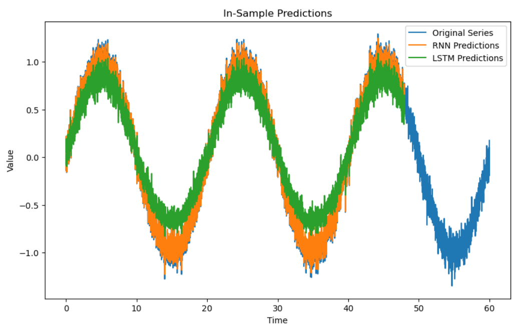
The epochs is chosen to be 1 because it can achieve good accuracy in this situation.

The input size is set to be 10 because the prediction plot illustrates that the performance of accuracy (Figure4 & 5) is much better in 10 input size. They all reach $MSE < 0.5$

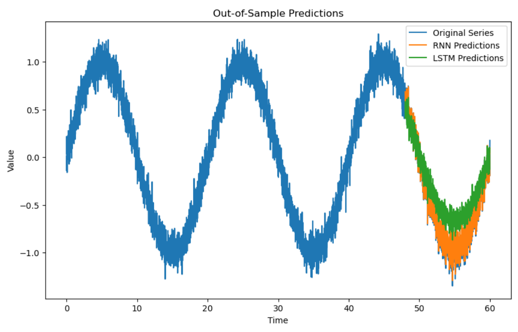
1.1.1 Discussion

1. How have you designed the neural network for each series? What hyper-parameters do you use in each case?

   Analysed in each subtask.

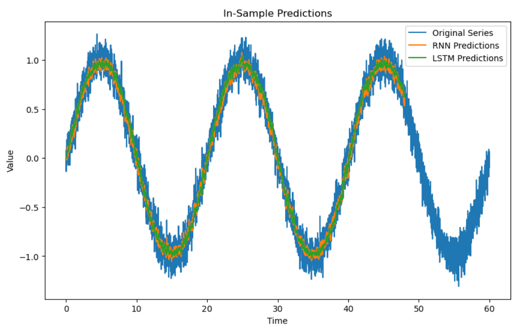


(a) In sample test

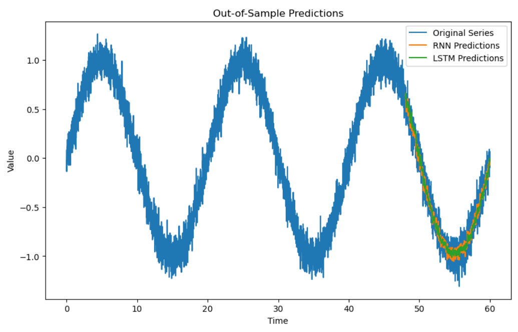


(b) Out sample test

Figure 4: Prediction results when input size is 1



(a) In sample test



(b) Out sample test

Figure 5: Prediction results when input size is 10

2. Can the neural network fit well to the specific series well? What are the accuracy merits?

   In some cases they can fit well, for example, in question 1,3. But the MSE in question 2,4 shows that neural networks cannot fit well to the white noise signal because the MSE are always close to the σ^2 no matter what parameters are changed to make it more accurate.

   White noise is inherently unpredictable and lacks any specific information that a neural network can effectively capture. As a result, trying to fit white noise with a neural network is unlikely to yield meaningful results.

3. How is the performance of LSTM in comparison with RNN? Is the LSTM outperforming the RNN in general?

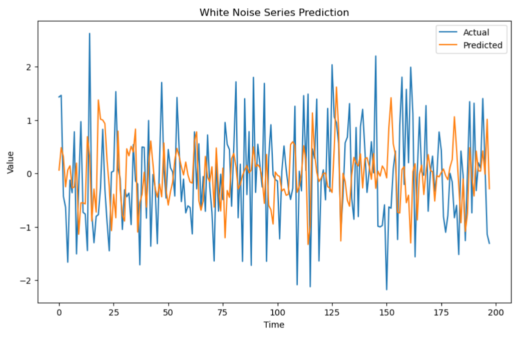
   RNN is better than LSTM due to my results of MSE in case 3, but table 5 shows that LSTM is better than RNN. In my opinion, the reason is that case3 is more simple than case4. This means RNN is sufficient to deal with the task because it has shorter dependencies and the amount of available training data is limited.

LSTM	RNN
0.01175	0.01227

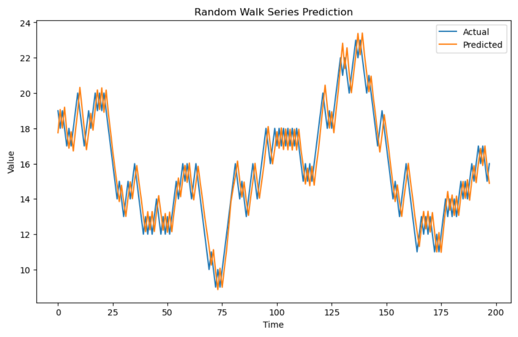
Table 5: MSE of RNN and LSTM in question 4

1.2 Predict white noise, random walk, an ARMA process using neural networks

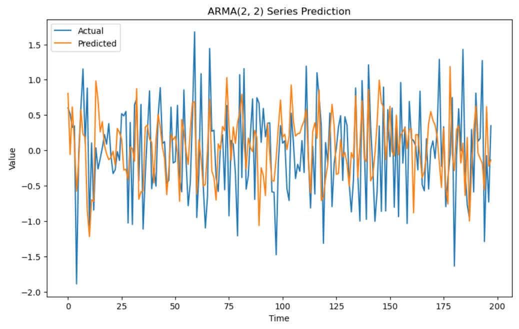
The fitting results are shown in Figure6



(a) White noise



(b) Random walk



(c) ARMA

Figure 6: Prediction results of different datasets with LSTM

The fitting results are similar when using RNN and LSTM. The accuracy and effectiveness can't be directly identified with visual figures.

The MSE results are shown in table7

1.2.1 Discussion

1. How have you designed the neural network? What hyper-parameters do you choose?

Dataset	LSTM	RNN	MLP
white noise	1.003	1.140	1.060
random walk	1.064	1.006	1.108
ARMA	0.380	0.474	0.491

Table 6: MSE of prediction in different dataset

It is shown is table7, when using MLP, I set activation function as 'relu' because it has the best accuracy.

Hyper	Loss function	Optimizer	Layer	epochs	batchsize
	MSE	Adam	2	100	32

Table 7: Hyper parameters used in this section

2. Can the neural network fit well to the white noise series?

   No, it can't.

3. Can the neural network fit well to the random walk series?

   It can just fit well to the trend of the random walk series

4. Can the neural network fit well to the ARMA process? Why or Why not?

   Yes. Because the data from table 7 indicates that the MSE are all less than 0.5 when using different neural networks.

1.3 Comparison with ARIMA-based modeling and prediction

1. The accuracy evaluation of four methods

   When the noise ratio is set to be 5%, the (p, d, q) of ARIMA is set to be (1,2,1) because the ADF test shows that the p-value of the original data is 1.0 which means it is not stationary. After a 2-order differencing, the p-value is 1.59e-14. Thus d=2. From the ACF and PACF plots shown in Figure 7, set p and q to be 1. When the noise ratio is set to be 10%, the (p, d,

Method	MSE	MAE	MAPE
MLP	0.107	0.158	0.016
RNN	0.003	0.042	0.004
LSTM	0.003	0.046	0.005
ARIMA	0.001	0.028	1.500

Table 8: MSE, MAE, MAPE of the prediction errors when noise ratio= 5% sing a length=10 out of sample test series

q) of ARIMA is set to be (1,1,1) because the ADF test shows that the p-value of the original data is 0.924 which means it is not stationary. After a 1-order differencing, the p-value is 2.33e-15. Thus d=1. From the ACF and PACF plots shown in Figure 8, set p and q to be 1.

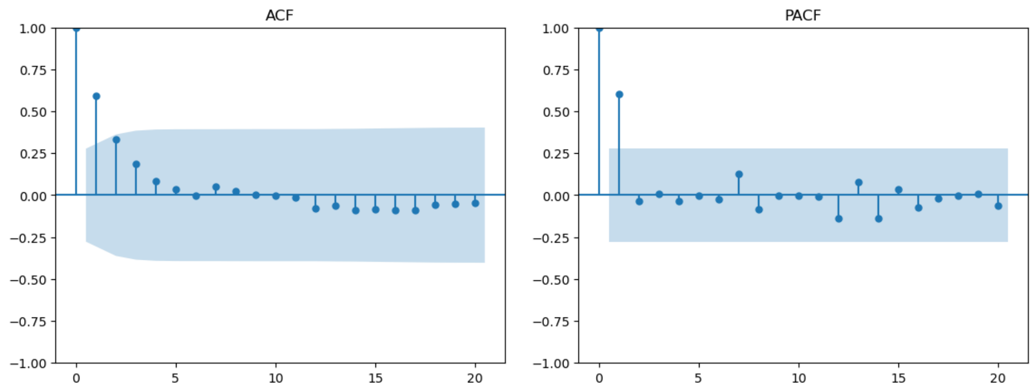


Figure 7: ACF and PACF of the Fibonacci series noise ratio=5%

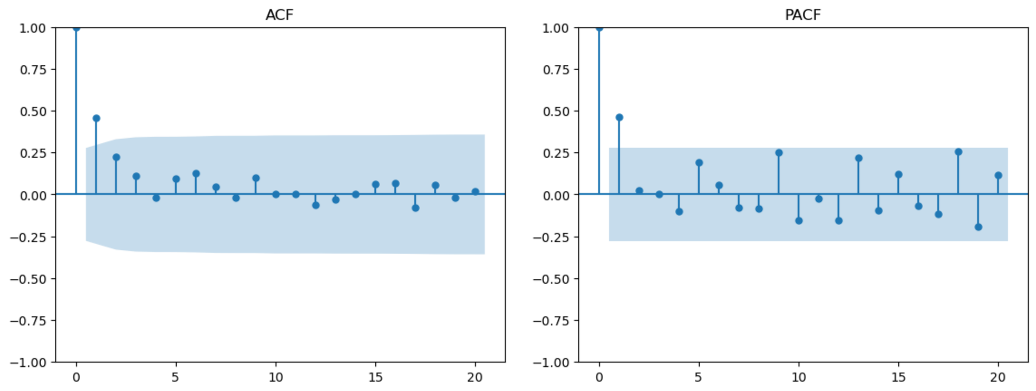


Figure 8: ACF and PACF of the Fibonacci series when noise ratio=10%

1.3.1 Discussion

1. How have you designed the neural networks? What hyper-parameters do you choose?

   Presented in table10

2. How have you trained your neural networks?

   I transformed the Fibonacci series into a set of data with a more average increasing slope in order to train the neural networks and to reach a higher accuracy. I randomly separate the training and the testing dataset because the Fibonacci data aren't uniformly distributed.

3. Can your MLP, RNN, LSTM networks fit well to the Fibonacci series? Which one is best?

   Yes, they can. I think RNN is the best one which is a bit better than LSTM, but these 2 are all much better than MLP approach.

4. Can your ARIMA model fit well to the Fibonacci series?

   No, it can't.

Method	MSE	MAE	MAPE
MLP	0.184	0.266	0.027
RNN	0.004	0.051	0.005
LSTM	0.005	0.057	0.006
ARIMA	0.002	0.027	0.797

Table 9: MSE, MAE, MAPE of the prediction errors noise ratio=10% using a length=10 out of sample test series

Hyper	Loss function	Optimizer	Layer	epochs	batchsize
	MSE	Adam	4	500	32

Table 10: Hyper parameters used in this section

5. Which modeling approach, neural network based or ARIMA based, gives a better performance? Why? Discuss the pros and cons of different modeling approaches.

   I think neural network is better. Because it has the ability to capture complex patterns, and it can handle various types of data and can be trained on a wide range of problems.

   However, neural networks typically require a large amount of training data to effectively learn complex patterns. If the Fibonacci series dataset is small, the neural network may struggle to generalize well and may overfit or underfit the data.

   ARIMA models provide statistical properties, such as confidence intervals and significance tests, which can be useful for interpreting the results and understanding the uncertainty in predictions.

   As for ARIMA's cons, they have limited memory of past observations.

6. Which model, ARIMA or NN, is more tolerant to noise?

   In general, neural network models tend to be more tolerant to noise compared to ARIMA models. The values from table 8, 9 can verify this.

2 Task 2. Decomposition-based anomaly detection

2.1 Procedure

- Use a decomposition method, e.g. the classic decomposition or STL, to split the components of the series. The classic decomposition method is used to split the series
- Visualize the components.
- Set a threshold, and find the time index and value of the anomalies. The threshold is set to be 3.
- Mark and visualize the anomalies in the series.

The decomposition plot is shown in Figure9, and the anomaly is labeled as the red point which is inserted by the code "series.loc["1998-12-1"] = 20". To highlight better the anomaly, I chose only a part of the data, from 1995-01-01 to 2000-01-01 (61 data points).

The anomaly is identified by the z-score function. I set the threshold of z-score as 3. The resulting z-score represents the number of standard deviations the data point is away from the mean. It can be positive or negative, indicating whether the data point is above or below the mean, respectively. A z-score of 0 indicates that the data point is equal to the mean.

```
#set the threshold
threshold = 3
#calculate z-score
z_score = np.abs((residual - residual.mean()) / residual.std())
#find the location of the anomaly
ano_location = np.where(z_score>threshold)[0]
```

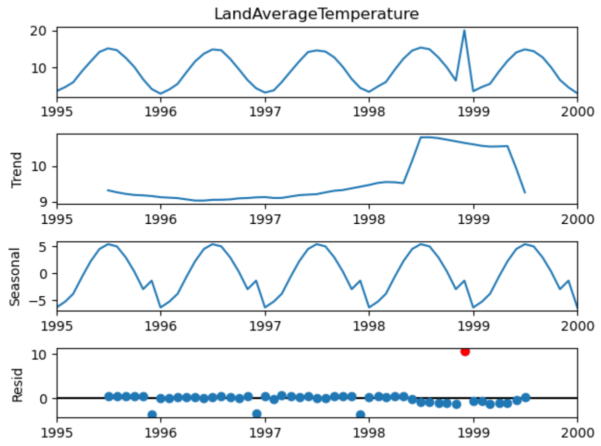


Figure 9: Decomposition result and anomalies

Use the box-plot function to identify the outliers, the box-plot is in Figure10:

```
q1=dataset.quantile(0.25)
q3=dataset.quantile(0.75)
iqr = q3 - q1
lower_threshold = q1 - 1.5 * iqr
upper_threshold = q3 + 1.5 * iqr
print(lower_threshold,upper_threshold)
anomaly_loc = np.where((dataset[:]<lower_threshold)|
    (dataset[:]>upper_threshold))[0]
anomaly = dataset.index[anomaly_loc]
plt.boxplot(sample_temp['LandAverageTemperature'],showfliers=True)
plt.show()
```

From the box-plot, it can be observed that there is no outliers identified with box-plot function because the output of lower_threshold,upper_threshold are -6.7115 and 25.9165. The inserted

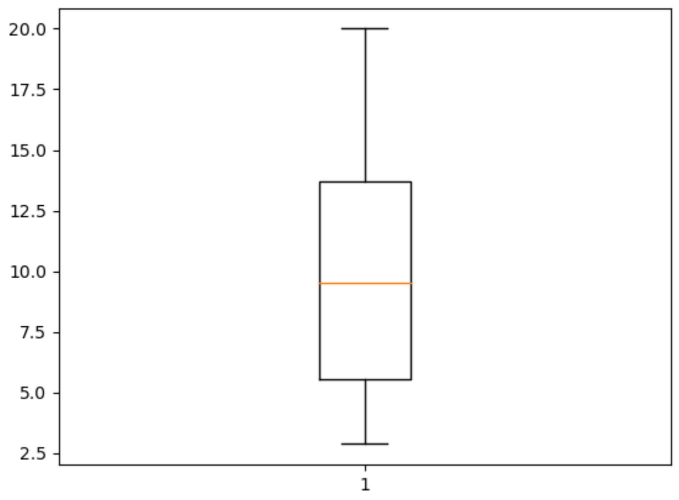


Figure 10: Box plot result

anomaly which is 20 is in the normal range.

The reason why the box-plot can't detect the outliers that the z-score approach can detect is that, boxplot detects outliers based on the IQR and uses a fixed threshold of 1.5 times the IQR to determine outliers. Any data points outside this range are considered outliers. Boxplot is effective in identifying outliers that deviate significantly from the central tendency of the data. z-score takes into account the distribution of the entire dataset and calculates the deviation from the mean, whereas boxplot focuses on the quartiles and uses a fixed threshold.

2.2 Discussion

1. Can the decomposition clearly separate the trend, season (constant period), and remainder components?

   From Figure11(the decomposition plot contains 392 sample data), it can be observed that the method can clearly separate season, but the trend is not that clearly. Probably because noisy or irregularly sampled data may introduce additional challenges in accurately identifying and separating the components.

2. When decomposing the series, is there a general rule to determine which part belongs to a trend, a season, or a remainder? Or is it embedded in and thus dependent on each individual algorithm?

   No. Different algorithms may have different rules or assumptions for identifying and separating these components.

   Generally, the trend component represents the long-term direction or overall pattern of the time series; the seasonality component captures regular, repeating patterns that occur within

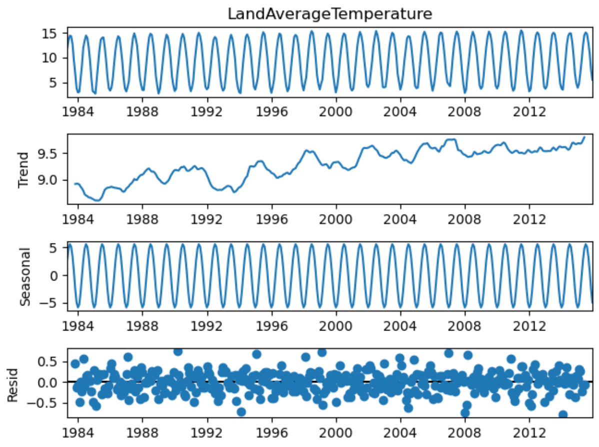


Figure 11: Decomposition result with more sample data

a fixed period; the residual component contains the random or unexplained fluctuations that cannot be attributed to the trend or seasonality.

3. Is there a growing tendency in the trend series?

   From Figure11, it is obvious that there is a growing tendency.

3 Task 3. Prediction-based anomaly detection

3.1 Task 3.1 Anomaly detection for uni-variate series with ARIMA

3.1.1 Procedure

1. Exploratory data analysis. Draw seven graphs: line plot, histogram, density plot, heatmap, box plot, lag-1 plot, and lag-2 plot for the series.

   Shown in Figure12

2. If the series is not stationary, differencing the series, until it is stationary, but not over-differenced.

   The series is not stationary at first, and after 1-order differencing it becomes stationary. I used ADFuller approach to test if it is stationary. Before differencing, the ADF result is:

```
ADF Statistic: 0.9382926419378802
p-value: 0.993570249862259
Critical Values:
1%: -3.479007355368944
5%: -2.8828782366015093
10%: -2.5781488587564603
The series is non-stationary.
```

(p, d, q)	AIC
(1,1,1)	-66.608
(1,1,3)	-86.174
(2,1,1)	-69.719
(2,1,3)	-89.477

Table 11: Results between (p, d, q) values and AIC

After 1-order differencing, the ADF result is:

```
ADF Statistic: -12.165503286659842
p-value: 1.4604869372130724e-22
Critical Values:
1%: -3.479007355368944
5%: -2.8828782366015093
10%: -2.5781488587564603
The series is stationary.
```

3. Summarize the statistical features of the series, such as mean, standard deviation. Plot the temporal correlation of the series by drawing its acf and pacf graphs.

   The output result is :

```
before differencing:
Mean:  Temperature    0.090638
dtype: float64
Standard Deviation:  Temperature    0.535909
dtype: float64
after differencing:
Mean:  Temperature    0.014929
dtype: float64
Standard Deviation:  Temperature    0.199584
dtype: float64
```

   Here are the ACF and PACF plots in Figure13

4. Model construction and selection. Follow the Box-Jenkins methodology step by step to construct an ARIMA(p, d, q) process to model the series.

   Base on the ACF of the differenced series, I choose p = 1 or 2. Base on the PACF of the differenced series, I choose q = 1 or 3. I choose d = 1 due to the first order differencing.

   I try different combinations (p, d, q) values and select an optimal model based on AIC. Base on the results shown in Table11, (2,1,3) is chosen to build ARIMA model.

5. In-sample prediction

   The results are shown in Figure14. The error are calculated and generate a plot in Figure15. It seems to be random.

   Then do the Ljung-Box test, the result illustrates the error series is random because $p\_value > 0.05$

6. Identify anomalies from the remainder series. If the difference is larger than the deviation threshold, this data point is marked as anomaly.

   Z-score: Set the anomaly ratio as 2%. Thus, I got $threshold = 2.48$. The anomalies are identified by z-score calculation and they are '2016-01-01', '2019-01-01', '2020-01-01'.

   Box-plot: The normal value range is $-1.27 < Temperature < 1.29$. The anomalies are identified by box-plot calculation and they are '2015-01-01', '2016-01-01', '2017-01-01', '2019-01-01', '2020-01-01', not the same as z-score but contains the anomalies identified by z-score.

3.1.2 Discussion

1. Since this task uses a different approach (prediction-based anomaly detection) from Task 2 which uses decomposition for anomaly detection, describe what the differences of the two methods are?

   ARIMA models focus on capturing temporal dependencies and forecasting, while decomposition methods aim to separate the time series into interpretable components. Both approaches can be used for anomaly detection.

   Anomaly detection with ARIMA models involves forecasting the future values based on the historical data and comparing the predicted values with the actual values.Decomposition methods can identify anomalies by examining the remainder component.

2. Do they achieve the same results? Why or Why not?

   Not exactly the same because of their different modeling approaches, assumptions, and treatment of components.

3. Given the anomaly ratio of 2%, what is the value of z-score?

   The value of z-score threshold should be 2.48. Greater than that are the anomaly points, while less are the normal points.

3.2 Task 3.2 Anomaly detection in ECG signals with LSTM

3.2.1 Procedure

1. Exploratory data analysis. Draw the line plot, lag-1 plot, and lag-2 plot for the two signals in the data set.

   The Figure16 is data analysis of MLII. The Figure17 is data analysis of V5.

2. Model construction. Split the data set into a training set (80%) and a test set (20%). Design and train an LSTM which can predict the next value given a few known values.

   Use the function:

```
def data_split(series, input_size, output_size,train_ratio):
    input = []
    output = []
    for i in range(len(series) - input_size - output_size + 1):
        input.append(series[i:i+input_size])
        output.append(series[i+input_size:i+input_size+output_size])
    input = np.array(input)
    output = np.array(output)
    train_size = int(train_ratio * len(input))
    input_train, input_test = input[:train_size], input[train_size:]
    output_train, output_test = output[:train_size], output[train_size:]
    return input_train, input_test, output_train, output_test
```

3. Validate the quality of the model. Calculate the MSE, MAPE etc. of the prediction model for the input size being 4, 8 and 16.

N	4	8	16
MLII	$1.75e-4$	$1.49e-4$	$1.51e-4$
V5	$2.68e-4$	$2.07e-4$	$1.97e-4$
bi-variate	$2.30e-2$	$2.33e-2$	$2.40e-2$

Table 12: MSE results

N	4	8	16
MLII	$3.50e-2$	$3.24e-2$	$3.14e-2$
V5	$5.70e-2$	$4.78e-2$	$5.48e-2$
bi-variate	$3.06e-1$	$3.11e-1$	$2.99e-1$

Table 13: MAPE results

The data in the table shows that the accuracy increases with the increasement of vector size. The performance doesn't improve a lot, but the time used in training becomes much longer.

4. Anomaly definition and detection. Calculate the prediction error series.

Assume 0.5% of error rate, find the anomaly points. Take N=8 as the analyze example. The anomaly points location are:

MLII: [60 368 369 372 1219 1494 1500 1782 2098 2403 2404 2406 2987 3260 3264 3268 3497 3543 3833 3845 3847 4139]

V5: [66 367 373 658 1225 1507 1510 2385 2386 2397 2402 2405 2945 2946 2956 2957 2959 2973 2979 3265 3832 3841]

bi-variate: [69 70 71 367 368 369 370 371 659 939 940 1226 1508 1792 1793 2099 2100 2101 2102 2404 2405 2406 2407 2701 2702 2983 2984 2985 3265 3266 3267 3268 3548 3549 3550 3551 3842 3843 3844 3845 4134 4135 4136 4137]

The results of anomalies are not same when using uni-variate and bi-variate.

5. Visualize the anomaly points in the prediction error series.

Take N=8 as an example, the error series are shown in Figure 18

3.2.2 Discussion

1. How have you designed the neural network? What hyper-parameters do you choose? Why?

   I used LSTM model.

   ```
   def lstm_build(input_shape):
       model = Sequential()
       model.add(LSTM(64,activation='relu', input_shape=input_shape))
       model.add((Dense(input_shape[-1])))
       model.compile(loss='mean_squared_error', optimizer='adam')
       return model
   ```

   The hyper-parameters chosen are shown in the above code. The parameter 64 refers to the number of LSTM units or memory cells in the LSTM layer. The number of units determines the complexity and capacity of the LSTM model. Choosing a higher number of units allows the model to capture more intricate patterns in the data, and the data amount of ECG series is large, so I chose 64. ReLU is a commonly used activation function in deep learning models due to its simplicity and effectiveness.

2. Can the neural network fit well to the ECG signals? What is the accuracy of your model?

   It reaches different accuracy with different approach. Using uni-variate is much better than bi-variate. The MSE and MAPE are shown in table12 and 13

3. How much is the influence of the input vector size on the prediction accuracy?

   Analyzed in part 3.2.1-question 3.

4. How many epochs do you use for training your LSTMs in order to achieve good accuracy? How much is the learning rate? What is your training optimization algorithm (e.g. SGD, Adam etc.)?

   I use epochs=100. The training optimization algorithm is Adam which has a default learning rate of 0.001 and I didn't modify that.

5. Which way of treating the time series data gives better accuracy: two uni-variate series or one bi-variate series? Why?

   Uni-variate is better. Maybe because these two signals are mutually independent. Treating them as separate uni-variate series allows the model to capture their own specific dependencies and patterns. By considering each lead independently, the model can better learn the temporal dynamics and specific correlations between the lead and the target variable, leading to improved accuracy. Also the complexity of calculation is reduced a lot. Thus with the same parameters, the uni ones get better accuracy.

4 Task 4. Clustering-based Anomaly detection

4.1 Procedure

1. Visualize the multi-variate series, plotting the line plot and scatter plot.

   The plot are shown in Figure19, 20,

2. Determine the number of clusters, and do clustering on the data.

   I set the number of cluster as 2. The cluster results by K-means are shown in Figure21 and 22,and the results by SOM are shown in Figure23 and 24

3. Calculate the distance between each point and its nearest centroid (the centroid of its belonging cluster).

   Function used to calculate distance in K-means:

   ```
   # calculate distance between each point and its nearest centroid
   distances = kmeans.transform(data) #Euclidean Distance with centroid
   ```

   Function used to calculate distance in SOM:

   ```
   # Euclidean distance between each data point and its
   # quantized representation
   distance_som = np.linalg.norm(data - som.quantization(data), axis=1)
   ```

4. Use the outlier ratio to calculate the total number of anomalous points (outliers). According to the point-wise distance, those points with largest distances are considered to be the outliers. This step leads naturally to set the distance threshold. By this step, the data points are split into a normal subset and an anomalous subset.

   The anomaly ratio is 2%, so I got 4 anomaly in the dataset of 200. The results of anomaly locations are the same using K-means and SOM: [7 28 69 111]

5. Visualize anomalies in a cluster view using the scatter plot (for 2D, 3D data), and in a time series view using the line plot. The results by K-means are shown in Figure21 and 22,and the results by SOM are shown in Figure23 and 24. The anomalies are labeled out with red points.

4.2 Discussion

1. How do you set the number of clusters? Why?

   I set the number of clusters as 2 because I think from the scatter plot, there is no obvious clusters and the points all distributes in an average way, but may be the two variable may influence the clustering.

2. Which distance metric do you use? Are there other distance metrics which might be useful for this task?

   I used Euclidean distance metric, which measures the straight-line distance between two points in Euclidean space. Some alternative distance metrics include:

   Manhattan distance (also known as city block distance or L1 norm): It measures the sum of the absolute differences between the coordinates of two points.

   Cosine distance: It measures the cosine of the angle between two vectors and is often used for text or high-dimensional data. For this task, it is not a good choice.

   Mahalanobis distance: It accounts for the covariance structure of the data and is suitable for dealing with correlated features.

Minkowski distance: It is a generalization of both the Euclidean and Manhattan distances, where the power parameter determines the type of distance metric.

3. Do the two different clustering methods (K-means and SOM) achieve the same results? Discuss why or why not.

   The results show that these 2 methods achieve the different clustering results, but seems really similar because they randomly initialize the centroid and even different time to execute the same cluster with the same dataset will lead to a different result. but the anomalies found in the 2 methods are the same. Because

   (1) Both K-means and SOM operate on the same input data, which is typically a set of features or variables describing each data point. As long as the input data and its representation are consistent across the methods, they have the potential to identify similar patterns and anomalies.

   (2)Both K-means and SOM rely on same distance metrics (Euclidean) to measure the similarity between data points.

   (3) They both attempt to partition the data into clusters, with each cluster representing a distinct pattern or behavior. Anomalies, by definition, deviate from the normal patterns, and both methods can identify such deviations as points that do not fit well into any cluster.

5 Task 5. Summary

5.1 Discussion between NN and ARIMA

1. Describe the two approaches in your own words.

   A neural network is a computational model inspired by the structure and function of the human brain. It consists of interconnected nodes called neurons that work together to process and learn from input data.

   ARIMA is a statistical modeling technique used for time series analysis and forecasting. It is based on the assumption that future values of a time series are a linear combination of past observations and past forecast errors.

2. Compare the two approaches and discuss their strength and weakness.

   Strengths of NN are its ability of Non-linear modeling, its flexibility to deal with many types of input, and its scalability to add more layers in the process. Weaknesses are its high computational complexity, and hard to interpret and hard to determine the hyper parameters.

   As for ARIMA, it's good because it is interpretable and well-suited for tasks involving trend, seasonality, and autocorrelation. But it has assumptions and limitations regarding linearity and memory.

3. List key points of the two approaches and key pros and cons.

   How modeling & prediction are done?

   ARIMA: Data Analysis -> Data Preprocessing -> Model Identification -> Model Estimation -> Model Diagnostic Checking -> Model Forecasting

NN: Data Preparation -> Model Architecture Design -> Model Training -> Model Evaluation -> Model Testing and Prediction

5.2 Discussion of anomaly detection methods

1. Describe the three different methods.

   Decomposition-based: Anomalies are then detected by analyzing the residuals or the difference between the observed values and the estimated values based on the decomposed components.

   Prediction-based: It predicts future values of the time series based on historical data and then compare the predicted values with the actual observations. When the difference is significant, the anomalies are detected.

   Clustering-based: Clustering-based anomaly detection approaches aim to group similar patterns or behaviors in the time series data and identify data points or clusters that deviate significantly from the majority.

2. Pros of the 3 methods

   Decomposition-based: Effective in detecting anomalies that exhibit patterns or deviations from expected trends and seasonal variations.

   Prediction-based: Can detect anomalies in real-time or near real-time by continuously updating the predictions based on new observations.

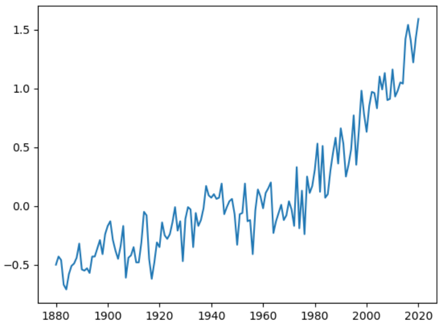
   Clustering-based: Can handle complex data structures and capture anomalies that are not well-defined or follow non-linear patterns.It is suitable for unsupervised anomaly detection.

3. Cons of the 3 methods

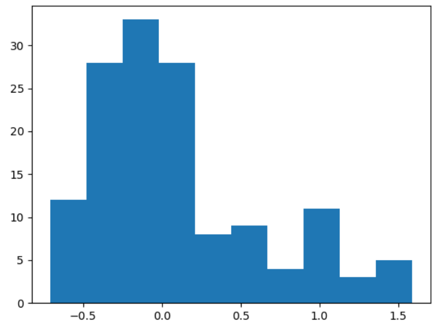
   Decomposition-based: Limited effectiveness in detecting anomalies that do not exhibit clear patterns or trends. Maybe sensitive to noise or outliers in the data.

   Prediction-based: Performance heavily depends on the quality of the prediction model and the availability of sufficient training data.

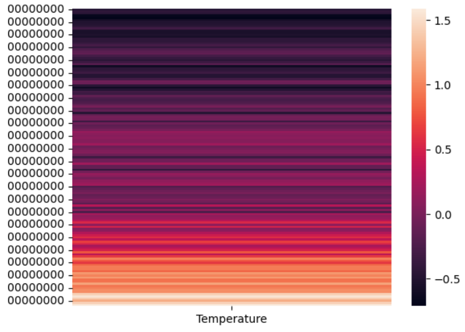
   Clustering-based: May struggle with detecting anomalies in high-dimensional or sparse data.



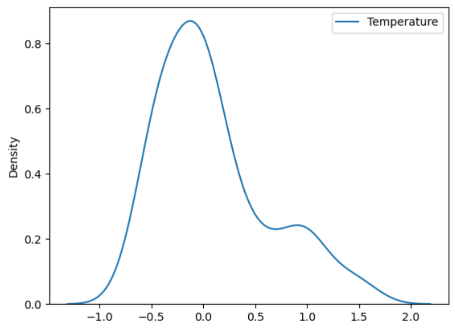
(a) Line plot



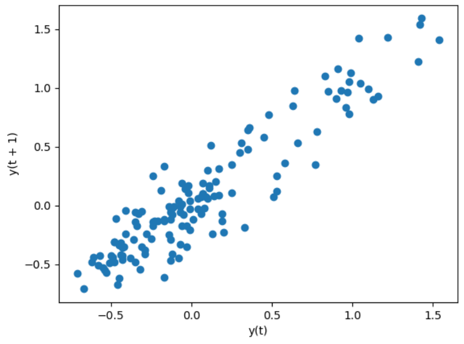
(b) Histogram



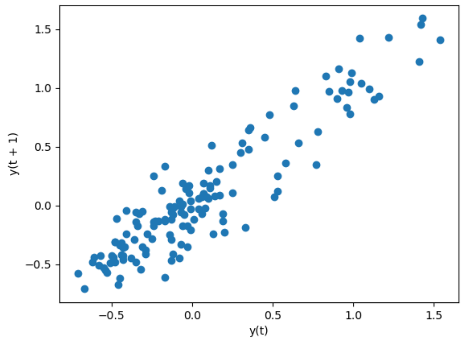
(c) Heatmap



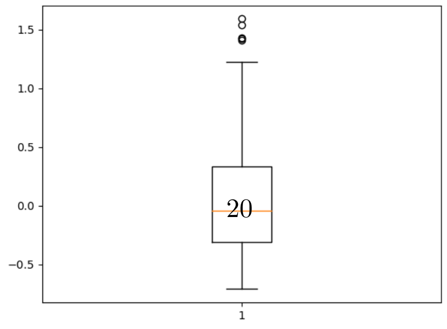
(d) Density plot



(e) Lag-1 plot

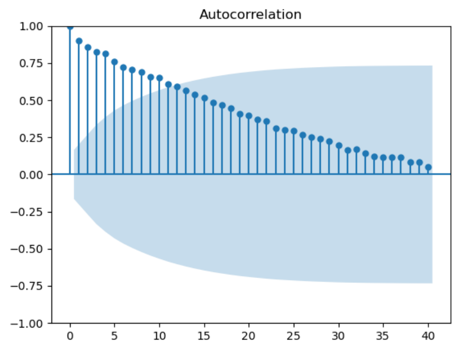


(f) Lag-2 plot

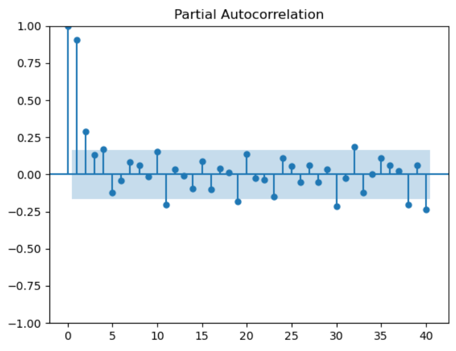


(g) Box plot

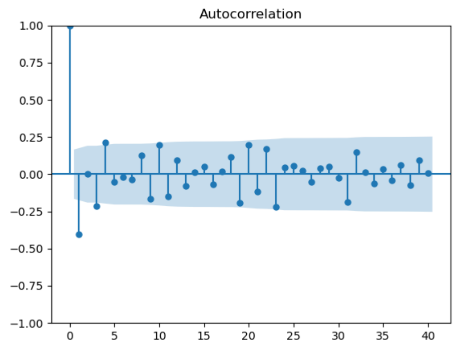
Figure 12: Data analysis graphs



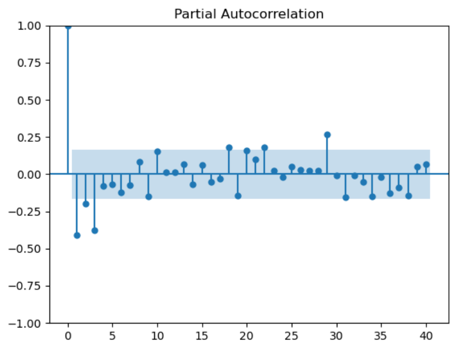
(a) ACF before differencing



(b) PACF before differencing



(c) ACF after differencing



(d) PACF after differencing

Figure 13: ACF and PACF plots of the series before and after differencing

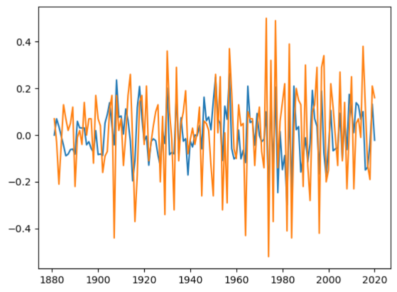


Figure 14: Predicted and actual series

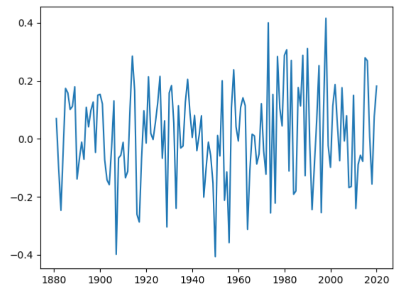
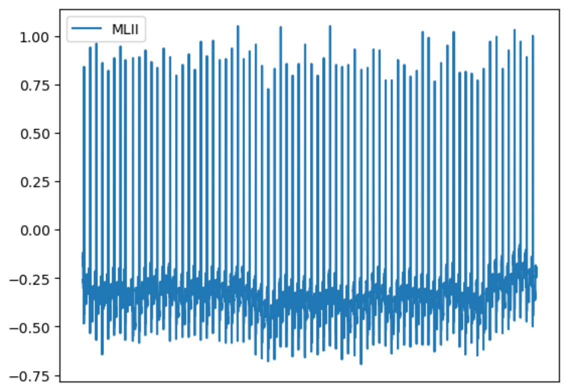
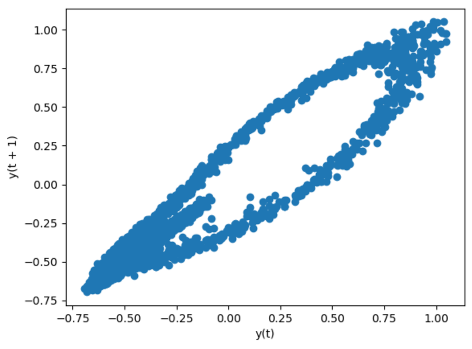


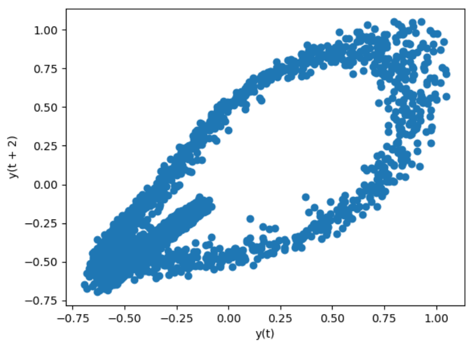
Figure 15: The results of error between prediction and actual series



(a) MLII line plot

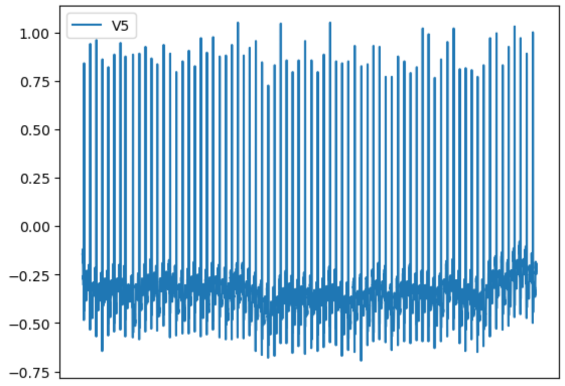


(b) MLII lag-1 plot

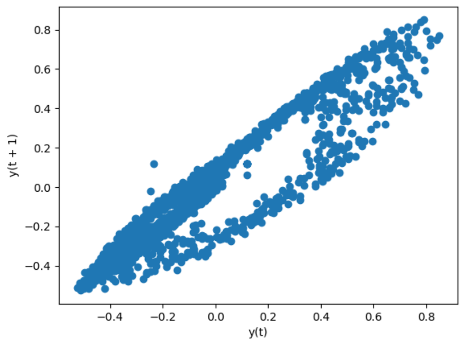


(c) MLII lag-2 plot

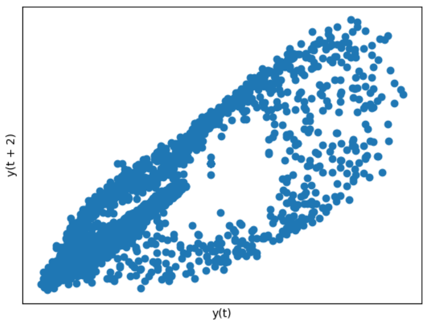
Figure 16: MLII data analysis



(a) V5 line plot

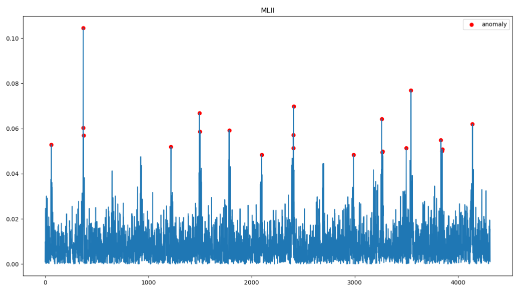


(b) V5 lag-1 plot

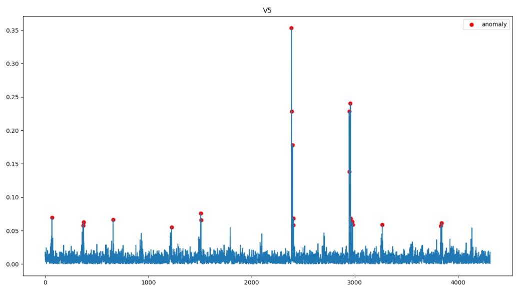


(c) V5 lag-2 plot

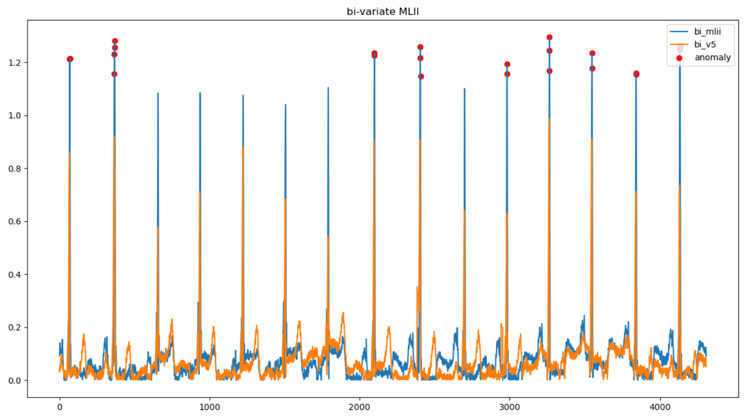
Figure 17: V5 data analysis



(a) Anomalies and error series by MLII



(b) Anomalies and error series by V5



(c) Anomalies and error series by bi-variate

Figure 18: Anomalies and error series using different ways

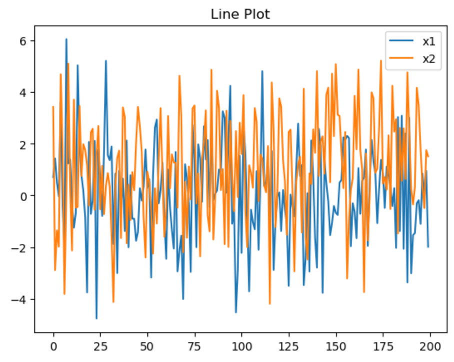


Figure 19: Line Plot

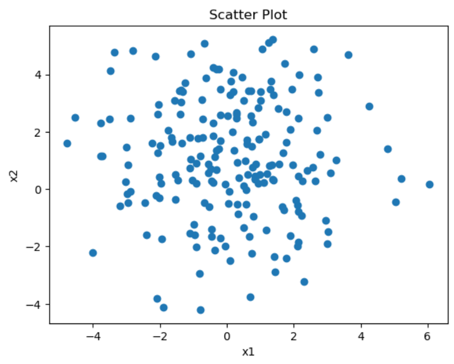


Figure 20: Scatter plot

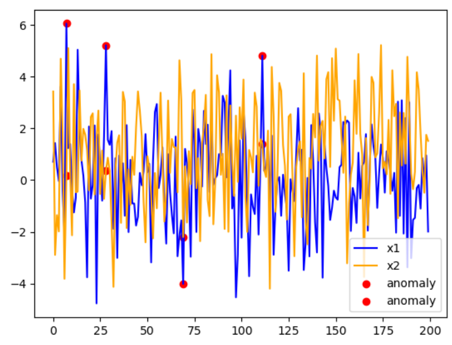


Figure 21: Line Plot by K-means

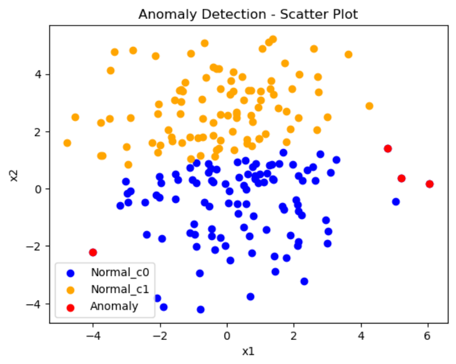


Figure 22: Scatter plot by K-means

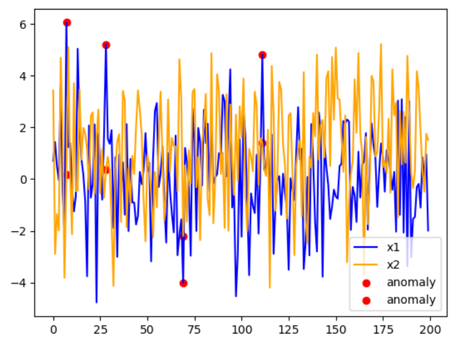


Figure 23: Line Plot by SOM

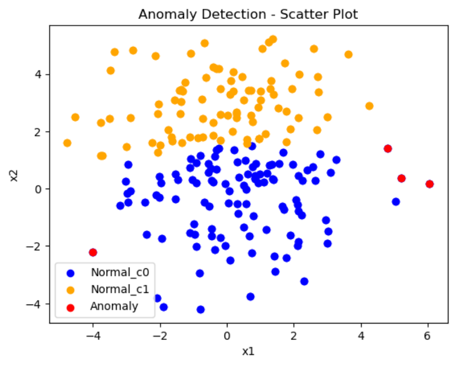


Figure 24: Scatter plot by SOM