GROUP WORK PROJECT #1 MScFE 642: Deep Learning for Finance
Group Number: 5281

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Marin Stoyanov	Bulgaria	azonealerts@gmx.com	
Brian Luna Patino	United Kingdom	brian1311@hotmail.co.uk	
Long Chen	China	chenlongpku@163.com	

Statement of integrity: By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above).

Team member 1	Marin Stoyanov
Team member 2	Brian Luna Patino
Team member 3	Long Chen

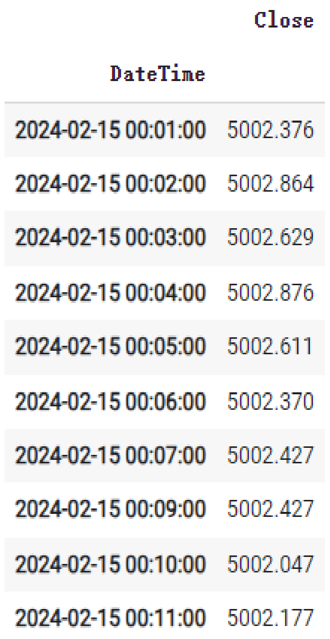
Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed.
Note: You may be required to provide proof of your outreach to non-contributing members upon request.

N/A

Step 1: Data Preparation

Original Time Series

The dataset used in this project contains 1-minute bars of S&P 500 during Feb 15, 2024. downloaded from [1]. The file is in Excel format and has been zipped alongside the accompanying notebook.



DateTime	Close
2024-02-15 00:01:00	5002.376
2024-02-15 00:02:00	5002.864
2024-02-15 00:03:00	5002.629
2024-02-15 00:04:00	5002.876
2024-02-15 00:05:00	5002.611
2024-02-15 00:06:00	5002.370
2024-02-15 00:07:00	5002.427
2024-02-15 00:09:00	5002.427
2024-02-15 00:10:00	5002.047
2024-02-15 00:11:00	5002.177

Table 1. Time series data description

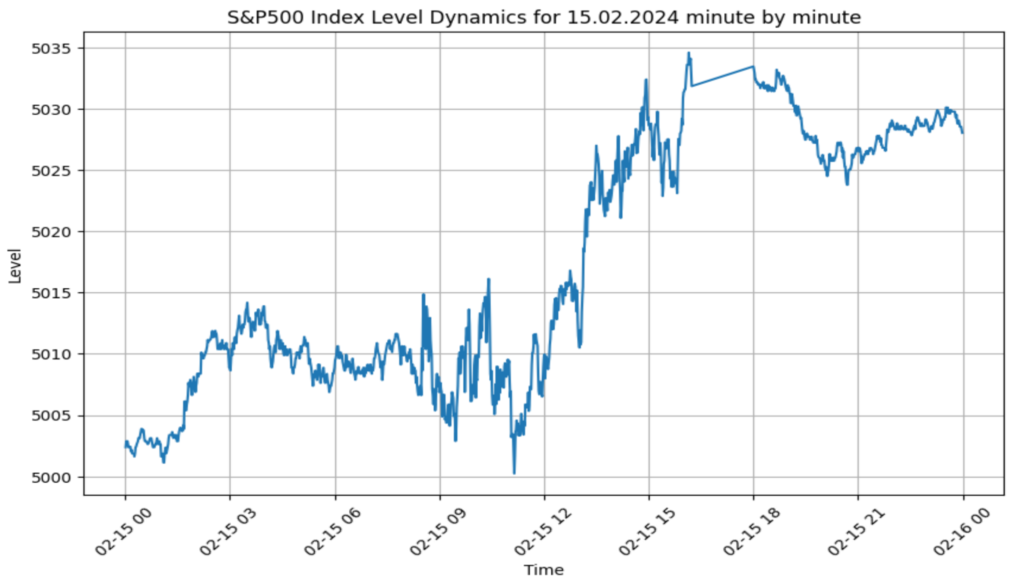


Fig.1. Plotting the raw data

The full 1-min price data of a day should contain 24×60=1440 rows. After some inspection, there are missing data points, such as at 00:08:00. Furthermore, there is no data between 16:15 and 18:00 each day, during which the U.S. stock market is closed.

To avoid any implication of this discontinuation, we decided to separate this data into two time series, one for training and the other for testing. And we filled in the missing data point by copying the previous minute one.

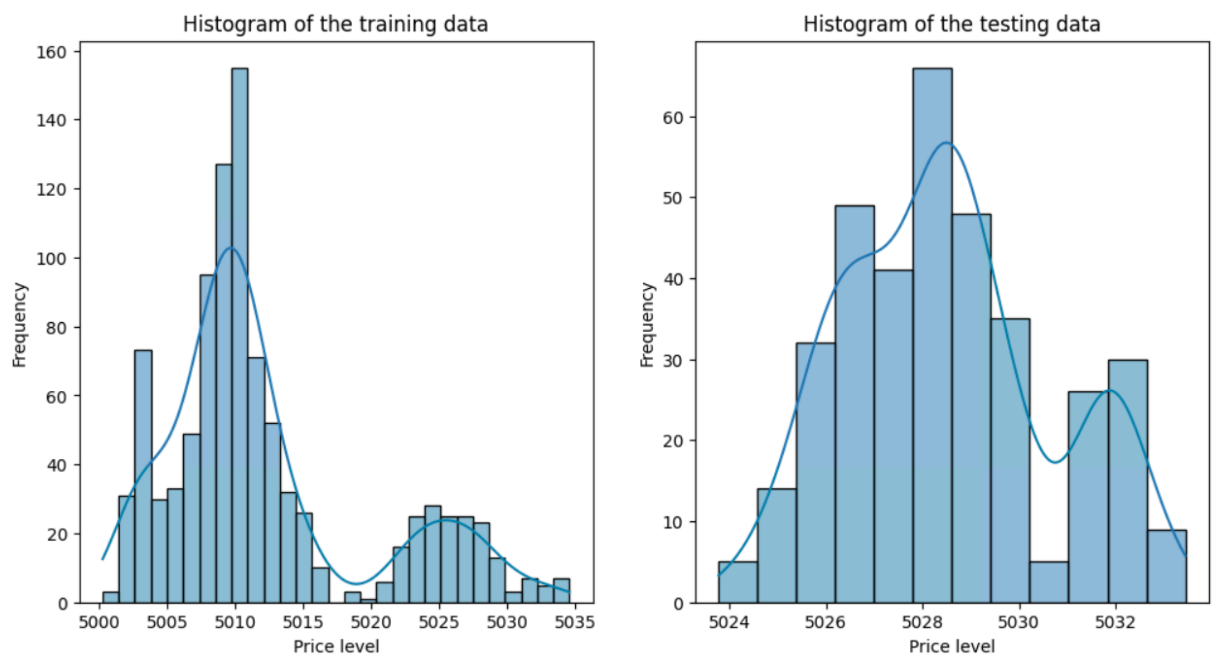


Fig.2. Histograms of training and testing data accordingly

The histograms show that both time series contain two modes, though at different levels.

We use Augmented Dickey-Fuller test to test unit root.

For the training set:

```
Results of ADF Test:
Test Statistic                  -0.625607
P-value                          0.865090
#Lags Used                       4.000000
Number of Observations Used    969.000000
Critical Value (1%)             -3.437116
Critical Value (5%)             -2.864527
Critical Value (10%)            -2.568361
dtype: float64
```

Table 2 ADF test of training data

For the test set:

```
Results of ADF Test:
Test Statistic                  -2.326153
P-value                          0.163672
#Lags Used                       0.000000
Number of Observations Used    359.000000
Critical Value (1%)             -3.448697
Critical Value (5%)             -2.869625
Critical Value (10%)            -2.571077
dtype: float64
```

Table 3 ADF test of testing data

Both p-values are greater than 0.05, so we fail to reject the null hypothesis, and this suggests that the time series is not stationary.

Stationary Time Series

To make the time series stationary, we simply take the 1st order difference, and run ADF test again.

For the training set:

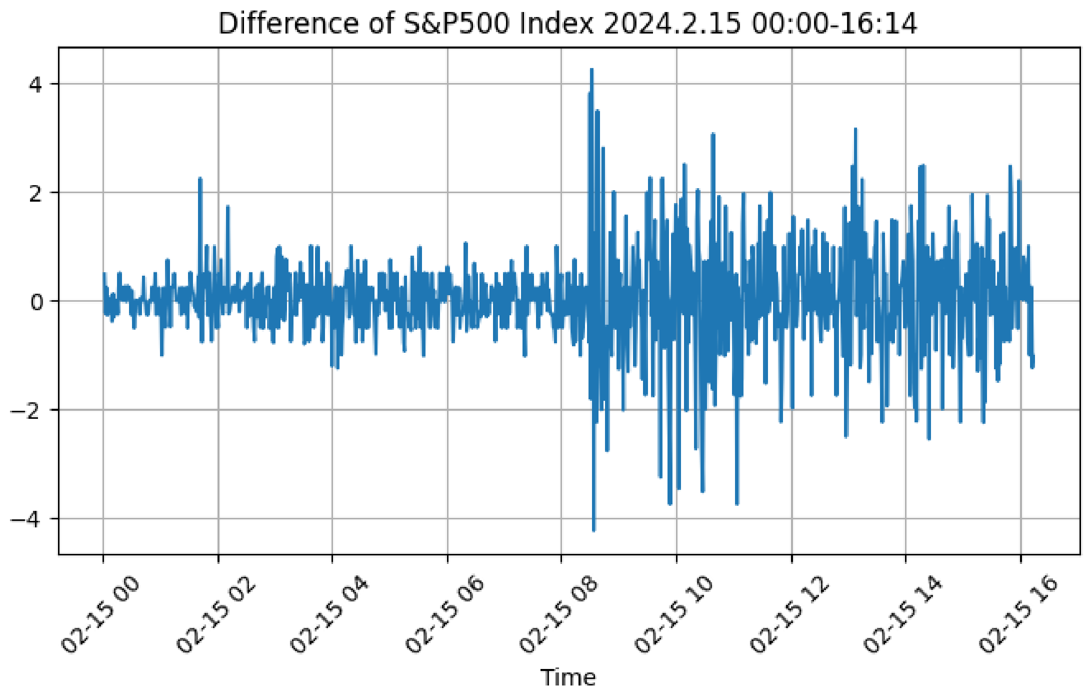


Fig 3. Detrended dataset of training data.

```
Results of ADF Test:
Test Statistic                 -1.755825e+01
P-value                         4.109165e-30
#Lags Used                      3.000000e+00
Number of Observations Used     9.690000e+02
Critical Value (1%)            -3.437116e+00
Critical Value (5%)            -2.864527e+00
Critical Value (10%)           -2.568361e+00
dtype: float64
```

Table 4 ADF test of the training data

The p-value is smaller than 0.05, so we reject the null hypothesis, meaning that the new series has no unit root and thus is stationary. [2] [5]

For the test set:

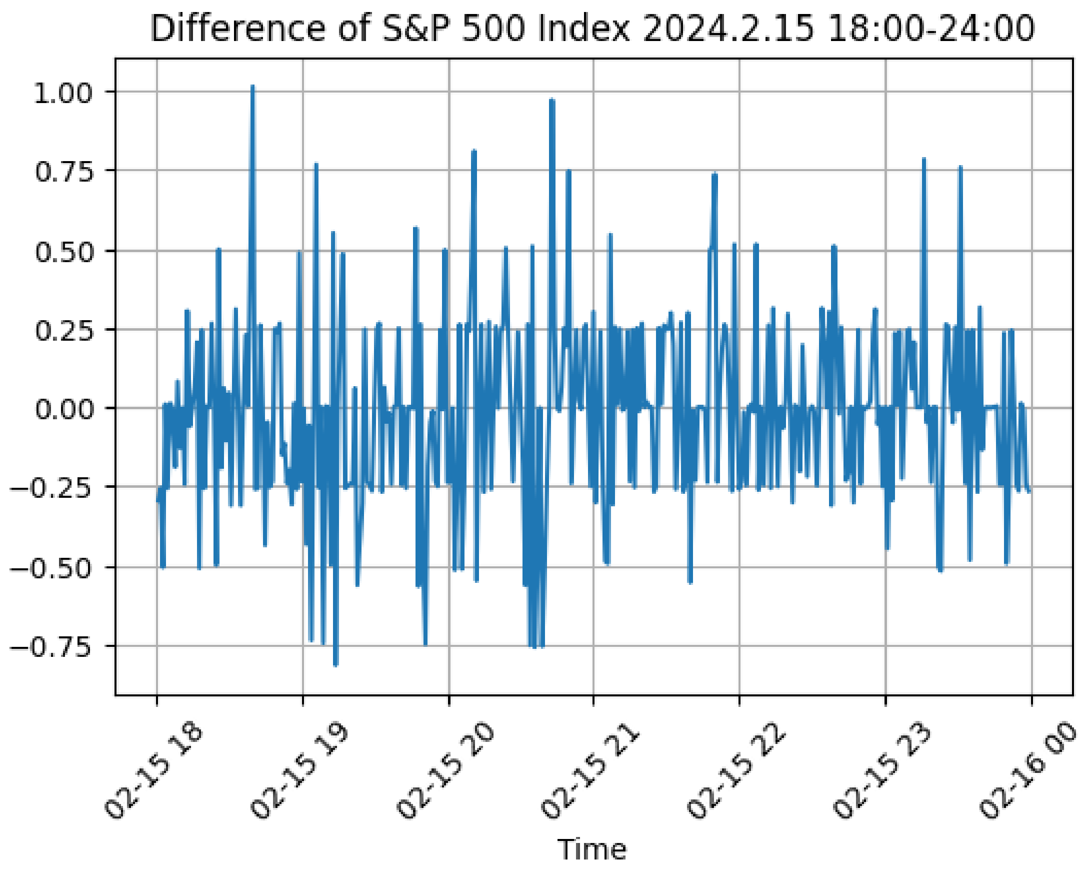
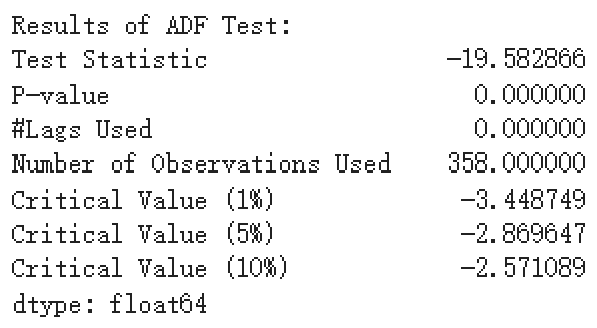


Fig 4. Detrended testing data.



```
Results of ADF Test:
Test Statistic                 -19.582866
P-value                          0.000000
#Lags Used                       0.000000
Number of Observations Used    358.000000
Critical Value (1%)             -3.448749
Critical Value (5%)             -2.869647
Critical Value (10%)            -2.571089
dtype: float64
```

Table 5 ADF test of the testing data

The p-values is zero, so we reject the null hypothesis, meaning that the new series has no unit root and thus is stationary. [2] [5]

Fractional Differencing

We draw graphs of ADF P-value by differencing order, as instructed by [3]. Since our data set is small, we choose the cutoff value a bit higher at 0.001.

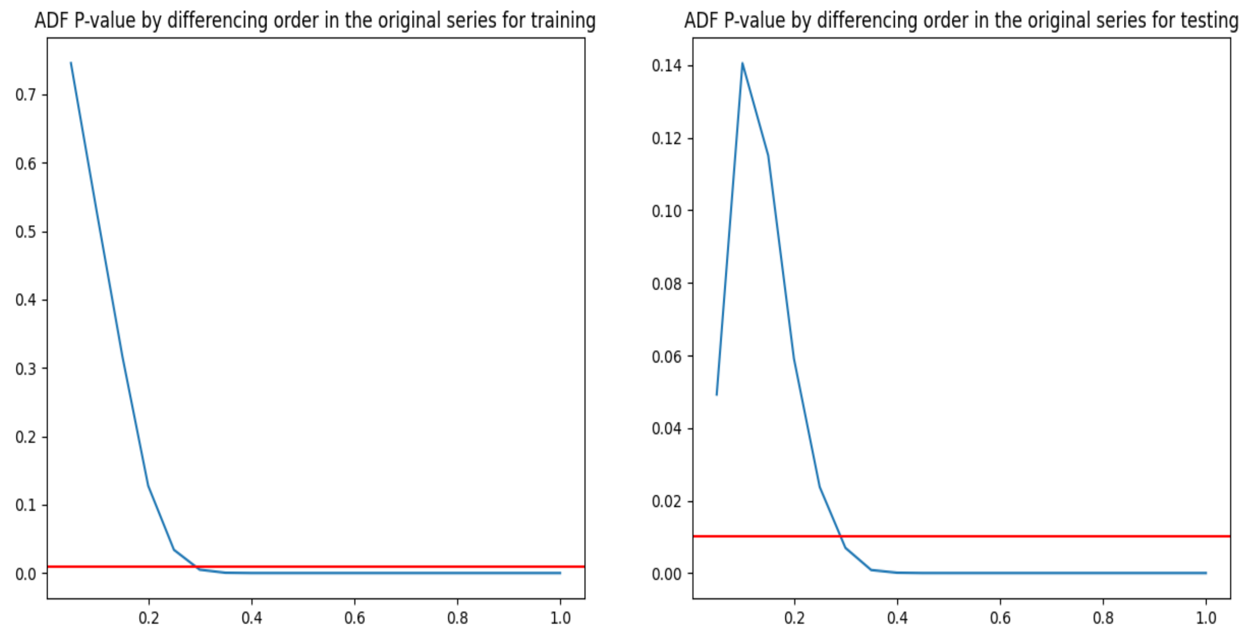


Fig. 5. Plots of ADF P-value by differencing order for training and testing data

From the graph we can see that 0.4 seems to be a good choice for the order of differencing. For analysis purpose, we still choose 0.6 as the order so that the cutoff lag would not be too large to greatly decrease the size of the dataset, which is 36.

We then draw the time series after factional differencing and run ADF test.

For the training set:

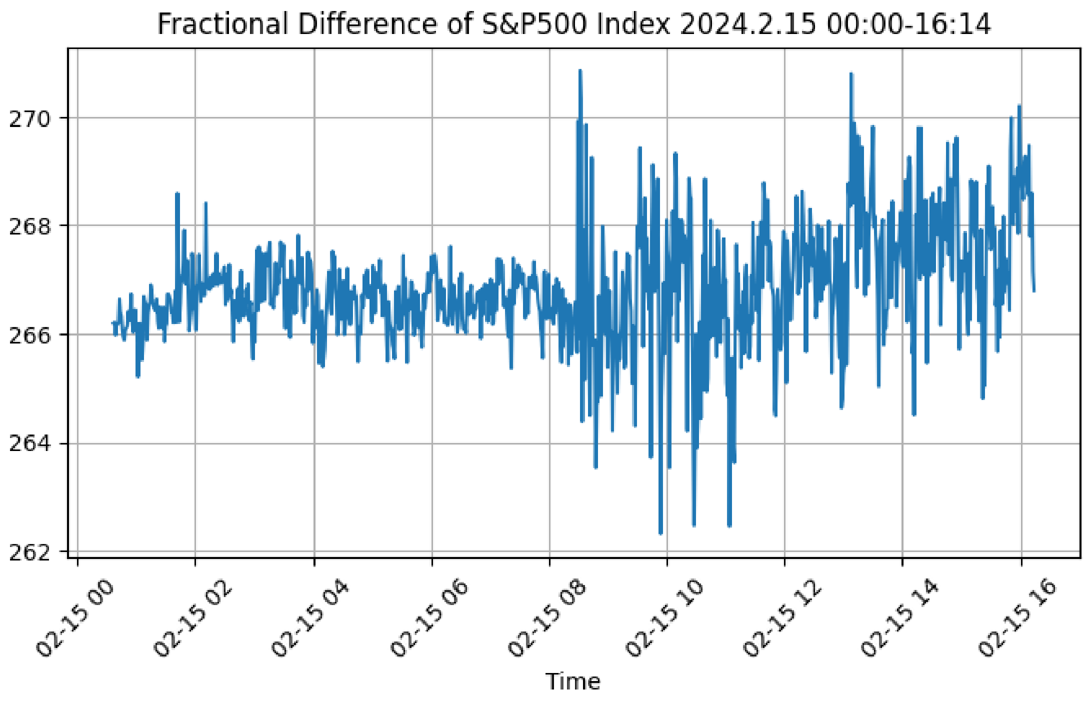
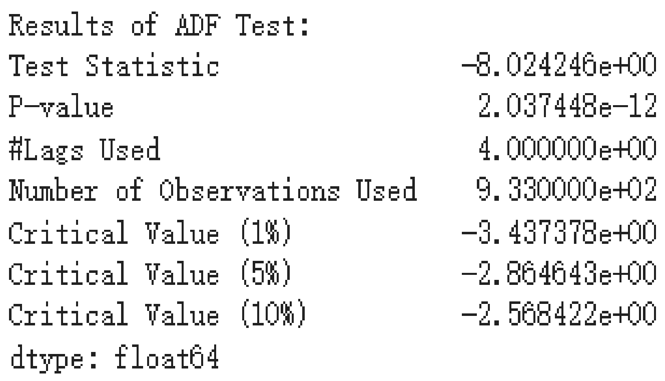


Fig. 6. Fractionally differenced training data



```
Results of ADF Test:
Test Statistic                -8.024246e+00
P-value                        2.037448e-12
#Lags Used                     4.000000e+00
Number of Observations Used    9.330000e+02
Critical Value (1%)           -3.437378e+00
Critical Value (5%)           -2.864643e+00
Critical Value (10%)          -2.568422e+00
dtype: float64
```

Table 6. ADF test of the Fractionally differenced training data

For the test set:

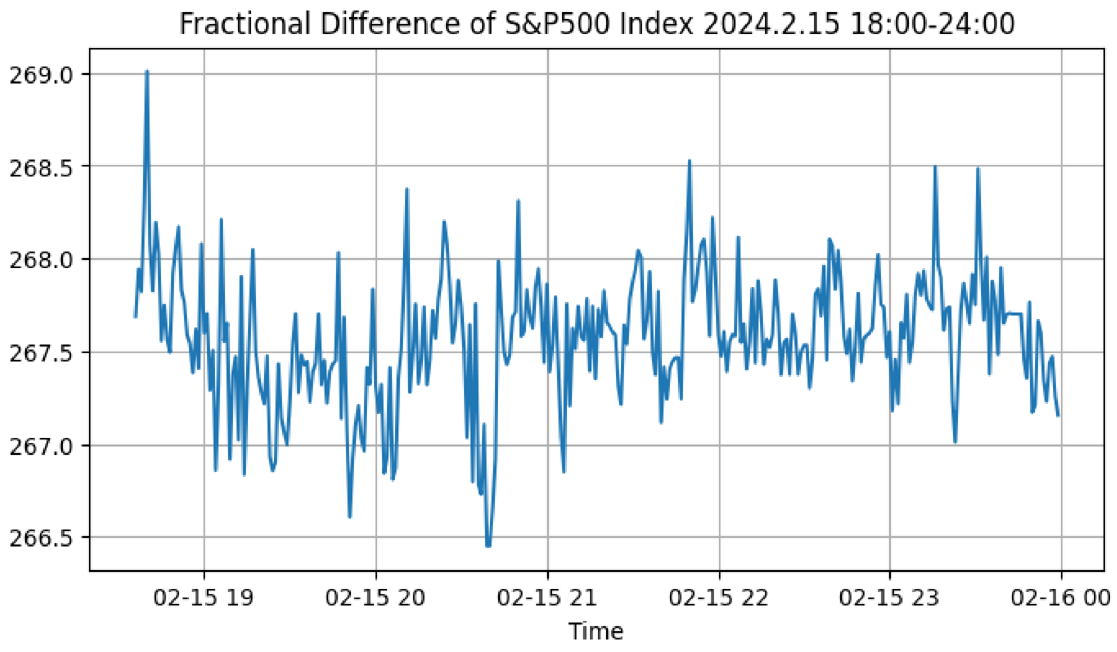
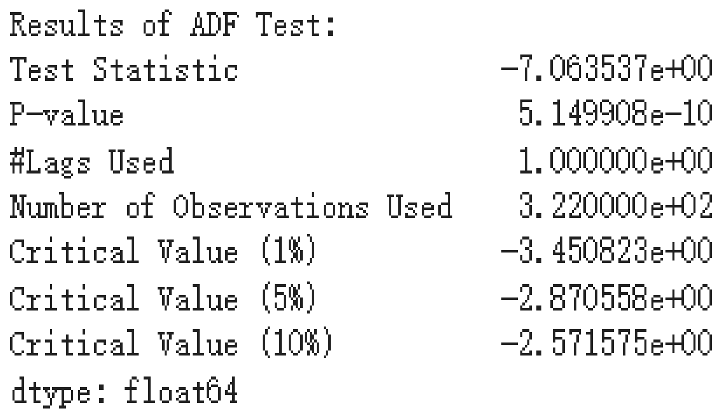


Fig. 7. Fractionally differenced testing data



```
Results of ADF Test:
Test Statistic                 -7.063537e+00
P-value                         5.149908e-10
#Lags Used                      1.000000e+00
Number of Observations Used     3.220000e+02
Critical Value (1%)            -3.450823e+00
Critical Value (5%)            -2.870558e+00
Critical Value (10%)           -2.571575e+00
dtype: float64
```

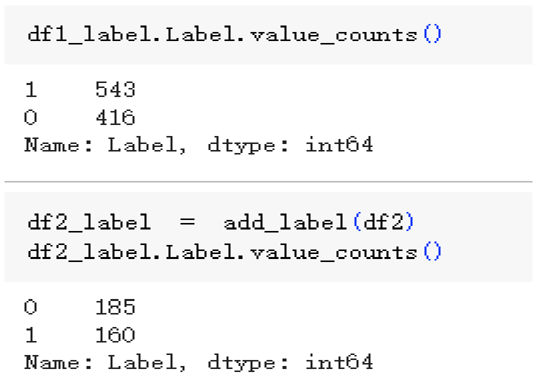
Table 7. ADF test of the Fractionally differenced testing data

The p-values are small, which suggests that both time series after fractional difference are stationary.

Step 2: MLP

Set-up

We define the label as whether the next 15 min return is positive. We use df1 to denote the dataframes associated with the training set, and df2 those associated with the test set.



```
df1_label.Label.value_counts()
```

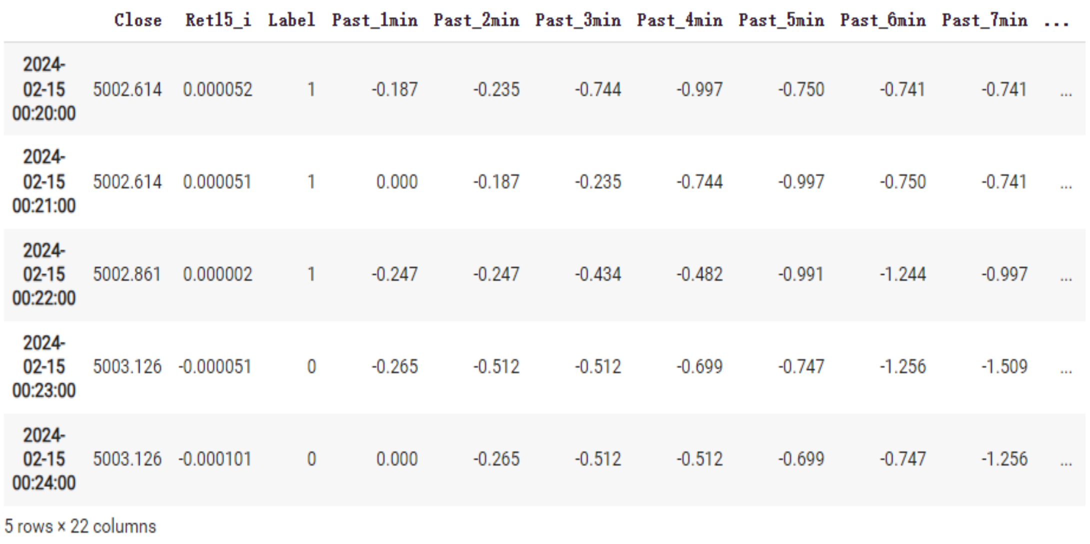
```
1    543
0    416
Name: Label, dtype: int64
```

```
df2_label = add_label(df2)
df2_label.Label.value_counts()
```

```
0    185
1    160
Name: Label, dtype: int64
```

As we need to compare MLP with CNN on GAF later, we will use the same length of past information. For demonstration purpose, if we want to use window size 20 in the GAP representation, then we can use 20 data points as length of past information. Therefore, we simply use the past 19 minutes' data plus the current minute data as features (not taking rolling averages as usual).

Notice that for the original time series, the absolute value is quite large. Given limited training data, we can subtract the "Close" from all past values so that neural networks can be trained more easily. The training data looks like the table below where "Past_{}min"s are features.



	Close	Ret15_i	Label	Past_1min	Past_2min	Past_3min	Past_4min	Past_5min	Past_6min	Past_7min	...
2024-02-15 00:20:00	5002.614	0.000052	1	-0.187	-0.235	-0.744	-0.997	-0.750	-0.741	-0.741	...
2024-02-15 00:21:00	5002.614	0.000051	1	0.000	-0.187	-0.235	-0.744	-0.997	-0.750	-0.741	...
2024-02-15 00:22:00	5002.861	0.000002	1	-0.247	-0.247	-0.434	-0.482	-0.991	-1.244	-0.997	...
2024-02-15 00:23:00	5003.126	-0.000051	0	-0.265	-0.512	-0.512	-0.699	-0.747	-1.256	-1.509	...
2024-02-15 00:24:00	5003.126	-0.000101	0	0.000	-0.265	-0.512	-0.512	-0.699	-0.747	-1.256	...

5 rows × 22 columns

Table 8. Training data for the NN (the past 19 minutes' data and the current minute data)

For the other two stationary time series, we may choose to keep the way it is.

Original Time Series

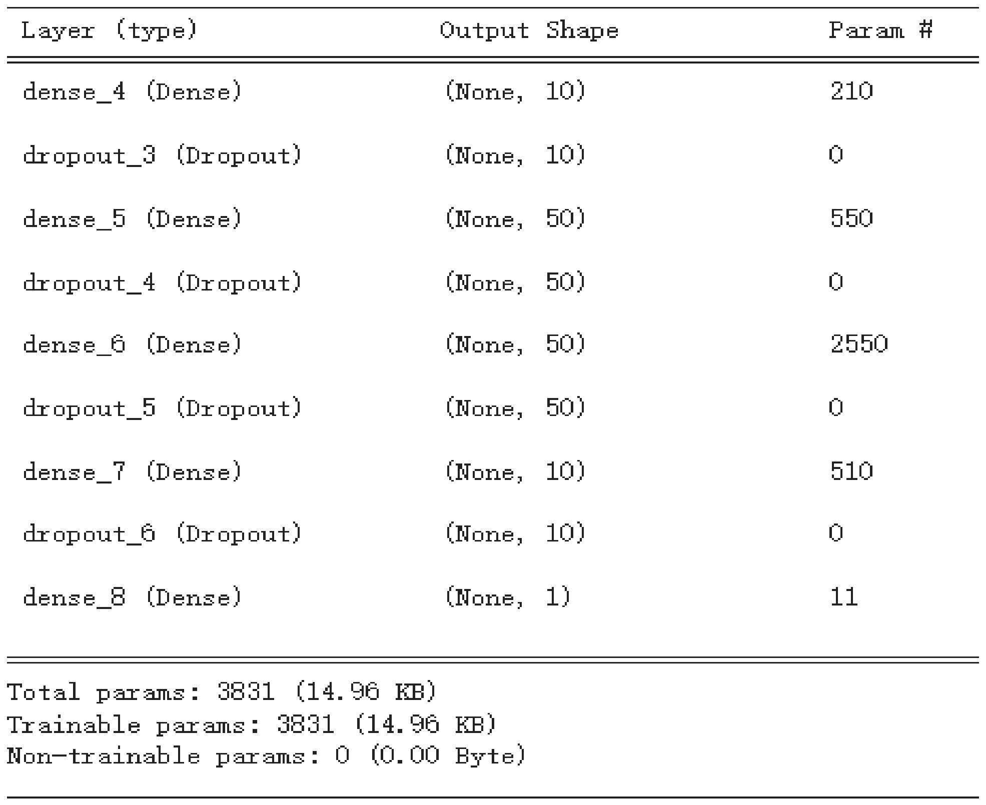
For the original time series, we have subtracted "Close" from the past information and therefore we can exclude it from the features.

We use Keras Tuner [4] for hyperparameter tuning. We ask the Keras Tuner to choose:

- The optimal dropout rate after each hidden layer of 0, 0.2 or 0.3
- The optimal number of layers (between 1 and 5)
- The optimal number of units that each of the selected layer (between 10 and 50)

We use 30% of the training set as validation set during hyperparameter tuning, and the best set of hyperparameters is selected according to the minimum validation loss. We then re-train the model with the best set of hyperparameters and evaluate the model on the test set in terms of accuracy, F1-score and confusion matrix.

Due to the nature of the problem setup, this hyperparameter tuning process is very unstable. We found that the following network structure can produce good results. **Therefore, we will stick with this structure as shown in the figure below and will try it for the other time series as well.**



```
Layer (type)                 Output Shape              Param #
==================================================================
dense_4 (Dense)              (None, 10)                210

dropout_3 (Dropout)          (None, 10)                0

dense_5 (Dense)              (None, 50)                550

dropout_4 (Dropout)          (None, 50)                0

dense_6 (Dense)              (None, 50)                2550

dropout_5 (Dropout)          (None, 50)                0

dense_7 (Dense)              (None, 10)                510

dropout_6 (Dropout)          (None, 10)                0

dense_8 (Dense)              (None, 1)                 11

==================================================================
Total params: 3831 (14.96 KB)
Trainable params: 3831 (14.96 KB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________
```

Table. 9. MLP neural network structure

So, the test results are:

```
11/11 [==============================] - 0s 3ms/step - loss: 0.7098 - accuracy: 0.5307
Accuracy over test: 53.07%
11/11 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

         0.0       0.60      0.28      0.39       169
         1.0       0.51      0.80      0.62       157

    accuracy                           0.53       326
   macro avg       0.55      0.54      0.50       326
weighted avg       0.56      0.53      0.50       326
```

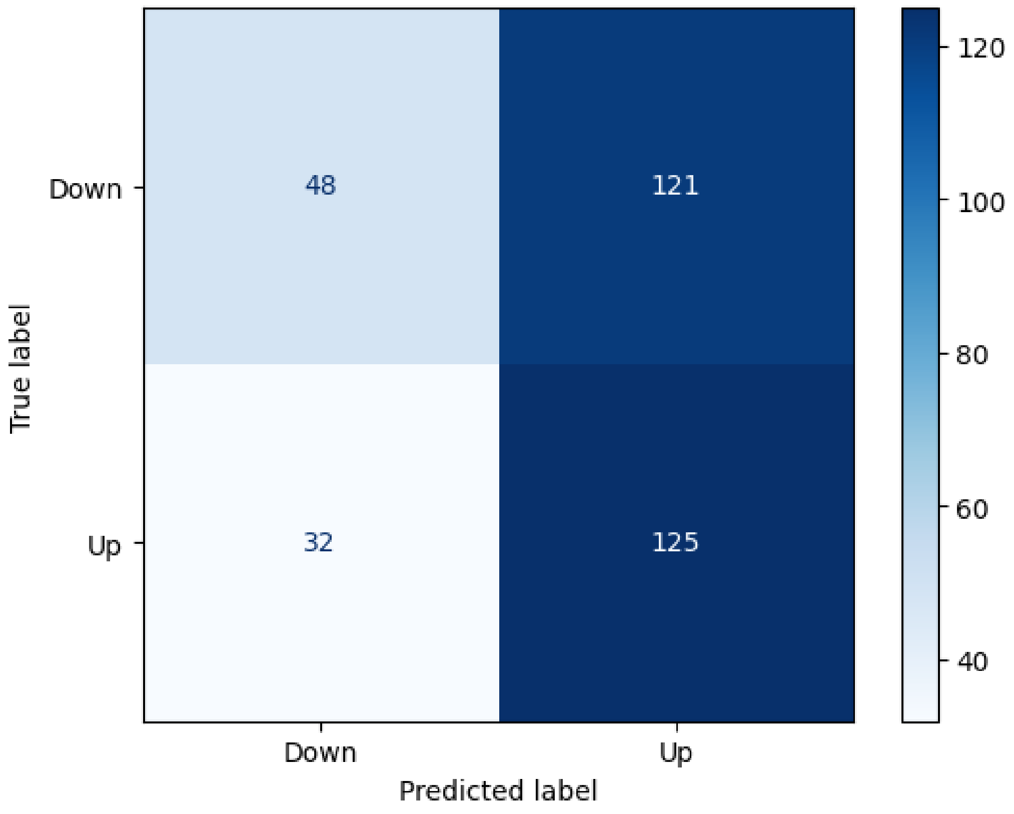


Fig. 8. Confusion matrix of the classification results by MLP neural network trained with the original data.

Stationary Time Series

Since the values of the 1st order differenced data are small, we do not subtract "Close" from features and keep "Close" in the features.

The test results are:

```
11/11 [==============================] - 0s 3ms/step - loss: 0.6938 - accuracy: 0.4954
Accuracy over test: 49.54%
11/11 [==============================] - 0s 3ms/step
              precision    recall  f1-score   support

         0.0       0.55      0.13      0.21       168
         1.0       0.49      0.89      0.63       157

    accuracy                           0.50       325
   macro avg       0.52      0.51      0.42       325
weighted avg       0.52      0.50      0.41       325
```

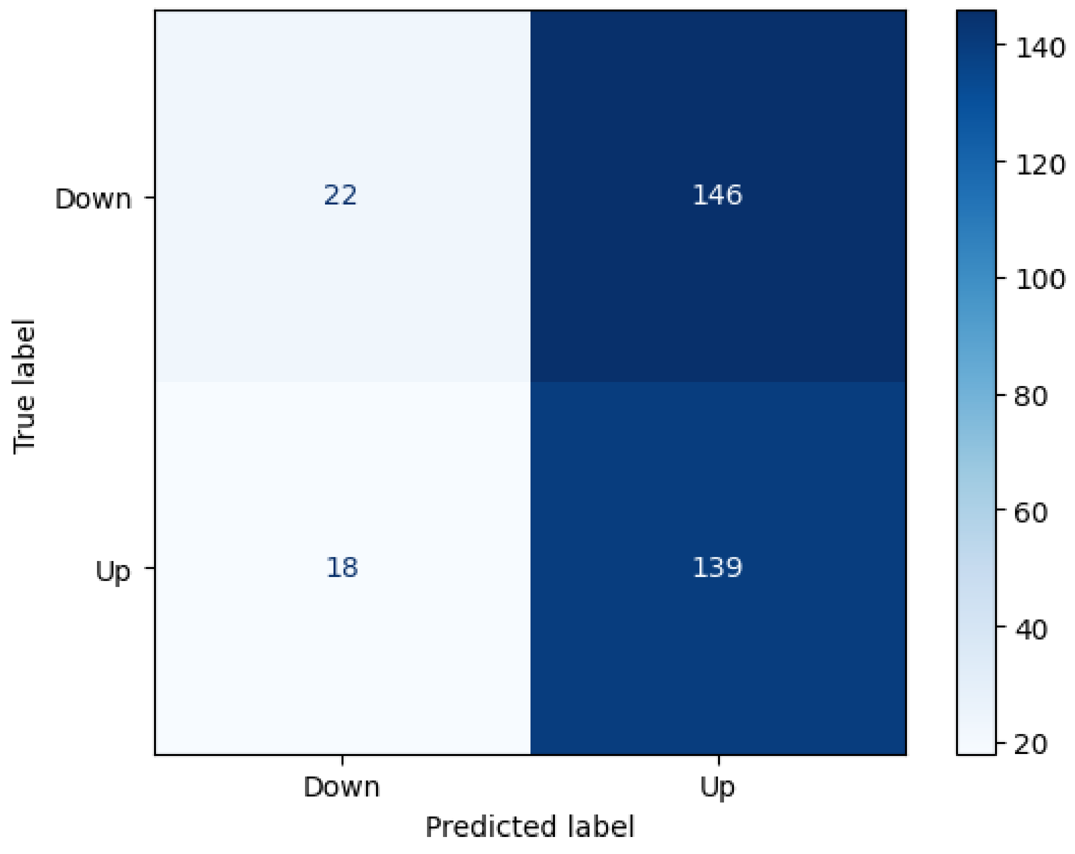


Fig. 9. Confusion matrix of the classification results by MLP neural network trained with the detrended data. (the Stationary Time Series)

Fractional Differencing

We first tried modeling without subtracting "Close" from the features. The test results are:

```
10/10 [==============================] - 0s 9ms/step - loss: 0.7283 - accuracy: 0.4828
Accuracy over test: 48.28%
10/10 [==============================] - 0s 3ms/step
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00       150
         1.0       0.48      1.00      0.65       140

    accuracy                           0.48       290
   macro avg       0.24      0.50      0.33       290
weighted avg       0.23      0.48      0.31       290

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedM
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedM
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedM
  _warn_prf(average, modifier, msg_start, len(result))
```

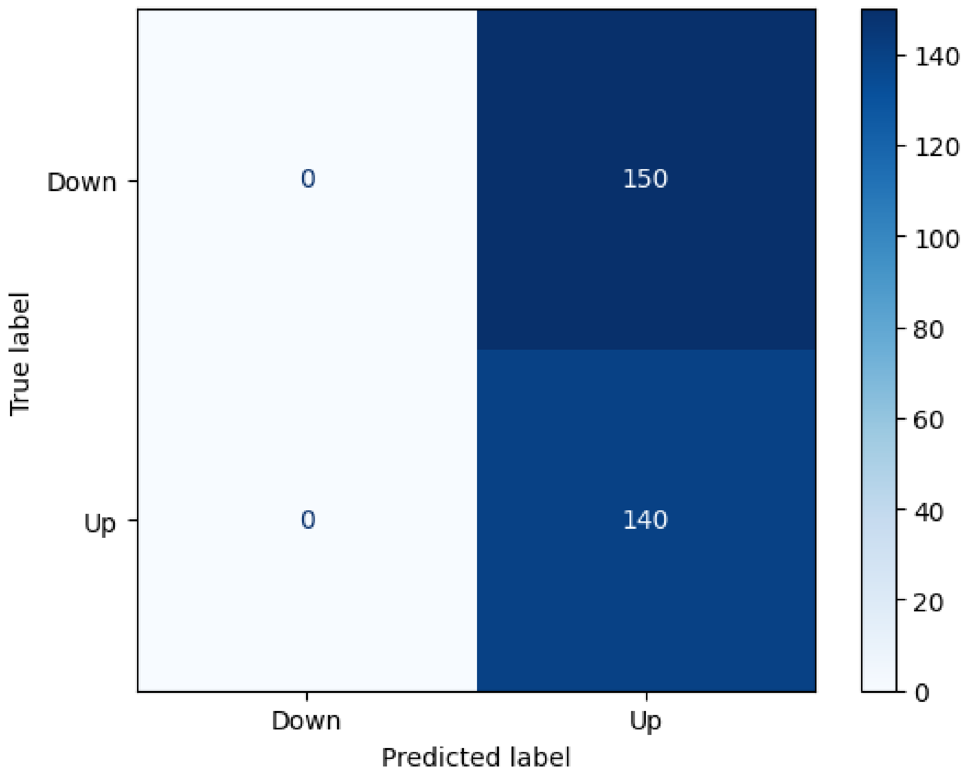


Fig. 10. Confusion matrix of the classification results by MLP neural network trained with the detrended data. (the Fractional Differenced Time Series with 'Close' feature)

The model was not properly trained since it cannot make "Down" predictions. We tried hyperparameter tuning but it produces the same result.

Then we train the model with the selected structure again but subtracting "Close" from the features. The test results are:

```
10/10 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5241
Accuracy over test: 52.41%
10/10 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

         0.0       0.61      0.23      0.33       150
         1.0       0.50      0.84      0.63       140

    accuracy                           0.52       290
   macro avg       0.56      0.53      0.48       290
weighted avg       0.56      0.52      0.48       290
```

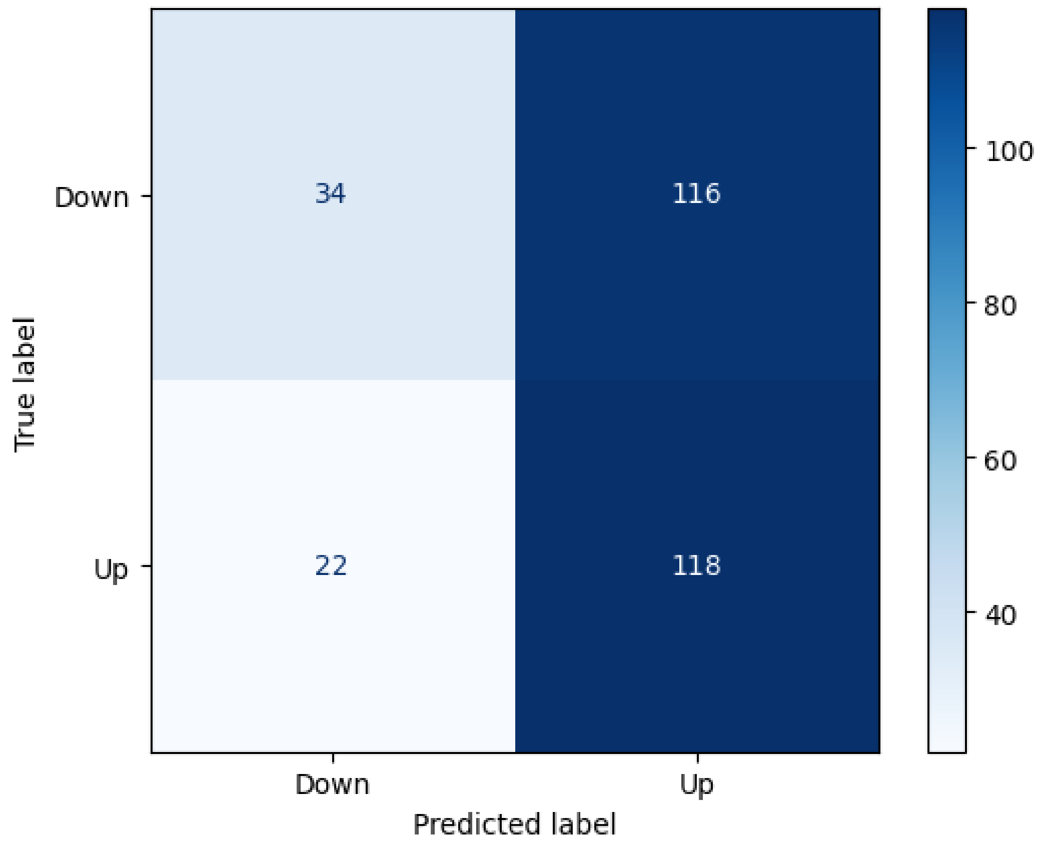


Fig. 11. Confusion matrix of the classification results by MLP neural network trained with the detrended data. (the Fractional Differenced Time Series without 'Close' feature)

Comparison of the results of MLP

We summarize the test results in the following table to compare the performances. Notice that all models summarized below are of the same structure shown before.

Input Data	**Accuracy**	**F1-score**	**Pct of Down Prediction**
Original	0.5307	0.62	$\frac{80}{80+246}=0.2454$
1st Order Difference	0.4954	0.63	$\frac{40}{40+285}=0.1231$
Fractional Difference	0.5241	0.63	$\frac{56}{56+234}=0.1931$

Table. 10. Comparison of the results of MLP

The time series from 1st order differencing has the worst results, which is exactly what [3] points out that such method loses too much past information. The other two models yield similar results in terms of accuracy, which indicates that the partial past information retained by fractional differencing does have some merit.

The comparison result seems to indicate that neural networks can work with non-stationary data. However,

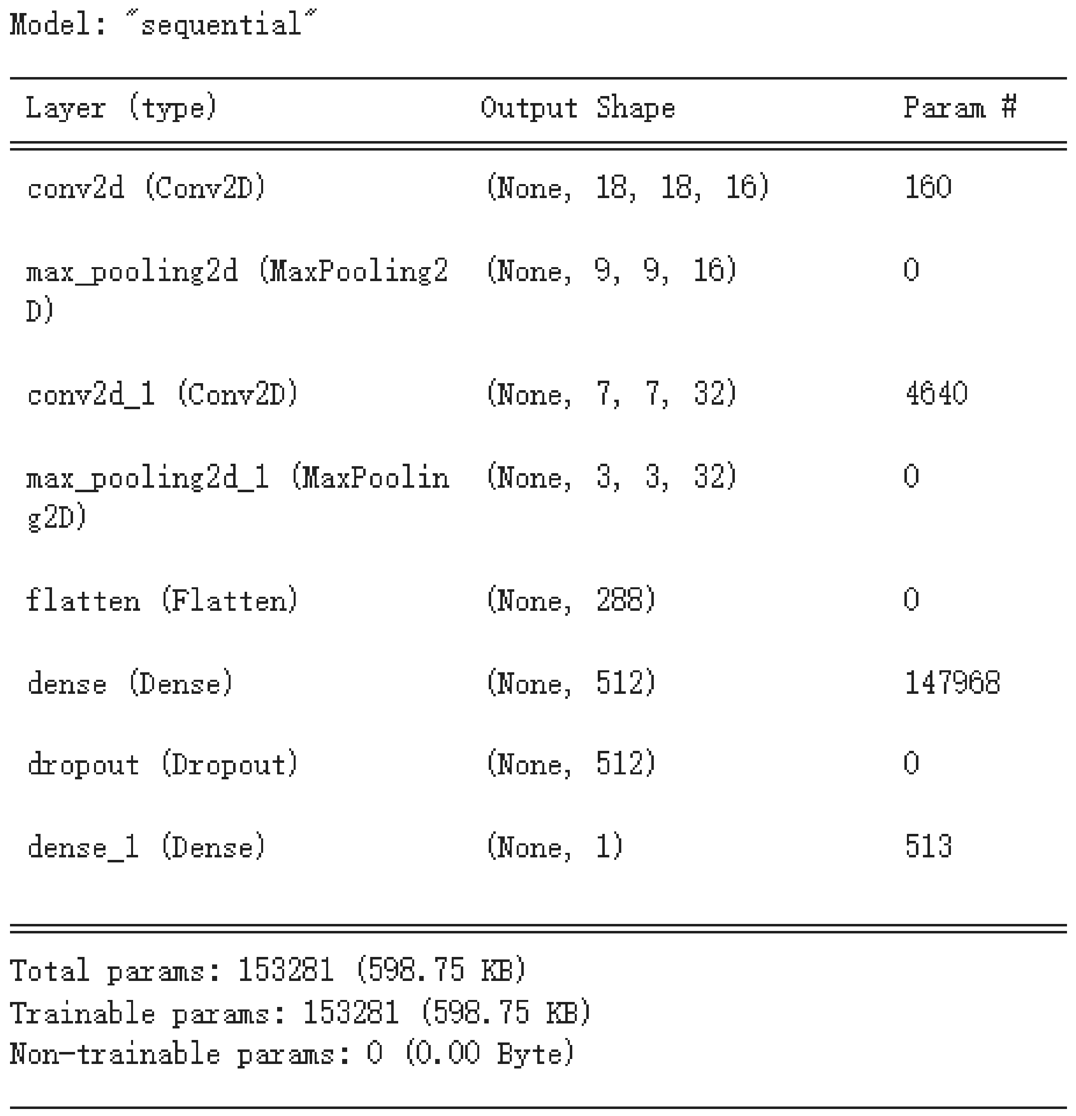
- We do need to keep the values of input features small to facilitate training MLP.
- The original time series has a narrow range, which may overshadow the merit of stationarity.

Another thing worth mentioning is that the percentage of "Down" predictions is very small, considering both the training and test data are quite balanced, which may indicate that the models haven't learned many useful patterns, or it is nearly impossible to tell the labels apart.

Step 3: CNN on GAF

Set-up

We set the window size of GAF as 20, and use the following CNN structure:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 18, 18, 16)        160

 max_pooling2d (MaxPooling2  (None, 9, 9, 16)          0
 D)

 conv2d_1 (Conv2D)           (None, 7, 7, 32)          4640

 max_pooling2d_1 (MaxPoolin  (None, 3, 3, 32)          0
 g2D)

 flatten (Flatten)           (None, 288)               0

 dense (Dense)               (None, 512)               147968

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 153281 (598.75 KB)
Trainable params: 153281 (598.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Table 11. CNN with GAF neural network structure

Notice that it has way more trainable params (153281) than the MLP used in step 2, which has only 3831 ones.

We will use GAF result * 255 as the input image for CNN.

Original Time Series

GAF representation of training samples:

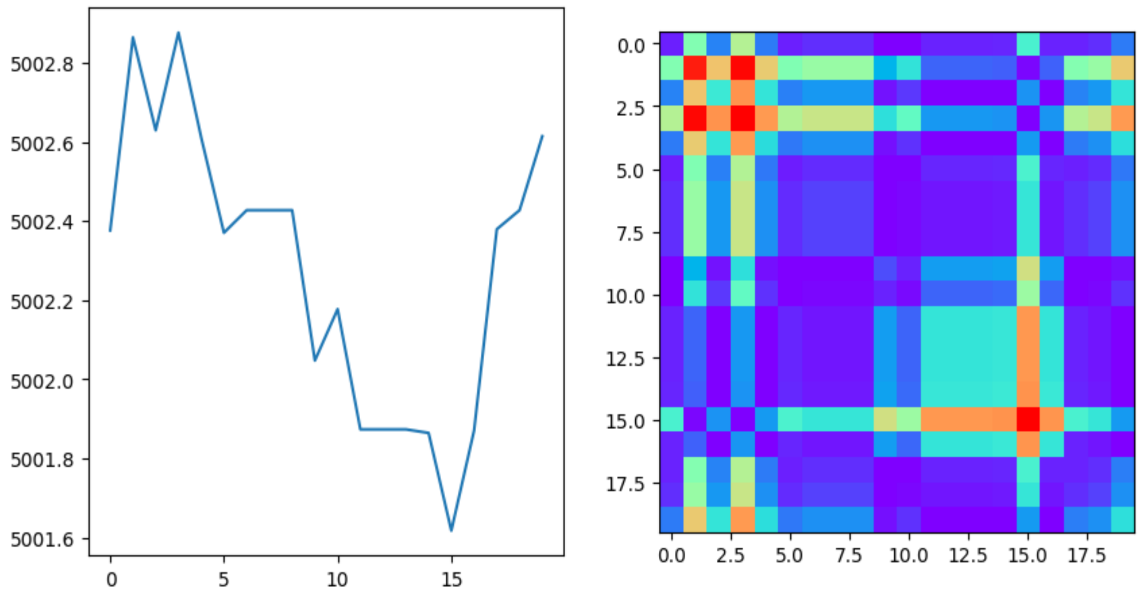


Fig. 12. Results after GAF transformation of the original data for the first time series

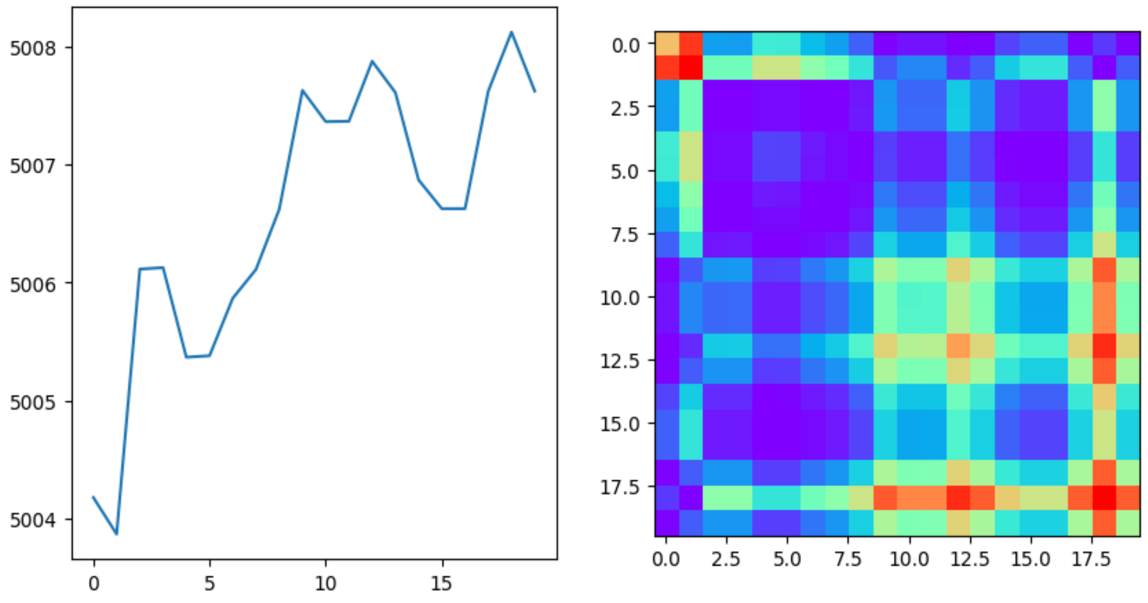


Fig. 13. Results after GAF transformation of the original data for the 100th time series

Red-ish values on the diagonal (top left to bottom right) represent extreme values in the original time series.

The evaluation on the test set:

```
11/11 [==============================] - 0s 4ms/step - loss: 1.0633 - accuracy: 0.5046
Accuracy over test: 50.46%
11/11 [==============================] - 0s 4ms/step
              precision    recall  f1-score   support

         0.0       0.52      0.49      0.50       168
         1.0       0.49      0.52      0.50       157

    accuracy                           0.50       325
   macro avg       0.51      0.51      0.50       325
weighted avg       0.51      0.50      0.50       325
```

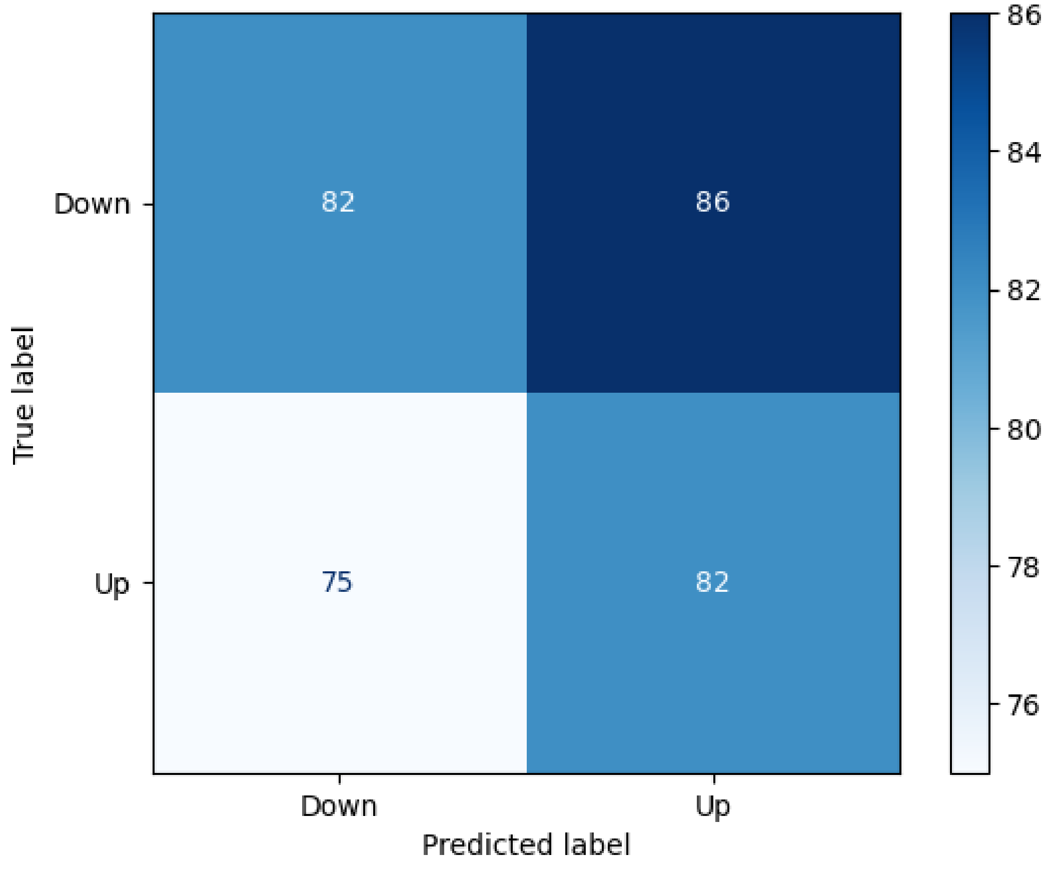


Fig. 14. Confusion matrix of the result from CNN after GAF transformation of the original data

Stationary Time Series

GAF representation of training samples:

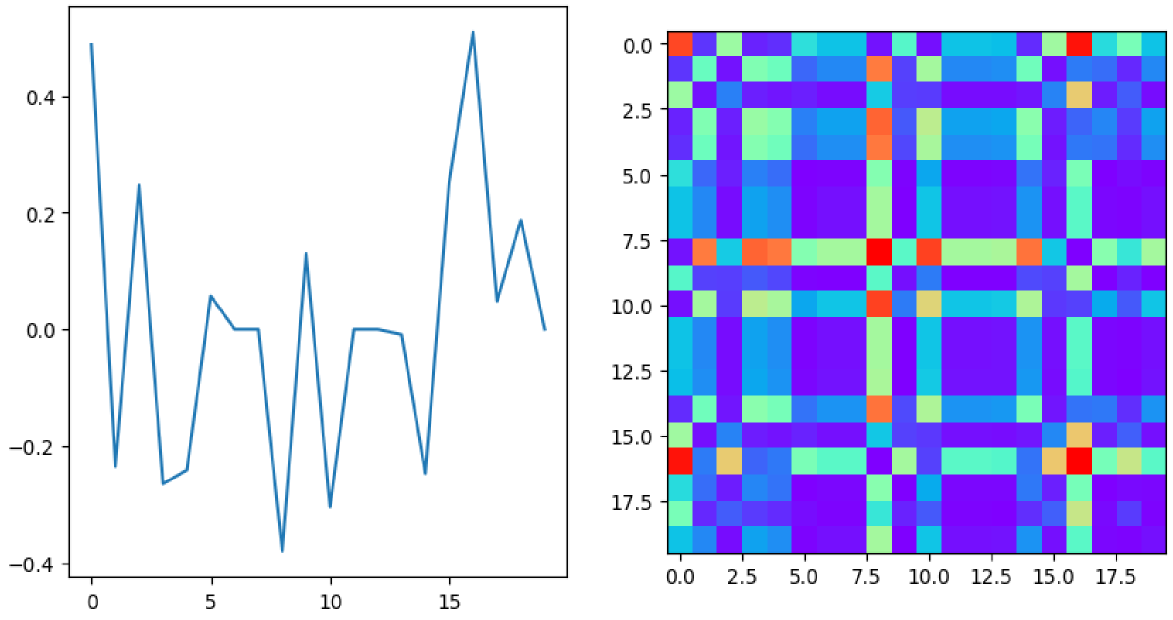


Fig. 15. Results after GAF transformation of the stationary data for the first time series

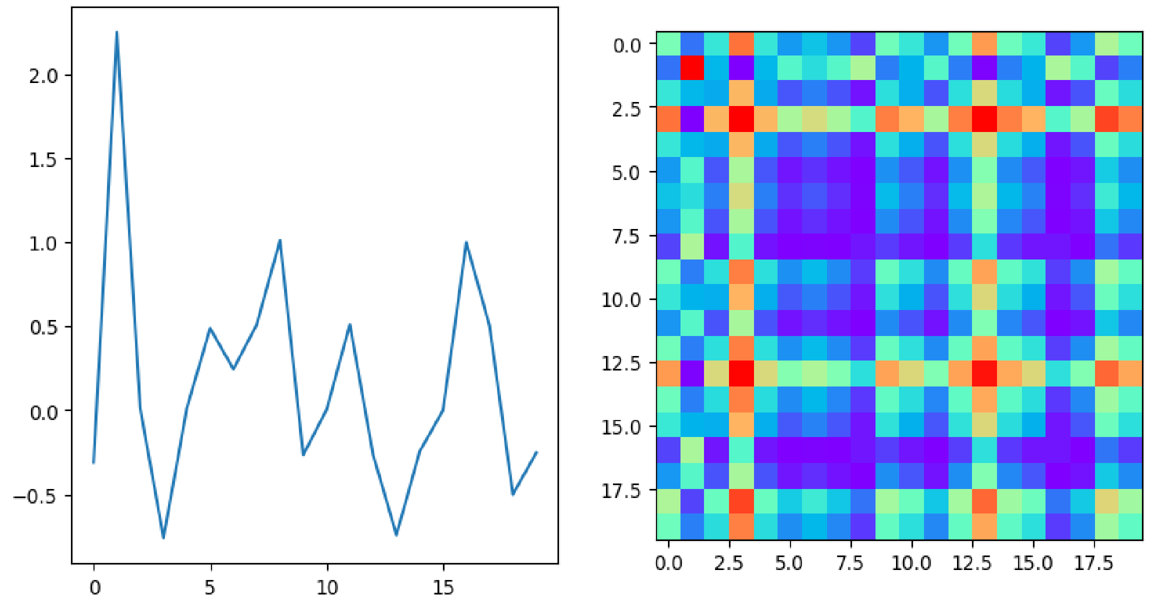


Fig. 16. Results after GAF transformation of the stationary data for the 100th time series

The signals are quite many and scattered, which is not a good sign.

The evaluation on the test set:

```
11/11 [==============================] - 0s 8ms/step - loss: 0.7333 - accuracy: 0.4599
Accuracy over test: 45.99%
11/11 [==============================] - 0s 6ms/step
              precision    recall  f1-score   support

         0.0       0.45      0.22      0.29       167
         1.0       0.46      0.72      0.56       157

    accuracy                           0.46       324
   macro avg       0.46      0.47      0.43       324
weighted avg       0.46      0.46      0.42       324
```

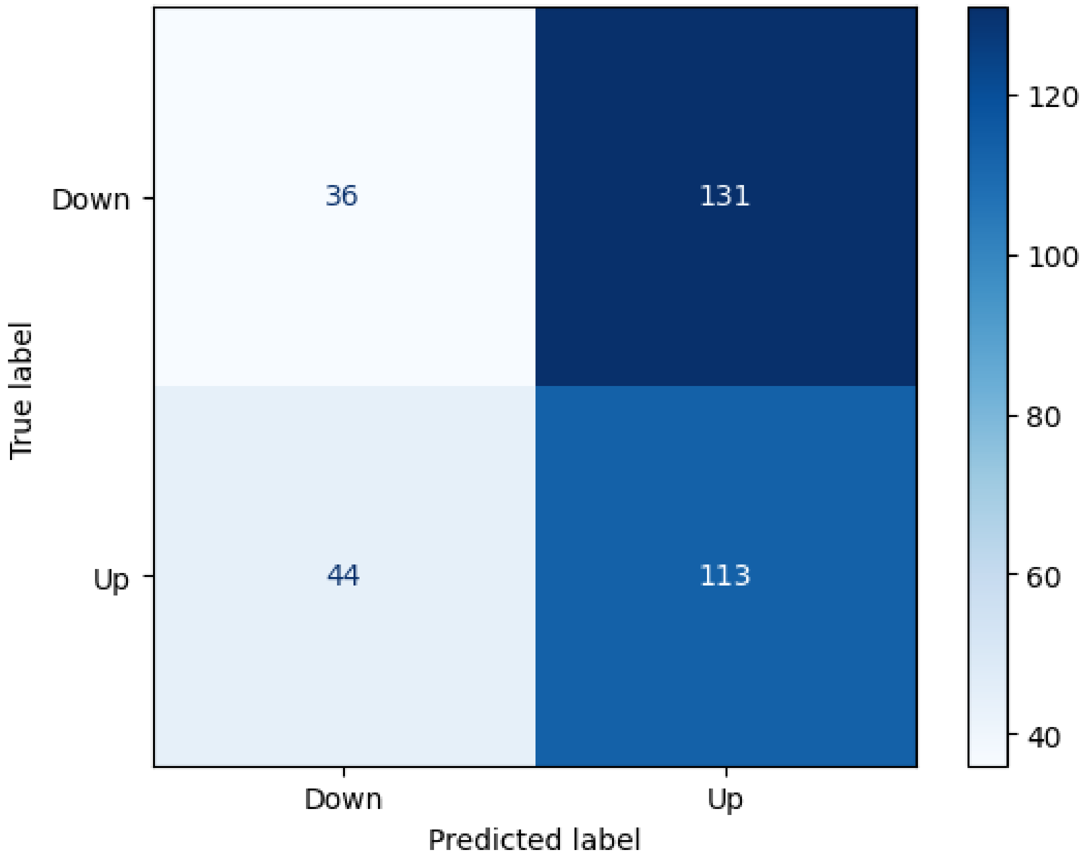


Fig. 17. Confusion matrix of the result from CNN after GAF transformation of the stationary data

The performance is poor.

Fractional Differencing

GAF representation of training samples:

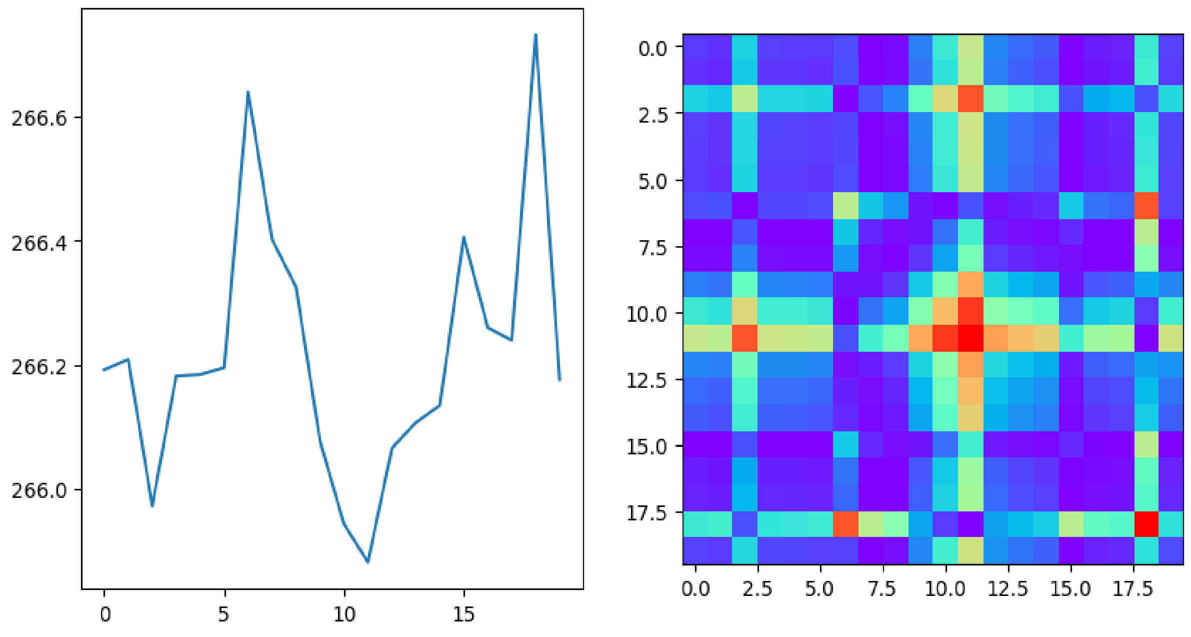


Fig. 18. Results after GAF transformation of the Fractionally Differenced data for the first time series

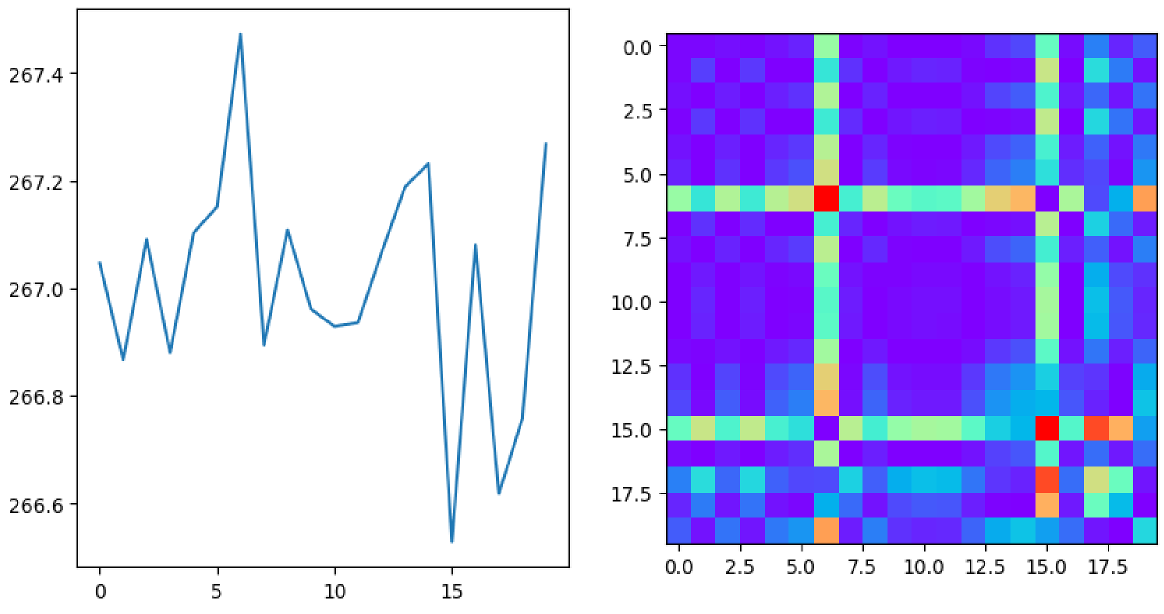


Fig. 19. Results after GAF transformation of the Fractionally Differenced data for the 100th time series

The evaluation on the test set:

```
10/10 [==============================] - 0s 4ms/step - loss: 0.8838 - accuracy: 0.4983
Accuracy over test: 49.83%
10/10 [==============================] - 0s 4ms/step
              precision    recall  f1-score   support

         0.0       0.57      0.11      0.18       149
         1.0       0.49      0.91      0.64       140

    accuracy                           0.50       289
   macro avg       0.53      0.51      0.41       289
weighted avg       0.53      0.50      0.40       289
```

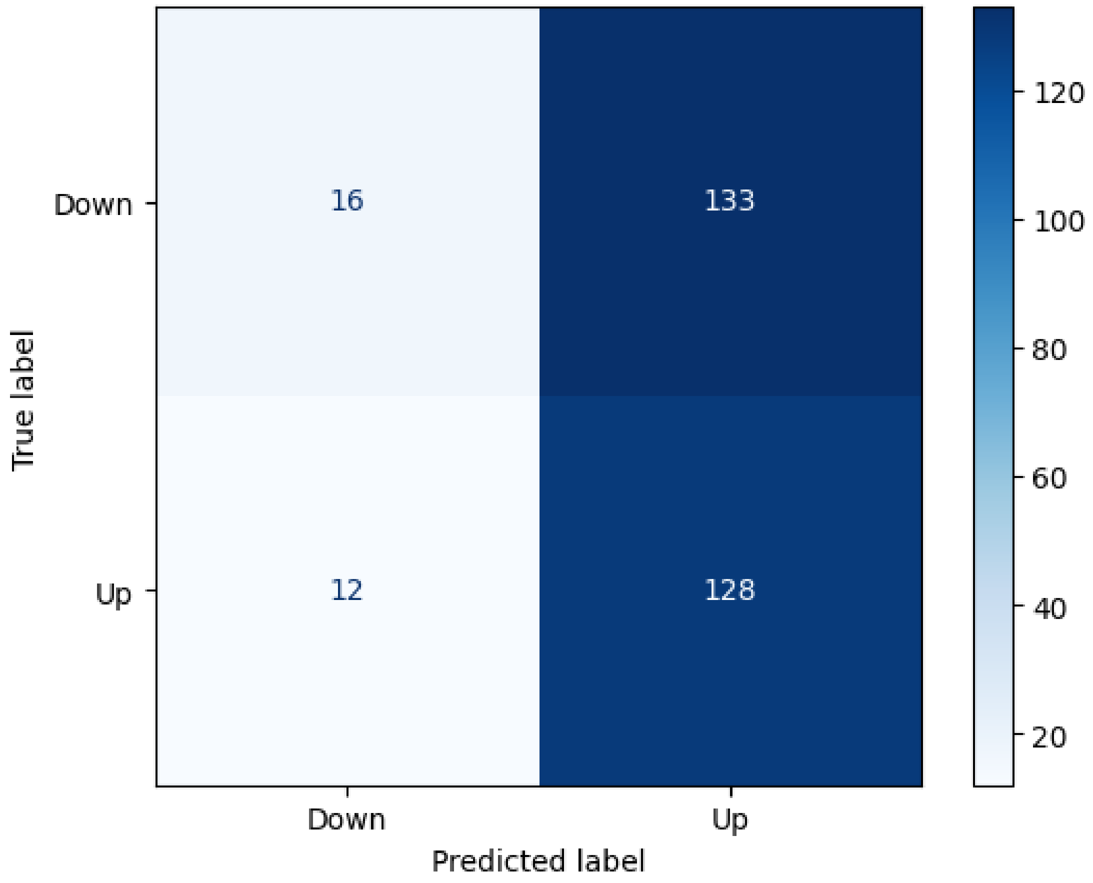


Fig. 20. Confusion matrix of the result from CNN after GAF transformation of the Fractionally Differenced data

Comparison of the results from CNN with GAF transformation

Input Data	Accuracy	F1-score	Pct of Down Prediction
Original	0.5307	0.50	$\frac{82+75}{82+75+82+86} = 0.4831$
1st Order Difference	0.4599	0.56	$\frac{36+44}{36+44+131+113} = 0.2469$
Fractional Difference	0.4983	0.64	$\frac{16+12}{16+12+133+128} = 0.0969$

Table 12. Comparison of the results from CNN with GAF transformation

We may need to have more training samples to train CNN properly since it has 153,281 trainable parameters, though about 70% of them are in the last fully connected layer.

Based on the results we have, again, the model with 1st order difference as input yields the worst performance in terms of accuracy, similar to MLP.

The model with the original time series as input outperforms the other two in terms of accuracy and balanced predictions. It indicates that with CNN, we prefer to keep the time series at it is, which needs not to be stationary. One possible explanation is that neural networks can find and extract useful features pertinent to the task at hand on their own, and differencing can be seen as part of feature engineering, which may not be optimal for the task. Also, as [3] points out, information is lost after differencing, whether fractional or not.

Notice that among all six models summarized in step 2 & 3, CNN on the GAF representation of the original time series is the only one that can have a balanced prediction on a balanced dataset, which seems to be the best choice for this task and data.

References:

1. https://www.histdata.com/download-free-forex-historical-data/?/metatrader/1-minute-bar-quotes/spxusd/2023
2. R.I.D. Harris, Testing for unit roots using the augmented Dickey-Fuller test: Some issues relating to the size, power and the lag structure of the test, Economics Letters Volume 38, Issue 4, April 1992, Pages 381-386, ScienceDirect, Elsevier, https://www.sciencedirect.com/science/article/abs/pii/016517659290022Q
3. https://www.kaggle.com/code/elvisesp/time-series-analysis-using-fractional-differencing/notebook
4. https://www.tensorflow.org/tutorials/keras/keras_tuner?hl=en
5. Wolters, J., Hassler, U. Unit root testing. 43–58 (2006). https://doi.org/10.1007/s10182-006-0220-6