GROUP WORK PROJECT # 01
GROUP NUMBER: 1 ANONYMIZED FOR PRIVACY

MScFE 642: Deep Learning for Finance

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTR MEMBER
ETIENNE LANDRY ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	
Daniel ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	
Kago ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	

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	ETIENNE LANDRY ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY
Team member 2	Daniel ANONYMIZED FOR PRIVACY
Team member 3	Kago ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY

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

1 Introduction

This project puts us in the shoes of an optimisation engineer in a statistical arbitrage team. Our job is to understand a financial time series, transform it into more model–friendly versions, and then compare how different deep learning architectures perform when tasked with forecasting it.

We chose the SPDR S&P 500 ETF (ticker: **SPY**) as our underlying security.This is a liquid exchange-traded fund (ETF) which trades on the NYSE Arca under the ticker "SPY". The ETF tracks the S&P 500 index by holding a portfolio comprising all 500 companies on the index. As of 25th November 2025, the ETF had 503 holdings and a weighted average market capitalization of $1.44 billion (State Street Investment Management) Using daily adjusted closing prices from 1 January 2018 to 31 October 2025 (1,969 observations), we:

- analysed the raw price level series and two stationary transformations (first difference and fractional difference);
- trained Multi–Layer Perceptron (MLP) models on all three representations;
- converted lagged sequences into Gramian Angular Field (GAF) images and trained Convolutional Neural Networks (CNNs) on those images; and
- compared MLP versus CNN performance and reflected on what this means for practical forecasting.

Across the board, the models struggle to beat a naive benchmark (all out–of–sample R^2 values are negative), but the comparative results are still informative. The MLPs consistently outperform the CNNs, especially when the input data are expressed in stationary form.

2 Step 1: Time Series Construction and Stationarity

2.1 Step 1(a): Price Level Time Series

We downloaded daily adjusted closing prices of SPY between 2018–01–01 and 2025–10–31. The sample size and basic descriptive statistics from the notebook are:

- Number of observations: 1,969
- Mean: 389.20
- Standard deviation: 119.20
- Minimum: 206.11
- 25th percentile: 273.26
- Median: 385.77

- 75th percentile: 444.69
- Maximum: 687.39

Visually, the SPY price series trends upward over the sample, with periods of volatility clustering around market events. This trend and volatility structure already hint that the level series is unlikely to be stationary.

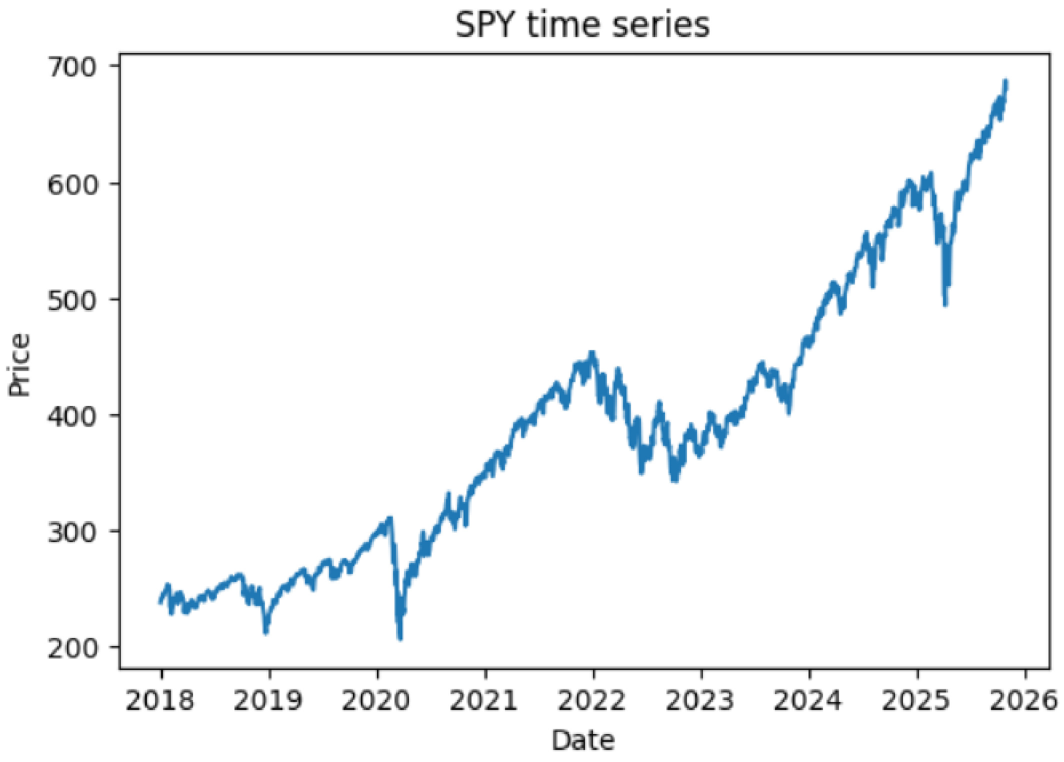


Figure 1: SPY price level time series.

To formalise this, we ran an Augmented Dickey–Fuller (ADF) test on the price levels:

- ADF statistic: 0.7356
- p–value: 0.9905
- Number of lags used: 9

At conventional significance levels, we strongly *fail* to reject the null hypothesis of a unit root. In other words, the price level behaves like a non–stationary process. This is consistent with financial theory, where prices often follow a random walk with drift.

From a modelling perspective, using the raw price levels directly in a predictive model risks spurious regressions and unstable parameter estimates.

2.2 Step 1(b): First Difference of SPY

We then constructed the first difference of SPY, defined as

$$\Delta P_t = P_t - P_{t-1},$$

which can be interpreted as a one–day nominal return in price units.

Key summary statistics for the first differences (1,968 observations) are:

- Mean: 0.225
- Standard deviation: 4.487
- Minimum: −31.24
- 25th percentile: −1.62
- Median: 0.32
- 75th percentile: 2.48
- Maximum: 51.84

Compared to the level series, the first differences:

- centre much more tightly around zero;
- show no obvious deterministic trend; and
- exhibit volatility clustering typical of daily returns.

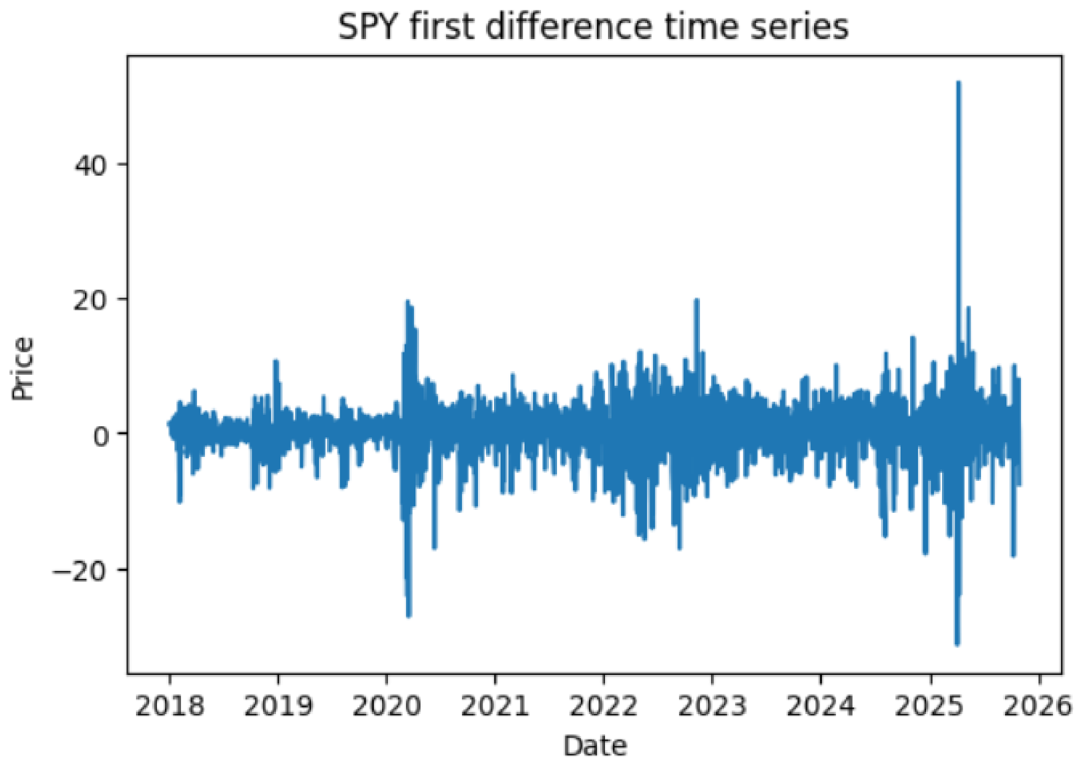


Figure 2: First-differenced SPY series (daily changes).

The ADF test on the first differences gives:

- ADF statistic: -14.6085
- p–value: 0.0000

This is strong evidence to *reject* the null of a unit root. The first–differenced series is stationary in the usual sense, which makes it more suitable as input to many time series models.

2.3 Step 1(c): Fractional Differencing of SPY

Classical first differencing ($d = 1$) often achieves stationarity, but at the cost of removing a lot of the long–memory structure of the series. To strike a balance between stationarity and memory preservation, we applied **fractional differencing** with order $d = 0.6$.

We constructed fractional differences using the standard binomial expansion weights and functions from (ELVISESPINAL),

$$w_k = -w_{k-1} \cdot \frac{d-k+1}{k},$$

with $w_0 = 1$, and truncated the weights when their absolute value fell below a tolerance of $\tau = 10^{-4}$. For $d = 0.6$, this resulted in a lag cutoff of 142 terms. The fractionally differenced series is therefore a weighted sum of the current and past 141 observations.

The resulting series has 1,827 observations (we lose initial points due to the lag structure). Its descriptive statistics are:

- Mean: 10.38
- Standard deviation: 6.18
- Minimum: −30.29
- 25th percentile: 7.01
- Median: 10.30
- 75th percentile: 14.42
- Maximum: 48.09

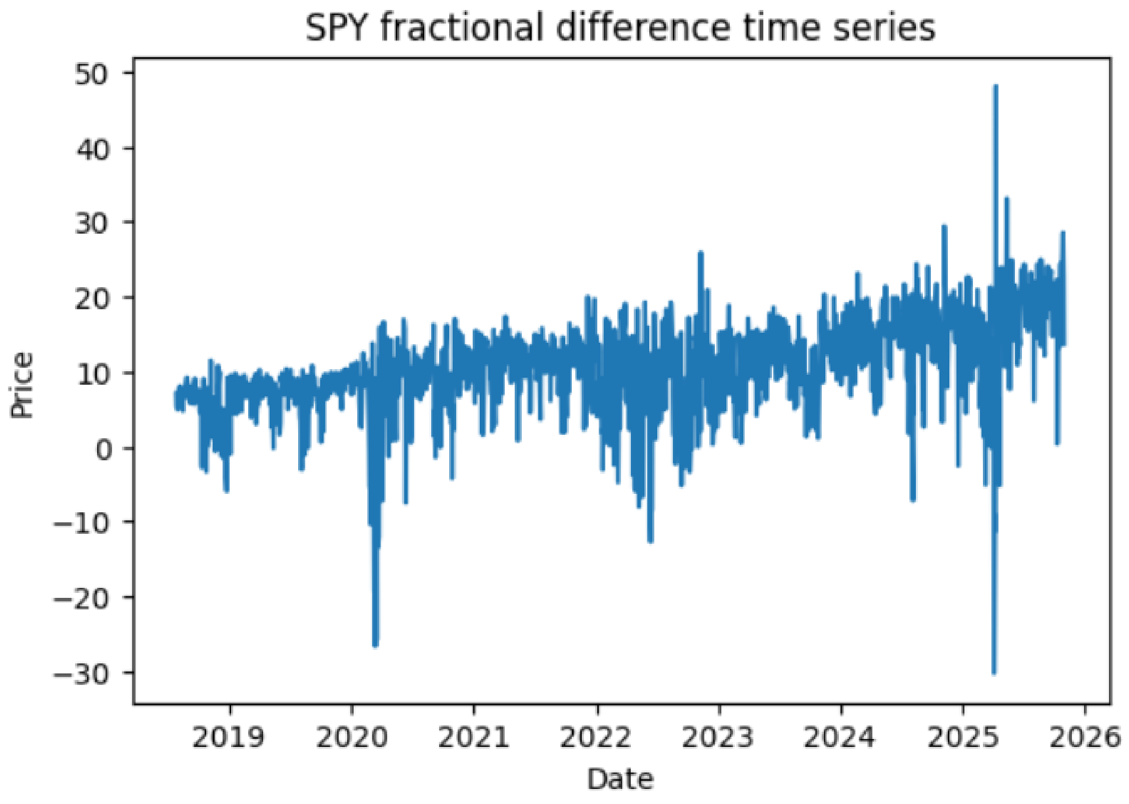


Figure 3: Fractionally differenced SPY series ($d = 0.6$).

An ADF test on the fractionally differenced series yields:

- ADF statistic: −4.2511
- p–value: 0.0005

We again reject the null of a unit root, so the fractional difference is stationary. At the same time, the series shows smoother behaviour than the fully differenced returns and appears to retain more of the low–frequency structure of the original SPY levels.

2.4 Step 1(d): Comparison of the Three Representations

Summarising Step 1:

- **Price levels (SPY)**: Non–stationary, with a clear upward trend and changing volatility over time. Economically meaningful in terms of absolute price, but problematic for many forecasting models due to the presence of a unit root.
- **First differences**: Stationary and centred near zero, with behaviour similar to daily returns. The trade–off is that some of the long–run information about the level of the index has been stripped away.
- **Fractional differences**: Also stationary, but constructed with $d = 0.6$ so that part of the low–frequency information is preserved. This representation aims to keep enough memory for predictive power while still satisfying stationarity requirements.

From a modelling point of view, the first difference is the safest choice for standard time series tools, while the fractional difference is a compromise that might help more flexible models (like deep networks) exploit longer–range dependence. The rest of the project tests whether that theoretical advantage shows up in practice.

3 Step 2: MLPs on Time Series Representations

In Step 2, we trained MLP models on each of the three representations: price levels, first differences, and fractional differences this is similarily to how it was done in (Zhang et al).

3.1 Data Preparation and Input Design

For each series, we used a sliding window of past observations as inputs. Specifically:

- Window length (*look–back*): 20 days.
- Train–test split: 80% of the windows for training, 20% for testing.
- Scaling: Min–Max scaling of the target series to $[0, 1]$ within the training set, then applying the same transformation to the test set(Tensorflow).

The task is set up as a regression problem: given the last 20 values of the series, predict the next value.

3.2 MLP Architecture and Training Setup

To forecast the SPY series, we constructed a feed-forward Multi–Layer Perceptron (MLP) using a sliding window of the past 20 daily closing prices to predict the next day's value. All inputs were scaled using Min–Max scaling fitted only on the training set to prevent information leakage. The architectural choices number of layers, hidden units, activation functions, and the use of dropout follow standard guidance in the hyperparameter optimisation literature, particularly the discussions in Yang and Shami (2020) on balancing model complexity and generalisation .

All three MLPs (corresponding to price levels, first differences, and fractional differences) share the same architecture:

- **Input layer:** 20 features (one for each lag in the look-back window).
- **Hidden layer 1:** 64 ReLU units, followed by 20% dropout to reduce overfitting.
- **Hidden layer 2:** 32 ReLU units.
- **Output layer:** 1 linear unit producing the one-step-ahead forecast.

The training configuration was as follows:

- **Optimizer:** Adam with a learning rate of 0.001.
- **Loss function:** Mean Squared Error (MSE).
- **Epochs:** 50.
- **Batch size:** 32.
- **Data ordering:** No shuffling, ensuring the temporal structure of the series is preserved.

This setup provides enough model capacity to capture nonlinear relationships while maintaining regularisation appropriate for a dataset of this size.

3.3 Out-of-Sample Performance of MLP Models

After training, we evaluated each MLP on the held–out test set. Predictions and targets were both transformed back to the original scale of the corresponding series before computing metrics.

Model	R^2 (Out-of-Sample)	RMSE	MAE	MSE
Price Level MLP	-1.4105	64.8974	63.8514	4211.6758
First Difference MLP	-0.4529	7.5991	5.4484	57.7465
Fractional Difference MLP	-0.5454	8.8364	7.3453	78.0818

Table 1: Out–of–sample performance of MLP models on the three SPY representations.

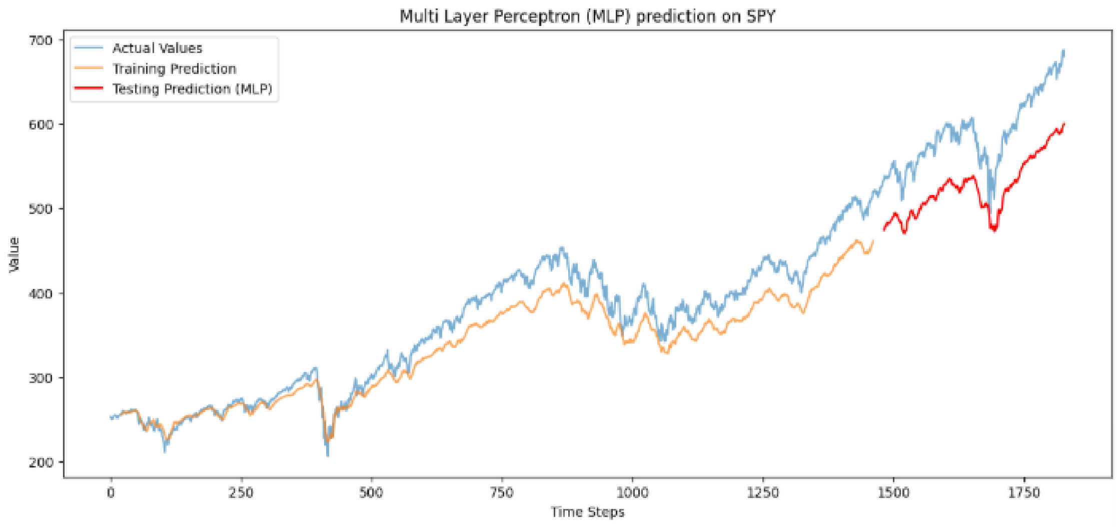


Figure 4: Out-of-sample predictions vs actual values for the MLP on SPY price levels.

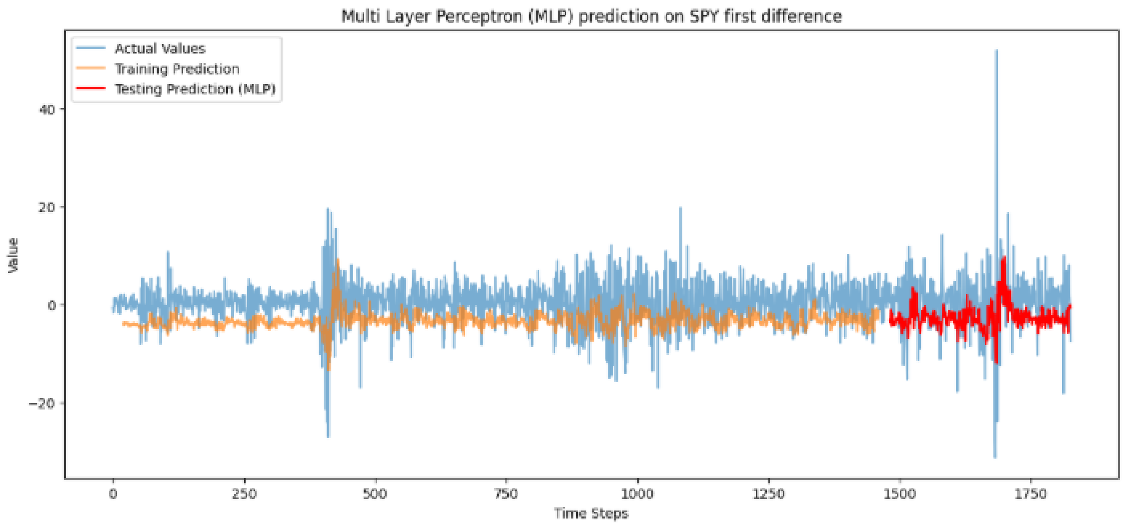


Figure 5: Out-of-sample predictions vs actual values for the MLP on first-differenced SPY.

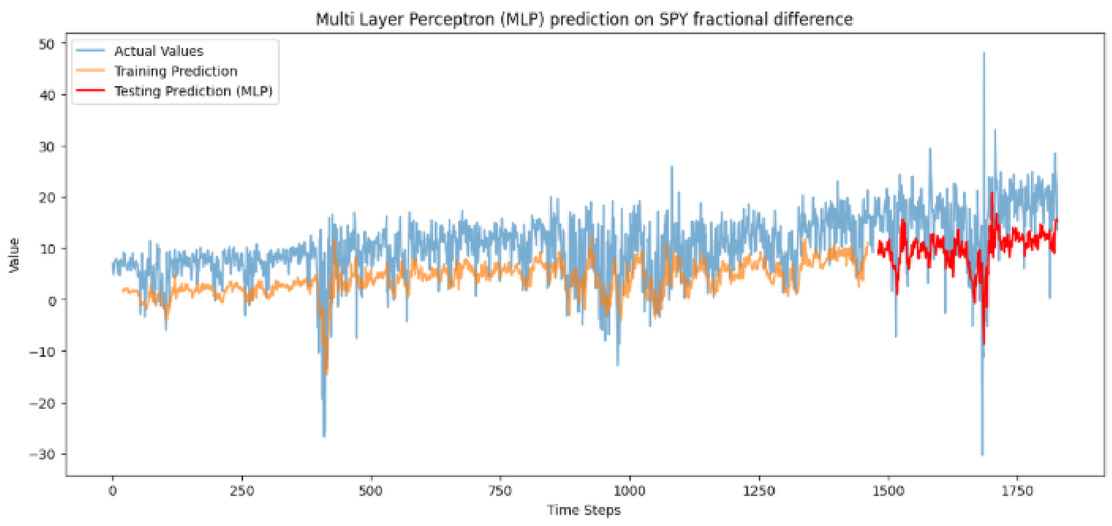


Figure 6: Out-of-sample predictions vs actual values for the MLP on fractionally differenced SPY.

A few points stand out:

- All three R^2 values are negative, which means that, on the test set, each MLP is worse than simply predicting the mean of the target series.
- The price level model performs particularly poorly ($R^2 \approx -1.41$ and RMSE ≈ 65). Intuitively, this is not surprising: forecasting the absolute level of a noisy, trending financial series with a short look–back window is extremely difficult.
- The first difference MLP is the least bad ($R^2 \approx -0.45$ and RMSE ≈ 7.6), indicating that the stationary return–like series is somewhat easier to model.
- The fractional difference MLP sits between the two: it performs worse than the first difference but better than modelling the levels directly.

3.4 Interpretation of MLP Results

Although none of the MLPs truly "crack" the forecasting problem, the relative ordering is informative:

1. **Working with stationary inputs helps.** The large gap between the price level MLP and the other two models shows that removing the unit root is important. Feeding the network a non–stationary series leads to forecasts that are both unstable and uncompetitive.
2. **Simple differencing is hard to beat here.** The first difference MLP slightly outperforms the fractional difference MLP. One possible explanation is that financial return series are notoriously dominated by noise, so gains from preserving additional long–memory structure (via fractional differencing) may not be enough to offset the added complexity.

3. **Capacity versus signal.** The MLP architecture is moderately flexible, but the underlying signal in daily SPY data is weak relative to noise. In such settings, even sophisticated nonlinear models can end up overfitting noise rather than extracting stable patterns.

From a practical perspective, these results are a warning: simply increasing model capacity does not guarantee better performance in financial prediction, especially if the signal–to–noise ratio is low.

4 Step 3: CNNs on GAF Image Representations

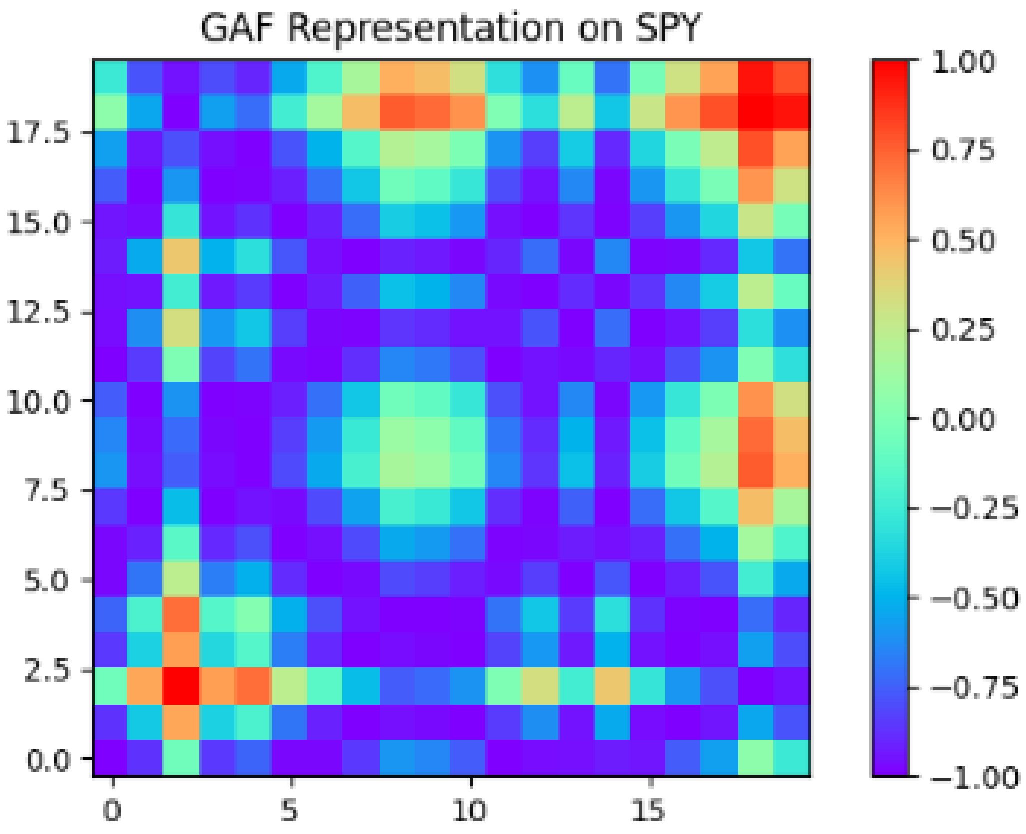
In Step 3, we moved from one–dimensional sequences to image–like representations by using Gramian Angular Fields (GAF). This effectively turns the time series into images that can be processed by CNNs.

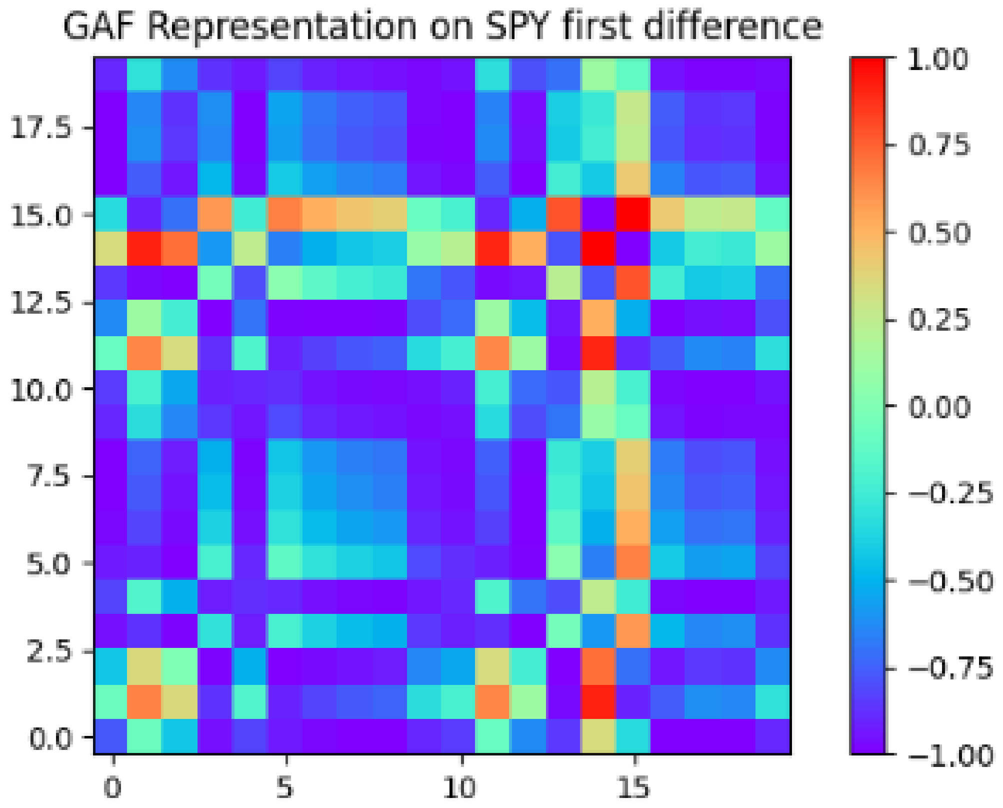
4.1 GAF Transformation

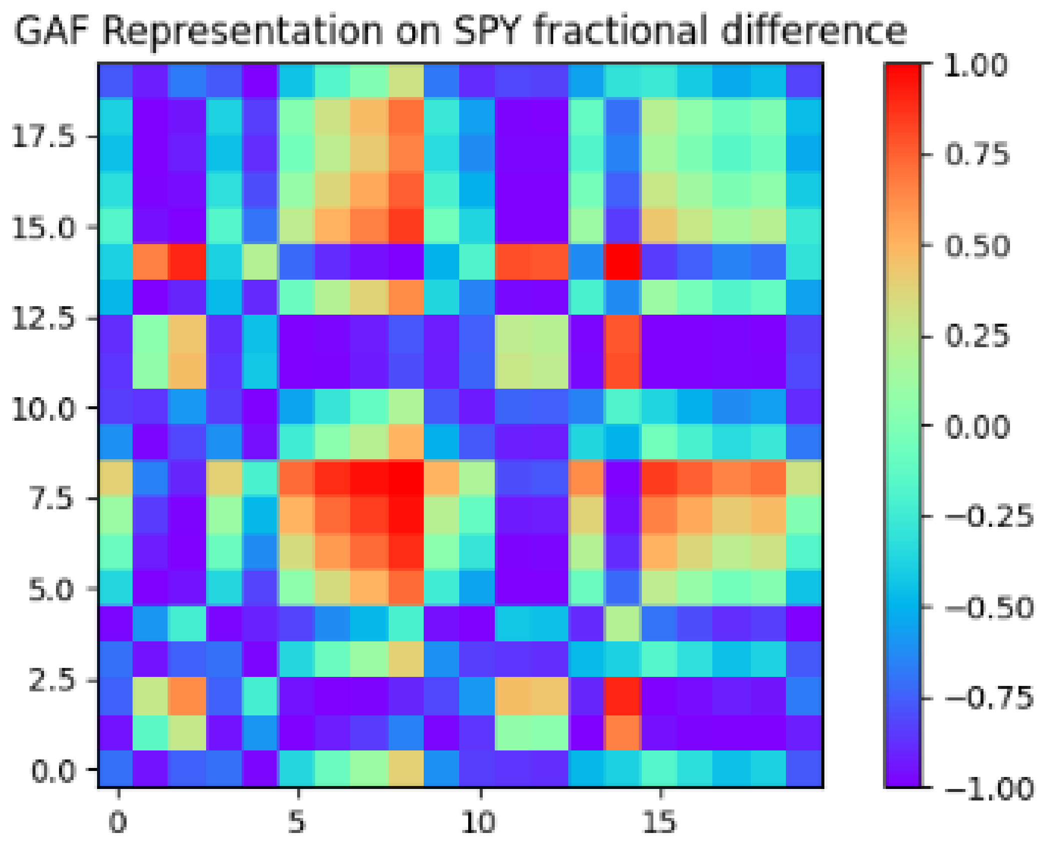
For each of the three time series (levels, first differences, fractional differences), we:

- used a look–back window of 20 lags to form short sequences;
- scaled each series to $[0, 1]$;
- transformed each 20–dimensional sequence into a 20×20 GAF image using the summation method.

Each pixel in the GAF captures interactions between points in the sequence in polar coordinates, encoding temporal information into a two–dimensional structure. Intuitively, smoother and more persistent series produce more structured, regular GAF images, while noisy series produce more irregular patterns.


GAF Representation on SPY
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


GAF Representation on SPY first difference
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
5
10
15
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



4.2 CNN Architecture and Training

We used the same CNN architecture for all three data representations:

- Input: $20 \times 20 \times 1$ GAF image.
- Convolutional layer 1: 32 filters, 3×3 kernel, ReLU.
- Max pooling: 2×2.
- Convolutional layer 2: 64 filters, 3×3 kernel, ReLU.
- Second max pooling: 2×2.
- Flatten layer.
- Dense layer: 64 units, ReLU.
- Dropout: 20%.
- Output layer: 1 unit, linear activation.

Training details mirror the MLP setup:

- Optimizer: Adam, learning rate 0.001.
- Loss function: MSE.
- Epochs: 50.
- Batch size: 32.

4.3 Out-of-Sample Performance of CNN Models

The CNNs were evaluated on the held–out test sets, again after mapping predictions back to the original scale. The results are summarised in Table 2.

Model	R^2 (Out-of-Sample)	RMSE	MAE	MSE
Price Level CNN	−28.4414	226.8033	222.2975	51439.7276
First Difference CNN	−0.9256	8.7484	6.8808	76.5348
Fractional Difference CNN	−5.5668	18.2148	16.7501	331.7782

Table 2: Out–of–sample performance of CNN models on GAF images.

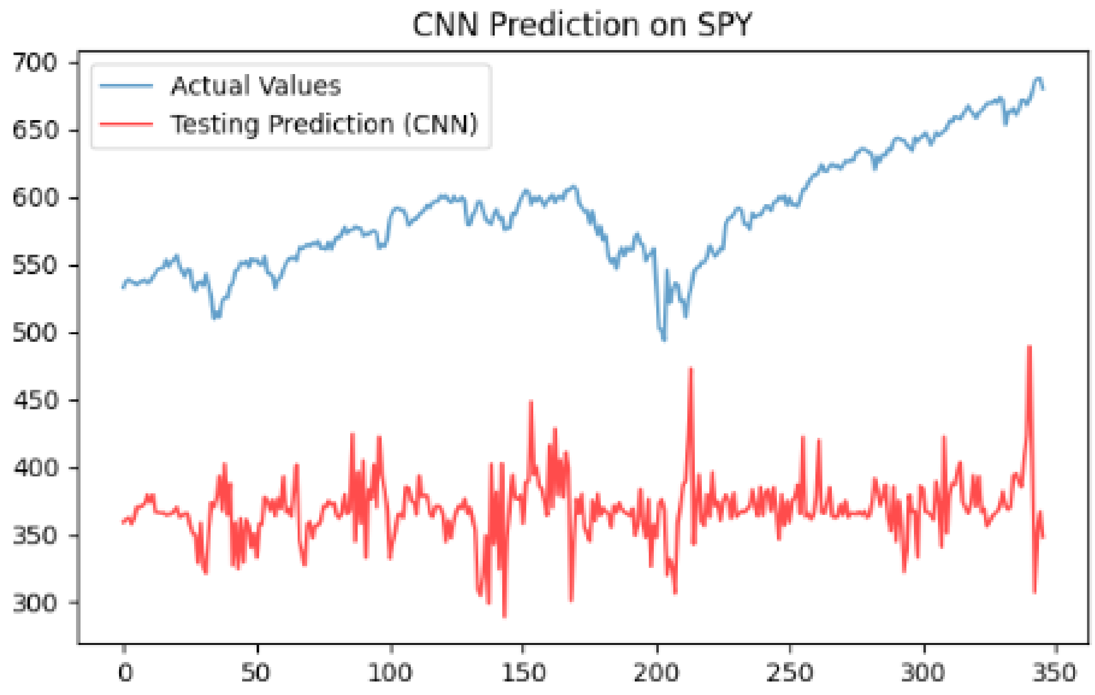


Figure 7: Out-of-sample predictions vs actual values for the CNN on GAF images of SPY price levels.

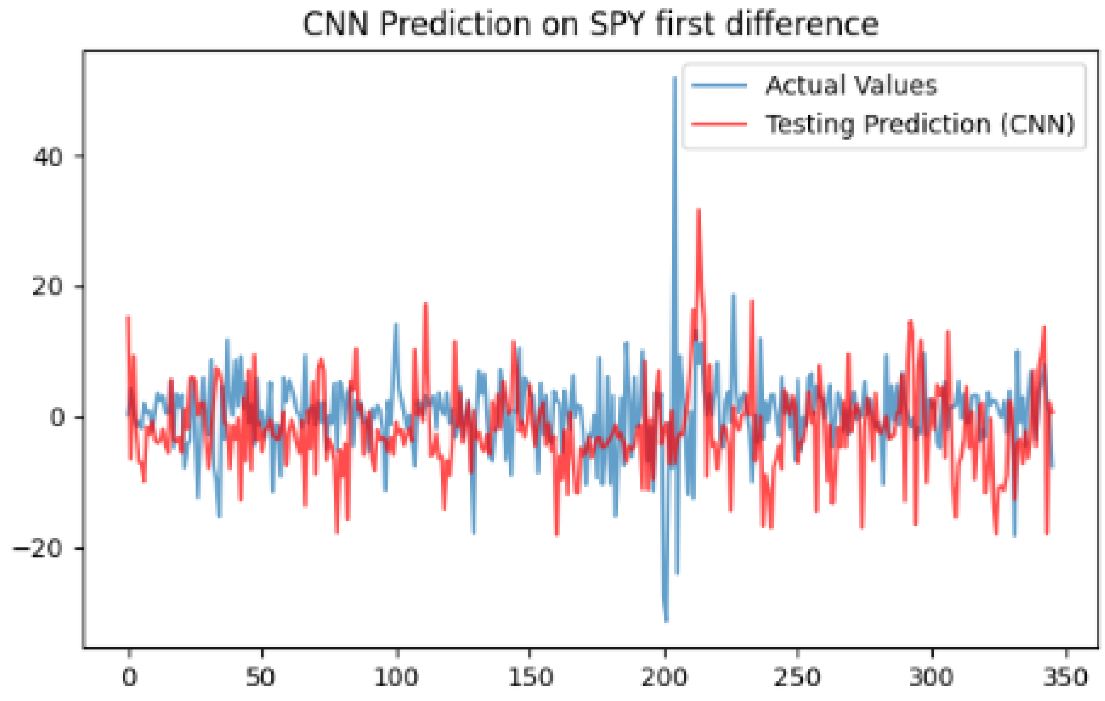


Figure 8: Out-of-sample predictions vs actual values for the CNN on GAF images of first-differenced SPY.

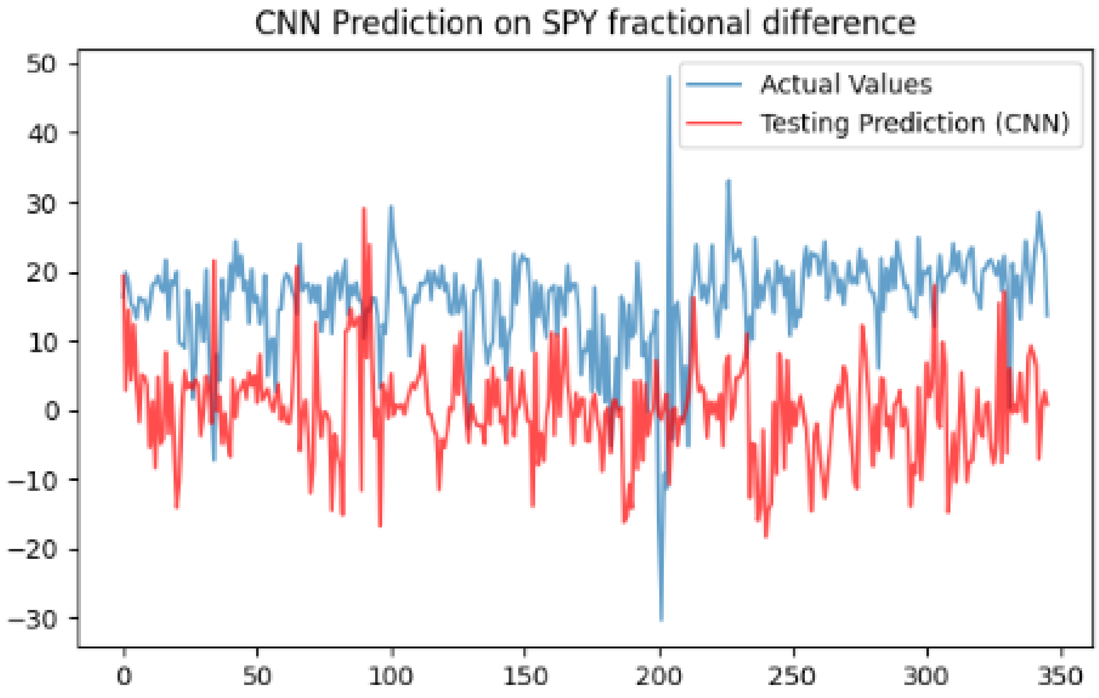


Figure 9: Out-of-sample predictions vs actual values for the CNN on GAF images of fractionally differenced SPY.

Several observations are noteworthy:

- All CNNs perform worse than their MLP counterparts. The price level CNN is particularly poor, with $R^2 \approx -28.44$ and extremely large errors.
- Even for the more regular stationary inputs (first difference and fractional difference), the CNN models do not manage to improve on the MLP baselines.
- The best CNN (on first differences, with $R^2 \approx -0.93$) still has worse metrics than the best MLP (on first differences, with $R^2 \approx -0.45$), both in terms of R^2 and absolute error magnitudes.

4.4 Interpretation of CNN Results

While CNNs have been very successful in image recognition, their advantage does not automatically translate to this GAF–based time series setting. Some plausible reasons:

1. **Sample size versus model complexity.** SPY daily data from 2018 to 2025 yield fewer than 2,000 points, which translate into a relatively small number of GAF images. CNNs with multiple convolutional layers and dozens of filters can easily overfit in such low–data regimes.

2. **Noisy financial signals.** Financial time series often have weak predictive structure. Converting them into images does not magically create signal; it just redistributes the information. In our case, the CNN seems to learn patterns that do not generalise.

3. **Loss of direct temporal interpretation.** The GAF representation makes it harder to reason directly in terms of recent lags and trend/mean–reversion structure. The MLP, by contrast, works directly on lagged values and may therefore be better suited to this simple one–step–ahead forecasting task.

Overall, the CNNs appear to be an overly complex solution for this particular problem and data set.

5 Step 4: Comparing MLP and CNN Architectures

Step 4 asks us to reflect on the differences between the MLP and CNN results and relate them to the way each architecture processes information and to the properties of our SPY time series.

5.1 Performance Comparison at a Glance

Putting Tables 1 and 2 side by side, we see a clear pattern:

- For all three representations (levels, first differences, fractional differences), the MLP outperforms the CNN in out–of–sample R^2, RMSE, and MAE(Tong and Zhu).
- The gap is especially large for the price level and fractional difference series, where CNN errors are several times larger.
- Even the best CNN (on first differences) is noticeably weaker than the best MLP (also first difference).

So, in this setting, **simpler wins**: a relatively modest feedforward network on lagged data beats a deeper convolutional network on image representations.

5.2 How the Architectures Process the Data

MLP. The MLP treats the last 20 observations as a 20–dimensional feature vector. The network can learn nonlinear combinations of these lags, but it does not explicitly encode spatial or temporal locality. This is both a limitation and a strength:

- The model is simple enough not to over–react to small idiosyncratic patterns in a short training sample.
- It is well–matched to the task: one–step–ahead prediction based on a short look–back horizon.

CNN on GAF. The CNN, by contrast, operates on 20×20 GAF images. It looks for local patterns in the two–dimensional angular field of the time series:

- Convolutional filters and pooling layers are powerful when there are stable local structures (e.g., edges in photos).
- In our case, the GAF images encode interactions between lags in a fairly indirect way, and the small amount of training data makes it hard for the CNN to learn robust filters.

When the underlying dataset is small and noisy, this additional layer of abstraction can hurt more than it helps.

5.3 Role of the Time Series Properties

The SPY series we analysed has several characteristics that shape model performance:

- **Non–stationary levels with weak predictability.** At the daily frequency, SPY is close to a random walk. This inherently limits the potential forecasting accuracy of any model, no matter how sophisticated.
- **Stationary but noisy returns.** Once we difference, we obtain a stationary series with volatility clustering but relatively limited linear predictability. MLPs can capture mild non-linear relationships in such data; CNNs, however, may simply overfit noise in the image domain.
- **Limited data.** Under 2,000 observations is small for training deep networks. It is enough for a regularised MLP but arguably too small for a two–layer CNN on images to reach its potential.

The combination of weak signal, small sample size, and additional transformation complexity explains why the CNNs underperform.

5.4 Practical Takeaways for a Stat Arb Team

If we were genuinely advising a statistical arbitrage desk, the main messages from this project would be:

- **Do the basics first.** Ensuring stationarity (via differencing or fractional differencing) and choosing an appropriate forecasting horizon are more important than jumping directly to complex architectures.
- **Match model complexity to data richness.** CNNs and other heavy architectures may be justified for high–frequency data with millions of observations or for cross–sectional problems with many assets. For a single daily series like SPY, simpler models are often more robust.

- **Focus on economic signal, not just deep learning.** The negative R^2 values remind us that if the underlying process is close to a martingale, no amount of deep learning will reliably extract alpha. In practice, one would need richer features (macro variables, cross–sectional information, order book data, etc.) rather than just more layers.

6 Conclusion

This project walked through the full pipeline a quantitative team might follow when building a forecasting model for a financial time series:

1. We diagnosed non stationarity in SPY price levels and constructed both first–differenced and fractionally differenced (with $d = 0.6$) series that passed ADF stationarity tests.
2. We trained MLP models on each representation and found that stationarity matters: models on differenced data outperform those on raw levels, though all models still struggle to beat a naive mean predictor.
3. We transformed lagged sequences into GAF images and trained CNNs, which, in this setting, performed worse than the MLPs across all metrics.
4. We interpreted these results in light of how MLPs and CNNs process information and of the inherent properties of daily SPY data.

The main lesson is not that CNNs are "bad" or MLPs are "good", but that in finance, model choice must be grounded in data characteristics, sample size, and economic plausibility. For a relatively short, noisy daily time series such as SPY, a well–designed and carefully interpreted MLP on stationary inputs is a more sensible starting point than a highly complex CNN architecture on transformed images.

In a real trading environment, we would extend this analysis by incorporating additional features, experimenting with longer horizons, and benchmarking against simpler statistical models (ARIMA, random walk, etc.) to ensure that any claimed predictive power is both statistically and economically meaningful(Hutter et al).

References

- Zhang, Ashton, et al. “Dive into Deep Learning.” *arXiv*, 2023.
  `https://doi.org/10.48550/arXiv.2106.11342`

- TensorFlow. “Introduction to Keras Tuner.” 2025.
  `https://www.tensorflow.org/tutorials/keras/keras_tuner?hl=en`

- Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning*. Springer, 2019.
  `https://link.springer.com/book/10.1007/978-3-030-05318-5`

- Tong, Yu, and Hong Zhu. “Hyper-Parameter Optimization: A Review of Algorithms and Applications.” *arXiv*, 2020.
  `https://doi.org/10.48550/arXiv.2003.05689`

- Yang, Li, and Abdallah Shami. “On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice.” *Neurocomputing*, vol. 415, 2020, pp. 295–316.
  `https://doi.org/10.1016/j.neucom.2020.07.061`

- State Street Investment Management. “SPY: SPDR® S&P 500® ETF Trust.” 2025.
  `https://www.ssga.com/us/en/intermediary/etfs/spdr-sp-500-etf-trust-spy`

- ELVISESPINAL. “Time Series Analysis Using Fractional Differencing.” *Kaggle*, 2019.
  `https://www.kaggle.com/code/elvisesp/time-series-analysis-using-fractional-differencing/notebook`