Predicting Stock Movement with Neural Networks —— A Data-Driven Approach Using AAPL Return Trends

Stephanie Daniella Hernandez Prado
Nanzhu Li
Serena Zhou

April 8, 2025

Contents

1 Introduction

1.1 Background and Motivation

Traditional financial theories, such as the Efficient Market Hypothesis (EMH), claim that asset prices fully reflect all available information, making it impossible to consistently predict future movements based on past data. However, behavioral finance challenges this view by identifying market anomalies and cognitive biases that influence investor behavior. One of the most widely observed phenomena is the **momentum effect**, where assets that have performed well in the recent past tend to continue performing well in the short to medium term, and vice versa for underperforming assets. This effect contradicts the EMH and suggests that investors may underreact to new information or follow patterns without reassessing fundamental values.

This behavioral perspective is supported by psychological biases such as **confirmation bias**, where investors selectively focus on information that confirms their existing beliefs, and **herding behavior**, where individuals follow market trends driven by group actions rather than independent analysis. Together, these biases can reinforce price trends and create opportunities for predictive modeling based on historical return data.

Given these insights, this project explores a neural network approach for multi-class trend prediction using technical indicators derived from past returns. The goal is to capture potential momentum patterns in financial time series while acknowledging the behavioral forces that may drive them.

1.2 Research Objective

The primary objective of this research is to develop a data-driven model capable of classifying short-term stock price movements into three categories: **uptrend**, **downtrend**, or **neutral**, based on historical return patterns.

To achieve this, we aim to:

- Design a threshold-based multi-class labeling approach to reflect short-term market trends.
- Develop relevant features from raw financial time series data, incorporating technical indicators and return windows.
- Implement and train a neural network model tailored for multi-class time series classification.
- Evaluate the model's ability to predict trend direction through performance metrics and visual comparisons against actual market movements.

Ultimately, This project seeks to explore whether deep learning can capture behavioral market phenomena, such as momentum and reversal patterns, which are not fully explained by traditional financial theories.

1.3 Contribution

- A statistically interpretable method for indicating short-term trends.

  We propose a trend classification approach based on the evolution of future returns and dynamic thresholds derived from empirical volatility. This method provides a principled way to convert sudden price data into discrete and actionable trend categories (uptrend, neutral trend, downtrend), allowing reliable monitoring for training models.

- Simple, fully stand-alone forecasting framework based solely on historical returns.

  The architecture is compact, interpretable and achieves high performance through the exclusive use of public price data.

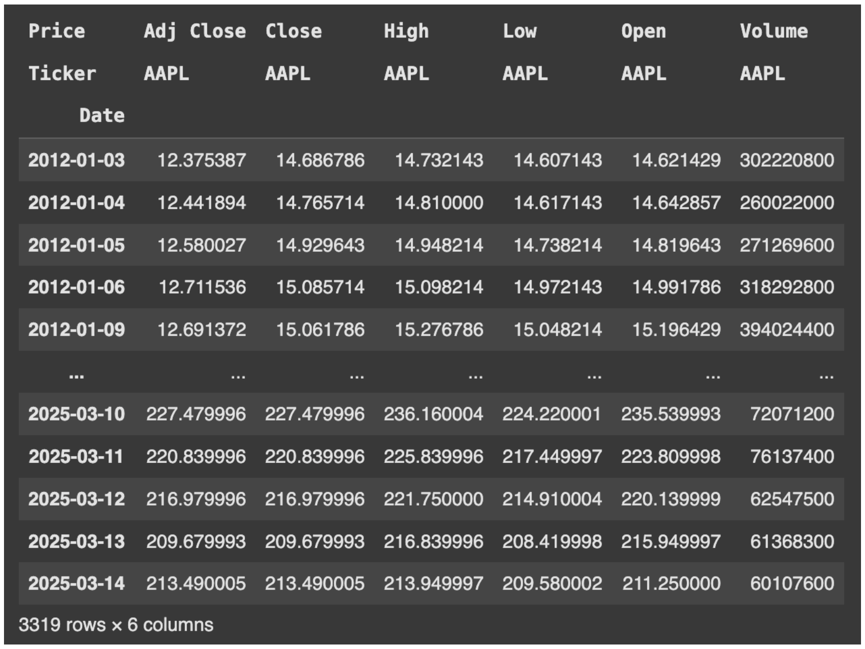
- A reusable foundation for lightweight financial prediction models

  It's minimal design makes it easy to deploy, evaluate, and integrate into broader decision-making systems.

2 Data

2.1 Overview

We collected daily stock data for **Apple Inc. (AAPL)** from **January 1, 2012 to March 15, 2025** using the `yfinance` Python library. The dataset includes open, high, low, close, adjusted close prices, and trading volume.



Price	Adj Close	Close	High	Low	Open	Volume
Ticker	AAPL	AAPL	AAPL	AAPL	AAPL	AAPL
Date						
2012-01-03	12.375387	14.686786	14.732143	14.607143	14.621429	302220800
2012-01-04	12.441894	14.765714	14.810000	14.617143	14.642857	260022000
2012-01-05	12.580027	14.929643	14.948214	14.738214	14.819643	271269600
2012-01-06	12.711536	15.085714	15.098214	14.972143	14.991786	318292800
2012-01-09	12.691372	15.061786	15.276786	15.048214	15.196429	394024400
...	...	...	...	...	...	...
2025-03-10	227.479996	227.479996	236.160004	224.220001	235.539993	72071200
2025-03-11	220.839996	220.839996	225.839996	217.449997	223.809998	76137400
2025-03-12	216.979996	216.979996	221.750000	214.910004	220.139999	62547500
2025-03-13	209.679993	209.679993	216.839996	208.419998	215.949997	61368300
2025-03-14	213.490005	213.490005	213.949997	209.580002	211.250000	60107600

3319 rows × 6 columns

Figure 1: Raw dataset downloaded using `yfinance`. Each row represents one trading day.

In this project, we aim to model and predict short-term market trends. Instead of using raw prices, we compute **daily returns** based on closing prices. Returns represent the relative change in price from one day to the next and are more suitable for capturing short-term dynamics.

Using returns provides several advantages:

- **Scale-independence:** Returns represent proportional changes, enabling comparisons across time periods and assets.
- **Volatility sensitivity:** Return series better reflect short-term market fluctuations and investor reactions—essential for trend classification.

2.2 Data Processing

(a) Return Calculation

Daily returns were computed using the percentage change in adjusted closing prices:

$$\text{Return}_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \texttt{pct\_change()}$$

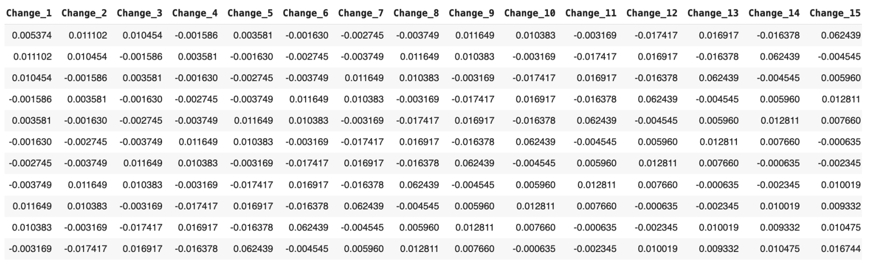
(b) Sliding Window for Feature Construction

To capture short-term return dynamics, we implemented a **sliding window** approach. A sliding window is a technique where a fixed-size window (here, 15 days) moves forward one day at a time along the return series, creating overlapping samples. Each window consists of:

15 consecutive daily returns:

$$\begin{gathered}\text{Sample}_1 = [r_1, r_2, \ldots, r_{15}] \\ \text{Sample}_2 = [r_2, r_3, \ldots, r_{16}] \\ \vdots \\ \text{Sample}_{3304} = [r_{3302}, r_{3303}, \ldots, r_{3316}]\end{gathered}$$

This process resulted in a dataset of **3,304 samples**, each consisting of **15 features**.



Change_1	Change_2	Change_3	Change_4	Change_5	Change_6	Change_7	Change_8	Change_9	Change_10	Change_11	Change_12	Change_13	Change_14	Change_15
0.005374	0.011102	0.010454	-0.001586	0.003581	-0.001630	-0.002745	-0.003749	0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439
0.011102	0.010454	-0.001586	0.003581	-0.001630	-0.002745	-0.003749	0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545
0.010454	-0.001586	0.003581	-0.001630	-0.002745	-0.003749	0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960
-0.001586	0.003581	-0.001630	-0.002745	-0.003749	0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960	0.012811
0.003581	-0.001630	-0.002745	-0.003749	0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960	0.012811	0.007660
-0.001630	-0.002745	-0.003749	0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960	0.012811	0.007660	-0.000635
-0.002745	-0.003749	0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960	0.012811	0.007660	-0.000635	-0.002345
-0.003749	0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960	0.012811	0.007660	-0.000635	-0.002345	0.010019
0.011649	0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960	0.012811	0.007660	-0.000635	-0.002345	0.010019	0.009332
0.010383	-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960	0.012811	0.007660	-0.000635	-0.002345	0.010019	0.009332	0.010475
-0.003169	-0.017417	0.016917	-0.016378	0.062439	-0.004545	0.005960	0.012811	0.007660	-0.000635	-0.002345	0.010019	0.009332	0.010475	0.016744

Table 1: Example of input samples

(c) Rolling Sum for Trend Labeling

To assign trend labels to each window, we computed a **rolling sum** S_t of the returns in the 5 days *after* each input window:

$$S_t = \sum_{i=t+1}^{t+5} r_i$$

This value captures the cumulative price movement immediately following each return window and will be used in the classification rule described in the next section.

3 Model Design and Implementation

3.1 Neural Network

A neural network (ANNs) is a computational model composed of layers of interconnected nodes that are capable of learning nonlinear mappings between input features and target outputs. Inspired by the structure of biological neurons, each artificial neuron performs a weighted sum of its inputs followed by a nonlinear activation function. Through a process of iterative optimization using labeled data, neural networks are able to learn complex patterns and decision boundaries that traditional linear models may fail to capture.

The most basic type of neural network is the feedforward neural network, also known as the multilayer perceptron (MLP). In this architecture, data flows in one direction—from the input layer through one or more hidden layers to the output layer—without any recurrent or convolutional connections. Each neuron in a given layer is fully connected to all neurons in the subsequent layer.

In classification tasks, the final layer of the network typically uses a softmax activation function to convert the outputs into a probability distribution over predefined classes. During training, the model parameters (weights and biases) are optimized to minimize a loss function such as categorical or sparse categorical cross-entropy using a gradient-based algorithm, such as the Adam optimizer.

Figure 2 below shows a schematic of the feedforward network structure.

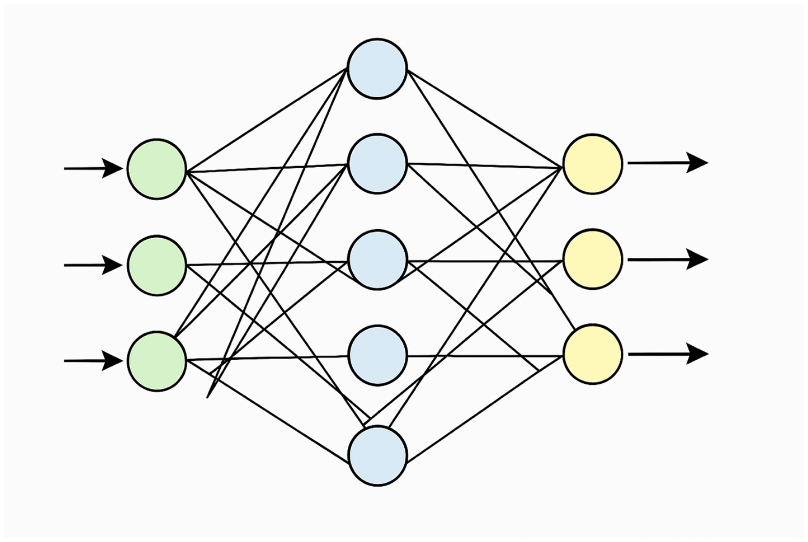


Figure 2: Feedforward Neural Network Architecture

3.2 Labeling Strategy

To assess the distributional properties of the daily return series, we first computed daily returns from adjusted closing prices. We then constructed a Q-Q plot comparing the empirical quantitative values of the daily returns with the quantitative values of a standard normal distribution. The plot is shown below.

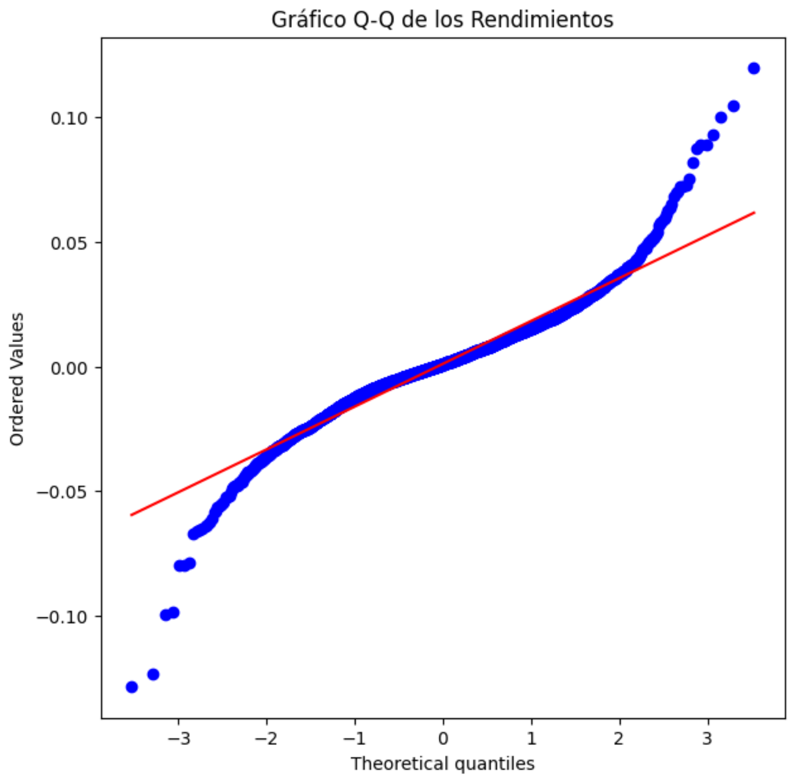


Figure 3: Q-Qplot of daily return series

From this graph we can assume that daily returns approximately follows a normal distribution function, with the center data of the graph essentially overlapping the 45-degree line. It is known that daily returns follows a normal distribution, then any linear combination of it also follows a normal distribution. Therefore, sum of the daily returns also follows something similar to the normal distribution.

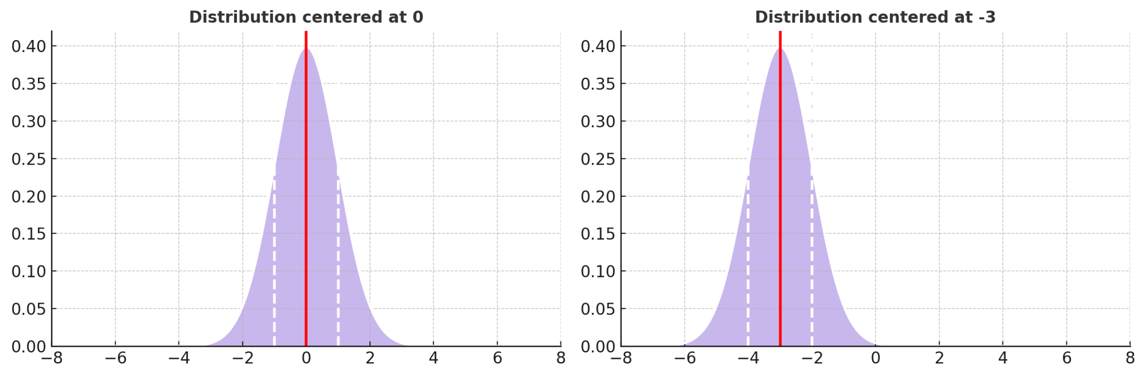


Figure 4: Effect of Distribution Centering on Threshold-based Trend Classification

It is important to note that the effectiveness of this labeling method relies on the assumption that the rolling sum of returns is centered around zero.

As illustrated in the right-hand plot, if the distribution is not centered at 0, due to drift or persistent bias—then the same fixed thresholds ($\pm 0.3 \times \sigma$) will no longer correspond to symmetric trend definitions. In such cases, negative return sums may lie in the region above the threshold and be misclassified as uptrends, even when the asset is still experiencing losses.

We then define a trend label for each sample by summing the returns of the next 5 days following each window. Based on this sum, we apply a threshold:

$$\theta = \pm 0.3 \times \sigma_S$$

where is the standard deviation of the rolling 5-day sums.

Then, each sample was categorized into one of three classes based on the following rule:

- **Uptrend (2):** if $S_t > \theta$
- **Downtrend (0):** if $S_t < -\theta$
- **Neutral (1):** if $-\theta \leq S_t \leq \theta$

This classification rule creates a buffer zone around zero and ensures that only statistically significant upward or downward movements are labeled as trends, while small fluctuations are treated as neutral.

To evaluate this labeling approach, we apply the obtained label categories to the aapl stock price data and visualize them by labeling them with colors according to the specified categories. Downtrend (0) is marked in red, Neutral (1) in yellow and uptrend (2) in green.

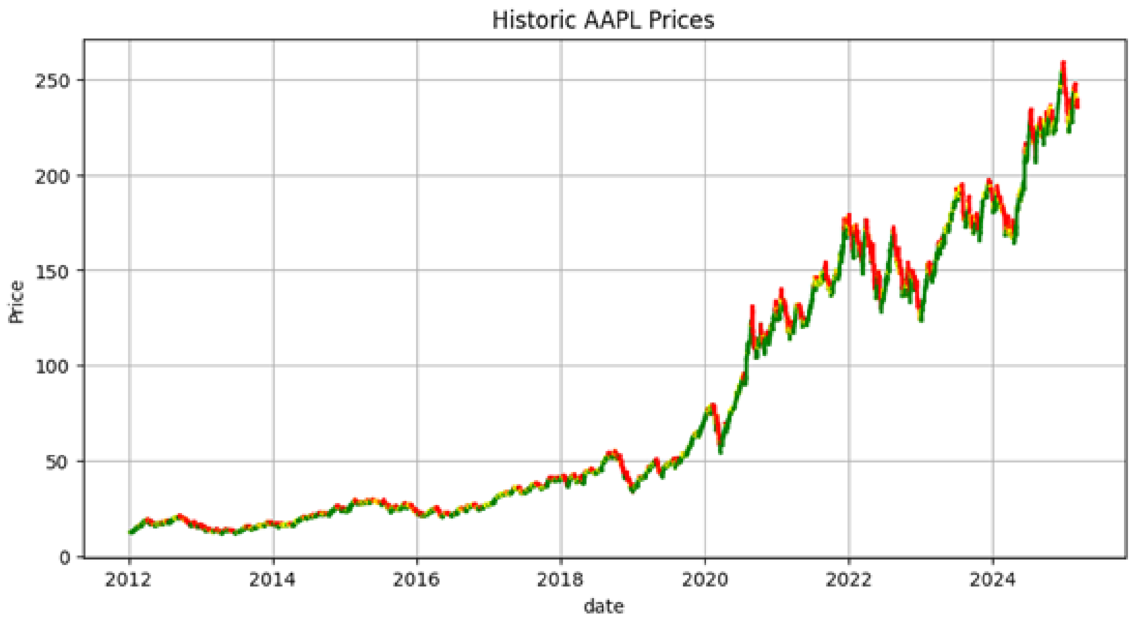
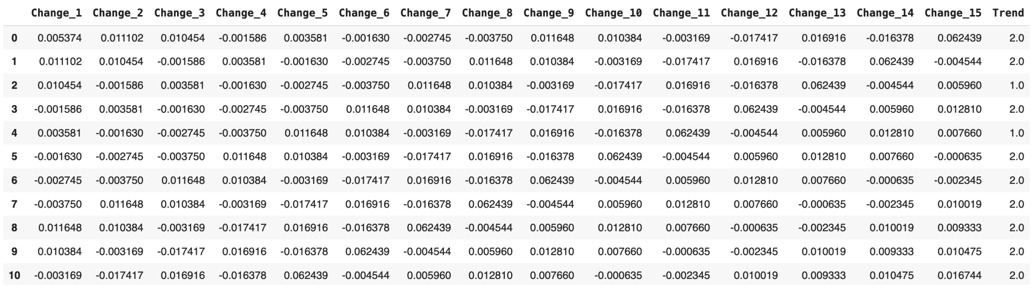


Figure 5: Historic AAPL Prices with Color-coded Trend Labels

As we can see from the picture above, our theshold is set up perfectly. When the price of the aapl stock goes down (e.g. at the beginning of 2020), it is clearly labeled as downtrend. when it goes up (e.g. in the middle of 2024), it is clearly labeled as uptrend. Thus, we can once again be sure that the assumption that daily returns follow a standard normal distribution is valid, and at the same time, our threshold perfectly reflects the trend behind the data.



	Change_1	Change_2	Change_3	Change_4	Change_5	Change_6	Change_7	Change_8	Change_9	Change_10	Change_11	Change_12	Change_13	Change_14	Change_15	Trend
0	0.005374	0.011102	0.010454	-0.001586	0.003581	-0.001630	-0.002745	-0.003750	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	2.0
1	0.011102	0.010454	-0.001586	0.003581	-0.001630	-0.002745	-0.003750	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	2.0
2	0.010454	-0.001586	0.003581	-0.001630	-0.002745	-0.003750	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	1.0
3	-0.001586	0.003581	-0.001630	-0.002745	-0.003750	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	0.012810	2.0
4	0.003581	-0.001630	-0.002745	-0.003750	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	0.012810	0.007660	1.0
5	-0.001630	-0.002745	-0.003750	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	0.012810	0.007660	-0.000635	2.0
6	-0.002745	-0.003750	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	0.012810	0.007660	-0.000635	-0.002345	2.0
7	-0.003750	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	0.012810	0.007660	-0.000635	-0.002345	0.010019	2.0
8	0.011648	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	0.012810	0.007660	-0.000635	-0.002345	0.010019	0.009333	2.0
9	0.010384	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	0.012810	0.007660	-0.000635	-0.002345	0.010019	0.009333	0.010475	2.0
10	-0.003169	-0.017417	0.016916	-0.016378	0.062439	-0.004544	0.005960	0.012810	0.007660	-0.000635	-0.002345	0.010019	0.009333	0.010475	0.016744	2.0

Figure 6: Example of Sliding Window Input Features and Assigned Trend Labels

As we mentioned in Section 2, we adopt a sliding window strategy. As shown in Figure 6, each observation consists of a 15-dimensional input vector, which is 15 consecutive daily returns, and a single output label that reflects the trend category (0,1,2). This window moves one day at a time exactly to the left, therefore generating overlapping samples that preserve temporal continuity in the data. This formulation enables the model to learn patterns from short-term historical sequences and generalize them into directional trend predictions.

3.3 Model Implementation and Training

Based on the labeled dataset created in Section 3.2, we used a fully connected forward-looking neural network to predict trend categories for each sample. The model was built using the Keras sequential API with multiple hidden layers and non-linear activations. This section provides an overview of the overall implementation of the classification model, including the neural network architecture, training process, loss function, optimization configuration, and final performance evaluation using numerical and visual methods.

3.3.1 Network Architecture

The overall neural network architecture consists of five sequential layers: one input layer, three hidden layers, and one output layer. The model accepts as input a 15-dimensional vector, representing the daily log returns from the past 15 trading sessions.

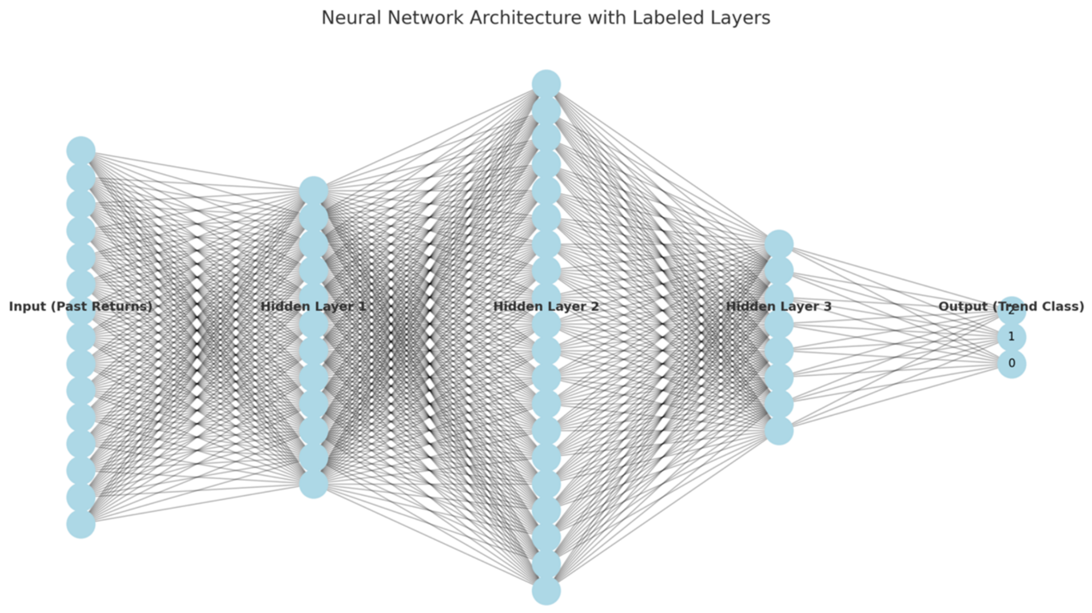


Figure 7: Neural Network Architecture with Labeled Layers

The input to the model is a 15-dimensional vector representing the daily returns over the last 15 trading days. This input passes sequentially through three hidden layers and then produces a probability distribution of three possible classes of trends.

The transformed output then enters a second hidden layer containing 20 neurons. Just like in the previous layer, each neuron is fully connected to all outputs of the previous layer. After another LeakyReLU activation, a layer with a dropout rate of 0.25 is introduced, which randomly removes a subset of the neurons during training. This regularization technique reduces the risk of overtraining by encouraging the network to learn redundant representations.

The third hidden layer consists of 8 neurons, which in turn are fully connected to the 20 outputs of the previous layer. As before, packet normalization is applied to the output of the dense layer, followed by LeakyReLU activation to maintain a stable and nonlinear transform. At this stage, the network has managed to transform the original return sequence into a more abstract representation that consistently filters and combines features relevant for trend prediction.

Finally, the output layer consists of 3 neurons corresponding to the three goal categories: downtrend (0), neutral (1), and uptrend (2). To convert the raw output into class probabilities, a softmax activation function is applied. The predicted category with the highest probability is then selected. This structure allows the network to learn nonlinear decision bounds in the reward space and efficiently distinguish between different market conditions.

3.3.2 Training Configuration

The complete dataset contains 3,304 labeled samples. Each sample consists of a 15-dimensional input vector representing the daily log returns of the past 15 trading days, paired with a categorical label indicating the market trend 5 days ahead: *downtrend* (0), *neutral* (1), or *uptrend* (2). To preserve the temporal structure of the data and simulate realistic deployment, the dataset was split chronologically: the first 80% of samples were allocated to the training set and the remaining 20% to the test set.

Prior to training, all input vectors were standardized using the `StandardScaler` , which transforms the data to have zero mean and unit variance. Standardization is particularly crucial for gradient-based optimizers and improves convergence speed and numerical stability, especially when paired with batch normalization.

The model was trained using the following configuration:

- Loss Function: Sparse categorical cross-entropy
- Optimizer: Adam
- Learning Rate: 0.001
- Batch Size: 16
- Epochs: 50
- Random Seed: 996

Loss Function. Given the multi-class classification nature of the problem, we use the sparse categorical cross-entropy loss, which is defined as:

$$L = -\sum_{i=1}^{N} \log(p_{i,y_i})$$

where N is the batch size, $y_i \in \{0, 1, 2\}$ is the true class label for the i-th sample, and p_{i,y_i} is the model's predicted probability (via softmax) for that class. This formulation is equivalent to categorical cross-entropy but is computationally more efficient because it operates directly on integer class labels rather than one-hot encoded vectors.

The function strongly penalizes incorrect predictions with high confidence while encouraging the model to assign high probabilities to correct classes. When used in tandem with softmax output, it provides a smooth probabilistic interpretation and helps calibrate uncertainty in predictions.

Adam Optimizer. To minimize this loss, we adopted the Adam optimizer—an adaptive gradient-based optimization method that combines the advantages of both momentum and RMSProp. Adam maintains exponentially decaying averages of past gradients (first moment) and squared gradients (second moment), adjusting the learning rate dynamically for each parameter at every iteration. Its update rules are given by:

$$\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1 - \beta_1)\frac{\partial L}{\partial w_t} \\
v_t &= \beta_2 v_{t-1} + (1 - \beta_2)\left(\frac{\partial L}{\partial w_t}\right)^2 \\
\hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\
w_{t+1} &= w_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}
\end{aligned}$$

The default hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ were used in our training. Adam is especially well-suited for problems involving sparse gradients and noisy data, both of which are characteristics observed in financial time series like stock returns.

Training Dynamics. To ensure robust generalization and mitigate overfitting, several regularization techniques were incorporated into the model:

- **Batch Normalization** was applied to each dense layer to stabilize gradient flow and reduce internal covariate shift.
- **Dropout** was applied after the second hidden layer with a dropout rate of 0.25, randomly deactivating neurons during training to encourage redundancy and prevent co-adaptation.
- **LeakyReLU activations** were used instead of ReLU to allow small gradients when inputs are negative, mitigating the vanishing gradient problem.

This configuration—combining adaptive optimization, probabilistic loss, and architectural regularization—enabled the network to learn complex non-linear boundaries within the return space and make trend predictions that reflect genuine patterns in financial data.

4 Experimental Results

To assess the effectiveness of our neural network model, we evaluate both quantitative accuracy metrics and qualitative visual alignment between predicted and actual trends.

4.1 Quantitative Performance

The trained model was evaluated on the test set with an overall accuracy of 92.28% and a loss of 0.2228%. These results show that the model is able to correctly classify most trend labels in unseen data.

The following prediction results show the first few trend category predictions made by the model:

```
[1 2 1 2 2 2 2 2 2 2]
```

4.2 Trend Prediction Visualization

To further validate the model's performance, we visualized the predicted trend labels overlaid on the historical AAPL price data. In this visualization:

- Red denotes predicted downtrends
- Blue denotes predicted neutral trends
- Green denotes predicted uptrends

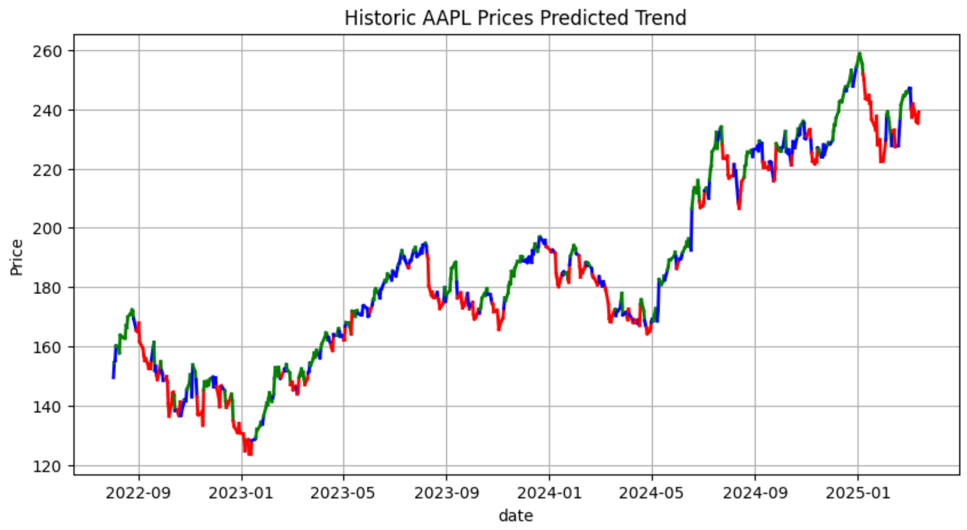


Figure 8: Predicted Trends on AAPL Price Series

As shown in Figure 8, the model has very good prediction results and it is fascinating that we don't need to evaluate the model by other complex metrics, which can be found directly by observing the picture with the naked eye. Stocks are successfully labeled by the model as green when they are rising and red when they are falling.

4.3 Comparison

We also plotted the real trend labels, derived from the labeling strategy in Section 3.2, using the same color scheme:

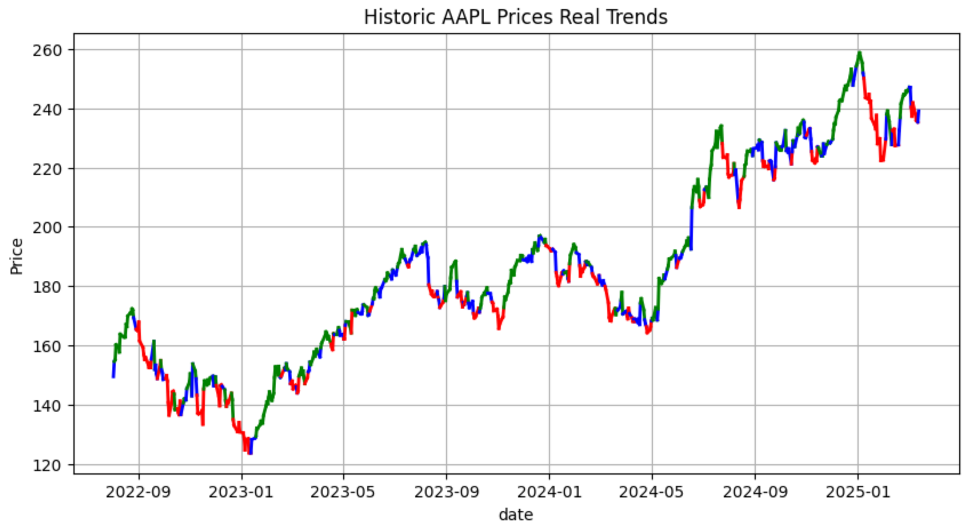


Figure 9: Real Trend Labels on AAPL Price Series

Comparing Figure 8 and Figure 9, in particular, looking at the period of decline data before September 2023, we can find that if the label classification basis is directly applied to the data, it is classified as a neture class, but the model training successfully transformed it into a decline class, which indicates that our model is very good, and is able to correct and refine the classification basis to arrive at the optimal result.

5 Conclusion and Future Work

5.1 Key Findings

This study proposes a neural network framework that uses only historical closing stock price data to classify short-term price movements and help shareholders decide whether to buy, sell or wait.

The network's design - a three-layered hidden forecasting model - identifies price movements in direction with a proven accuracy of over 92%. Visual comparisons also confirm that the predicted trend signals closely match the past behavior of AAPL prices. Interestingly, the model achieved this level of accuracy without the use of external data, highlighting the benefits of the performance-based learning capabilities of time series models and deep networks. The stability of the learning is achieved by using an Adam optimizer and a sparse transitive loss classification, which allows for efficient convergence and generalization of the model to known data.

5.2 Limitations and Future Work

Although the proposed neural network shows promising results in classifying short-term stock market trends, there are a number of limitations that restrict its application on a larger scale.

First, the current model is trained and evaluated on only a single asset, Apple Inc (AAPL). While this approach allows for controlled and consistent experiments, it limits the generalizability of the results. Market behavior can vary significantly from one asset to another due to differences in liquidity, sector impact or investor base. As a result, the robustness of the model may be questioned if it is not tested across multiple securities or asset classes.

Another important limitation relates to the trend labeling strategy. Our classification rule is based on fixed statistical thresholds derived from the standard deviation of the moving sum of future returns. While this method is simple and transparent, it assumes that the distribution of returns is relatively stationary and that the thresholds remain meaningful across different market regimes. In practice, however, financial markets are known to exhibit non-stationary behavior; volatility clusters, structural breaks and exogenous shocks can lead to changes in the distribution of returns.

In addition, the model uses only the previous day's earnings as input, which ignores many informative signals. Indicators such as trading volume, volatility indices, macroeconomic indicators and text sentiment can provide valuable context for understanding price movements. The lack of these additional inputs limits the model's ability to capture subtle nuances or market behavior driven by external factors, especially during periods of strong news or systemic change.

There is much room for future development and extension of the model. A natural next step would be to evaluate the model's performance across a wider range of assets, including stocks with different capitalization levels, volatility profiles and sector classifications. This will allow for a more comprehensive assessment of the overall applicability and limitations of the model.

Also, the labeling methodology could be adapted or improved by adding learned boundary mechanisms. Future work could include, for example, volatility-adjusted dynamic thresholds, probabilistic trend inference, or even self-supervised labeling techniques that allow the model to identify regime boundaries based on hidden data samples.

In conclusion, although our current model is a valid and interpretable baseline model for trend forecasting, there is still room for improvement through feature extensions, dynamic labeling, and more expressive architectures. Future work on these issues will significantly improve the robustness and usefulness of neural network-based trend classification systems.

Appendix

Source Code Repository

All implementation code and related materials can be found at our GitHub repository:

`https://github.com/Stephanie-Daniella/Stock-Returns-Neural-Nerwork`