GROUP WORK PROJECT 01
GROUP NUMBER: 8458

MScFE 642: Deep Learning for Finance

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Sujan Antony S	India	sonnyantony05@gmail.com	
Regulavalasa Krishna Vamsi	India	krishnavamsi8262@gmail.com	
Phan Van Nguyen	Vietnam	nphanvan10@gmail.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	**Sujan Antony S**
Team member 2	**Regulavalasa Krishna Vamsi**
Team member 3	**Phan Van Nguyen**

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.

Step 1

a. Original Time Series (Bank Rakyat Indonesia)

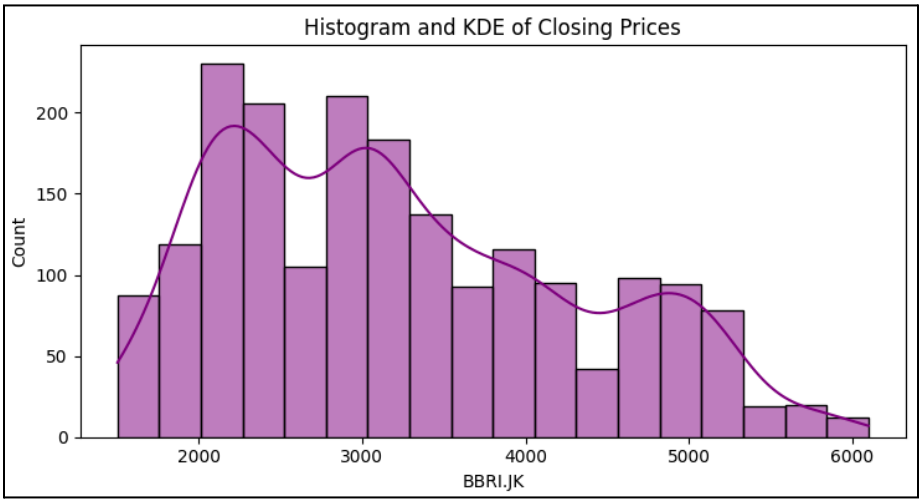
The chosen Time Series for this analysis is Bank Rakyat Indonesia listed on the Indonesia Stock Exchange.

Summary Statistics:

Mean	3242
Standard Deviation	1074
Minimum	1504
25th Percentile	2324
50th Percentile	3054
75th Percentile	3995
Maximum	6098
Skewness	0.49

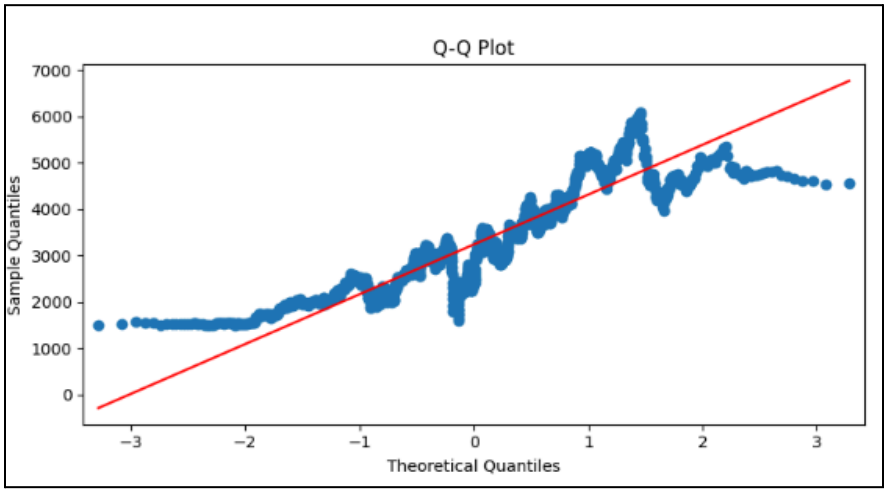
The values mentioned in the table suggest that the price range is wide with a slight positive skew.

Histogram & KDE:



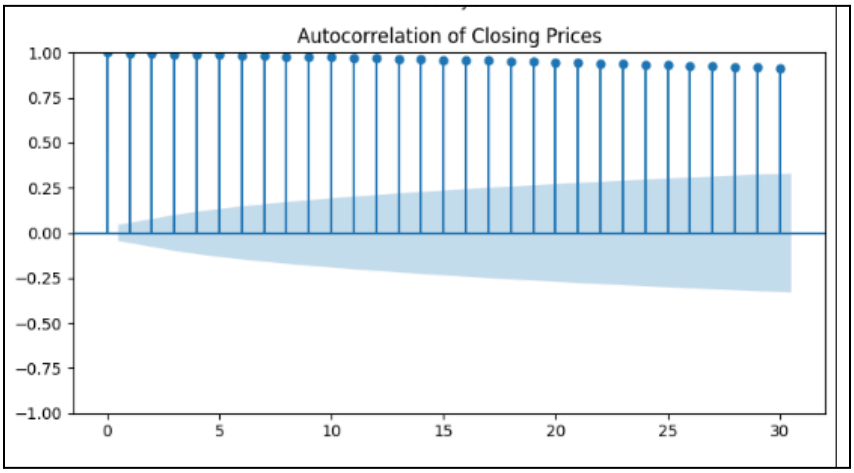
The Histogram plot appears to be roughly bimodal but with a right skew.

QQ-Plot:



The divergence from the straight line in QQ- Plot indicates that the distribution is not perfectly normal (particularly in the tails).

Persistence & Autocorrelation:



The Autocorrelation Function (ACF) figure demonstrates considerable positive autocorrelation for multiple lags, indicating a high level of persistence. This is consistent with price movements that resemble a random walk.

Box Plot:

The box plot shows that there are no extreme outliers above the maximum value observed, thus while the range is wide, the data are generally contained within a reasonable spread around the median.

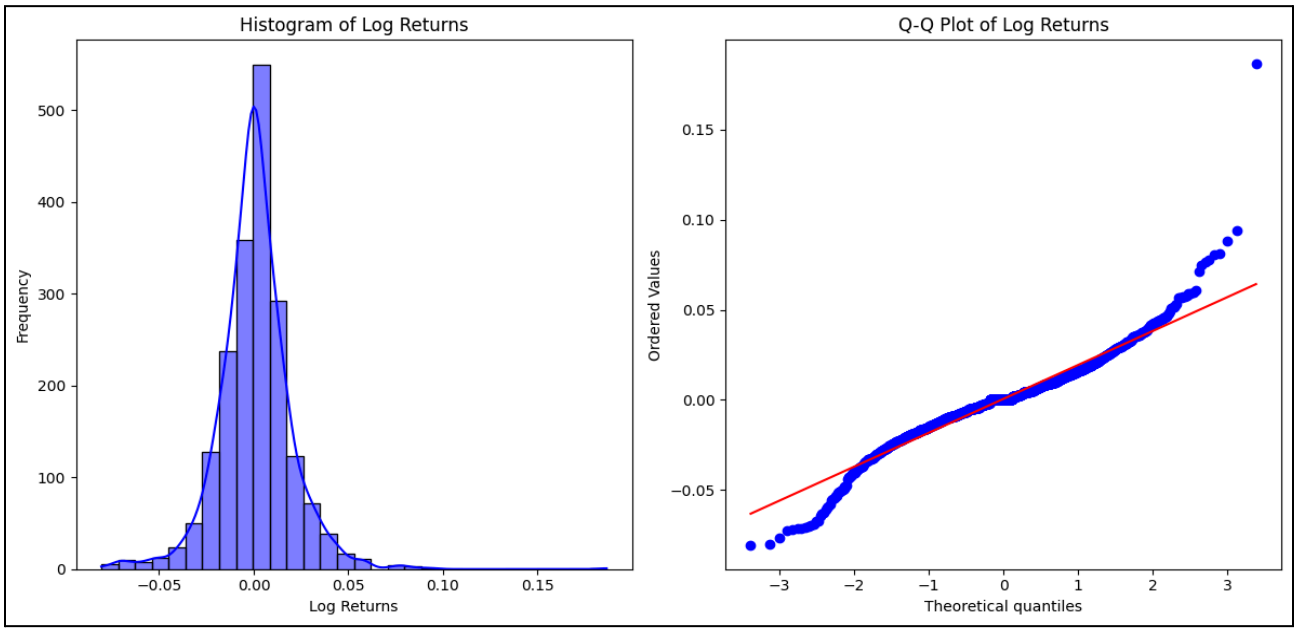
b. Transformed Series (Stationary - Log Returns):

Summary Statistics:

Mean	0.00057
Standard Deviation	0.019
Skewness	0.41
Kurtosis	6.63

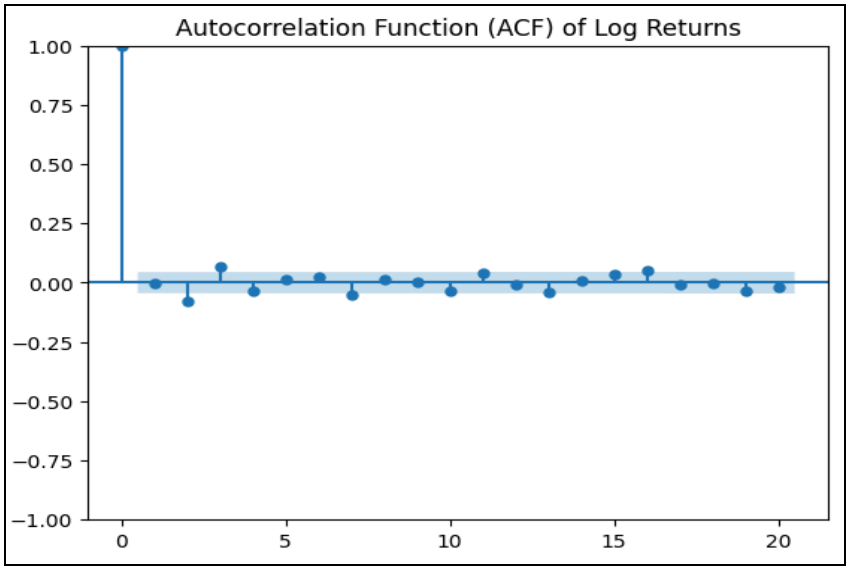
The above suggests that the distribution of log returns is slightly right-skewed with near-zero mean and constant variance. A kurtosis > 3 indicates heavy tails relative to the normal distribution (i.e., more extreme events).

Histogram & QQ- Plot:



The above histogram plotting and Q-Q plot visualize the statistical data above, confirming the positive skewness distribution and there is an outlier. The outlier based on the log returns plots happened in 2020 in the COVID era.

ACF Plot & Ljung-Box test:



The plot suggests that, visually, there's little to no linear autocorrelation in the log returns at the tested lags. This aligns with the idea that financial returns are often close to a random walk, where past returns don't predict future returns.

The Ljung-Box test (lags=10) when performed on the data, showed a p-value of 0.000258, The low p-value indicates that the data has some form of dependence, although it could be very weak, nonlinear, or only visible with a high sample size. It does not necessarily imply that the autocorrelation is strong or practically meaningful for prediction; it may just be statistically significant due to the test's sensitivity.

Augmented Dickey-Fuller Test:

When performed on the data, the ADF test returned a p-value of 0. This signifies that the log return time series shows that the dataset is stationary.

c. Fractionally Differenced Series

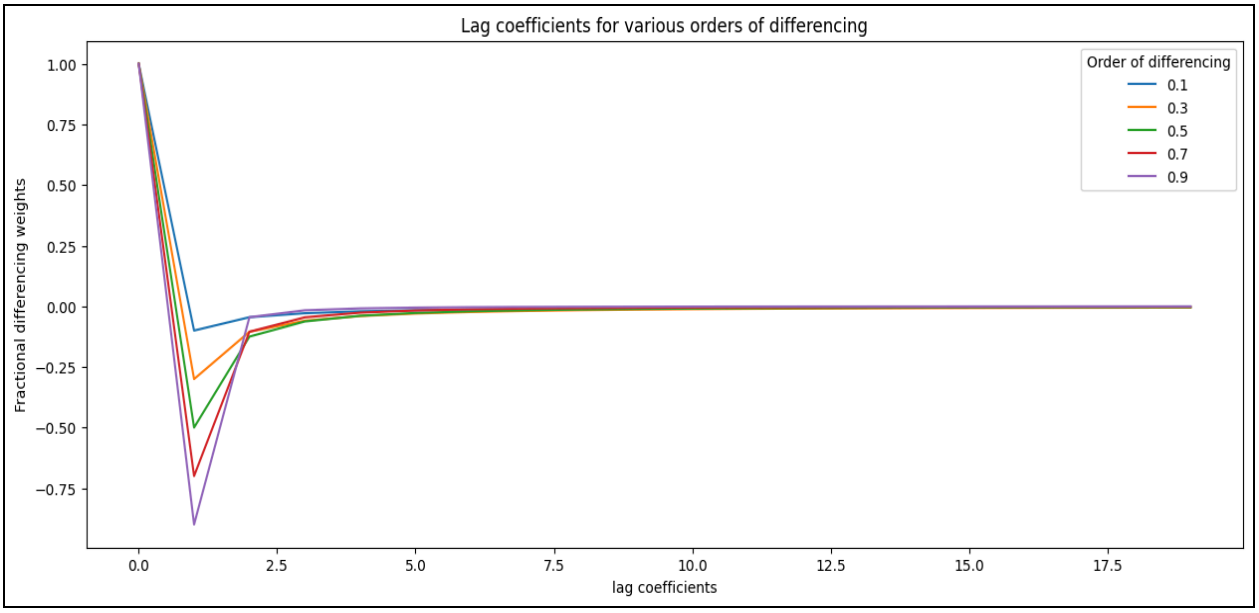
Fractional differencing converts the original non-stationary time series to a stationary series while retaining as much long-term dependence (or "memory") as possible. Traditional differencing (integer order) can eliminate valuable information from time series, but by employing a fractional differencing strategy, we achieve a balance between stationarity and information retention.

The theory behind Fractional Differencing

- Fractional differencing uses a fractional order 'd' where (0<d<1)
- Weights are computed using the binomial expansion of (1−B)^d, where B represents the backshift (lag) operator.
- These weights steadily decrease in magnitude, indicating incomplete differentiation.

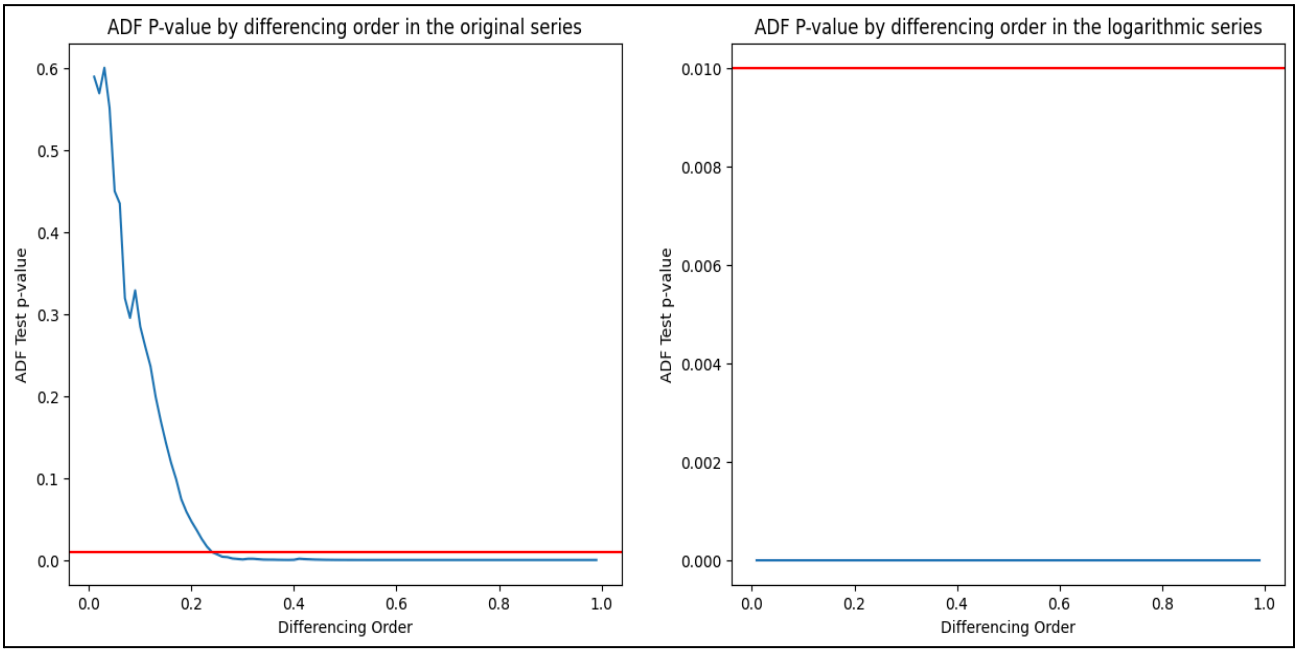
Determining Fractional Differencing Weights

- Defining a function called getWeights(d, lags), we compute the coefficients for the series expansion up to a given number of lags.
- Plotting the weights for various d values (e.g., 0.1, 0.3, 0.5, 0.7, 0.9) reveals their rapid convergence to zero.



Selecting the Fractional Order and Lag Cutoff

- We use a function (cutoff_find) to find the number of lags needed until the weights fall below a threshold which we take to be 10^{-4}.
- Then multiple d values (from 0.01 to 0.99 in 0.01 increments) are evaluated and the ADF test is applied to the fractionally differenced series.
- This reveals which orders the series becomes stationary (e.g., 0.3) which is shown in the plot below.



Implementing Fractional Differencing

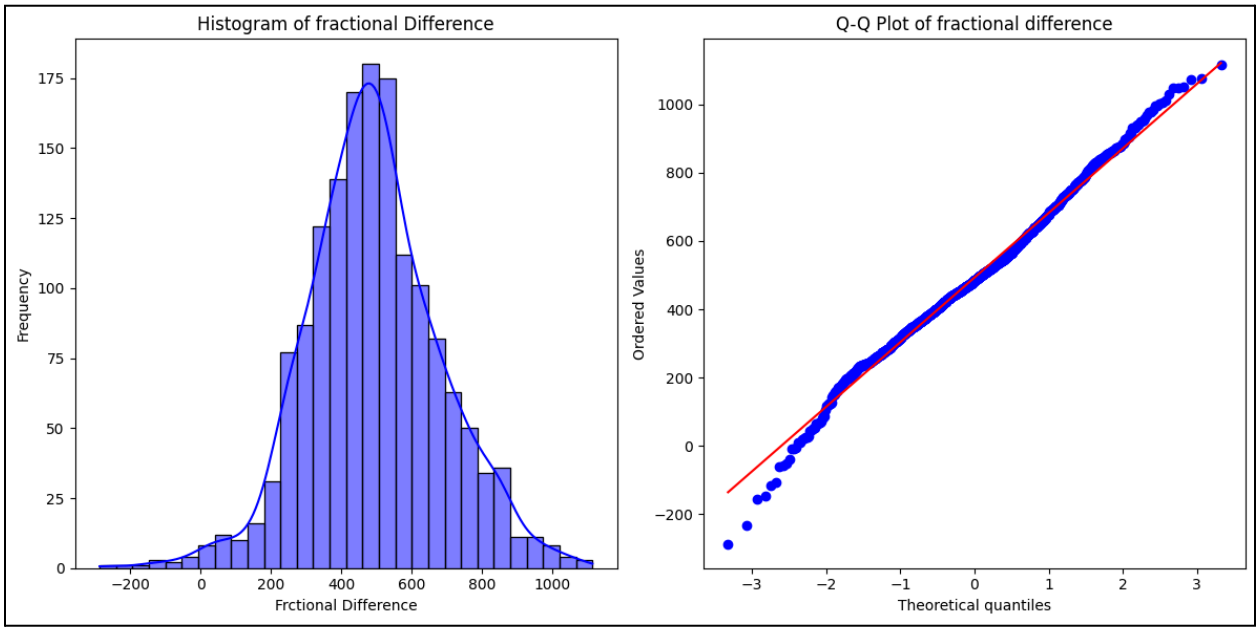
- We then create a Fractional Differencing function that calculates the fractionally differenced values by adding the product of each weight and the series' lagged values which is used to create the Fractionally differenced series.
- d is chosen as 0.3 after analyzing the ADF results. The 'cutoff_find' function is used to determine how many lags to include based on our threshold τ = 10^{-4}.

Stationarity (ADF Test):

Running the ADF test on the fractionally differenced series yielded a p-value of 0.0006 which is well below 0.05, thus we reject the null hypothesis of non-stationarity denoting that our fractionally differenced series is now stationary.

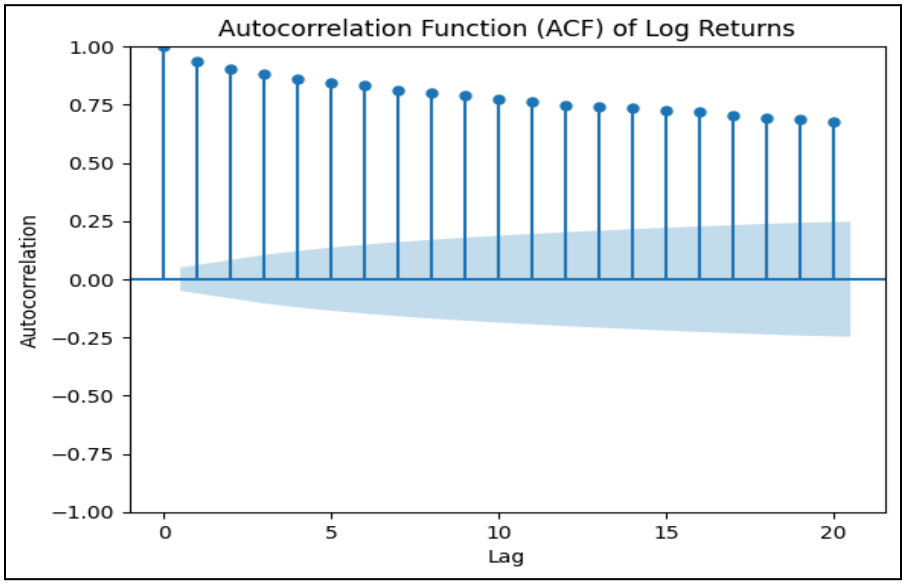
Summary Statistics, QQ-Plot & Histogram:

Mean	492.7
Standard Deviation	189.4
Skewness	0.06
Kurtosis	0.61



The Q-Q plot and histogram show that the distribution is generally symmetric but not perfectly normal because skewness is not equal to 0 and kurtosis is not equal to 3.

ACF & Ljung Box Test



The Autocorrelation Function (ACF) figure demonstrates considerable positive autocorrelation for multiple lags, indicating a high level of persistence. This is consistent with price movements that resemble a random walk.

The Ljung-Box test determines if considerable autocorrelation remains. In our example, the p-value is fairly low, implying that autocorrelation exists—this can be useful for model development.

Augmented Dickey-Fuller Test:

When performed on the data, the ADF test returned a p-value of 0. This signifies that the series is stationary.

d. Comments on the three representations

1. Original series (BBRI.JK):

- Non-stationary (p-value = 0.58).
- Demonstrates a trend and autocorrelation over time.

2. Transformed Series (Log Returns):

- Rapidly reaches stationarity (ADF p-value= 0).
- Loses some long-term memory in the process.

3. Fractionally Differenced Series

- Also stationary (ADF p-value < 0.01).
- It retains more of the original series' memory than simple integer differencing.
- Potentially better for models that rely on historical dependencies.

Step 2

a. MLP on the Non-Stationary Series

Data Preparation

1. The Original time series 'BBRI.JK' from 2017 to 2024 is taken as the data.
2. The rolling averages for 10, 25, 60, 90, 120, and 240 days are calculated to be used as input features.
3. The target is defined as the future 10-day rolling average, and a binary label is created based on whether it is above 0.

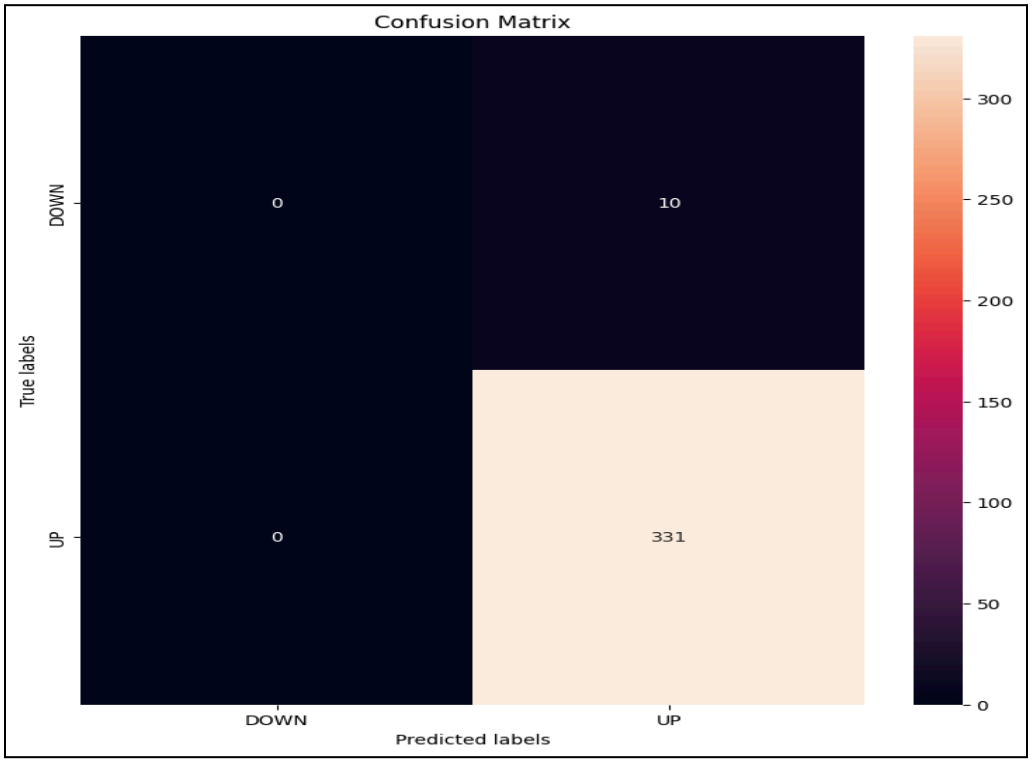
Train/Test Split

1. The data is split into 80% training and 20% testing,
2. The order of the time series is preserved

Model Architecture

1. Three hidden layers with 25, 15, and 10 neurons, ReLU activation, and a 20% dropout.
2. The output layer has one neuron (Sigmoid activation).
3. Optimizer: Adam with a learning rate of $1 * 10^{-5}$,
4. Loss function: Binary cross-entropy loss.
5. Early halting if there's no improvement in accuracy for 20 epochs
6. Method of addressing class imbalance: class weighting.

Results & Confusion Matrix

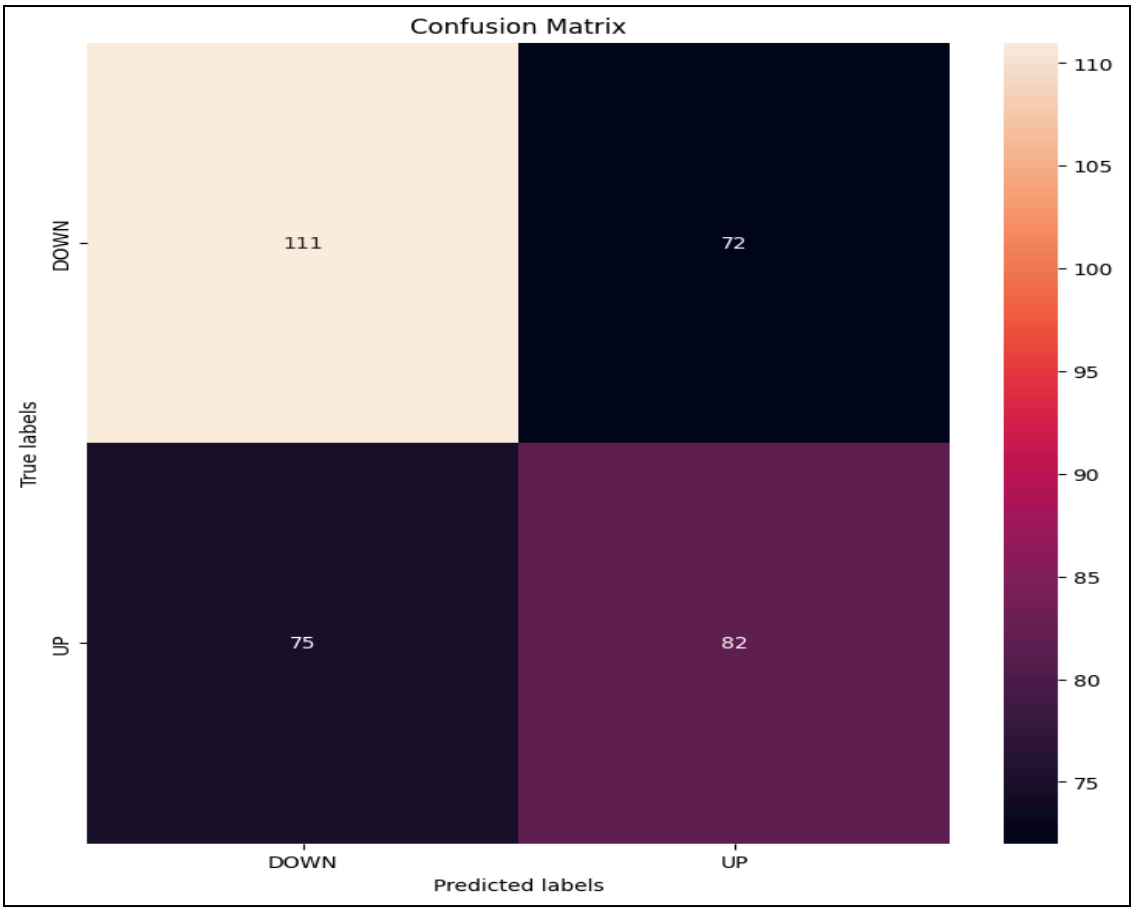


1. The model exhibited an accuracy of 97%.
2. Based on the confusion matrix, the model never forecasts "Down." All "Down" observations are mislabeled as "Up," although practically all "Up" observations are properly labeled.
3. This indicates a class imbalance, with the dataset substantially skewed toward "Up" and the algorithm defaulting to predicting "Up"
4. Despite its remarkable accuracy, the model has zero recall in the "Down" class.

b. MLP on Log Transformed Series

All the procedures from Data preparation to Train/Test Split to Model architecture are the same as before. The only difference is that the data used is the log-transformed stationary data (log returns)

Result & Confusion Matrix

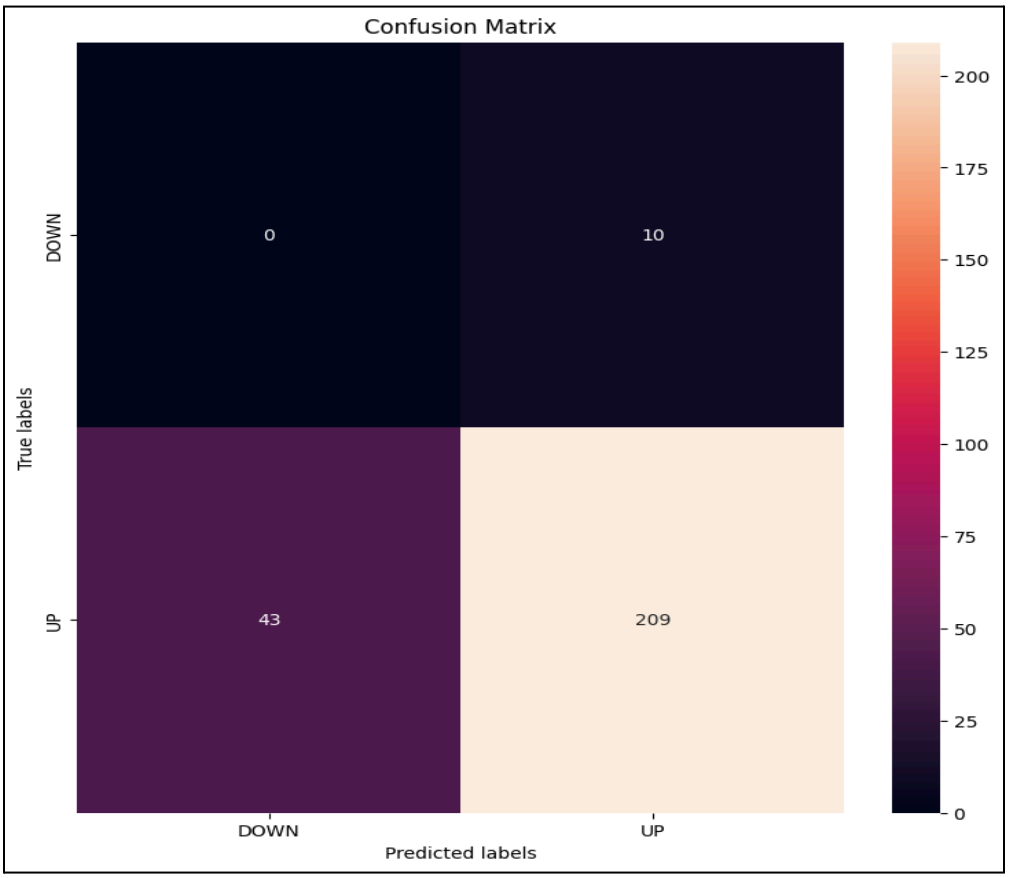


1. The model's test-set accuracy is roughly 56%, which is just slightly better than random guessing (50%).
2. Unlike the previous scenario, the model's predictions are balanced (it predicts both "DOWN" and "UP").
3. However, overall accuracy is not high.

c. MLP on Fractionally Differenced Series

In Step 1(c), we used a fractionally differenced time series to achieve stationarity while maintaining long-memory properties. In this step, we utilize the fractionally differenced data to create and train an MLP that predicts whether the future fractionally differenced value will be "UP" or "DOWN." All the procedures from Data preparation to Train/Test Split to Model architecture are the same as before. The only difference is that we use that fractionally differenced data to build and train an MLP.

Results & Confusion Matrix



- The model's test-set accuracy is 79.7%, which is better than the previous model.
- The model does rather well in predicting the majority class (UP), but struggles with minority (DOWN) observations.

d. Model Comparisons

MLP on Non-Stationary Series

- High accuracy (95-99%) may be misleading if data is substantially slanted towards "UP."
- The model may simply learn to predict "UP" most of the time (with low recall for "DOWN").
- The model can take advantage of any increasing trend in raw prices.
- Non-stationarity frequently produces misleading patterns; the model may overfit or not generalize well.

MLP on Stationary Series (Log Transformed Series)

- Accuracy ranges between 50-60%, implying that it is just slightly better than random guessing.
- The data is steady, but the simpler log returns may lose some of the long-term dependencies.
- Ensures stationarity, which is a common condition for forecasting models.
- The transformation may remove too much information, making it difficult for a basic MLP to gain strong predictive signals.

MLP on Fractionally Differenced Series

- Achieves 75-80% accuracy, balancing memory retention and stationarity.
- The confusion matrix demonstrates that the model can still suffer from class imbalance, although it performs far better than the merely stationary log-returns model.
- Maintains long-memory features better than standard differencing while adhering to stationarity assumptions.
- More difficult implementation (needs choosing a fractional differencing parameter d).

Reasons some models perform better than others

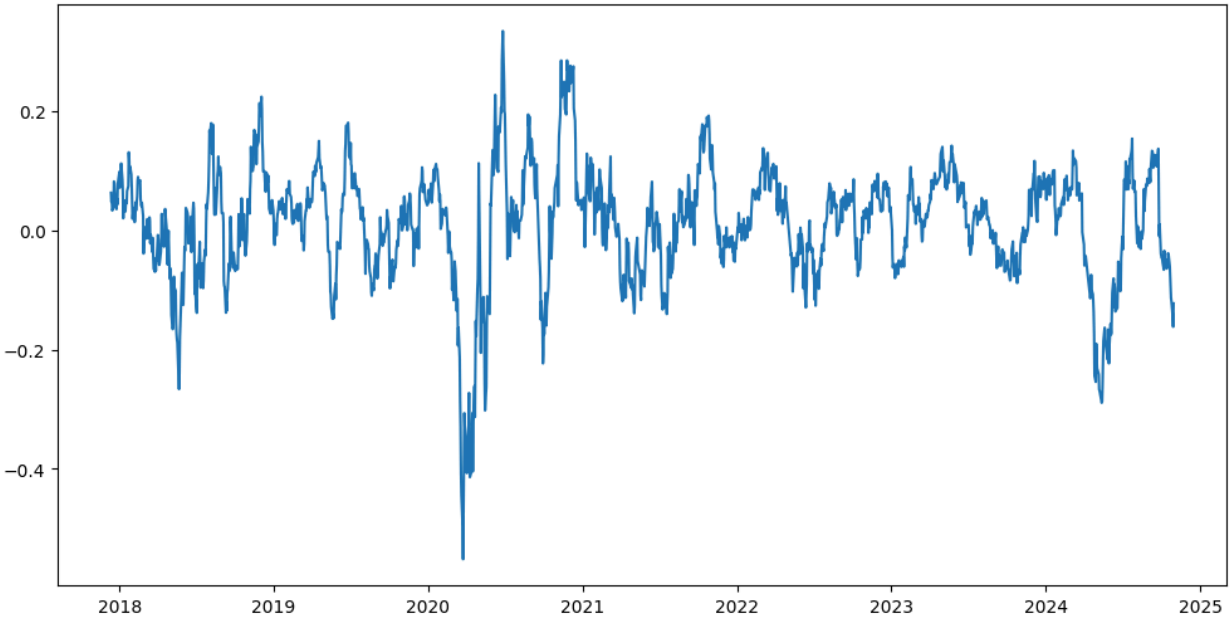
- Non-stationary data can provide falsely high accuracy if there is a strong trend (e.g., "UP" class). Stationary transformations (log returns or fractional differencing) eliminate the trend, making classification more difficult but statistically stronger.
- Using simple differencing or log returns can reduce historical dependence and improve information retention.
- Fractional differencing preserves more of the long-memory structure, allowing the model to recognize patterns.
- A naive classifier may appear "highly accurate" if there are more "UP" than "DOWN" points in the data.
- Proper treatment of imbalance (by SMOTE, altered class weights, or additional data) is critical for a meaningful assessment.

Step 3.

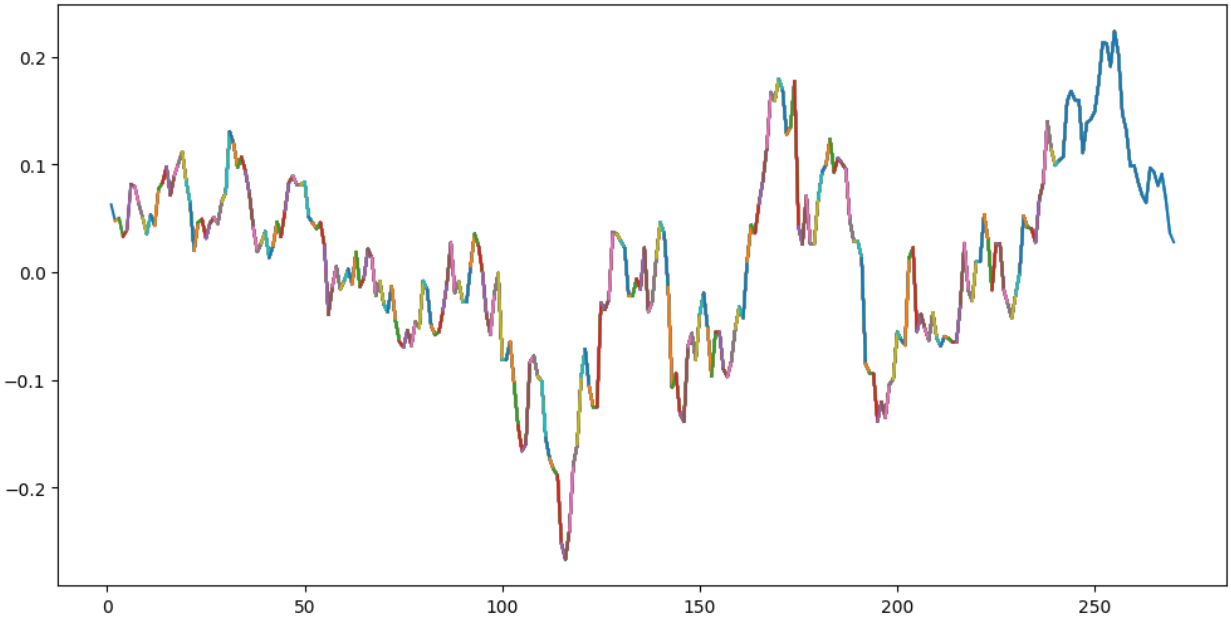
a. CNN on Non-stationary Series

Data Preparation

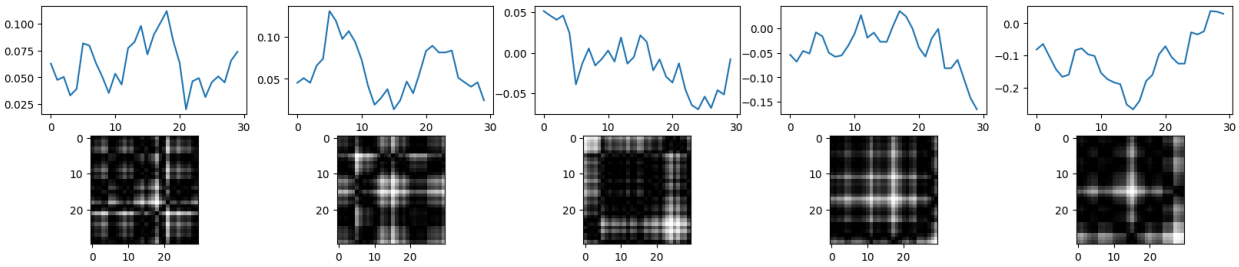
- The original time series 'BBRI.JK' from 2017 to 2024 is used.



- Create a series of 30-day windows from the input data



- Use the series of 30-day windows as input for GAF. The below graphs are samples of the results.



- Generate labels: 1 if the price goes up within the window, 0 otherwise

Train/Test Split

- The data is split into 80% training and 20% testing
- The order of the time series is preserved

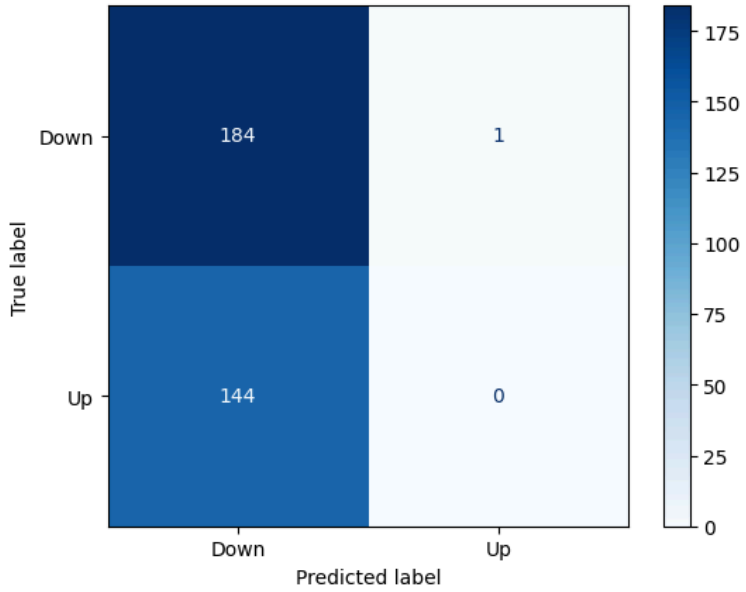
Model Architecture

- Create a sequential model, allowing the sequential linear stacking of layers
- The input layer defines input data as grayscale 30x30 images
- Three 2D convolutional layers with 16, 32, and 64 filters respectively, with the activation function being ReLU
- After each of the 2D convolutional layers is a MaxPooling2D 2x2 layer, effectively reducing the dimensions of the input feature maps by half.
- Flatten the 2D matrix into a 1D vector
- A dense layer with 1024 neurons using the ReLU activation function is added to model complex relationships
- A dropout layer of 0.5 is used to prevent overfitting
- The final dense layer has 1 neuron and uses the sigmoid activation function, suiting the classification task
- Similar to the MLP models, the loss function is binary cross-entropy and the method of addressing class imbalance is class weighting

Results & Confusion Matrix:

- The model has an overall accuracy of 56%.
- The model only predicted 1 Up label, which ended up being wrong, resulting in a precision and recall of 0 for the Up label
- As for the Down label, the model correctly predicted 184 out of 328, having a precision and recall of 0.56 and 0.99 respectively
- Despite there being little class imbalance, the model is heavily tilted toward predicting Down

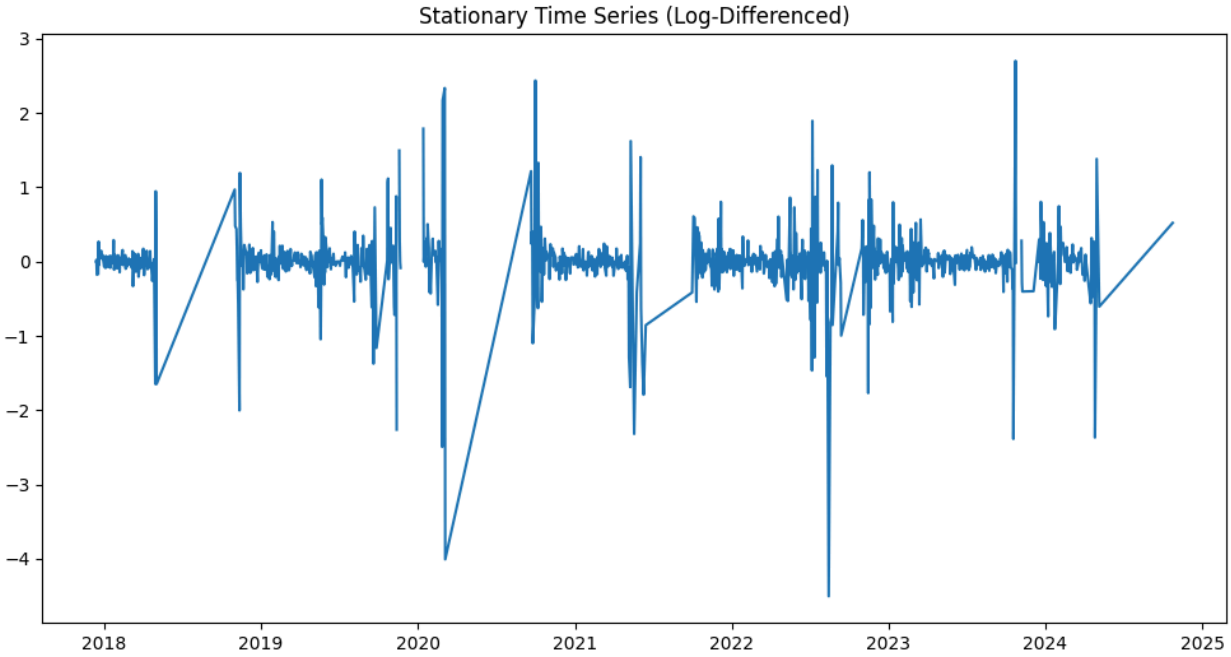
	precision	recall	f1-score	support
False	0.56	0.99	0.72	185
True	0.00	0.00	0.00	144
accuracy			0.56	329
macro avg	0.28	0.50	0.36	329
weighted avg	0.32	0.56	0.40	329



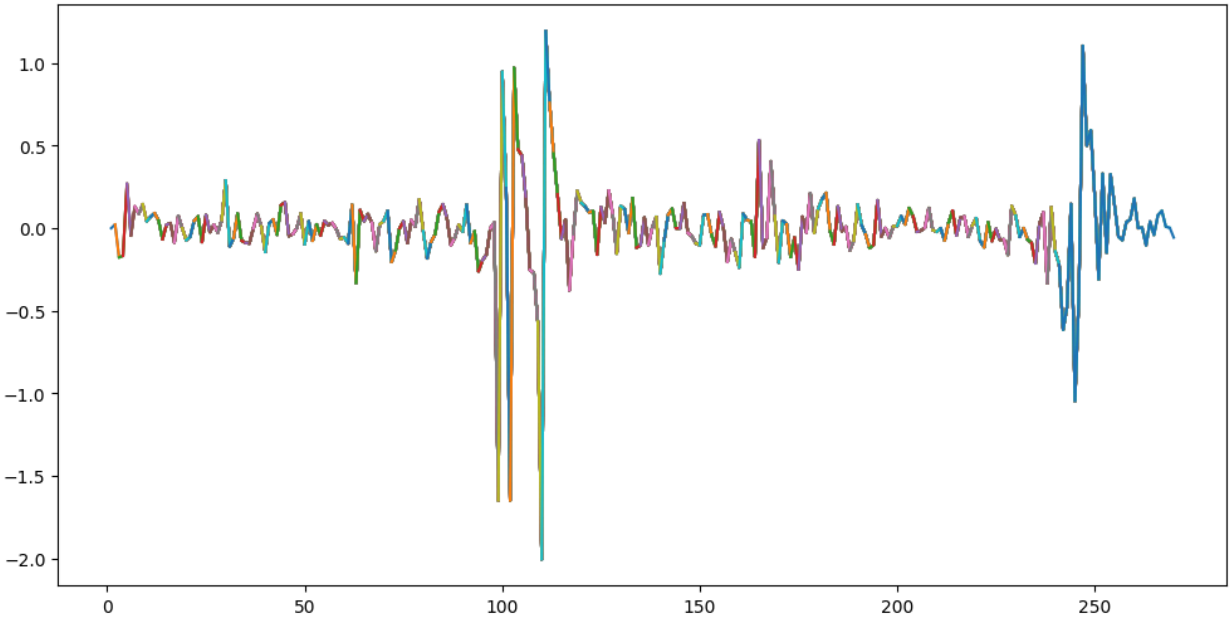
b. CNN on Stationary Series

Data Preparation

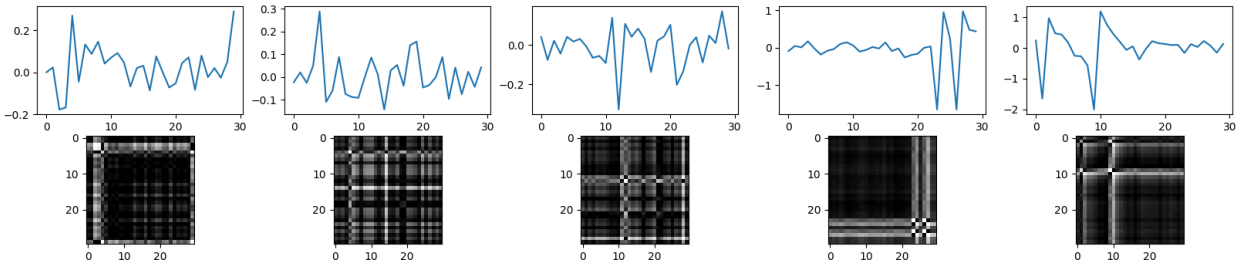
- Apply log transformation on the original time series 'BBRI.JK' from 2017 to 2024.



- Create a series of 30-day windows from the input data



- Use the series of 30-day windows as input for GAF. The below graphs are samples of the results.



- Generate labels: 1 if the price goes up within the window, 0 otherwise

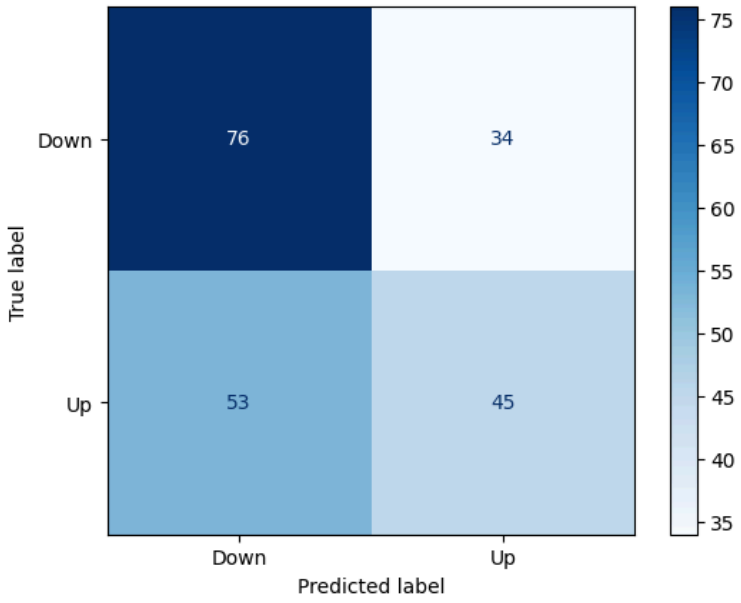
Train/Test Split and Model Architecture

Everything remains the same as Step 3a.

Results & Confusion Matrix

- The model's overall accuracy has improved a little, from 56% to 58%
- The model now predicts Up at a more balanced level compared to Step 3a.
- The Down label's result is still slightly better than its counterpart, having an F1 score of 0.64 to that of 0.51 for Up

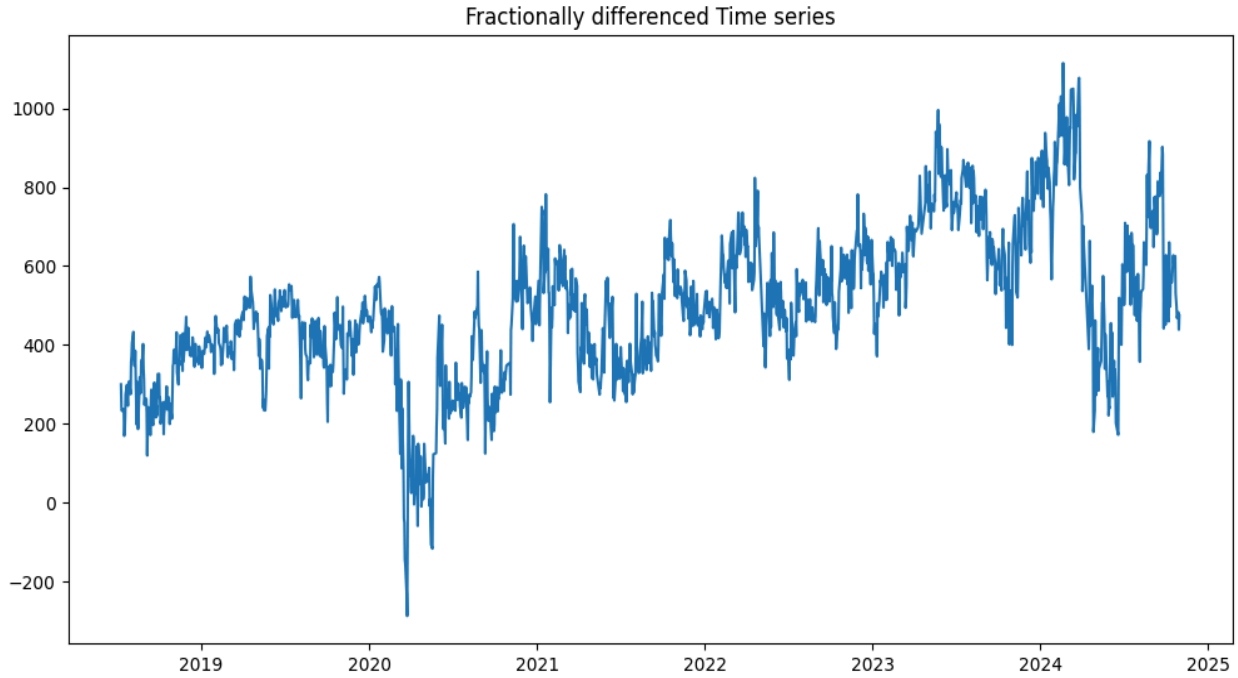
	precision	recall	f1-score	support
False	0.59	0.69	0.64	110
True	0.57	0.46	0.51	98
accuracy			0.58	208
macro avg	0.58	0.58	0.57	208
weighted avg	0.58	0.58	0.58	208



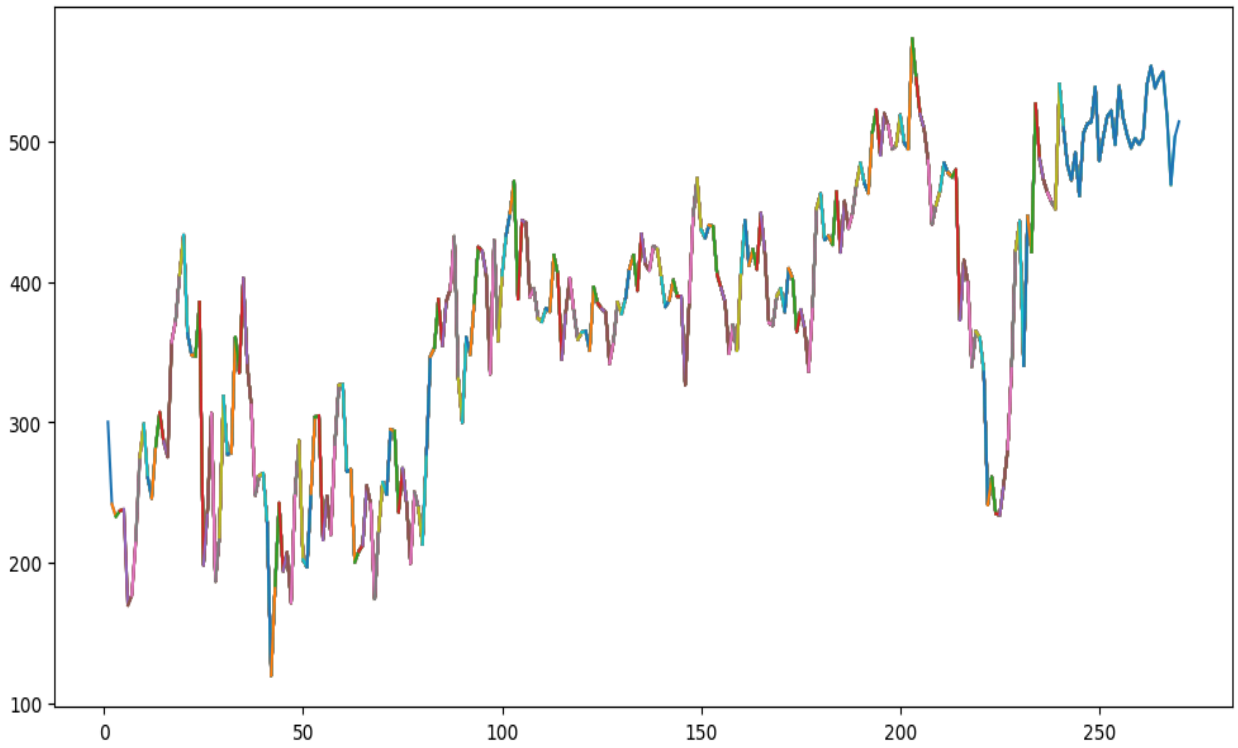
c. CNN on Fractionally Differenced Series

Data Preparation

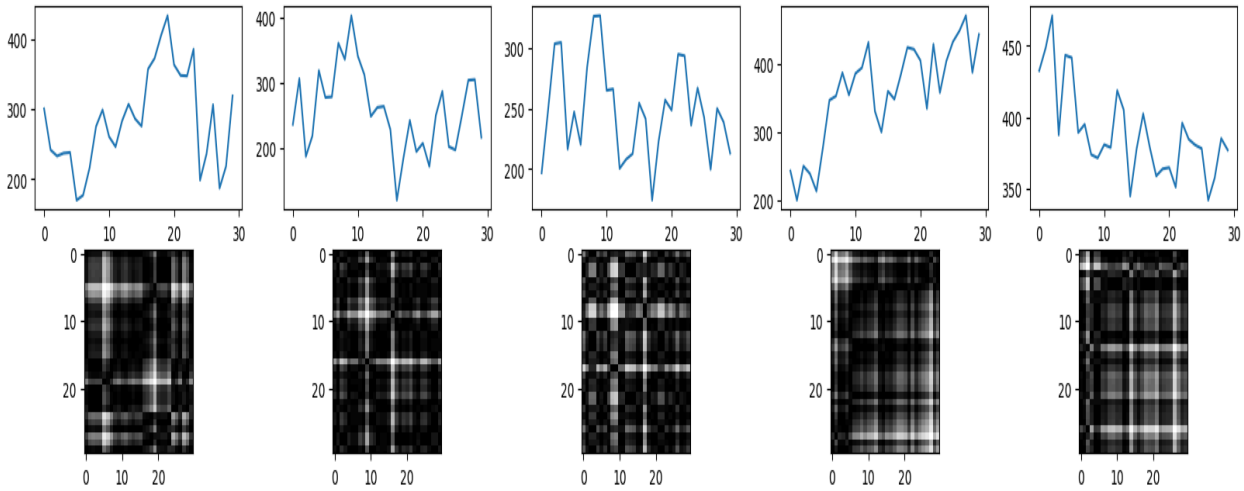
- The input data for this step is the Fractionally Differenced Series obtained in Step 1c.



- Create a series of 30-day windows from the input data



- Use the series of 30-day windows as input for GAF. The below graphs are samples of the results.



- Generate labels: 1 if the price goes up within the window, 0 otherwise

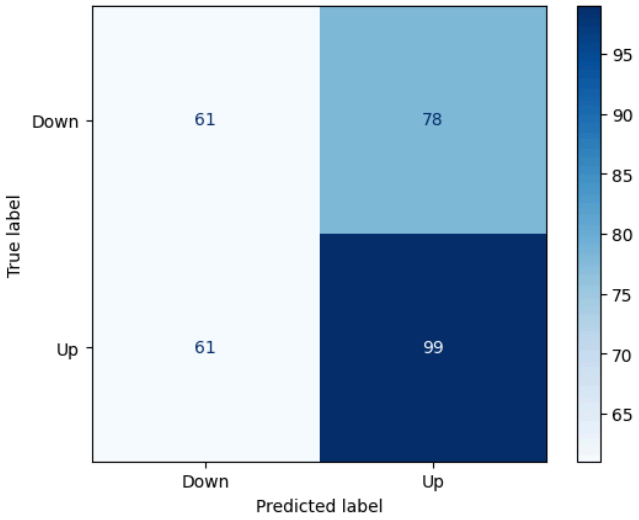
Train/Test Split and Model Architecture

- Everything remains the same as Step 3a.

Results & Confusion Matrix

- The model's overall accuracy has declined a little, down to 54%.
- The model now predicts Up more frequently than previously, and more than its counterpart.
- The Up label now also performs better than the Down label, demonstrating superior precision, recall, and F1 score.

	precision	recall	f1-score	support
False	0.50	0.44	0.47	139
True	0.56	0.62	0.59	160
accuracy			0.54	299
macro avg	0.53	0.53	0.53	299
weighted avg	0.53	0.54	0.53	299



d. Models comparison

- Overall, all three models perform relatively similarly to one another, ranging from 54% to 58%.
- In terms of overall accuracy, the model on Stationary data performs the best. Meanwhile, The CNN model on the Fractionally Differenced Series performs the worst out of the three, which is contrary to Step 2's findings, where it improved compared to the Stationary Series.
- The Stationary data model is also the best at predicting Down labels, with a 0.69 recall and 0.64 F1 score.
- However, when it comes to predicting Uptrends, the Fractionally Differenced model is the best model, with a recall of 0.62 and an F1 score of 0.59, despite having the worst overall accuracy.
- While the CNN model on Non-stationary data has an accuracy of 56%, it is the worst model in terms of practicality, where it almost only predicted Down. As a result, its accuracy is equal to the proportion of the Down label class within the dataset.

Step 4.

For the baseline time series (Steps 2A and 3A), MLP performs better than CNN. This is presumably because MLP directly handles raw data, enabling it to observe temporal relationships, whereas CNN is hampered by the complexity of GAF-transformed images. The non-stationary data might also have contributed to disrupting CNN's performance. Investigating different time series-to-image methods may yield better results in the future.

For the transformed time series (Steps 2B and 3B), MLP still performs better because the log return transformation stabilizes the data. CNN, which is intended for spatial pattern recognition, is not well suited for structured time series data.

For fractionally differenced time series (Steps 2C and 3C), MLP outperforms CNN because it handles long-term dependencies, whereas CNN, which is optimized for localized patterns, is unable to learn useful information from the GAF-transformed data. The transformation might have introduced noise or disturbed memory retention.

In general, MLP is more effective for time series modeling, whereas CNNs could do with better transformation techniques or hybrid models.

References

1. Espinal, E. (n.d.). *Time series analysis using fractional differencing*. Kaggle. Retrieved from https://www.kaggle.com/code/elvisesp/time-series-analysis-using-fractional-differencing

2. Maitra, & Sarit. (2023). *Time-series forecasting: Unleashing long-term dependencies with fractionally differenced data.* 2023 IEEE 64th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS). IEEE. Retrieved from https://arxiv.org/pdf/2309.13409

3. Sudre, C. H., et al. (2017). *Generalized Dice overlap as a deep learning loss function for highly unbalanced segmentations.* Lecture Notes in Computer Science, 240–248. https://doi.org/10.1007/978-3-319-67558-9_28