# Lab Project

## IL2233 - Embedded Intelligence

Axel Lindberg

# 1 Task 1: Time-series prediction with neural network

## 1.1 Task 1 part 1 prediction with synthetic series using MLP, RNN and LSTM

This part consists of generating the following uni-variate series, and then using neural networks to do in-sample and out-of-sample predictions.

### 1.1.1 Part 1.1

Creating an equal-difference series starting from 0 to 1 with 200 steps.

An MLP created for it with two layers and was tested with the test data using for input and one output with the tensorflow keras models evaluate function to calculate the MSE (mean square error) to be 1.1808862332429726e-08 which is a satifying result for this neural network

When predicting the next value of the input [0.02, 0.025, 0.03, 0.035] it predicted the value 0.03977244 which is close to the real value of 0.04. Looking at Figure 1 we can see the prediction compared to the original values.

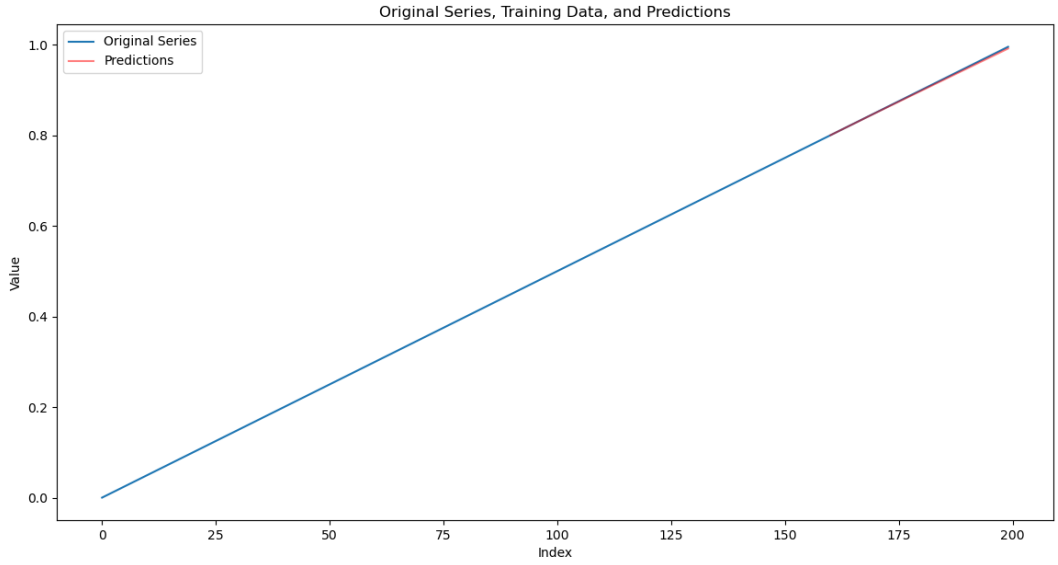



Figure 1: The figure shows the MLP predicting the last part of the series with satisfying accuracy

### 1.1.2 Part 1.2

Now we make 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.
With these new premises the result is expected to get worse which can be seen in the MSE test loss which is 0.01328169833868742 for the testing data. This is considerably worse than the result in part 1.1. The prediction can be compared to the actual values in Figure 2. It still find the trend and predicts that the changes are going to be unpredictable.

### 1.1.3 part1.3

This part starts with creating a deterministic series sampled from a sinusoidal wave with period 20 seconds, with a sample rate of 100 Hz. This using 3 periods to ensure a low MSE below 0.5.

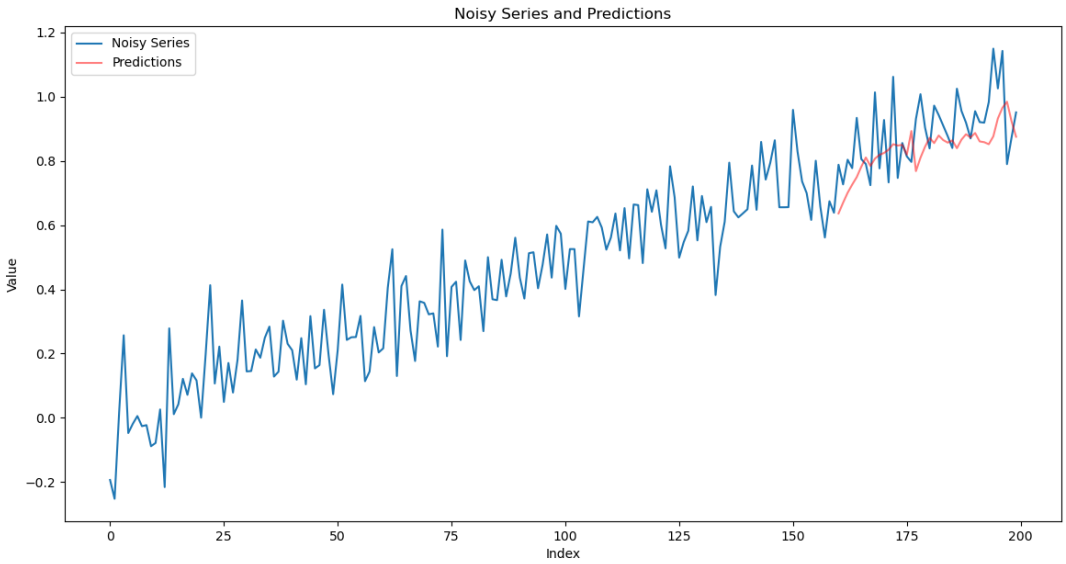



Figure 2: The figure shows the MLP predicting the last part of the series with a less satisfying accuracy

The performance of these models where in terms of MSE using the test data for RNN: 6.892473265907029e-06 and for LSTM: 5.087117074253911e-07. These values are considerably less than 0.5 which means the result is satisfying enough. In the Figure 3 the predictions for the last 20% of the graph can be seen.

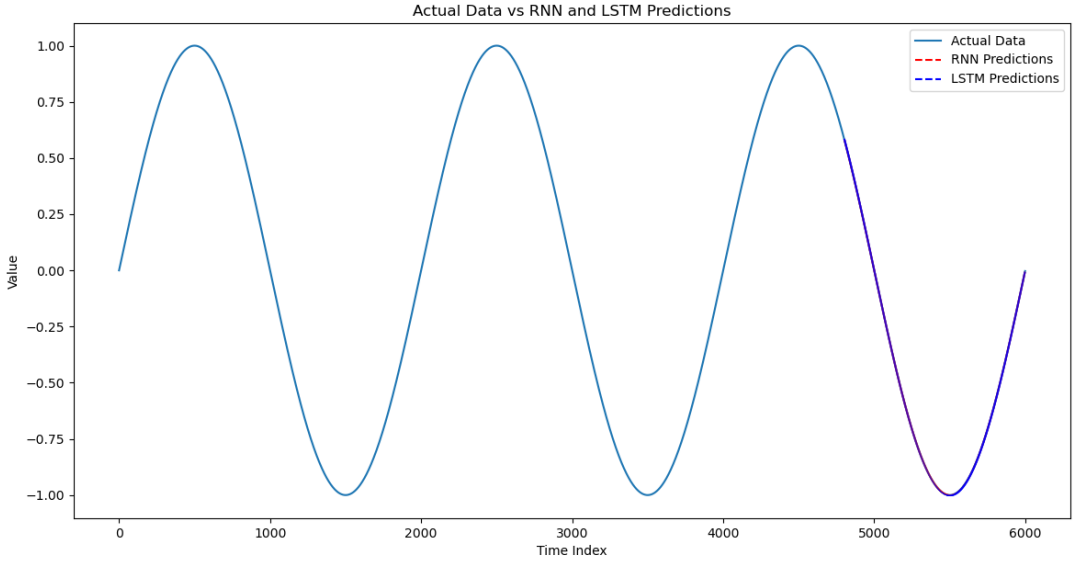



Figure 3: The figure shows the RNN and LSTM accurately predicting the upcoming values

### 1.1.4 Part 1.4

For this task we create 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. The amplitude is modified with the factor 0.1 as it was deemed an appropriate disturbance for the amplitude of the sinus wave. The MSE test loss was for RNN: 0.012212269008159637 and for LSTM 0.011694038286805153 which are significanly higher than in part 1.3 but still within the 0.5 specification of that part. The Figure 4 show the predictions predict the noise to be smaller than it actually is.

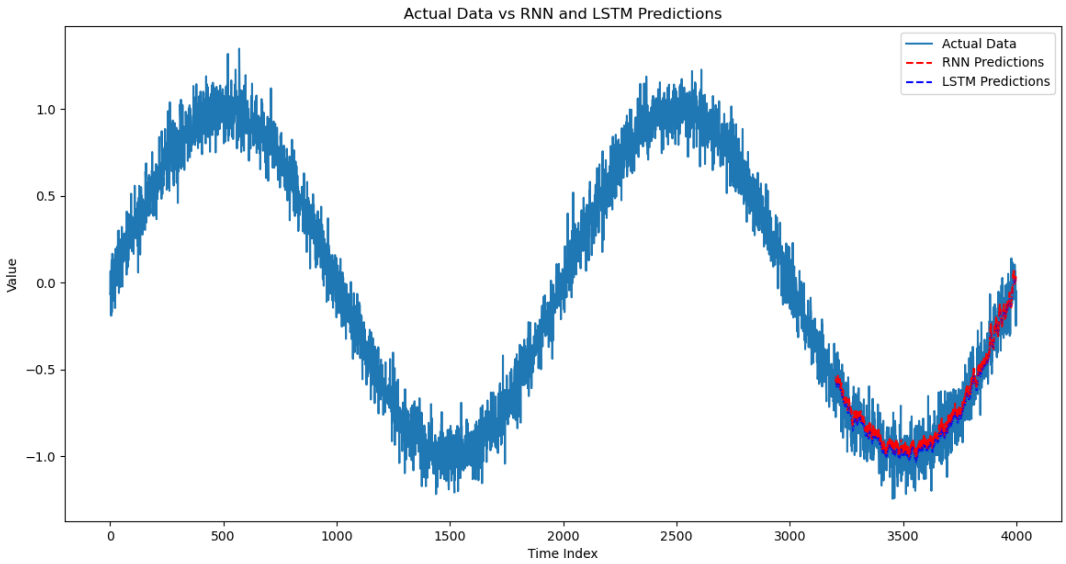



Figure 4: The figure shows the RNN and LSTM somewhat accurately predicting the upcoming values

#### 1.1.5 Discussion

How have you designed the neural network for each series?:

The neural networks were design in a pretty simple fashion using the TensorFlow framework. For the MLPs I used two Dense layers with 64 layers each. 64 Felt like good place to start and due to in giving out satisfying results i decided to keep it. For the RNN i used one RNN layer provided by the Tensorflow frame network with 50 units and the a Dense layer with two units to give a two step prediction. For the LSTM i used one LSTM layer with 50 unit provided by the Tensorflow framework and then one dense layer with two units to have a two step output.

What hyper-parameters do you use in each case?

For the first and second case: The epoch was 100, batch size 32, optimizer "adam", loss measurement: MSE, input dimension: 4, layers: 2, units per layer: 64, test split: 0.2.
For the third and fourth case RNN: The epoch was 100, batch size 32, optimizer "adam", loss measurement: MSE, input dimension: 4, RNN layer units: 50, test split 0.2
For the third and fourth case LSTM: The epoch was 100, batch size 32, optimizer "adam", loss measurement: MSE, input dimension: 4, LSTM layer units: 50, test split 0.2

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

Yes a neural network can be better fitted to a specific series depending on whats needs to be predicted and the training data available, layers can be varied and different models can be used. The metrics for accuracy used in these cases were using MSE loss metrics to calculate the accuracy and a plot prediction to compare the predicted results to the actual ones. One could also test the network on new data to see how it performs there.

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

In general the results suggests the LSTM is better at the tasks at hand and performing significantly better than the RNN. But training the LSTM network takes significantly longer than training the RNN network.

### 1.2 Task1 part 2 Predict white noise, random walk, an ARMA process using neural networks

I decided to use the RNN to test how it holds up to predict these different signals that have some form of randomness to them

#### 1.2.1 White noise

The first on test white noise that was made with a normal distribution with the mean being 0 and the variance being 1. The MSE test loss for this model and test was 1.4904602766036987 which considering the variance isn't a great result but it also difficult to predict white noise. In the Figure 5 the prediction can be seen and in Figure 6 the prediction errors can be seen forming what appears to be white noise.

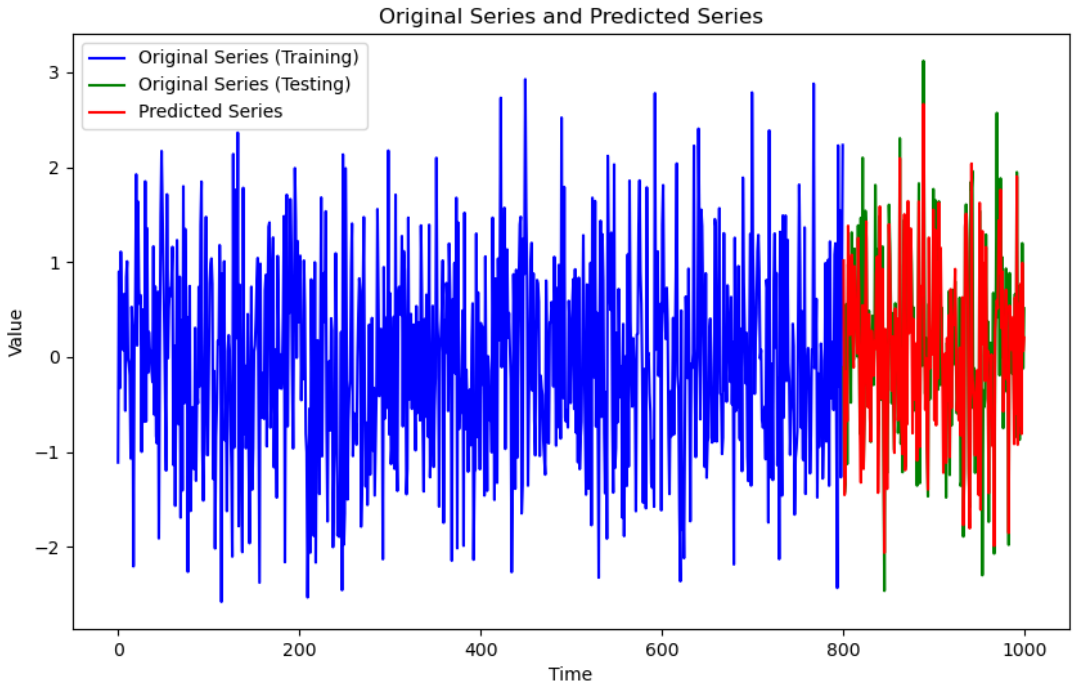



Figure 5: The RNN models attempt at predicting the tests white noise

#### 1.2.2 randomwalk

The first on test random walk with the step length being 1. The MSE test loss for this model and test was 1.0993973016738892 which isn't a great result but it isn't a bad one either. In the Figure 7 the prediction can be seen and in Figure 8 the prediction errors can be seen forming what appears to be white noise.

#### 1.2.3 arma(2,2)

The ARMA model was made using the AR coefficients: 1, -0.75 and 0.5 and the MA coefficients: 1, 0.5 and 0.3, to ensure a stable system. The MSE loss for this model was 1.0478651523590088 which matches up with the results from the previous tests. In Figure 9 we can se the prediction compared to the test and in Figure 10 we can see the prediction errors which appears to be white noise.

#### 1.2.4 Discussion

How have you designed the neural network?
I have designed a simple one step RNN neural network with one 64 unit layer, this making a simple neural network to test its effectiveness with the different input signals.
What hyper-parameters do you choose? Give short motivation.
The hyper parameter I choose were: epoch: 100: because its large but will not take to long to train, batch size 8 because the inputs were gonna be a bit more random, layers 1 to keep it simple, RNN units 64 becuase its a dectly big number for a random predictor, test split of 0.2 as the parts above this.
Can the neural network fit well to the white noise series?
No its cant really fit well onto a white noise series due to it being random which means that the network would have to be able to predict randomness.
Can the neural network fit well to the random walk series?
In the short term sense a neural network could figure out the next possible steps that a random walk could take but it would not be able to predict the next step consistently due to its random nature.
Can the neural network fit well to the ARMA process? Why or Why not?
The neural network could be able to fit the ARMA process, dues to the ARMA process having a simple structure and predictable behaviour once it starts to process signals.

### 1.3 Task 1 part 3 comparison with arima base modeling, fibonacci

The result of the MLP, RNN and LSTM can be seen in Figure 11 where they all are close to the curve. In Figure 12 the best version i could achieve can be seen which doesnt perfom anywhere near as well as the neural networks. The accuracy which can be seen below where the neural networks have around 15% MAPE and the Arima model 33% MAPE.

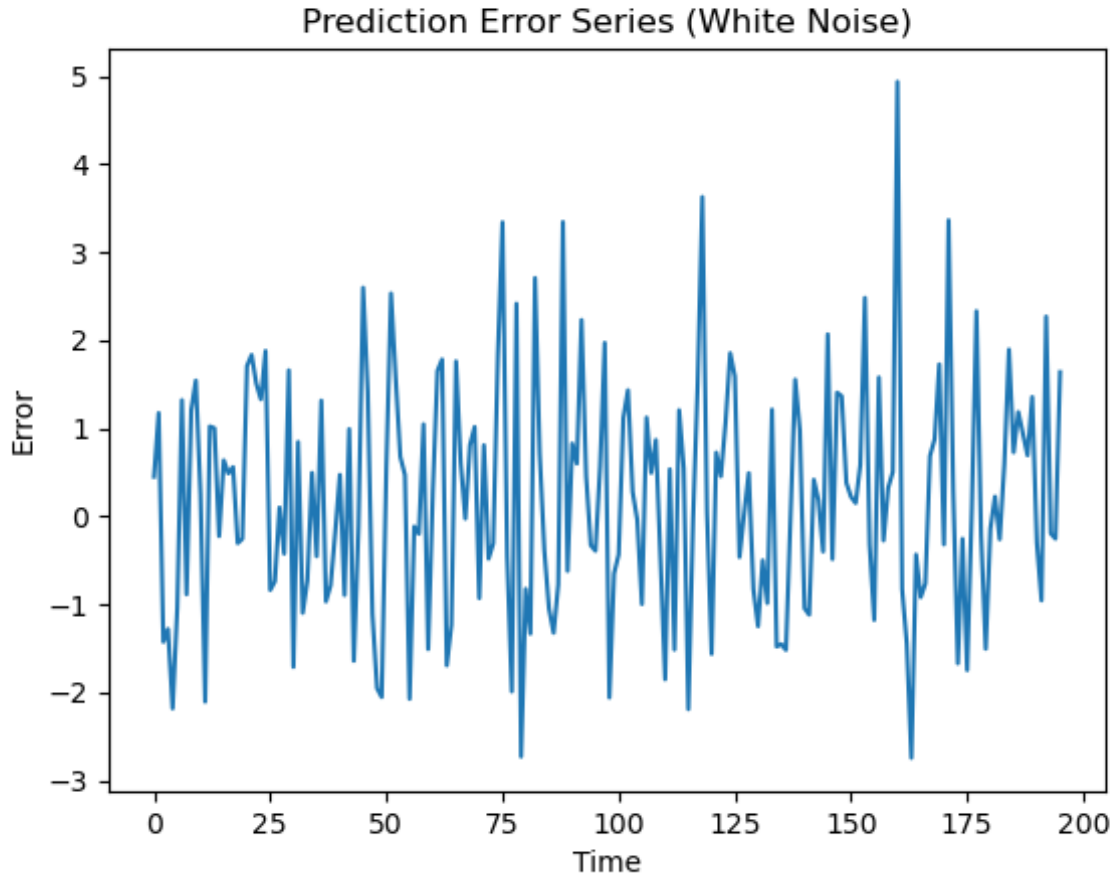



Figure 6: The prediction errors

ARIMA Model:
Mean Squared Error (MSE): 3.8833965201089183e+18
Mean Absolute Error (MAE): 1077955799.6155791
Mean Absolute Percentage Error (MAPE): 32.8517758455373%

MLP:
MSE: 4.6155666326299526e+17
MAE: 417349452.1763779
MAPE: 17.997085606257432%

RNN:
MSE: 4.1488304923116326e+17
MAE: 385458544.1763779
MAPE: 16.23553061363246%

LSTM:
MSE: 4.149424302700881e+17
MAE: 371610240.97637784
MAPE: 15.224502329443194%

### 1.3.1 Discussion

ARIMA Model:
MSE: 1.0154792294300082e+19
MAE: 1875504669.798579
MAPE: 57.8393040930985
MLP:
MSE: 8.633874552812399e+17
MAE: 447795760.17490757
MAPE: 30.185335226397946

RNN:
MSE: 1.8415623309175622e+17

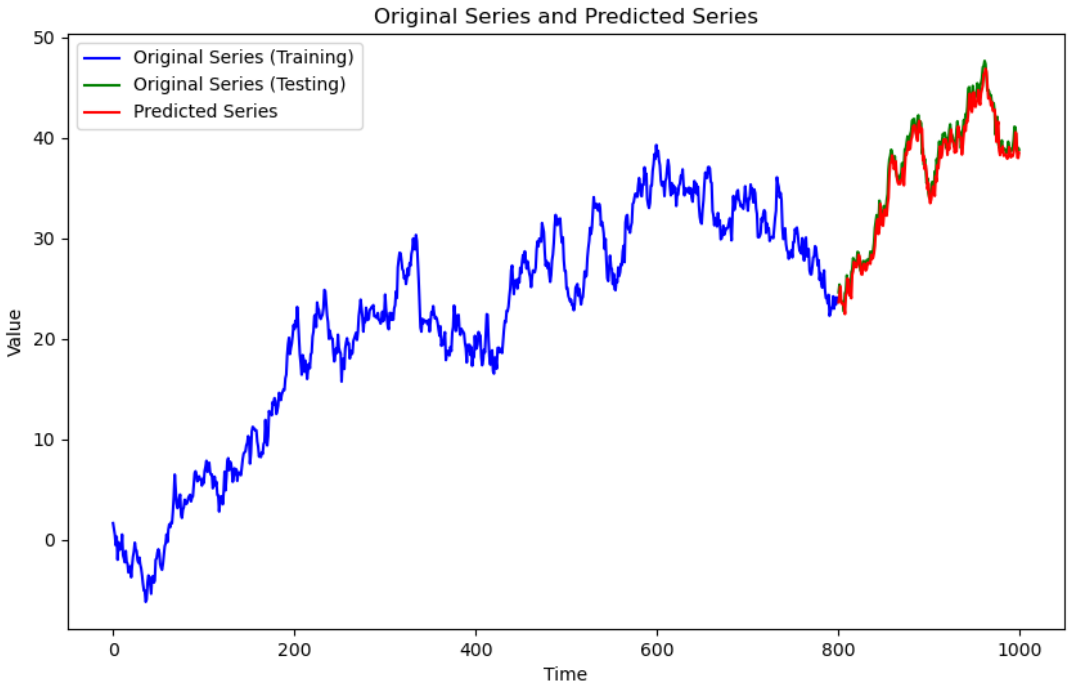



Figure 7: The RNN models attempt at predicting the tests Random walk

MAE: 294893361.93969023
MAPE: 25.711524472945733

LSTM:
MSE: 1.6913764141364416e+17
MAE: 280672994.5255923
MAPE: 19.64064634678172

# 2 Task 2

Analysing the data and insreting two anomalies to see if the anomaly detection works. This has the z-score threshhold set to 3 or above for it to count as an anomaly. The anomalies are located on 1994-12-01 with the value 20 and on 2000-06-01 with the value of -10 which can be observed in Figure 13 and 14.

The following months were considered anomalous and they are all around the inserted anomalies.
1994-12-01 19.468114
2000-02-01 3.160355
2000-03-01 3.096509
2000-04-01 3.375848
2000-06-01 -29.803577
1999-05-01 -3.233021

## 2.1 Discussion

Can the decomposition clearly separate the trend, season (constant period), and remainder components?
Yes it is clearly shown in Figure 13 how they have become clearly separated.
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?
While there are general rules how the decomposition happens depends a lot on the algorithm used. Is there a growing tendency in the trend series?
Yes within the Figure 13 an upwards trend can be observed.

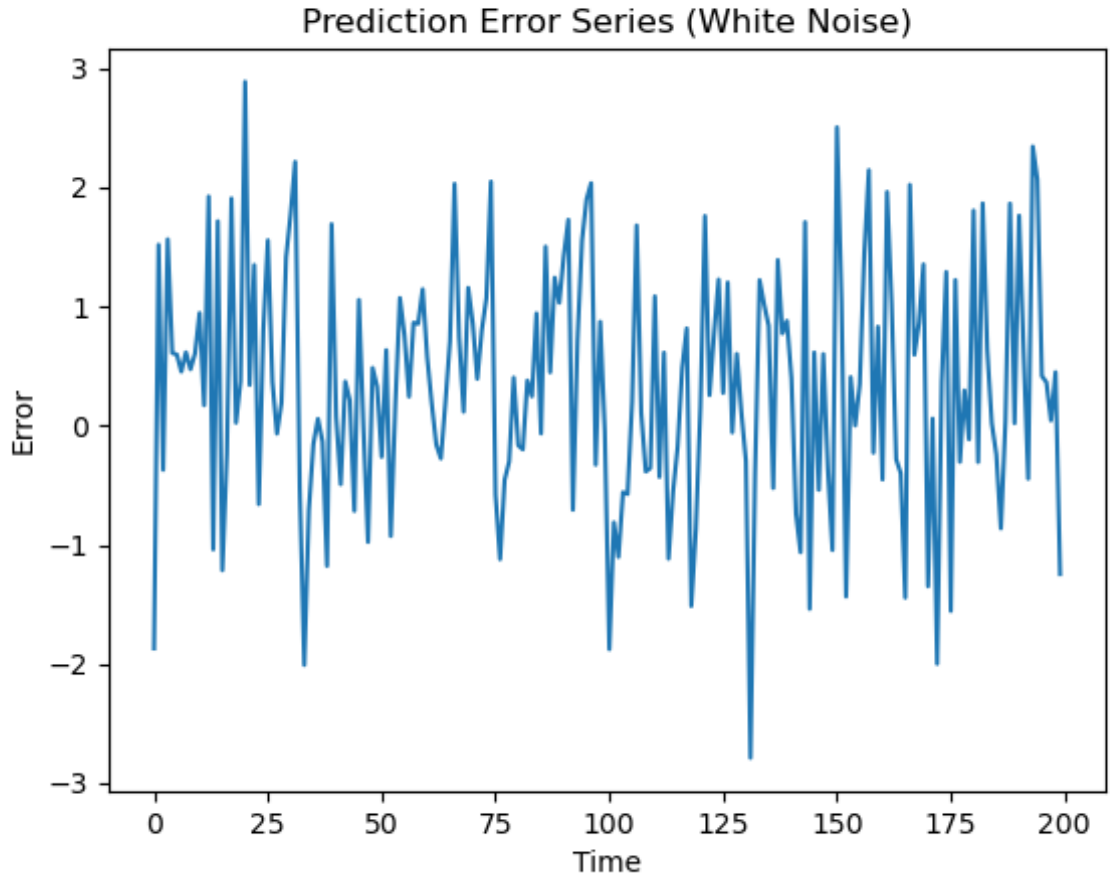



Figure 8: The prediction errors

# 3 Task 3

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

Looking at the exploratory data which cna be found in the Figure 15, 16, 17, 18, 19, 20 and 21. The mean value of this is 0.09063829787234039 while the standard deviation is 0.5359093910167457. Looking at the ACF and PACF in Figure 22 i cna be determined that the values for the ARIMA should be (3, 1, 3) with the threes being from the ACF and PACF and the 1 being from the one differentiation. Look at the residuals data in 23 we can see that is appears to be white noise. If we take a look at Figure 24 we can see the anomalies marked on the series.

### 3.1.1 Discussion

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?
While the decomposition focuses on splitting up the data into categories such as trend and seasonality to analyse the data and find the anomalies the prediction base models first builds a model then uses than model to make prediction and if the result is to far away from the prediction its considered anomalous. One focues on analysing old data while the other focuses on forecasting.
Do they achieve the same results? Why or Why not?
No they do not achieve the same results due to them focusing on different aspects of the data. If there are obvious anomalies they both will find it but cases where the difference isnt as big they may provide different results.
Given the anomaly ratio of 2%, what is the value of z-score? The values of the z-score was in this case 2.346953620477758.

## 3.2 Task 3.2 Anomaly detection in ECG signals with LSTM

The line plot and the lag plot can be seen in the Figures 25 and 26 where it is appaerent that there is a pattern that can be used to extract anomalies. The results can be seen in the Figures 3.2.1 to 3.2.1 where it can be seen on the residuals where the model finds anomalies at the different peaks.

MSE Scores:
Input Size: 4, Model: MLII, MSE: 0.0002696457592274914

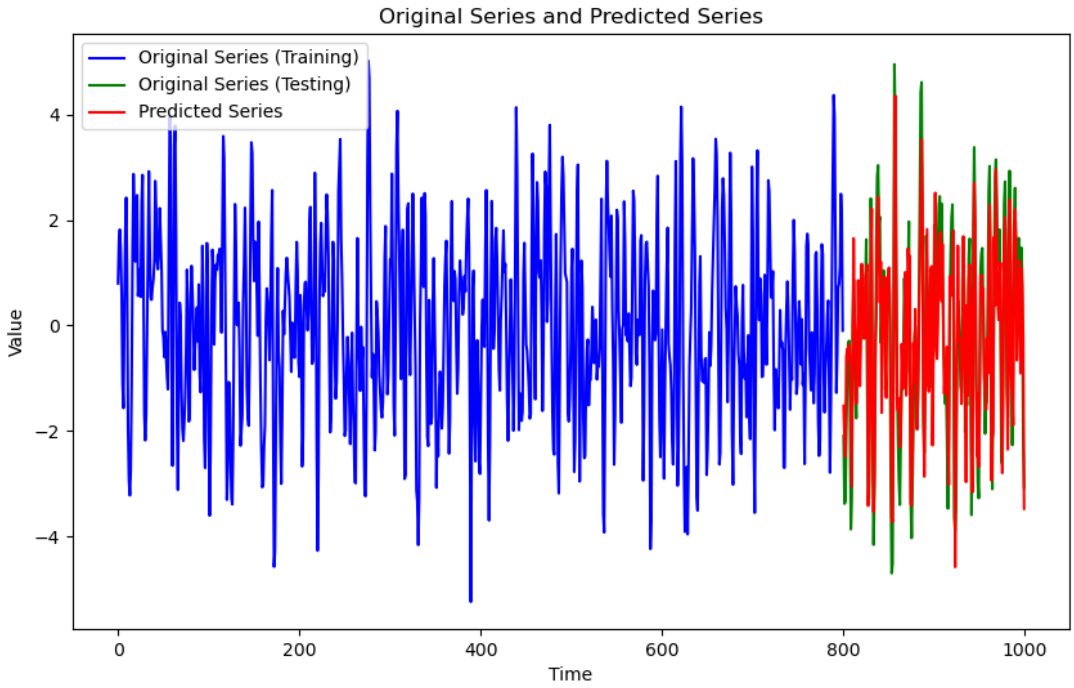



Figure 9: The RNN models attempt at predicting the tests ARMA process

Input Size: 4, Model: V5, MSE: 0.00035504318325118585
Input Size: 4, Model: Bi-variate, MSE: 0.00042569915819406503
Input Size: 8, Model: MLII, MSE: 0.00027133555235763233
Input Size: 8, Model: V5, MSE: 0.00038081652889350193
Input Size: 8, Model: Bi-variate, MSE: 0.00024879919812145 46
Input Size: 16, Model: MLII, MSE: 0.00028101592475425014
Input Size: 16, Model: V5, MSE: 0.0003362578477140594
Input Size: 16, Model: Bi-variate, MSE: 0.00029501250275489505

MAPE Scores:
Input Size: 4, Model: MLII, MAPE: 4.062635669854547%
Input Size: 4, Model: V5, MAPE: 6.952126325440765%
Input Size: 4, Model: Bi-variate, MAPE: 4.282003482212885%
Input Size: 8, Model: MLII, MAPE: 4.028795572523254%
Input Size: 8, Model: V5, MAPE: 7.085429350443126%
Input Size: 8, Model: Bi-variate, MAPE: 3.607581981285568%
Input Size: 16, Model: MLII, MAPE: 4.16534714110424%
Input Size: 16, Model: V5, MAPE: 6.7327099984878656%
Input Size: 16, Model: Bi-variate, MAPE: 4.0605435607049865%

### 3.2.1 Discussion

How have you designed the neural network?
I have designed them as one layer LSTM NNs that take in different sizes according to the task and returns a one step prediction. What hyper-parameters do you choose? Why? I choose to use Epoch: 10 which is a small amount, but this neural netwrok takes a long time to train which means that having 50 or 100 epoch would take over 10 minutes for each test and adjustment. The layer has 64 units which was a value i have use before in this laboratory with moderate success. The batch size 32 came through trial and error trying to minimize the MSE.
Can the neural network fit well to the ECG signals? What is the accuracy of your model?
Yes it fits well to the ECG signal and the MAPE accuracy varies around 5How much is the influence of the input vector size on the prediction accuracy?
The size of the input has close to no effect on the accuracy, depending which dataset was used had a bigger effect even though they were very similar.
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 used 10 epochs which may have not given the best accuracy possible but the tests showed that they were good enough. chatta learning rate, I use he Adam optimization algorithm.

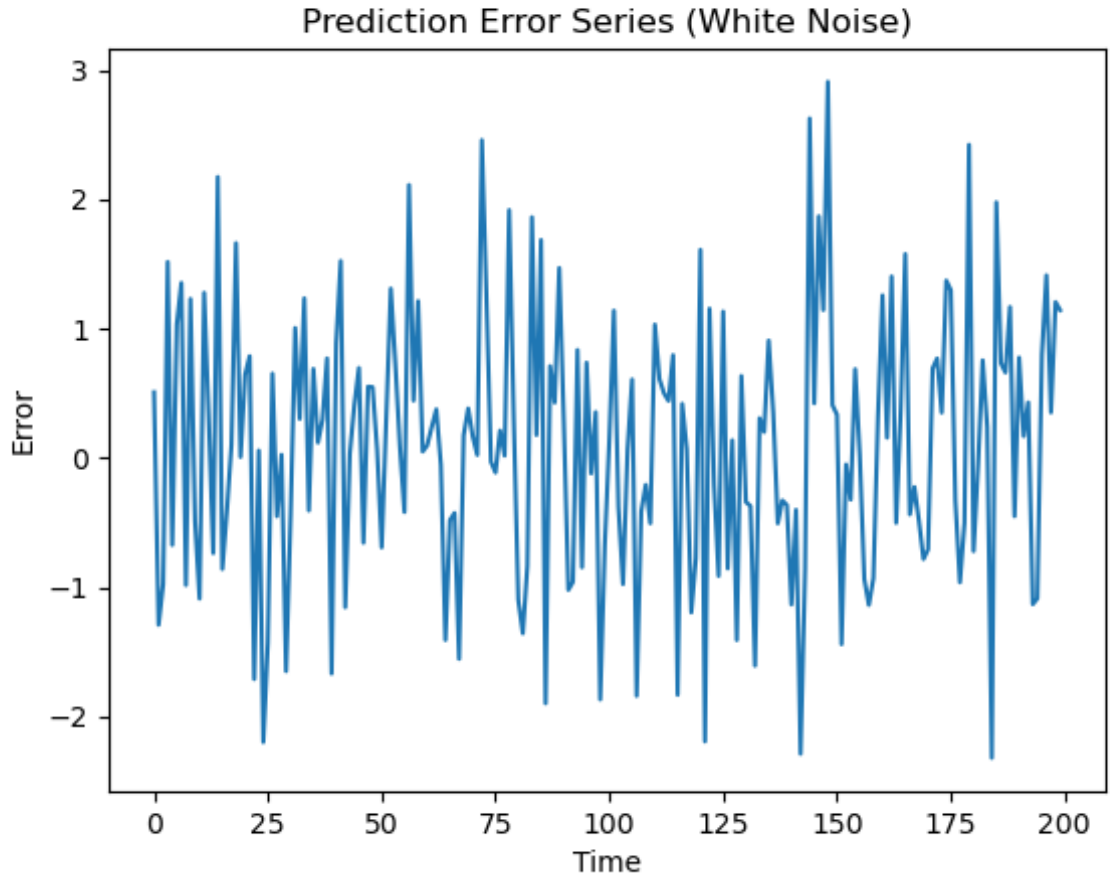



Figure 10: The prediction errors

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

The bi-variate series gave better accuracy. We have to series that have alot in common which means that using both of them makes it easier to predict both of them giving us better accuracy.

# 4 Task 4

The Figures 27 and 28 show the results of using Kmeans and SOM on the two series that was generated. The Kmeans used two clusters, whie the SOM used a 5 by 5 grid.

## 4.1 Discussion

How do you set the number of clusters? Why?

I set the number to 2 after bacuse we have o signals within the series that both have a mean point they are scattered around. Testing different values all gave different results but the anomalies were always on the edge

Which distance metric do you use? Are there other distance metrics which might be useful for this task?
The distance metric I used for this one was the simple Euclidean distance. The Manhattan distance or Minkowski distance could have also been utilized.

Do the two different clustering methods (K-menas and SOM) achieve the same results? Discuss why or why not.
No. They are fundamentally different approaches to clustering. We also know that Kmeans for example is worse at detecting clusters at higher dimensions which means they do not provide the same answer unless its a coincidence

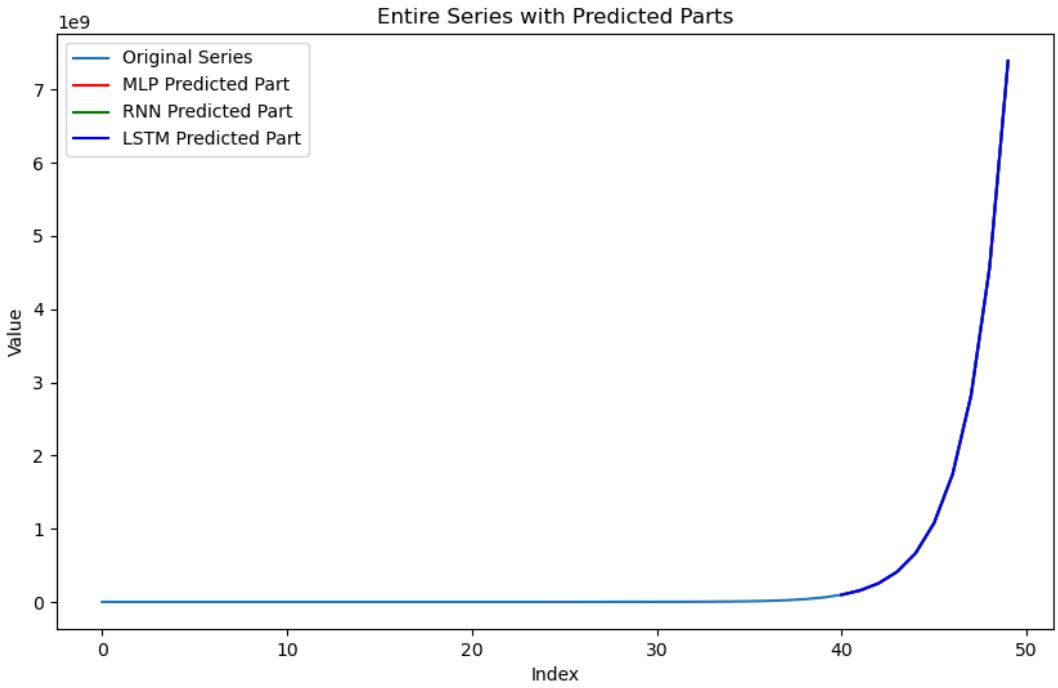



Figure 11: The predictions of The MLP, RNN and LSTM which are pretty close to the actual result

# 5 Task 5

## 5.1 Prediction approaches

ARIMA:
Assumption: I assume it is going to be good with linear data, like sound processing where the same pattern repeat.
Modeling: For modelling we need to remove trends then we look at the ACF and PACF to get the values for it then we can make predictions and add the trends to it afterwards.
Prediction: Done through classic signal processing feedback, feed to the front and with an integrated value
Strength: Its good for series with clear linear trends
Weakness: Series with hard to red non-linear trends.

NN:
Assumption: I assume its gonna be good at picking up subtle trends.
Modeling: Building a NN structure to be trained through training data. Changes to improve performance is hard to accurately predict so trial and error is needed.
Strength: Forms itself from the training data and is good at picking up all the subtle traits of the series.
Weakness: When it produces bad results we can now what exactly needs to fixed; we can only make educated guesses.

## 5.2 Anomaly Detection methods.

Decomposition:
Detection: Breaking down different patterns, finding anomalies there.
Assumption: It does not assume that there are anomalies.
Strength: Easy to interpret, helps finding what aspect the anomaly is in.
Weakness: It assume that the models are additive or multiplicative.

Prediction:
Detection: If the value is to far from the predicted value, it generates an anomaly
Assumption: It does not inherently assume that there are anomalies but the threshhold can be made so that a certain percentage
Strength: Can find small pattern for better detection, real-time usage such as in embedded systems
Weakness: Needs accurate training data, training is expensive, sensitive to bad data

Clustering:
Detection: The points furthest away from clusters are considered anomalies.

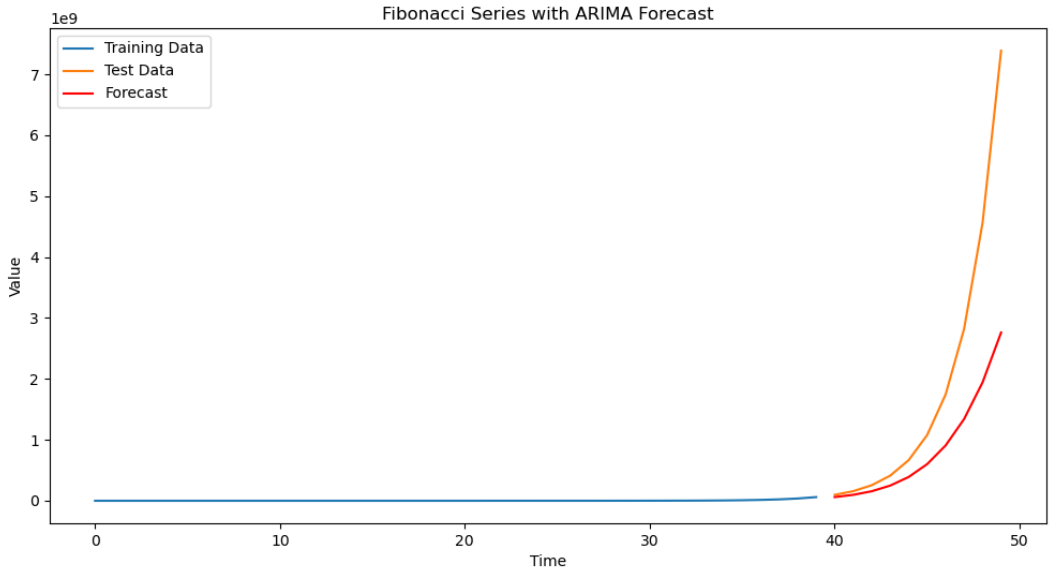



Figure 12: The predictions of the ARIMA model which isnt as good as the NNs

Assumption: This model assumes that there are anomlies within the data so if its all good data some may be considered anomalies.
Strengths: It can detect subtle patterns due the clusters. The threshhold can be adapted.
Weaknessess: Hard to interpret, anomalies are anomalous for some readon. Costly computation. Needs somewhat accurate idea of how the clusters form in this data.

Isolation Forest:
Detection: Distance from other values, creates trees and branches that are too long are anomalies.
Assumption: it doesnât assume that there are anomalies.
Strengths: Fast resource efficient, can find subtle patterns.
Weaknesses: Hard to interpret.

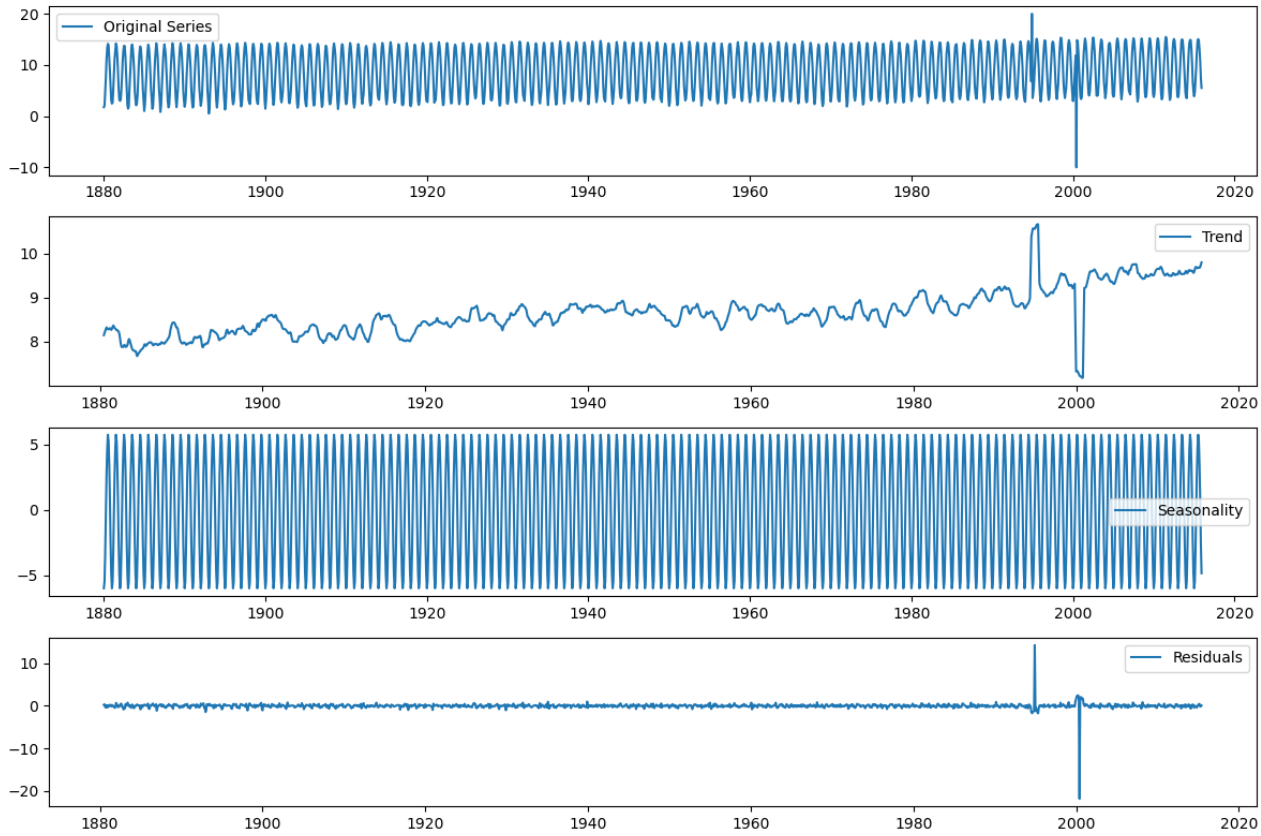



Figure 13: The different time series showing value, trend, seasonality and residuals

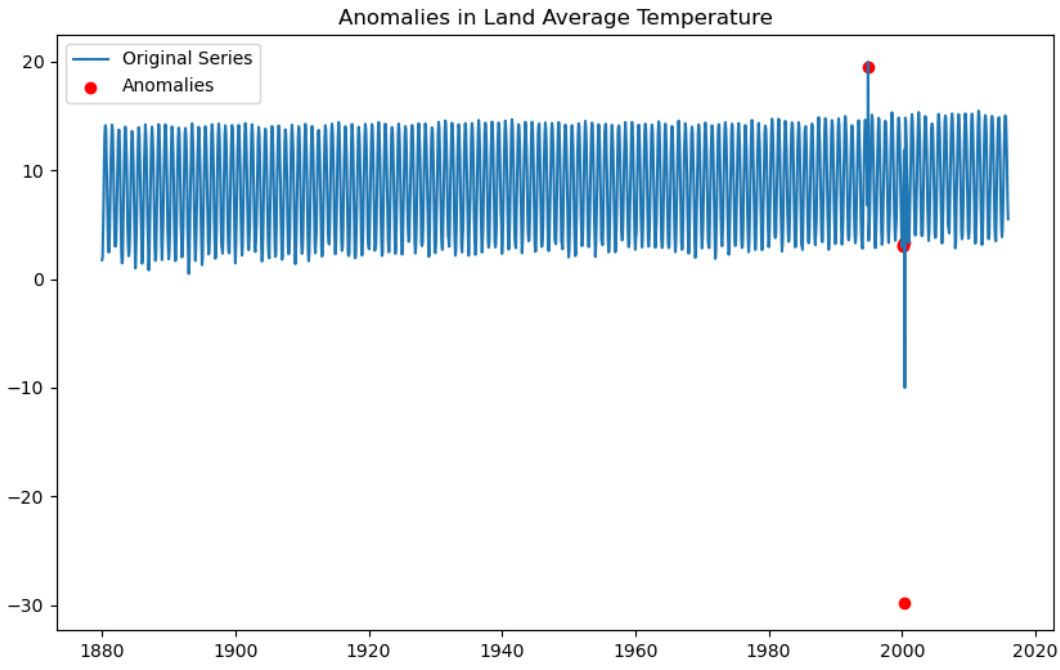



Figure 14: The time series showing the markings of the anomalies with a red dot at the level of their z-score

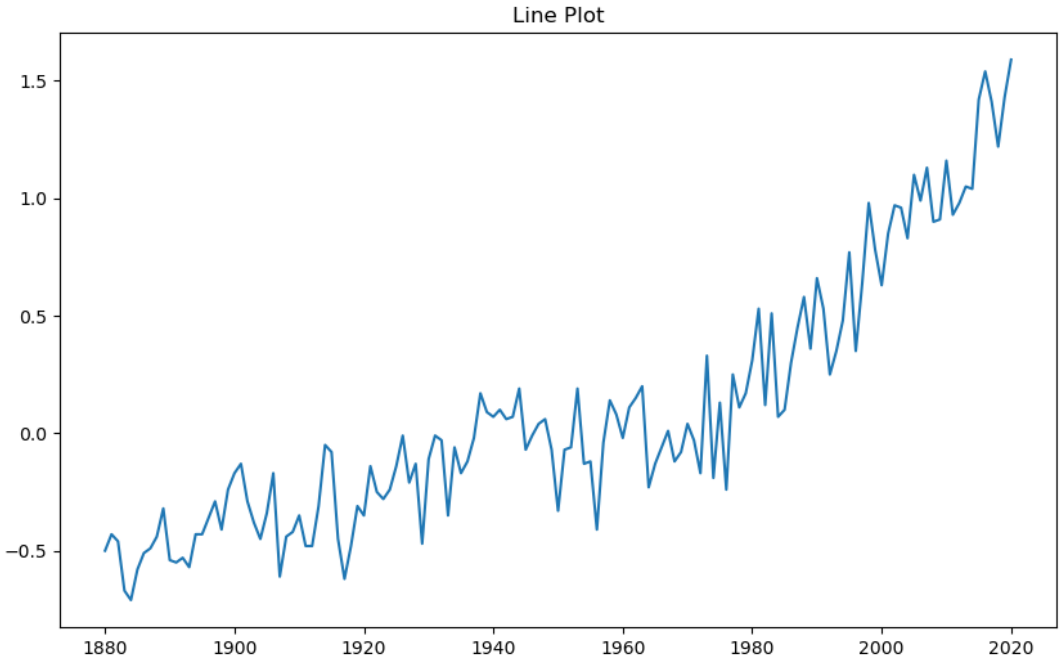



Figure 15: The line plot for the weather anomaly data

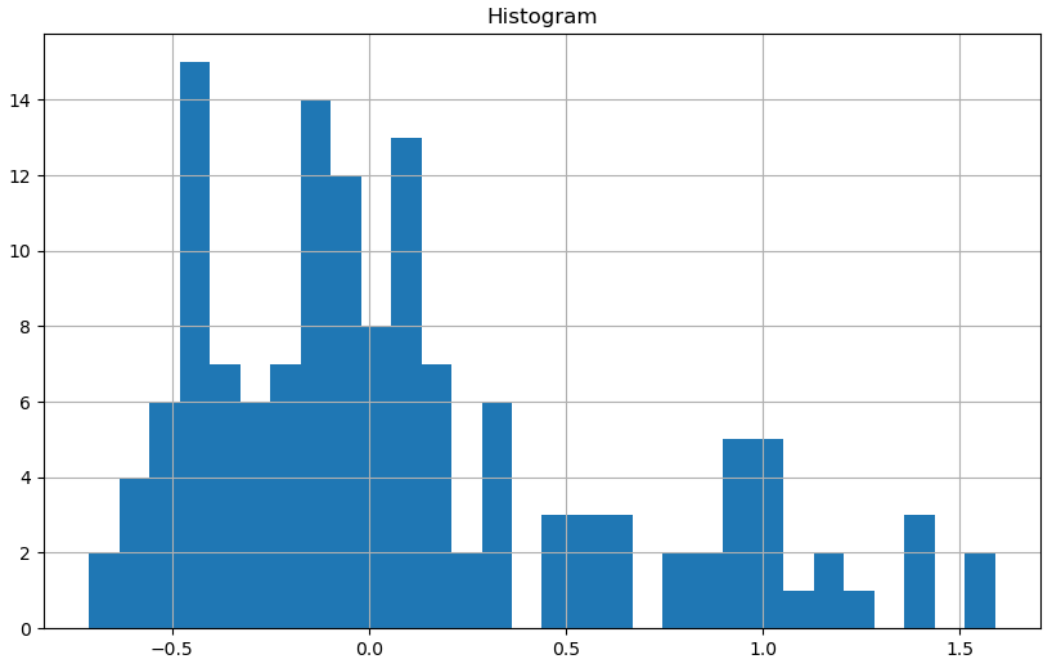



Figure 16: The histogram for the weather anomaly data

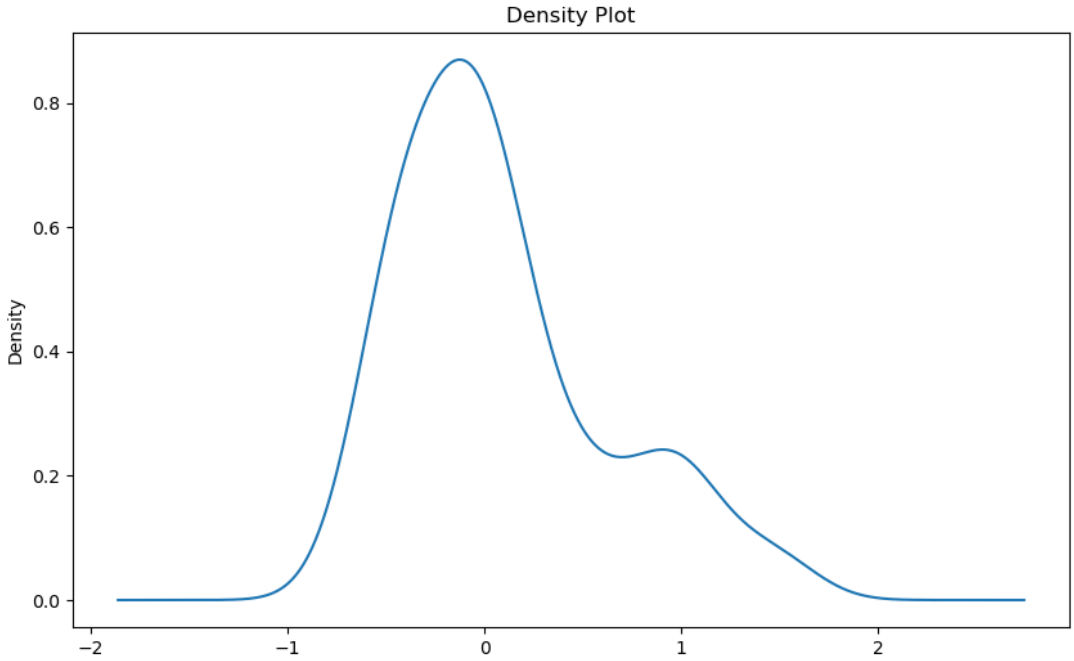



Figure 17: The density plot for the weather anomaly data

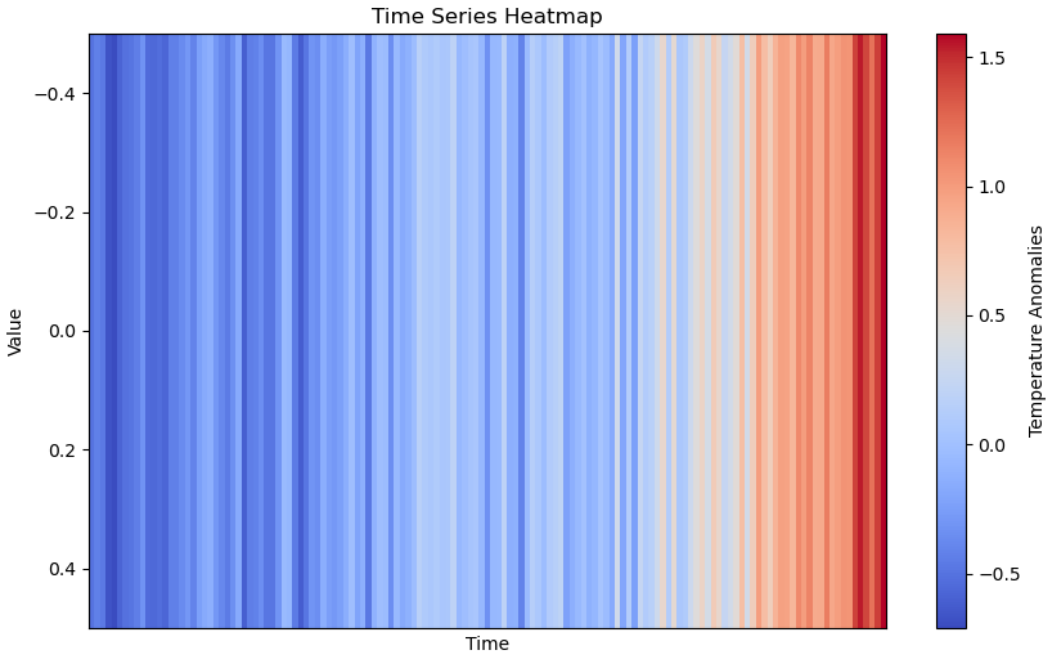



Figure 18: The heatmap plot for the weather anomaly data

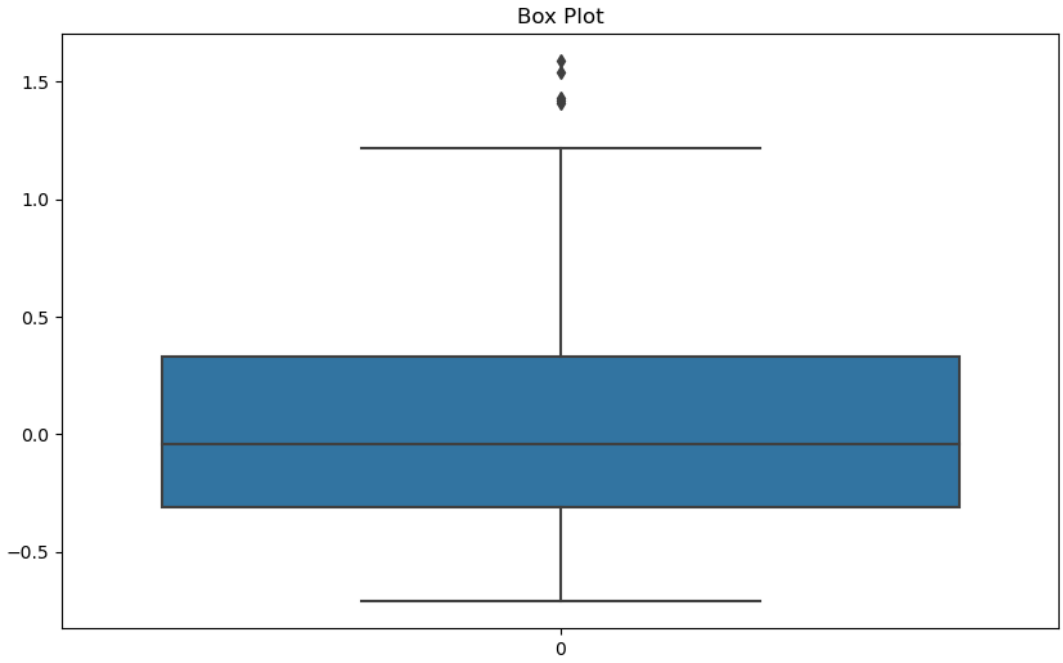



Figure 19: The box plot for the weather anomaly data

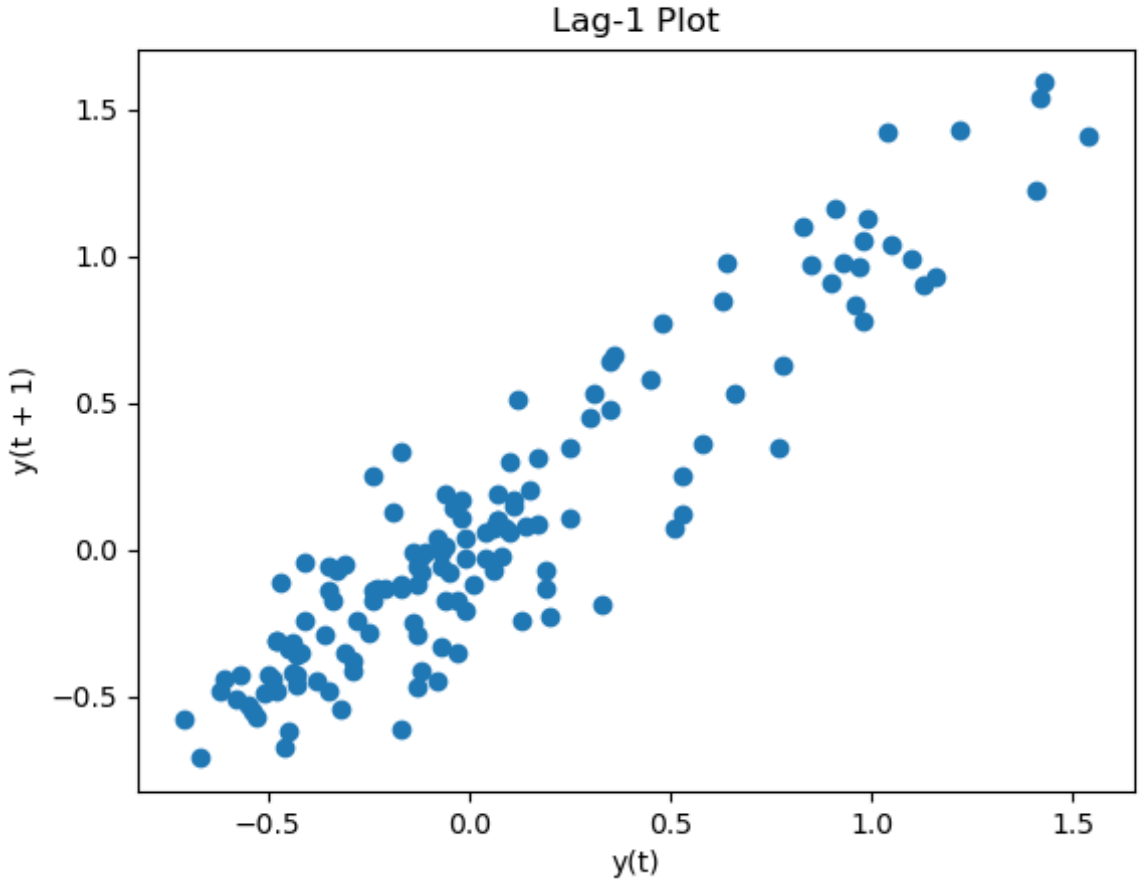



Figure 20: The lag-1 plot for the weather anomaly data

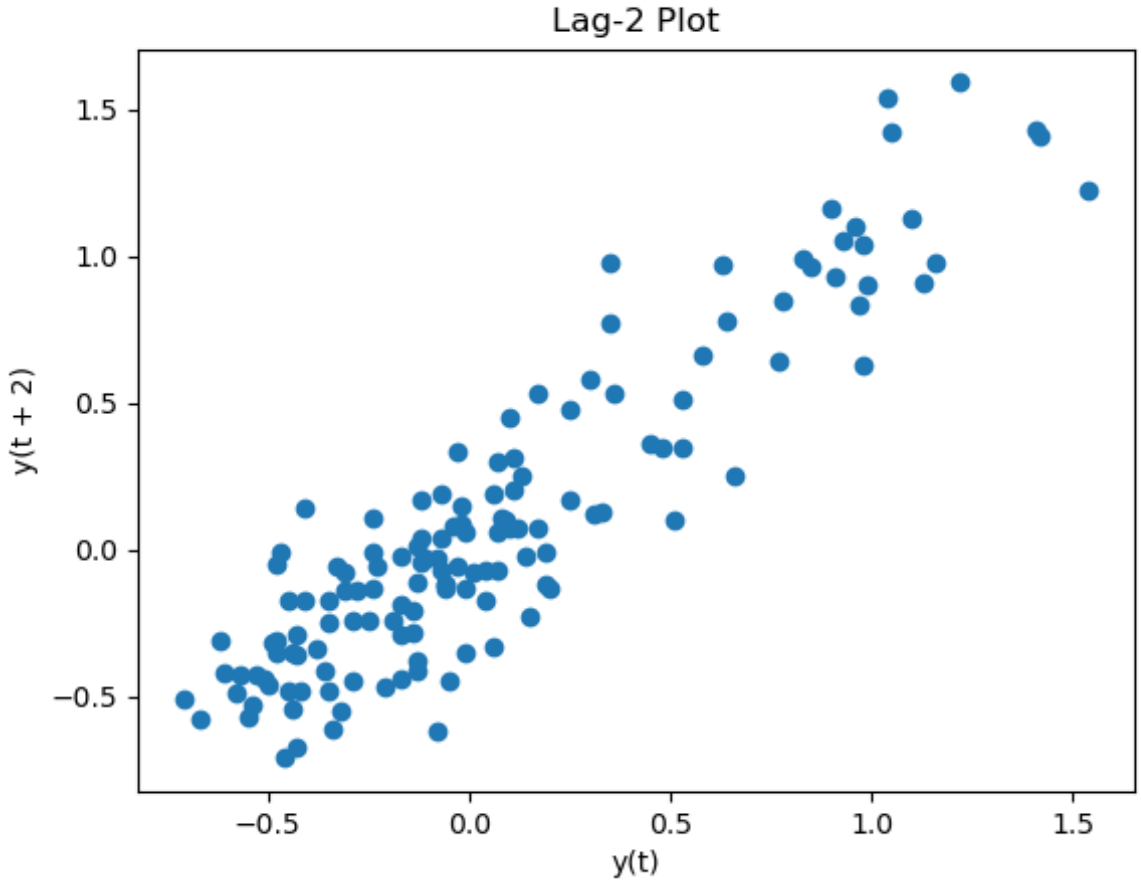



Figure 21: The lag-2 plot for the weather anomaly data

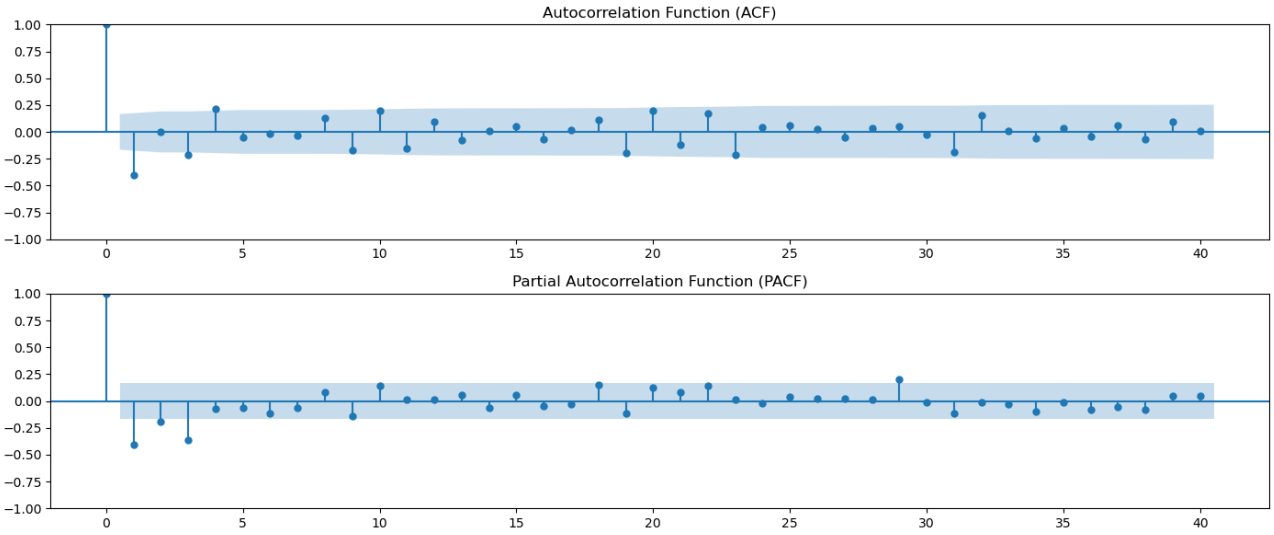



Figure 22: The acf and pacf for the differentialized data

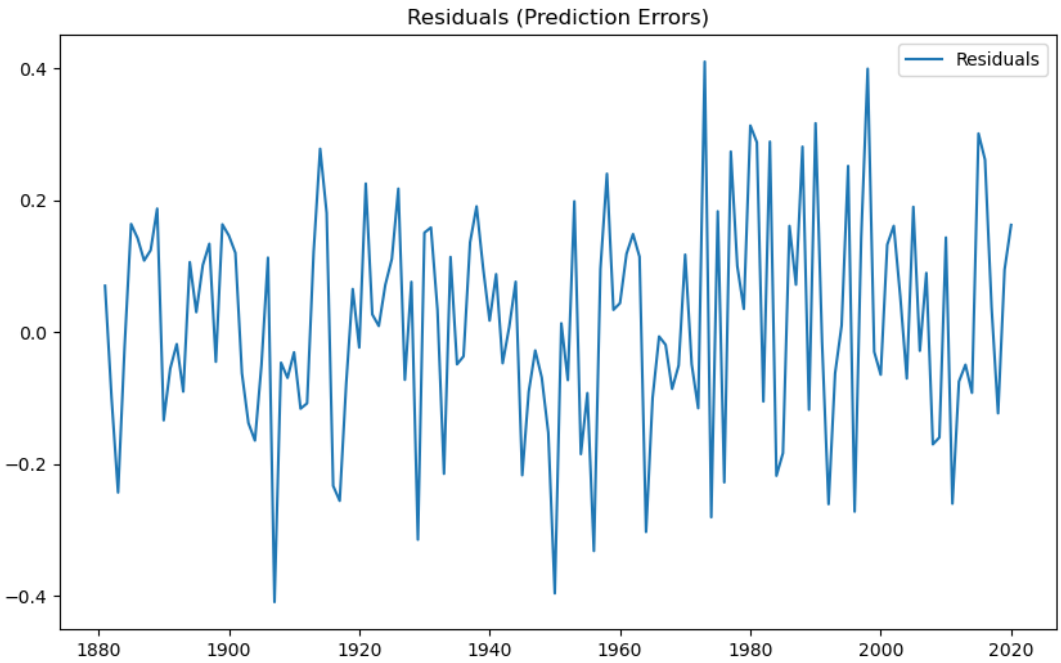



Figure 23: The residuals for the predictino data

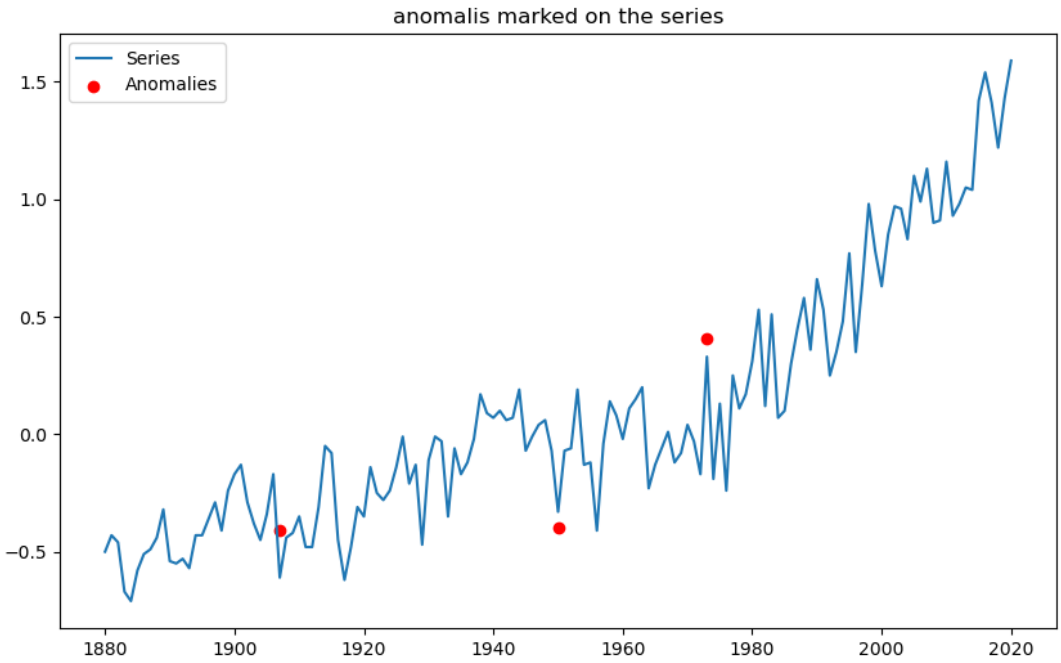



Figure 24: The lag-2 plot for the weather anomaly data

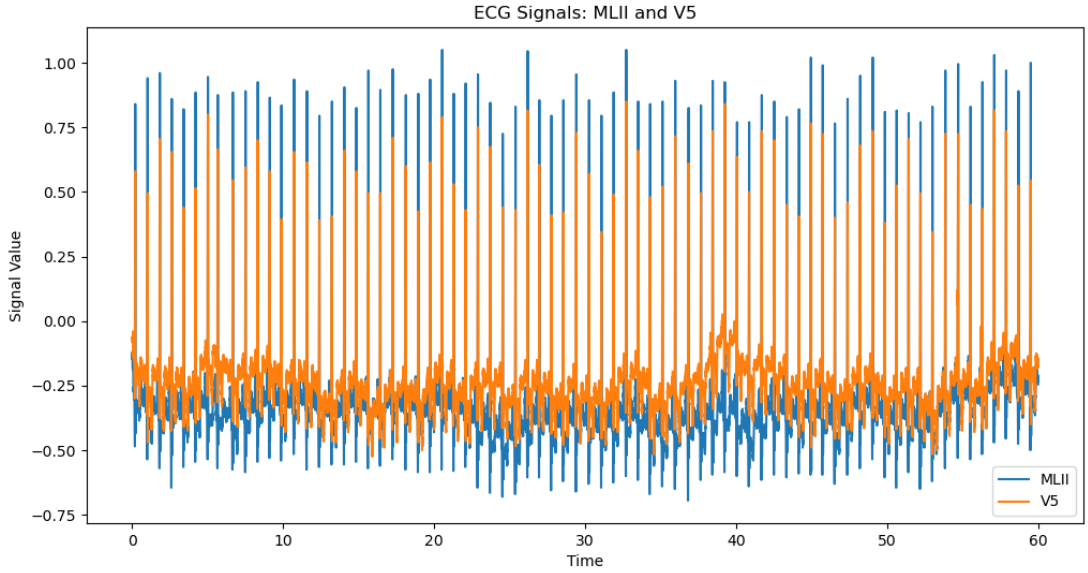



Figure 25: The line plot for the two different ECG values

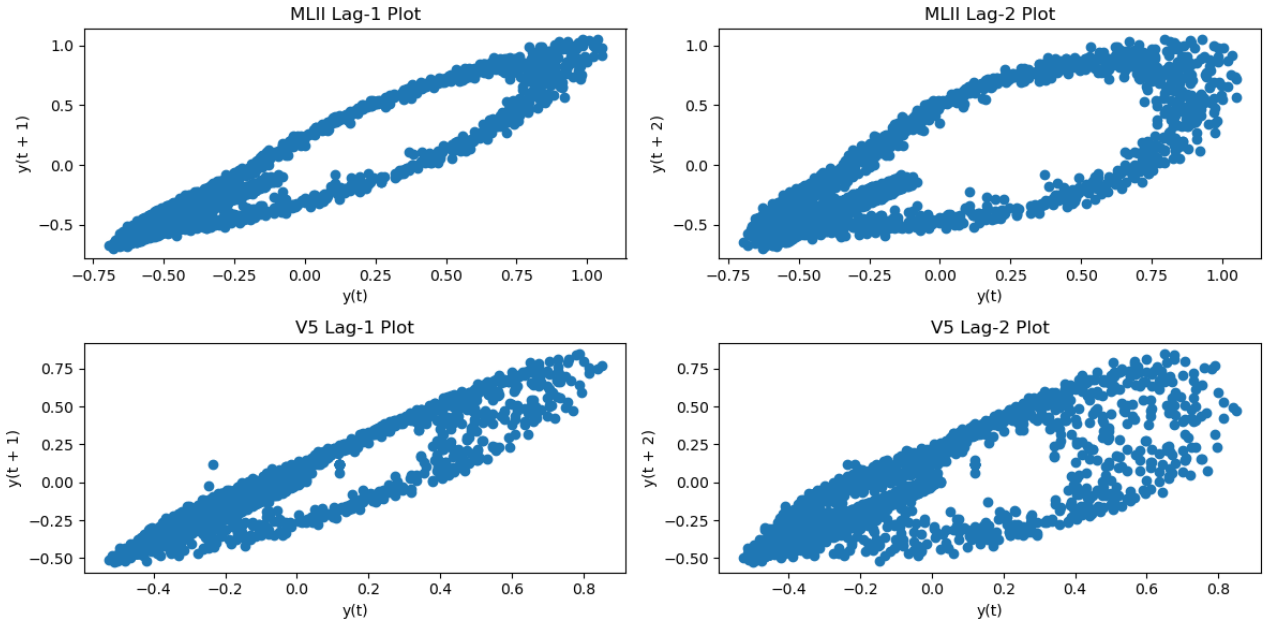
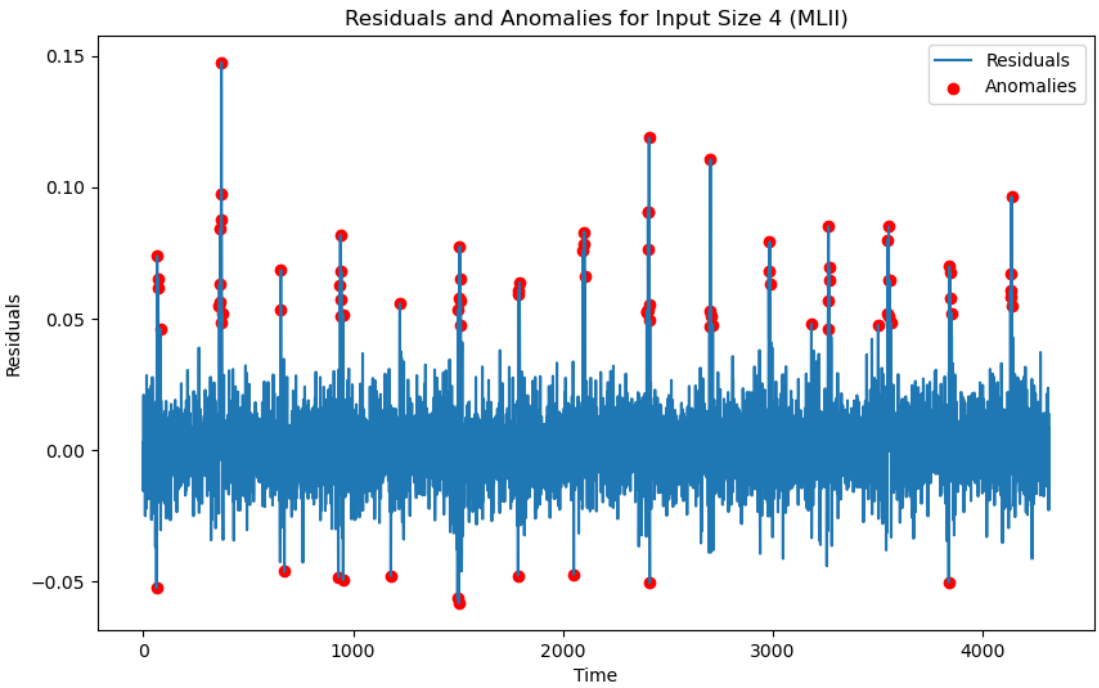
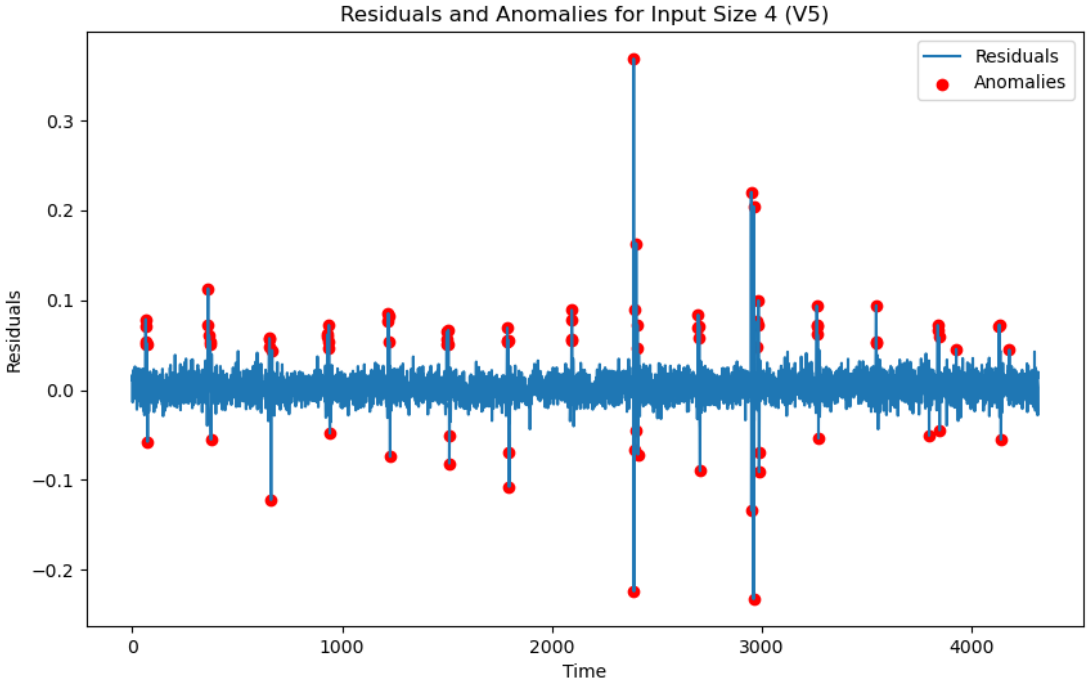
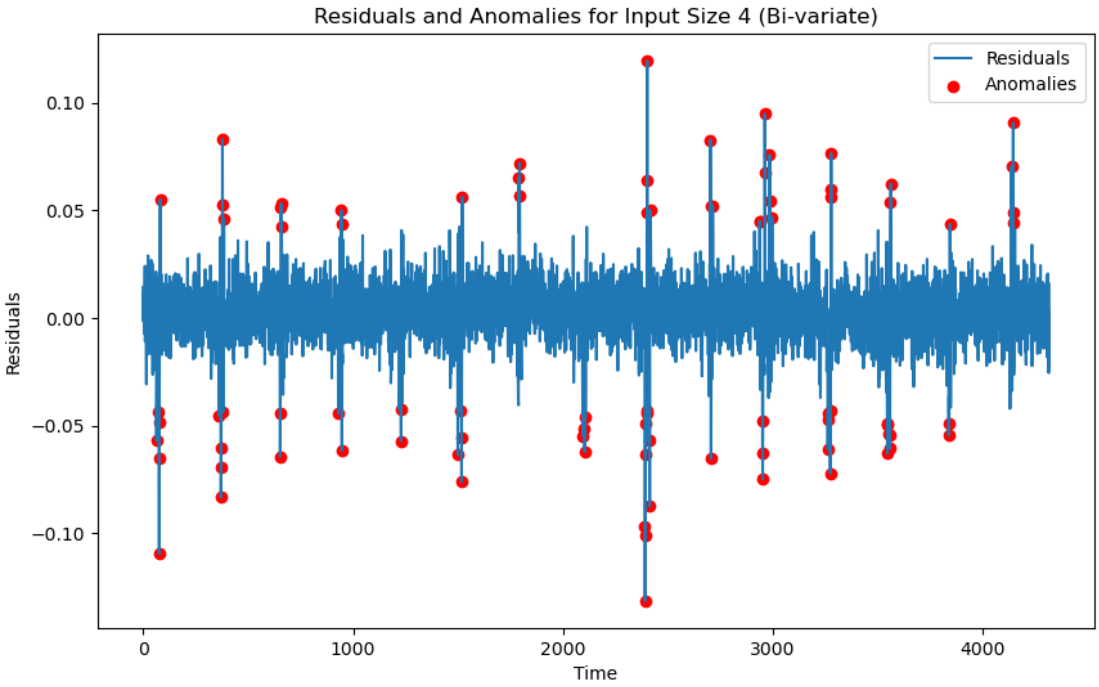
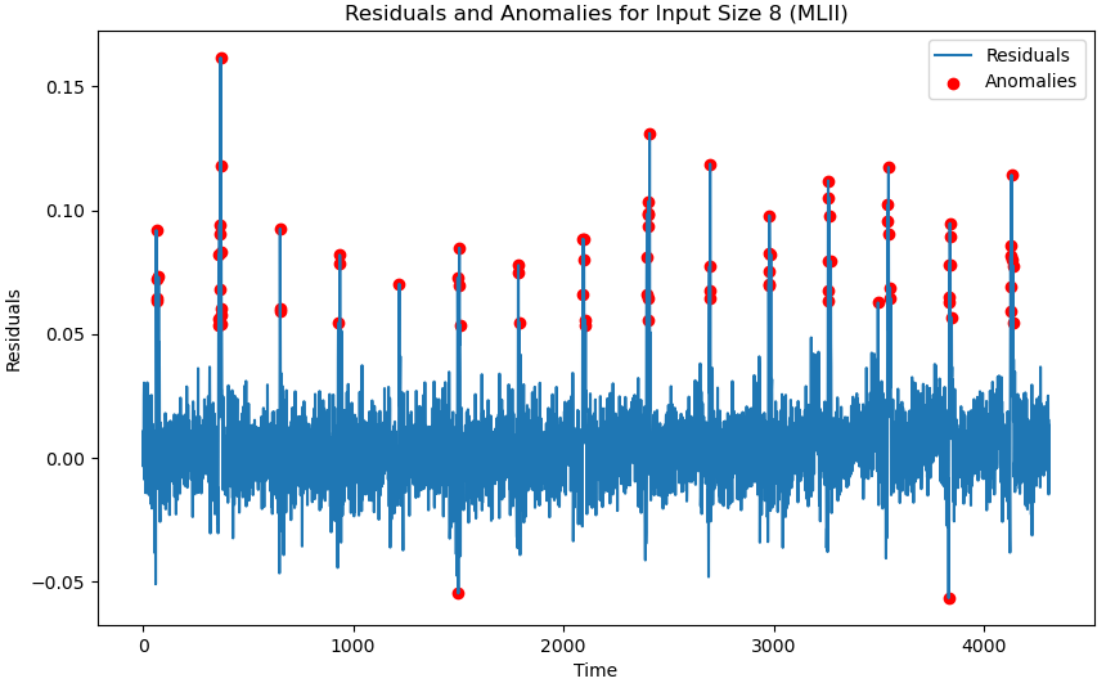


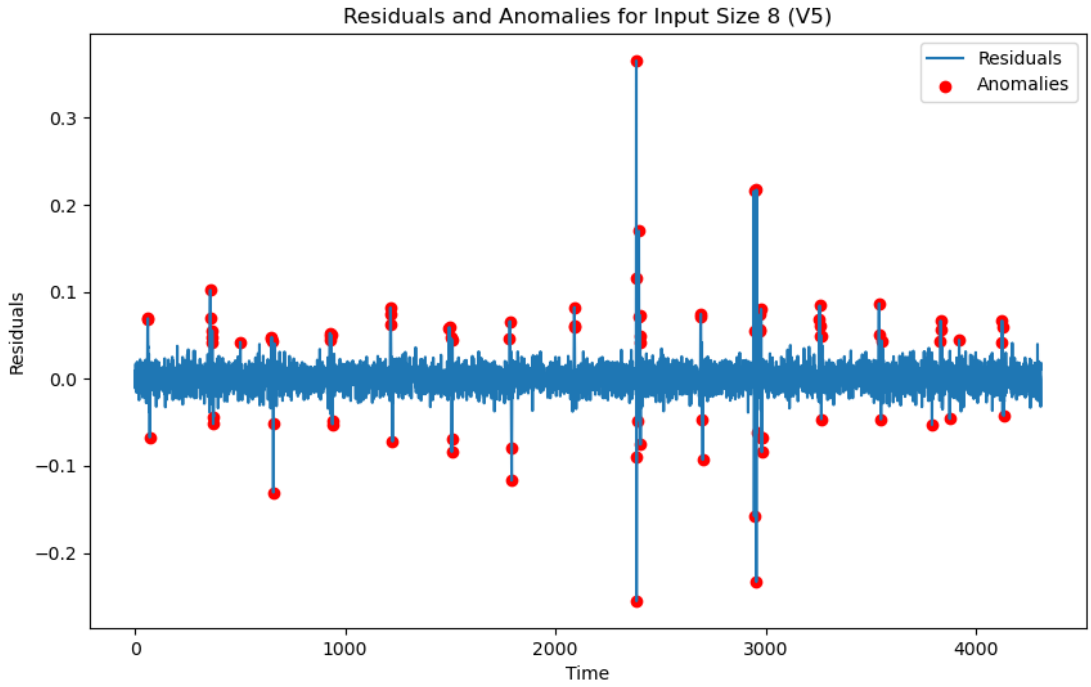

Figure 26: The lag1 and lag-2 plot for the two different ECG values

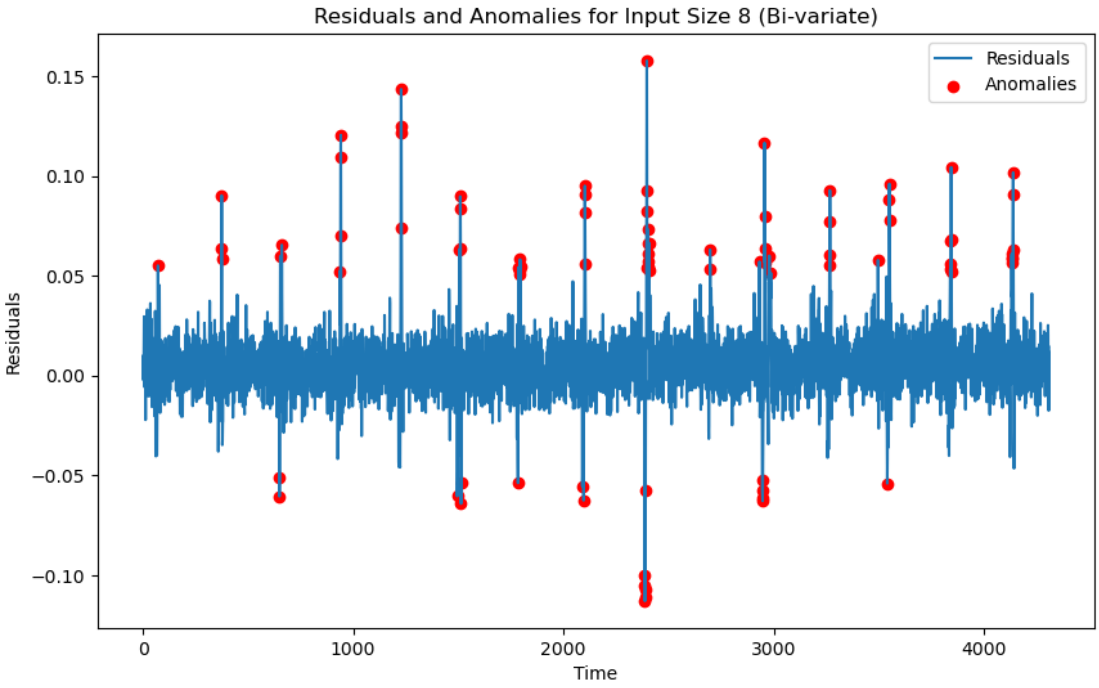

Residuals and Anomalies for Input Size 4 (MLII)
Residuals
Anomalies
Residuals
Time
0.15
0.10
0.05
0.00
−0.05
0
1000
2000
3000
4000

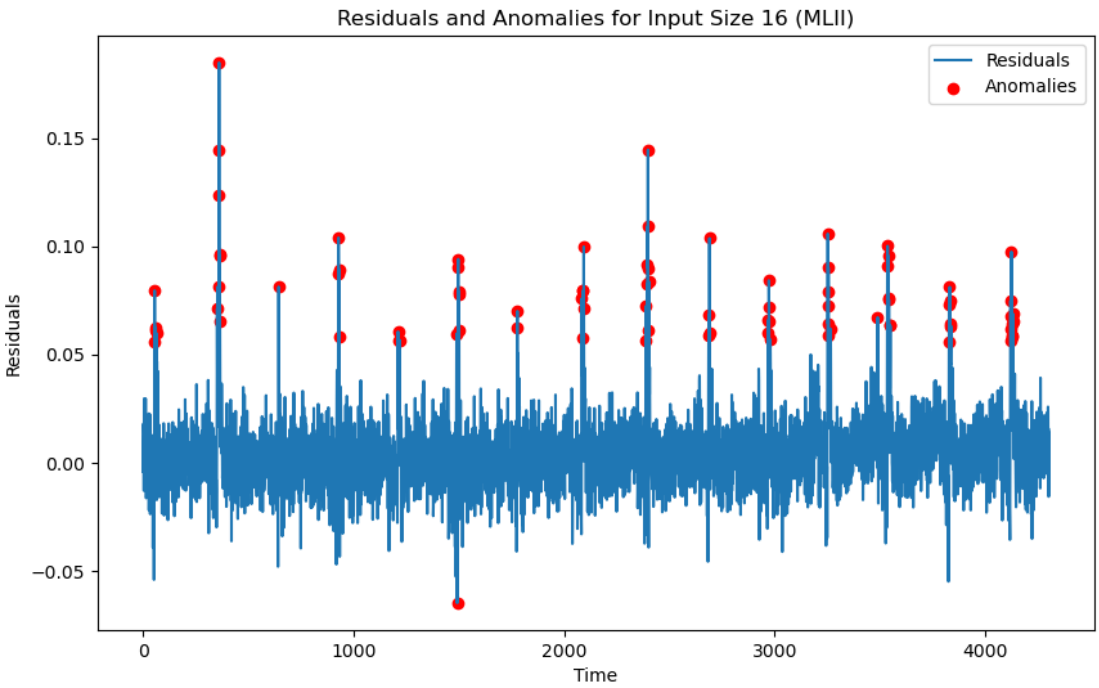

Residuals and Anomalies for Input Size 4 (V5)
Residuals
Anomalies
Residuals
Time
0.3
0.2
0.1
0.0
−0.1
−0.2
0
1000
2000
3000
4000

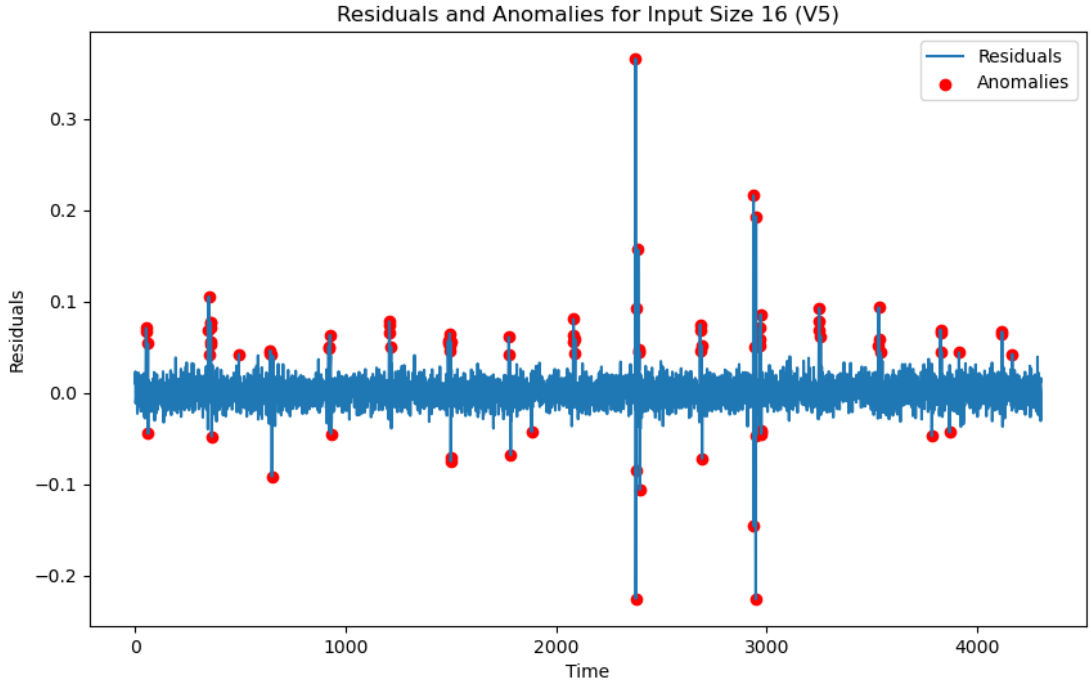

Residuals and Anomalies for Input Size 4 (Bi-variate)
Residuals
Anomalies
Residuals
Time
0.10
0.05
0.00
−0.05
−0.10
0
1000
2000
3000
4000

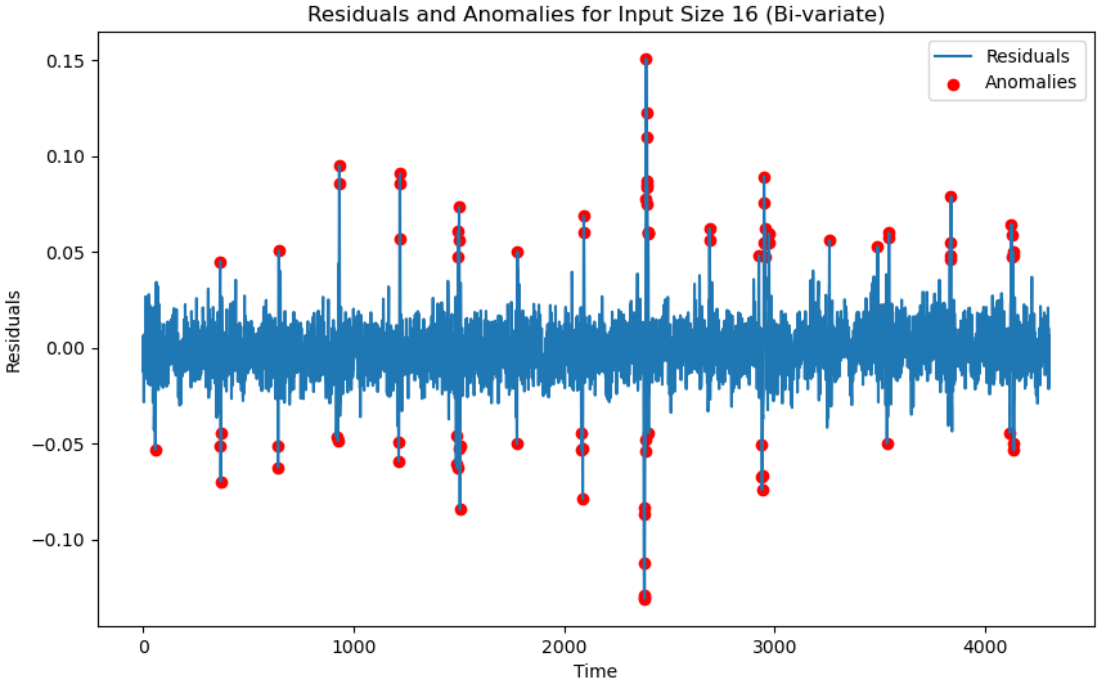

Residuals and Anomalies for Input Size 8 (MLII)
Residuals
Anomalies
Residuals
Time
0.15
0.10
0.05
0.00
−0.05
0
1000
2000
3000
4000


Residuals and Anomalies for Input Size 8 (V5)
Residuals
Anomalies
Residuals
Time
0.3
0.2
0.1
0.0
−0.1
−0.2
0
1000
2000
3000
4000


Residuals and Anomalies for Input Size 8 (Bi-variate)
Residuals
Anomalies
Residuals
Time
0.15
0.10
0.05
0.00
−0.05
−0.10
0
1000
2000
3000
4000


Residuals and Anomalies for Input Size 16 (MLII)
Residuals
Anomalies
Residuals
0.15
0.10
0.05
0.00
−0.05
0
1000
2000
3000
4000
Time


Residuals and Anomalies for Input Size 16 (V5)
Residuals
Anomalies
Residuals
0.3
0.2
0.1
0.0
−0.1
−0.2
0
1000
2000
3000
4000
Time


Residuals and Anomalies for Input Size 16 (Bi-variate)
Residuals
Anomalies
0.15
0.10
0.05
0.00
−0.05
−0.10
Residuals
0
1000
2000
3000
4000
Time

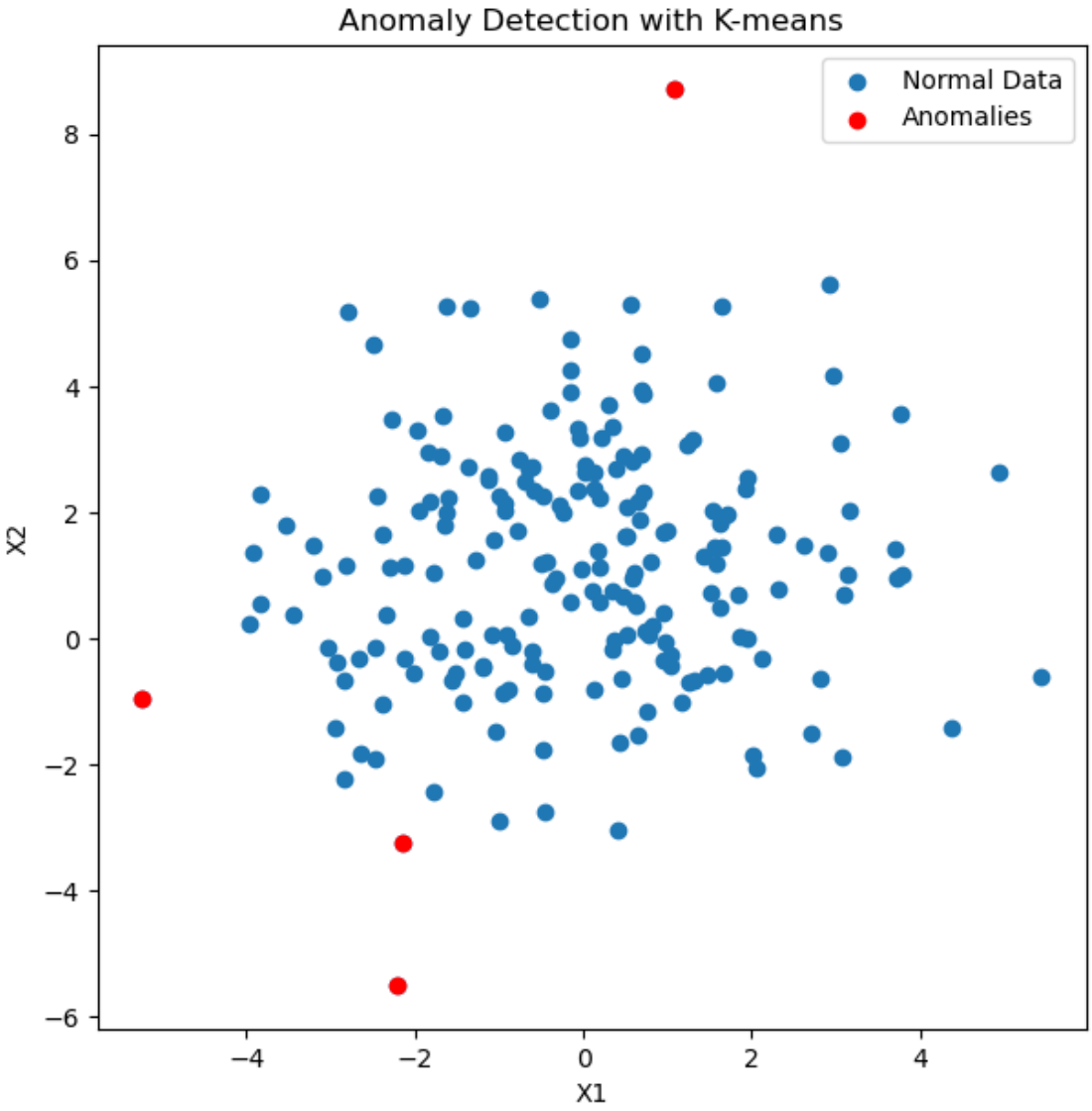



Figure 27: The anomalies showing in red dots

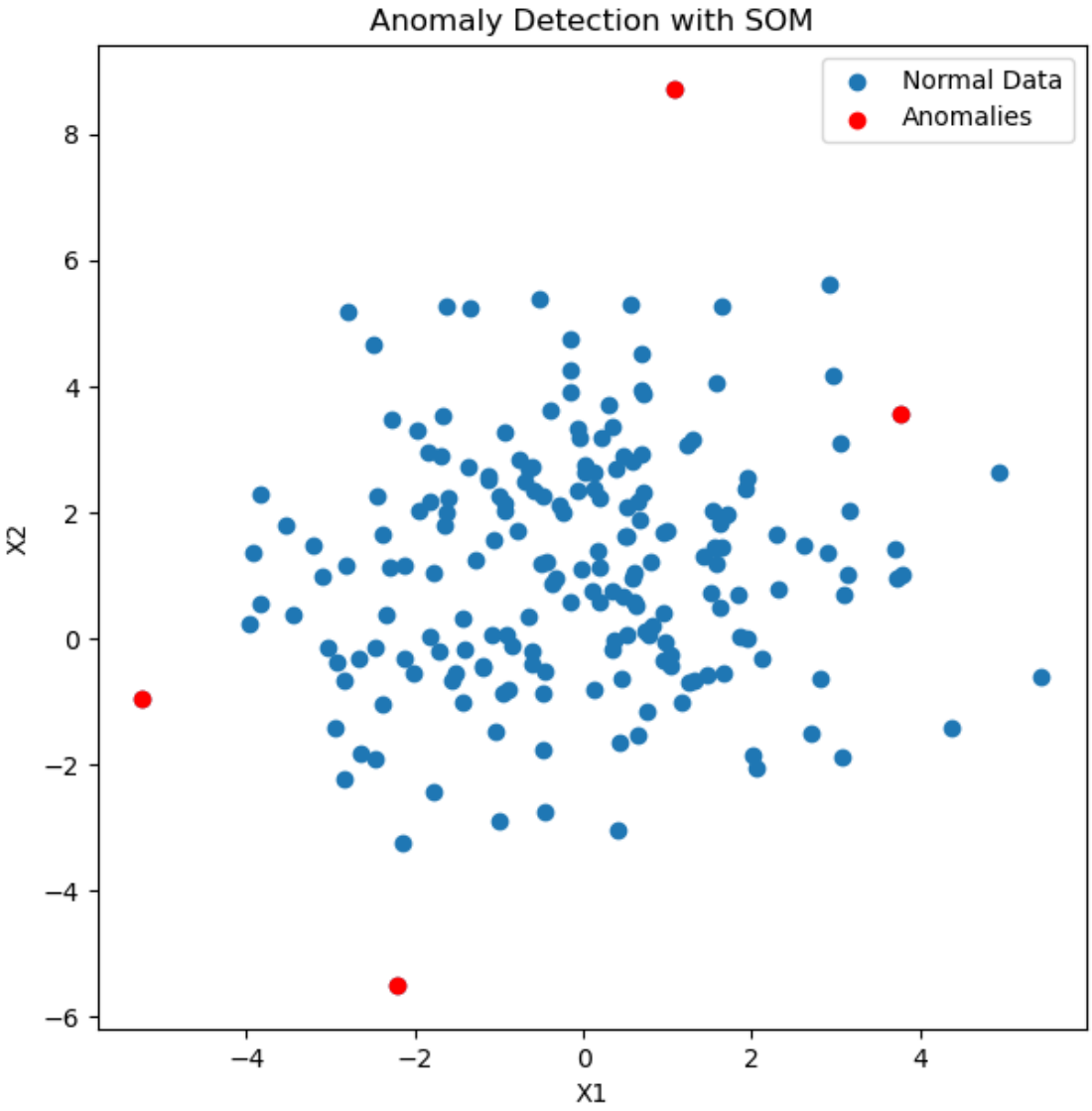



Figure 28: The anomalies showing in red dots