Forecasting Stock Performances with Hidden Markov Models

Andrew Nguyen
Donald Bren School of Information
& Computer Sciences
University of California, Irvine
adnguy16@uci.edu

Alan Pham
Donald Bren School of Information
& Computer Sciences
University of California, Irvine
alanp7@uci.edu

Michael Wang
Donald Bren School of Information
& Computer Sciences
University of California, Irvine
mzwang@uci.edu

Abstract

This study uses a Gaussian multivariate Hidden Markov Model (GHMM) to predict daily closing prices for five of the largest technology-industry stocks: Meta, Apple, Amazon, Netflix, and Google, also known as "FAANG". We train our model with 15 years of historical data and use the fractional changes between open and high, low, and closing prices. To determine the optimal number of hidden states for each company, we performed hyperparameter tuning with a range of different values for the number of states, selecting models with the lowest Akaike Information Criterion (AIC) value. We then utilized two different prediction methods, the traditional approach of predicting a day's closing price using the previous day, and a window-based method. To evaluate our predictions, we utilize the Mean Absolute Percentage Error (MAPE) as our metric.

1 Introduction

A Hidden Markov Model (HMM) is an ideal approach to modeling sequential data, where previous observations affect later ones. HMMs are defined by a set of hidden states, an initial probability distribution for starting states, a transition matrix of probabilities of moving from one state to another, and emission probabilities of observations for a given state. A multivariate Gaussian Hidden Markov Model (GHMM) models the observations using a Gaussian distribution, and observes multiple values at each state. HMMs are covered in Lecture 8 in our CS 275P course: Slide 47 outlines the recursion framework of the Forward-Backward algorithm, and Slide 50 summarizes the E and M steps used in the Baum-Welch algorithm. Both algorithms are combined in Slide 59, as Forward-Backward is mentioned as a good option to compute marginals and posteriors efficiently. As shown in Figure 1, the GHMM is visualized as a sequence of hidden states that each influence an observed value.

While many machine learning models and approaches have been explored in the past to forecast stock prices, the ambiguity, as well as the sequential nature of stocks, make GHMMs an effective option. In terms of ambiguity, stock figures depend on subjective factors, many of which cannot be measured in an organized manner in a dataset, including but not limited to: current events for the given company, industrial shifts, politics, and the general emotions of stock traders on a given day. The hidden states of the GHMM can represent the various factors that affect stock prices, and we make the assumption that these stock prices to be generated from Gaussian distributions.

Preprint.

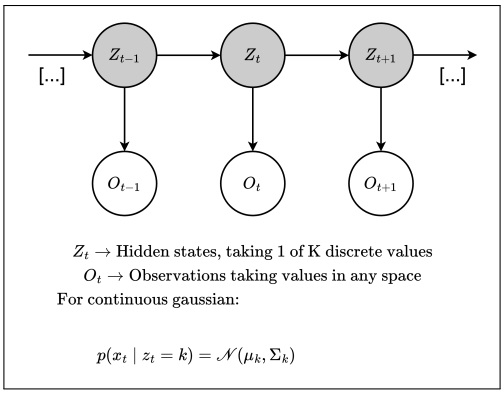

Figure 1: Graphical representation of Gaussian Hidden Markov Model.

To train our model, we obtained the high, low, opening, and closing prices for our companies of interest (Meta, Apple, Amazon, Netflix, and Google) for the last 15 years from Yahoo Finance. Referring to Figure 2, we see a clear difference between one subgroup: Netflix and Meta, and another subgroup: Google, Apple, and Amazon. With further research, we realized that the latter three companies have split their stocks within the measured period (e.g. splitting means that 1 share worth $200 can turn into 2 shares of $100) as described by Krauskopf and Mccrank (2022). This practice increases the accessibility of purchasing shares of those companies, as explained by Hussain (2024), but also results in the trendlines of those stocks appearing more steady than increasing.

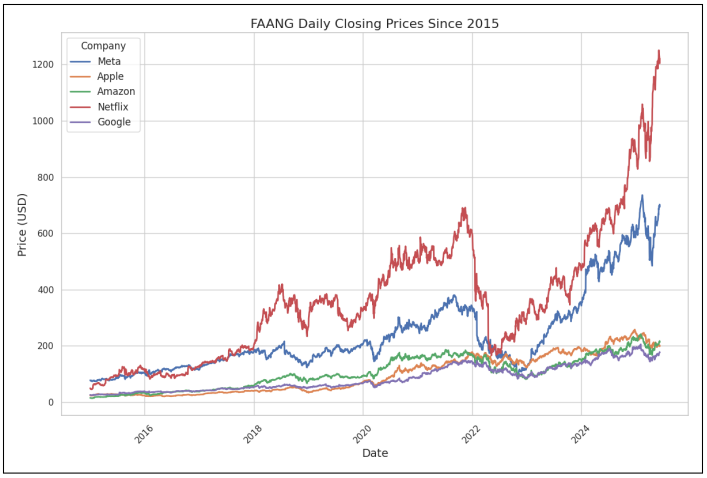

Figure 2: Visualization of FAANG company stocks since 2015.

2 Related work/Approach

Prior works have displayed promising results of HMMs for stock price prediction. One study by Gupta and Dhingra (2012) uses a 4-state HMM to predict the daily closing prices of Tata Steel, Apple, IBM, and Dell. While we adopt their strategy of calculating the fractional change in prices to use as observations, calculating predicted closing prices by adding the predicted returns to the day's opening price, and evaluating our models' test performance using MAPE, we differ from their approach in several regards. In terms of the dataset, we evaluate the biggest companies in the tech industry, as their underlying characteristics (influence in the modern world, sensitivity to world events) may yield interesting results. We explore a range of numbers of hidden states as well as modify the method of prediction. While their approach makes day-by-day predictions for observations based on a MAP estimate across a range of possible outputs, we predict our day-by-day observations using our emission probabilities.

Furthering this baseline model, we also implemented another prediction function that utilizes a similar window of previous days. We referenced work done by Hassan and Nath (2005), which utilized a GHMM with 4 states to predict airline stock prices. Their prediction method consists of calculating the log-likelihood of the observation sequence for a window before the day being predicted, finding the sequence with the closest log-likelihood in the training data, and finally using the observation for the following day as the prediction. We apply this approach to measure the performance of our models after hyperparameter tuning. We differ from their method by using the fractional change of prices as observations, whereas they directly use the low, high, opening, and closing prices.

An experiment by Nguyen (2018) also follows the window-based approach outlined by Hassan and Nath, but instead focuses on applying GHMM prediction to stock prices for the S&P 500. We followed Nguyen's approach of hyperparameter tuning using AIC as a metric of model performance, but while they used a range of 2 to 6 states and found an optimal number of 4, we used a much larger range, going up to 75 states. Additionally, while Hassan and Nath do not mention their specified window size, Nguyen uses a window size that matches the size of their test data, which we follow as well. Nguyen also introduces a sign coefficient in the final prediction step; if the log-likelihood of the matched training window is higher than that of the test window, the predicted price change is flipped in sign.

$$O_t = \left(\frac{\text{close} - \text{open}}{\text{open}}, \quad \frac{\text{high} - \text{open}}{\text{open}}, \quad \frac{\text{open} - \text{low}}{\text{open}} \right) = (\text{fracChange}, \text{fracHigh}, \text{fracLow}) \quad (1)$$

In our implementation of the GHMM, we adapt these approaches from the previous studies. We calculated the fractional change between the opening price and the high, closing, and low prices to obtain three observations per day, as shown in Equation 1. To standardize our features, we used StandardScaler from the sklearn preprocessing library. Finally, we use these three observations for each day to train our model using a 90/10 split between train and test data.

3 Experiments

3.1 Model Training/Selection

The parameters of GHMMS are represented as $\theta = \left\{ \pi, A, \{\mu_k, \Sigma_k\}_{k=1}^{N} \right\}$ where π represents a vector of initial state probabilities, transition matrix A, and a set of means and covariances $\{\mu_k, \Sigma_k\}_{k=1}^{N}\}$. To train these parameters, we utilize the Baum-Welch algorithm, which makes several passes on our training data using the expectation-maximization (EM) algorithm, repeatedly updating our parameters each time. The EM algorithm includes two steps: the expectation (E) step and the maximization (M) step.

3.1.1 E-step: forward-backward algorithm

The E-step uses the forward-backward algorithm, which computes the posterior probability of a certain state at time t given the observation data. To compute this, the algorithm computes probabilities in two passes, one forward and one backward.

$$\alpha_t(j) = P(O_1, \ldots, O_t, Z_t = j \mid \theta) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) A_{ij} \right] b_j(O_t) \quad (2)$$

The forward probabilities (as calculated in Equation 2) represent the joint probability of observing the sequence of observations $O_1, \ldots, O_t$ given all hidden states and being in hidden state j at time t. A_{ij} is defined as the transition probability from state i to j, and $b_j(O_t)$ represents the probability density of the response - which in this case is the multivariate Gaussian distribution with parameters μ_k and Σ_k.

$$\beta_t(i) = \sum_{i=1}^{N} A_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad (3)$$

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} = \frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{N} \alpha_T(i)}, \xi_t(i,j) = \frac{\alpha_t(i)A_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}. \tag{4}$$

The backward probabilities (Equation 3) are calculated starting at the end of the observation sequence, calculating the probability of observing the future sequence given the current state i at time t. These probabilities are then combined to calculate the probability of being in state j at time t ($\gamma_t(j)$) and the probability of transitioning from state i to j from time t to $t+1$ $\xi(t(i,j))$ (Equation 4).

3.1.2 M-step

$$\mu_i' = \frac{\sum_{t=1}^{T} \gamma_t(i) \cdot O_t}{\sum_{t=1}^{T} \gamma_t(i)}, \Sigma_i' = \frac{\sum_{t=1}^{T} \gamma_t(i)(O_t - \mu_i)(O_t - \mu_i)^T}{\sum_{t=1}^{T} \gamma_t(i)} \tag{5}$$

$$\log P(O \mid \lambda) = \log \left(\sum_{j=1}^{N} \alpha_T(j) \right) \tag{6}$$

The M step involves the approximation of our multivariate Gaussian parameters using maximum likelihood. This is calculated by calculating the weighted average of our observations as vectors, with $\gamma_t(j)$ used as the weights (Equation 5). With each iteration, we calculate the log-likelihood of the model by taking the log of the sum of the forward probabilities $\alpha_T(j)$ over states j, as shown in Equation 6. The EM algorithm is repeated until the log-likelihood converges and stops increasing significantly. This means that all parameters, including transition probabilities and multivariate Gaussian distributions, have stabilized.

To determine the best number of hidden states for each stock, we utilize hyperparameter tuning across multiple states. The range for hidden states we chose was [4, 6, 8, 10, 15, 20, 30, 40, 50, 75]. We chose this range as we wanted to start with a baseline number of states to be four, as that was what our related work experimented with, increasing until the model would be too large for our dataset size to support. The best model is defined as one with the lowest Akaike Information Criterion (AIC).

$$AIC = 2k - 2ln(L) \tag{7}$$

As shown in Equation 7, k is the number of parameters and L is the log-likelihood of the model on the training data. As the number of hidden states increases, the higher the number of parameters needed to be trained. This, in turn, penalizes the AIC score for more complex models. We select the model with the lowest AIC, which is defined as the model that best fits the data while also penalizing for the number of parameters.

Setting Apple as an example, hyperparameter tuning resulted in 20 hidden states yielding the lowest AIC. In Figure 3, we plotted the log-likelihood as a function of iterations and the resulting transition matrix. This shows not only that the log-likelihood has converged by 500 iterations, but also an intricate transition matrix featuring probabilities across all 20 states. This process is repeated for the rest of the companies, where we explore two different prediction methods.

3.2 Baseline: Traditional Prediction

As previously stated, our model utilizes Baum-Welch and inherits its fitting method from the hmmlearn library. The model then infers transitions using Viterbi, which gives us the most probable next state for the next day. The Viterbi algorithm is implemented in our model by the following: initially, our starting state is determined by finding the maximum of our initialization matrix as defined in Equation 8.

$$Z_1 = \arg\max_j [\pi_j \cdot b_j(O_1)] \tag{8}$$

Where π_j is the probability of starting at state j (using an initialization vector), and $b_j(O_1)$ is the probability of observing O_1 in state j. Recursively, we then determine subsequent states via a transition matrix as outlined in Equation 9.

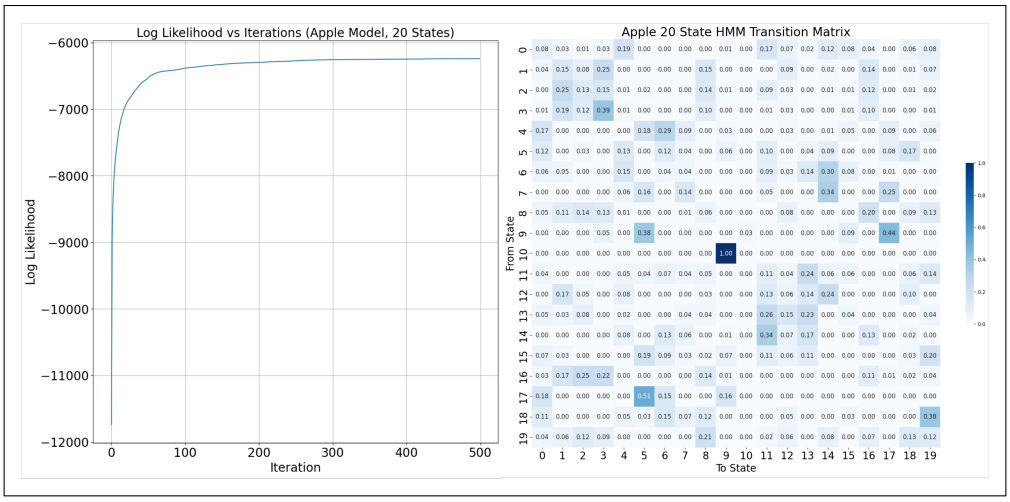

Figure 3: Log-likelihood vs. iterations and transition matrix of trained AAPL model (20 hidden states)

$$Z_{t+1} = \arg\max_j \left[\max_i \left(\text{Score}_t(i) \cdot A_{ij} \right) \cdot b_j(O_{t+1}) \right] \tag{9}$$

Where $\text{Score}_t(i)$ is the highest probability of any state endding in state i, A_{ij} is the transition probability of state i to j, and $b_j(O_t)$ is the marginal probability of observing O_t in state j. To forecast returns, we implemented a simple approach for the baseline. Given a day's inferred state, we utilize the transition matrix to predict the most probable state for the next day. We find the mean return for the most probable next-day state and project it onto the next day's opening price to predict the next day's closing price.

There is some naivety to this model and prediction strategy as it assumes the market will always transition to the highest probability next state. This means the predictions are not sensitive to volatile shifts such as crashes or rallies. It also assumes that the next state will always yield the mean return, disregarding any variances within each state. The model also only considers the current day's data concerning tomorrow's results, which, while correlated from prior days, may not capture longer-term patterns.

3.3 Exploratory: Window-based prediction

An alternative approach to predicting the next day's closing price is the window-based method defined by Hassan and Nath. For each data point in the test data set, a fixed-length window is constructed with the target point at the end. The log-likelihood of the sequence within the window is calculated, and the window is then compared to windows of the same length in the training set. until we select the training window with the closest log-likelihood. The observation following the window will be used to determine the closing price of the next day. This is repeated for each point in the test set.

4 Results

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|\text{actual}_i - \text{predicted}_i|}{\text{actual}_i} \tag{10}$$

To evaluate our model's test performance, we calculated the MAPE between the actual and predicted closing prices (as shown in Equation 10) for both our traditional prediction and window-based prediction methods. Table 1 shows the best number of hidden states we found through hyperparameter tuning for each company and the two MAPE values.

Hyperparameter tuning using AIC as the performance metric resulted in the best number of hidden states to be 15 for Meta and Google, and 20 for Amazon, Apple, and Netflix. The higher number of

Table 1: Best number of hidden states and MAPE for stocks

Stock Data	Best # Hidden States	Traditional Prediction MAPE (%)	Window Prediction MAPE (%)
META	15	1.68	2.12
AAPL	20	1.25	1.43
AMZN	20	1.43	1.67
NFLX	20	1.46	2.36
GOOGL	15	1.12	1.67

hidden states compared to our referenced works is likely due to the volatile factors we mentioned earlier. However, using a larger dataset may yield higher performance from higher numbers of hidden states.

There doesn't seem to be a clear difference in the number of hidden states between companies that have split shares (Apple, Amazon, Google) and companies that don't (Meta, Netflix). However, companies that split shares do tend to have a lower MAPE for both traditional and window-based prediction. This could be attributed to the fact that splitting shares results in less drastic changes in price across time. This could also explain why window-based prediction performs better for steadier stocks.

Here, we display the results of the prediction for one company that has not split its shares within the evaluation time (Meta) and one company that has split their shares (Google). See Appendix Figures 6, 7, and 8 for the rest of the prediction plots. Shown in Figure 4 are plots of predicted closing prices generated by both prediction methods against the actual closing prices for our test data for Meta.

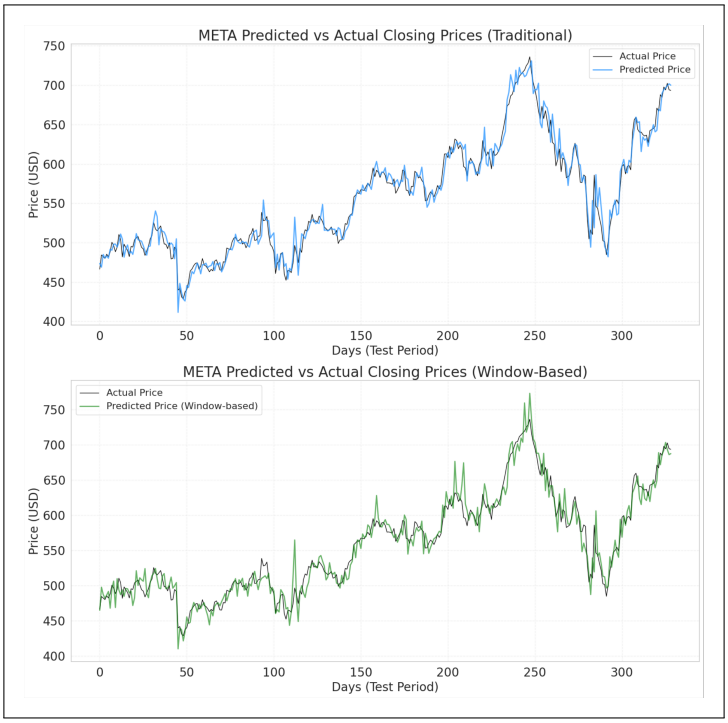

Figure 4: Plots of actual vs predicted closing price for Meta

From our observation of the predictions, we find the model to perform quite well, aligning with the trends in the actual results. We do notice, however, that at dips and peaks, the model tends to slightly overestimate the severity of those changes. While both methods of prediction match the general pattern of closing prices, the predictions using the window-based method are much less stable than the predictions using the traditional one-day method. While window-based prediction performs reasonably during moderate sections of the test set, it should be noted that performance starts to deteriorate during sharp upswings, suggesting that the model may be more applicable for more stable

stocks. Similarly, the predictions for Netflix, another company that has not split its shares, also show that window-based prediction performs worse than traditional prediction.

Shown in Figure 5 are the plots of predicted versus actual closing prices for Google generated by both prediction methods.

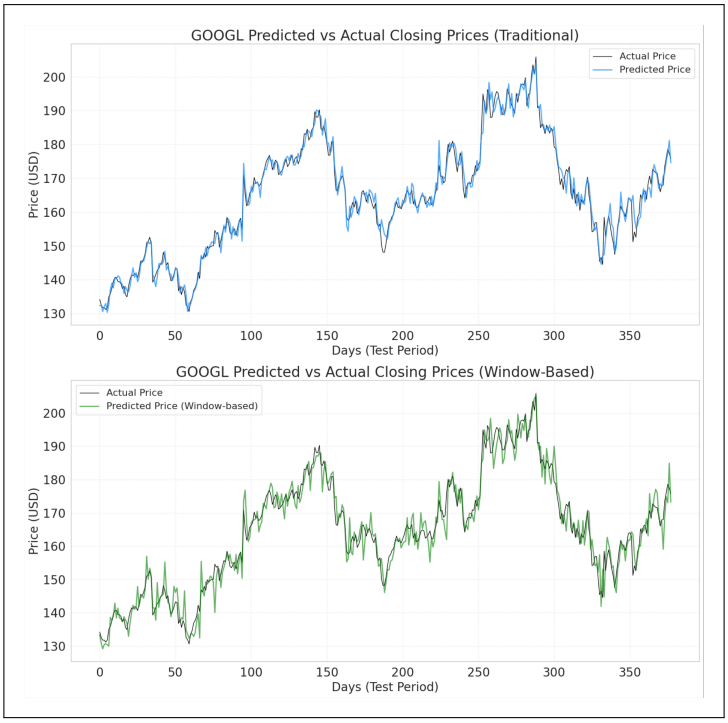

Figure 5: Plots of actual vs predicted closing price for Google

The traditional predictions for Google, a company that has split its shares within the evaluation window, match the actual closing prices much closer than those of Meta or Netflix. This same outcome can be seen to a lesser extent in the MAPE and plots for Amazon and Apple. The shape of the window-based predicted curve for Google also better captures the actual prices than it does for Meta, but it still underperforms compared to traditional prediction.

5 Conclusion

In past studies, GHMMs have been effective in predicting stock prices. In this work, we explored the usage of a multivariate GHMM in predicting closing prices for five of the largest technology companies. Through hyperparameter tuning, we found the optimal number of hidden states given each company, and we evaluated our models through two different prediction methods: a traditional prediction method using a single-day reference and one using a previous window of days. We found that while both methods perform well, the traditional prediction method resulted in a lower MAPE. We also noticed a small improvement in the model's prediction when training on and predicting stock prices from a company that splits its shares. In the future, our approach could be further expanded by finding larger datasets to better support the training of GHMMs with larger numbers of hidden states, or testing on more companies that split versus don't split their shares to see if the disparity in prediction persists.

References

1. Dhingra, B., & Dyer, C. (2018). Using Hidden Markov Models for Stock Trading. Duke University. https://users.cs.duke.edu/~bdhingra/papers/stock_hmm.pdf
2. Hasegawa-Johnson, M. (2021, Fall). Lecture 15: Baum–Welch [Lecture notes]. *ECE 417: Multimedia Signal Processing*, University of Illinois Urbana–Champaign. https://courses.grainger.illinois.edu/ece417/fa2021/lectures/lec15.pdf
3. Hassan, M. R., & Nath, B. (2005). Stock Market Forecasting Using Hidden Markov Models: A New Approach. In *Proceedings of the IEEE 5th International Conference on Intelligent Systems Design and Applications*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1578783&tag=1
4. Hussain, A. (2024). What Is a Stock Split? *Business Insider*. https://www.businessinsider.com/personal-finance/investing/what-is-a-stock-split
5. Krauskopf, L., & Mccrank, J. (2022). Alphabet Stock Split May Trigger Retail Buying, Dow Inclusion. *Reuters*. https://www.reuters.com/business/retail-consumer/alphabet-latest-megacap-split-its-stock-may-trigger-retail-buying-2022-02-02
6. Nguyen, N. (2018). Hidden Markov Model for Stock Trading. *Journal of Risk and Financial Management*, 6(2), 36. https://www.mdpi.com/2227-7072/6/2/36

Appendix

Figure 6: AAPL Traditional and Window Prediction

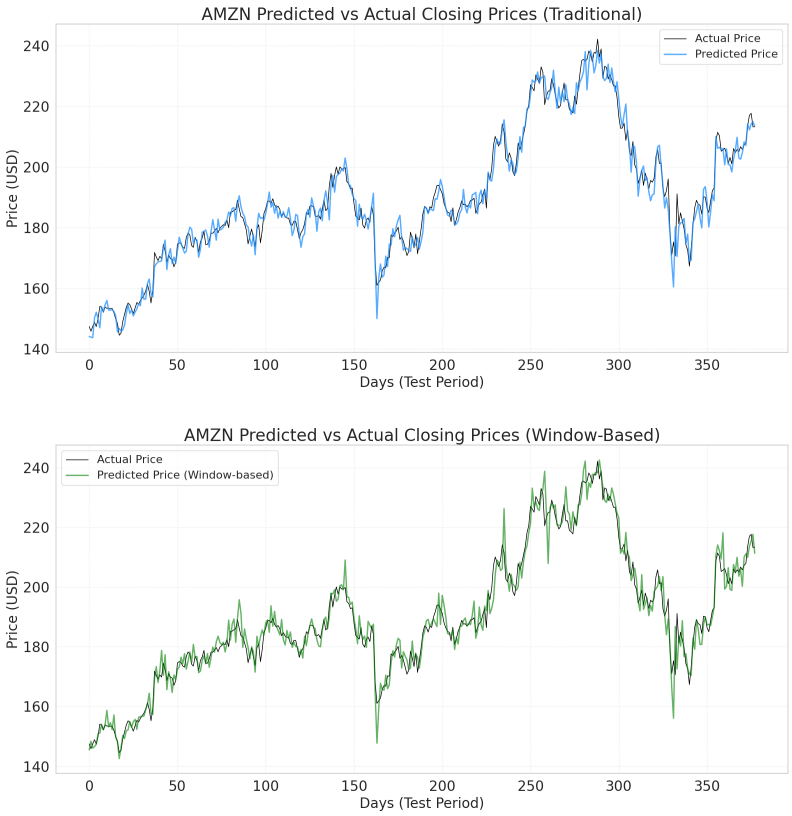

Figure 7: AMZN Traditional and Window Prediction

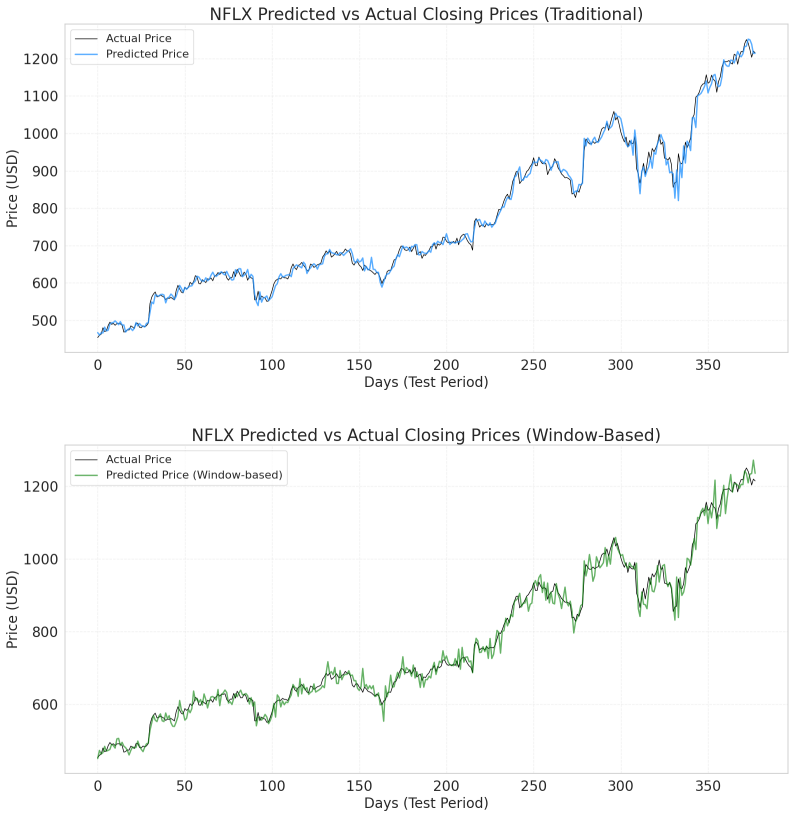

Figure 8: NFLX Traditional and Window Prediction