Time-Series Forecasting of NVDA Stock using Naive and Prophet Models

By Kourosh Ghahramani (2331288), Wendy Wang (2369115), Alana Smith (2460519)

1 Background

1.1 NVIDIA Stock

NVIDIA (NVDA) is one of the more influential companies in modern technology, playing a central role in artificial intelligence, data centers, and semiconductor innovations. Its stock has gained significant attention and has strong long-term growth, high short-term volatility. These characteristics make NVIDIA an interesting candidate for exploring stock forecasting methods.

1.2 The Naïve Model

The chosen naïve model is one of the simplest and most used baselines in time-series forecasting. It assumes that the best prediction for the next step is the most recent observed value: $\widehat{y_{t+1}} = y_t$

Because of its simplicity and strong baseline performance, the naïve model provides a meaningful point of comparison. Using the naïve model allows us to evaluate whether Prophet is truly adding predictive value or simply estimating yesterday's price.

1.3 Prophet Forecasting Model

Prophet is an open-source forecasting tool developed by Meta and is designed for time-series with strong trends and seasonality components. It uses an additive model consisting of trend (linear or nonlinear growth curves), seasonality (daily, weekly, yearly), holiday or event effects. $y(t) = g(t) + s(t) + h(t) + \varepsilon_t$. Prophet has several strengths when it comes to stock prediction like how it automatically handles missing data and irregular intervals However, as Prophet wasn't created with the intention of stock prediction it has weaknesses, for example Prophet tends to smooth trends heavily which means it overlooks short-term movements in the stock.

2 Methodology

We obtained daily price data for NVDA from Yahoo Finance using the 'yfinance' Python package. We keep the date and adjusted closing price and set ds as the trading date and y as NVDA adjusted close on that date To provide additional context for forecasting We did:

14-day Relative Strength Index (RSI14) We compute the daily price change delta = y.diff(), separate positive and negative changes into gains and losses, take 14-day rolling averages, and then convert them into RSI using the standard formula:

$$RSI14 = 100 - \frac{100}{1 + RS} \quad RS = \frac{avg\ gain}{avg\ loss}$$

We compute the daily percentage return and a 20-day rolling volatility vol_20. A binary feature high_vol is set to 1 on days where vol_20 is above the 80th percentile, marking high-volatility regimes. We also add two date-based flags: an is_ai_hype indicator for the recent AI boom period, and an export-control flag for a window where US restrictions affected NVIDIA's GPU sales.

To capture market context, we include daily returns for several related tickers (SPY, AMD, MSFT, etc.) aligned to the NVDA timeline. These act as simple multi-factor signals describing what the broader market, tech sector, and key competitors are doing on the same day. After constructing all features, we drop rows with missing values, keep a copy of the original price for the naïve model, and apply a log transformation to the Prophet target to stabilize variance.

We focus on forecasting the log of the adjusted closing price rather than the raw price. Let P be the adjusted closing price on trading day t. We define $y_t = \log(P)$. Using log prices helps stabilize the variance of the series and make the absolute error more comparable across time. Both the naive baseline and Prophet are trained and evaluated in log space. At the end, we convert predictions back to price space by applying the exponential function, so the errors can also be interpreted in actual dollars.

Each model was qualitatively evaluated using overlayed plots of the actual price vs the forecasted price as well as quantitively using the R^2 score.

Prophet hyperparameter exploration

Within the Prophet framework, we also performed a small hyperparameter exploration on the training and validation data. In particular, we varied the changepoint settings, including number of changepoints and the changepoint prior to scale, to control how quickly the trend is allowed to adapt to NVDA's rapid growth. We also toggled different seasonality components (weekly and yearly seasonality on or off) and experienced including or excluding holiday effects and important AI events by using Prophet's holiday utilities.

2.1.1 Relationship with Class Topics

This project is closely tied to several key ideas from class. For example, for our train/test split for time series instead of random splits we use a chronological split and hold out the most recent 500 days. This follows the time series of evaluation principles discussed in lectures. We also used our features and domain knowledge. We used a RSI14, volatility measures, regime flags, and peer return as an example of turning raw time series into more informative features. This mirrors the course focus on suing domain knowledge to make meaningful inputs

2.2 Results

Table 1: R^2 scores for NVDA Stock Price Forecasting Models

	Baseline Prophet	Log Prophet	Rolling Prophet	Naïve Model
R^2 score	-1.221264	-0.491538	0.797401	0.996256

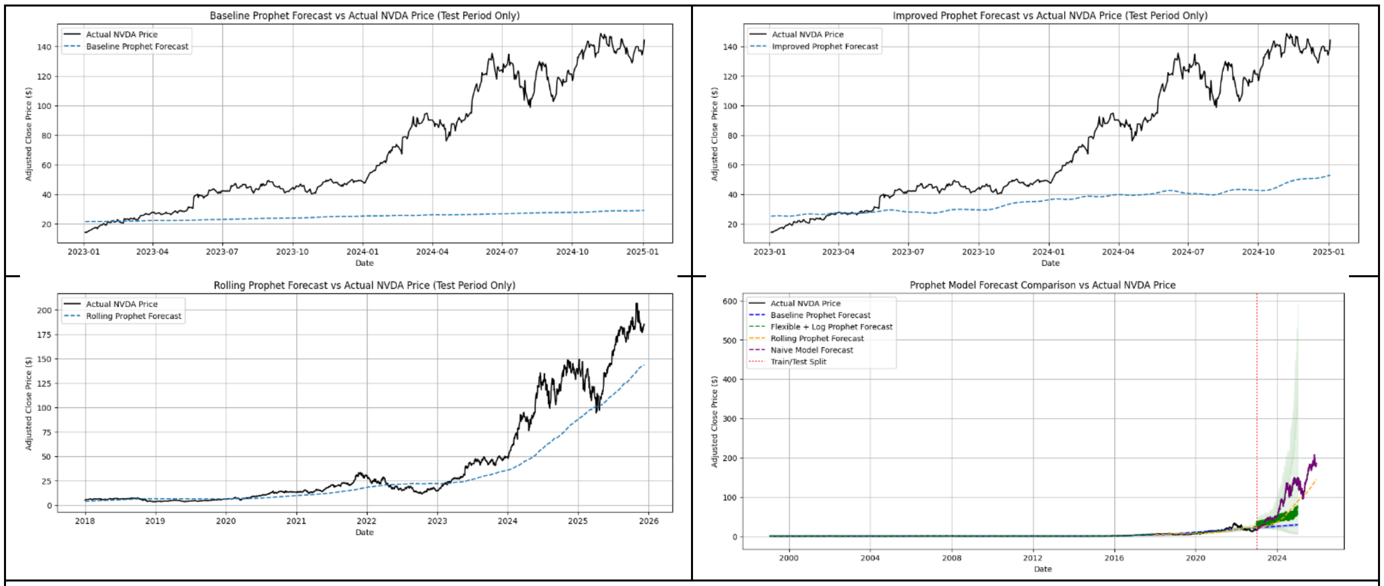

Figure 1: Baseline, Log transformed and Rolling Prophet prediction of the test period.

2.3 Interpretation

The Baseline Prophet model achieves an R^2 score of -1.221, indicating that it underperforms even a naïve mean predictor. This highlights Prophet's limited ability to extrapolate rapid changes in financial time series. Introducing log transforms and more flexible trends, improves the model's performance, increasing the R^2 score to -0.491. This shows that Prophet while can better adapt to changes with these modifications, it still falls short. Next, by implementing a rolling forecast, Prophet's predictive ability significantly improves, reaching a R^2 score of 0.797, demonstrating the benefit of updating the model iteratively. However, it still cannot surpass the naïve model with achieves a score of 0.996, emphasizing the surprising strong predictive power of simple last-value persistence for stock price forecasting.

3 Conclusion

In this project, we compared several forecasting approaches for NVIDIA stock prices, including baseline Prophet, log transforms, flexible trends, rolling forecasts, a Naïve model, and a few others. The results show that the baseline Prophet struggles to capture rapid market changes, with low R^2 scores, while log transforms and flexible trends offered modest improvement. Rolling forecasts significantly enhanced the performance by updating the model iteratively, though it was computationally intensive. Interestingly, the simple Naïve model outperformed all Prophet variances, highlighting the difficulty of predicting highly volatile stock prices. Key challenges in our approach include the time and computational intensity of the rolling method, as despite it being the best performing, the time constraints restrict further testing. Overall, these findings demonstrate that while advanced models can capture trends and seasonality, simpler methods may still provide strong predictive value, and careful consideration of model limitation and assumptions is essential in financial forecasting.