Matthew Joel

SPY Time Series Forecasting Project

Introduction and research question.

The goal of this project is to forecast the daily adjusted price of SPY (the SPDR S&P 500 ETF). I'm not treating this like a trading strategy, but rather I'm using it as a realistic time series to practice modeling, diagnostics, and fair model comparison. My research question is: if I train models on historical daily SPY data, how well can they forecast future prices, and what uncertainty range (prediction intervals) should I attach to those forecasts?

Data and preparation.

I built a single dataset that merges daily SPY market data with a handful of macro/risk indicators from FRED: the effective federal funds rate (DFF), 10-year Treasury yield (DGS10), 2-year Treasury yield (DGS2), 10-year breakeven inflation rate (T10YIE), and the VIX index level (VIXCLS). I also engineered a few simple regime features: the yield curve slope (10y minus 2y), daily changes in some macro series (deltas), and a 20-day rolling volatility of SPY log returns. The final dataset has 4012 daily observations and 19 variables, starting in 2010 and running through the most recent date in the pull.

Train/test split and evaluation metrics.

To keep the comparison clean, I held out the last 252 trading days as a test set (roughly one market year) and trained on everything before that. I evaluated each model on the same holdout window using RMSE, MAE, and MAPE, computed in the same way for every model. In the saved output, the final test date has a missing actual value (the pull included a day where adjusted price wasn't complete yet), so the metric calculations effectively ignore that one point.

EDA.

EDA made the main issue obvious: SPY prices in levels have a strong long-run upward trend, so the series is not stationary in levels. Even inside the training window, adjusted SPY ranges from about 77.6 to about 600.5, with a median around 212.4. That trend is exactly why differencing is usually needed for ARIMA, and it's also why simple follow the trend smoothly methods can perform surprisingly well. I also looked at log returns mainly as a volatility check; returns show calm periods and stress spikes, which is why I included VIX and rolling volatility as candidate predictors for the machine learning model, even though I expected price lags to dominate for next-day forecasting.

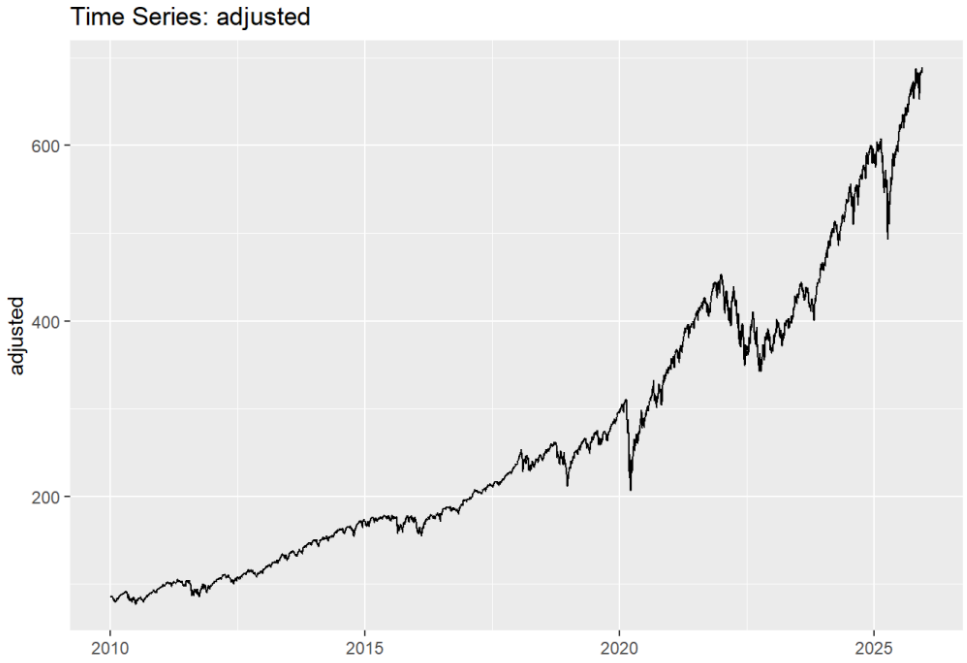
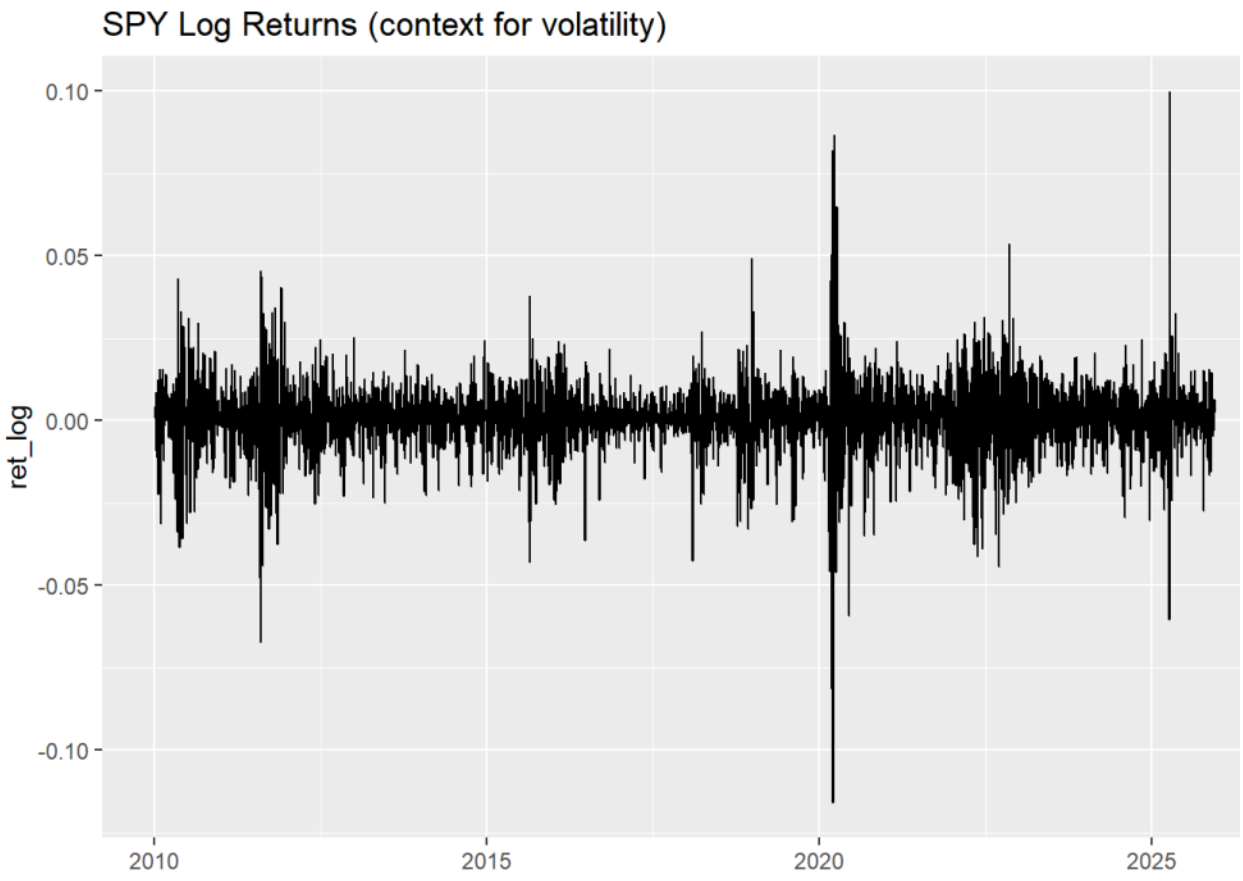


Figure 1 shows the adjusted SPY price level over time. The big takeaway is the obvious long-run upward trend, meaning the series is not stable around a constant mean. That's why a level-based model needs either smoothing (Holt) or differencing (ARIMA) to behave well. Figure 2 (below) shows log returns, which look much more 'centered' but with bursts of volatility (calm stretches followed by shocky periods). That's the reason I kept volatility-style predictors (VIX and rolling volatility) in the feature set for LightGBM, even though I expected price and price-lag information to dominate next-day forecasting.



Model 1 (midterm non-ARIMA baseline): Holt's method.

I started with Holt's exponential smoothing as the non-ARIMA before ARIMA baseline. I fit both linear-trend Holt and damped-trend Holt, then compared their holdout performance. The linear Holt fit ended up with alpha = 0.9423 and beta = 0.0001, which means the model updates the level quickly but keeps the trend very stable. On the test set, linear Holt was the strongest overall performer: RMSE ≈ 37.62, MAE ≈ 30.26, and MAPE ≈ 5.02% (and it beat the damped version). The forecast table also shows how the uncertainty band expands over time: on the first test day (2024-12-11), Holt predicted about 595.82 with a 95% interval about [590.18, 601.47]; near the end, it was predicting around 630 with a much wider interval.

Holt's linear trend method (ETS-style)

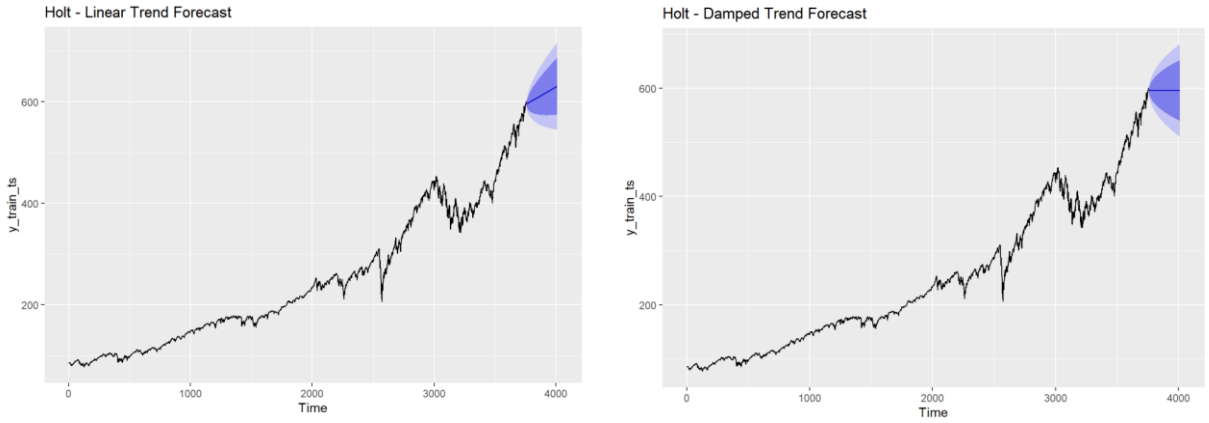
Level and trend updates:

$$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1})$$

$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1}$$

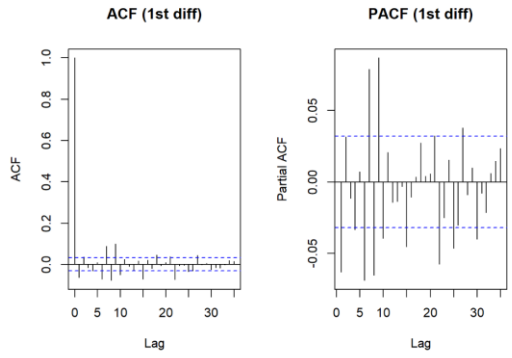
h-step forecast:

$$\hat{y}_{t+h|t} = \ell_t + hb_t$$



Model 2 (midterm): ARIMA / SARIMA.

For ARIMA, I first checked whether differencing was needed for stationarity. In levels, the ADF test had p ≈ 0.9656 and the KPSS trend-stationarity test had p ≈ 0.01, which both support not stationary in levels. After first differencing, the stationarity story improved (ADF p ≈ 0.01 and KPSS p ≈ 0.1), which supports using d = 1.



I also checked for seasonality using seasonal differencing tests and got 0 for seasonal differencing, so I stayed with a non-seasonal ARIMA instead of a SARIMA. Using auto.arima, the selected model was ARIMA(0,1,2) with drift.

ARIMA(0,1,2) with drift written out:

(compact form):

$$(1 - B)y_t = c + (1 + \theta_1 B + \theta_2 B^2)\varepsilon_t$$

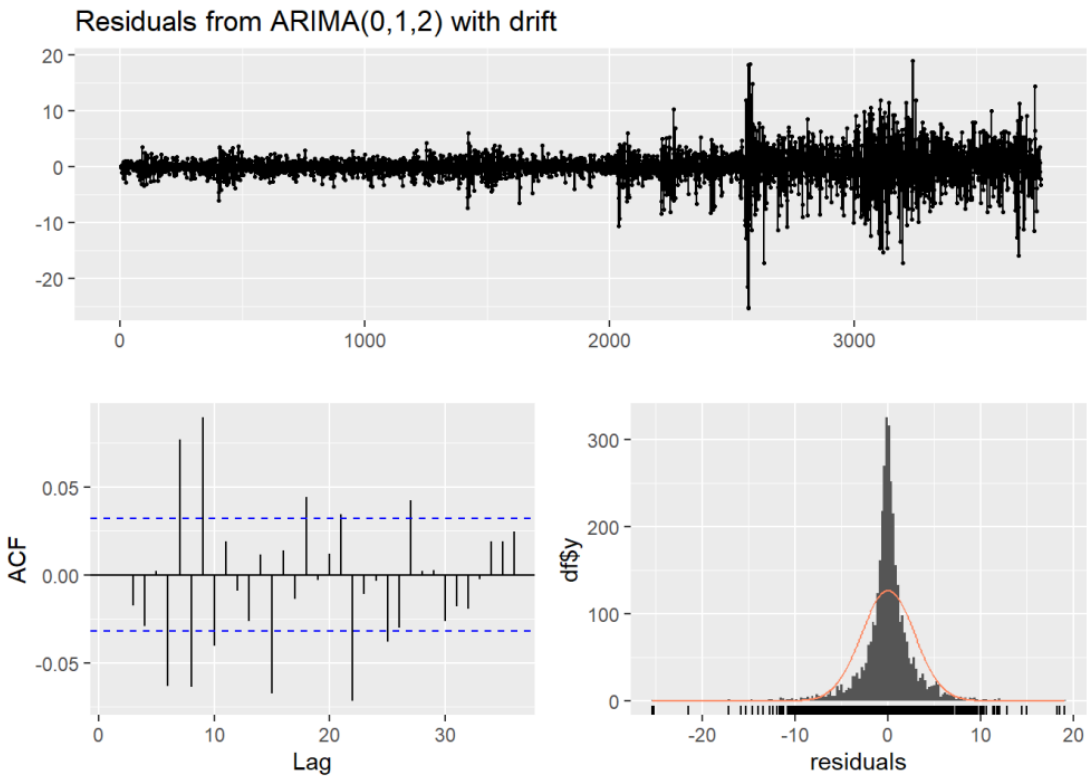
Expanded:

$$y_t = y_{t-1} + c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2}$$

From our output:

$$\hat{c} = 0.1357, \hat{\theta}_1 = -0.0600, \hat{\theta}_2 = 0.0353$$

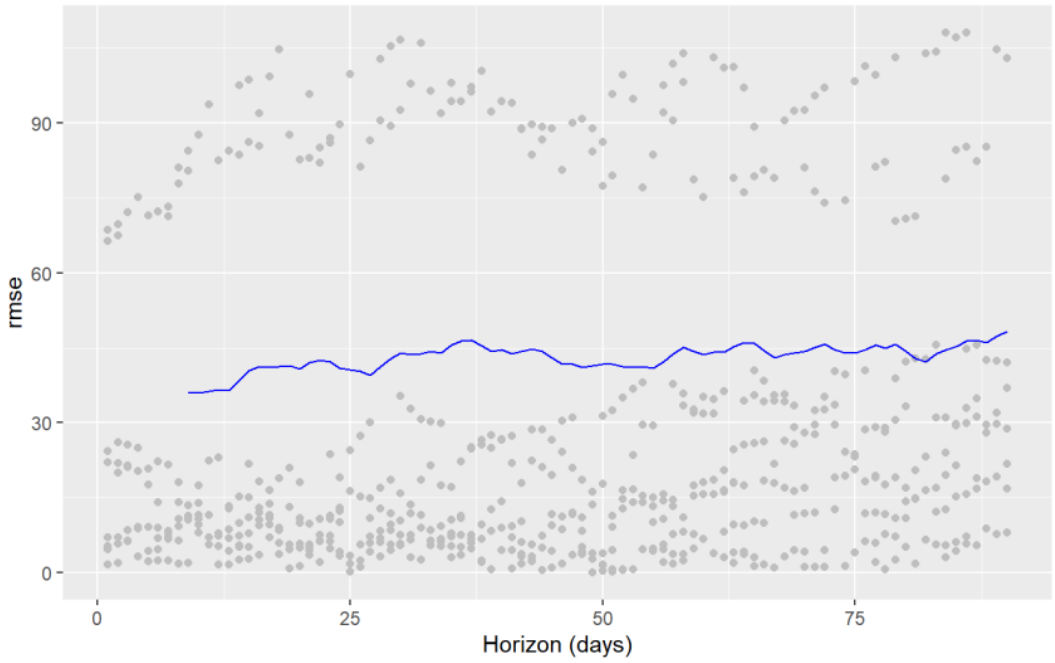
The estimated coefficients were significant (drift ≈ 0.1357; MA terms around -0.0600 and 0.0353 with small p-values). I checked residual diagnostics because ARIMA assumes that once the model captures the autocorrelation, residuals shouldn't show strong remaining structure. In my output, Ljung–Box tests were extremely significant ($p < 2.2e\text{-}16$), which tells me the residuals still have autocorrelation and the ARIMA isn't a perfect "everything left is white noise" fit. Even with that limitation, ARIMA performed almost the same as Holt on the holdout year (RMSE ≈ 37.76, MAPE ≈ 5.04%).



Model 3 (final non-ARIMA): Prophet.

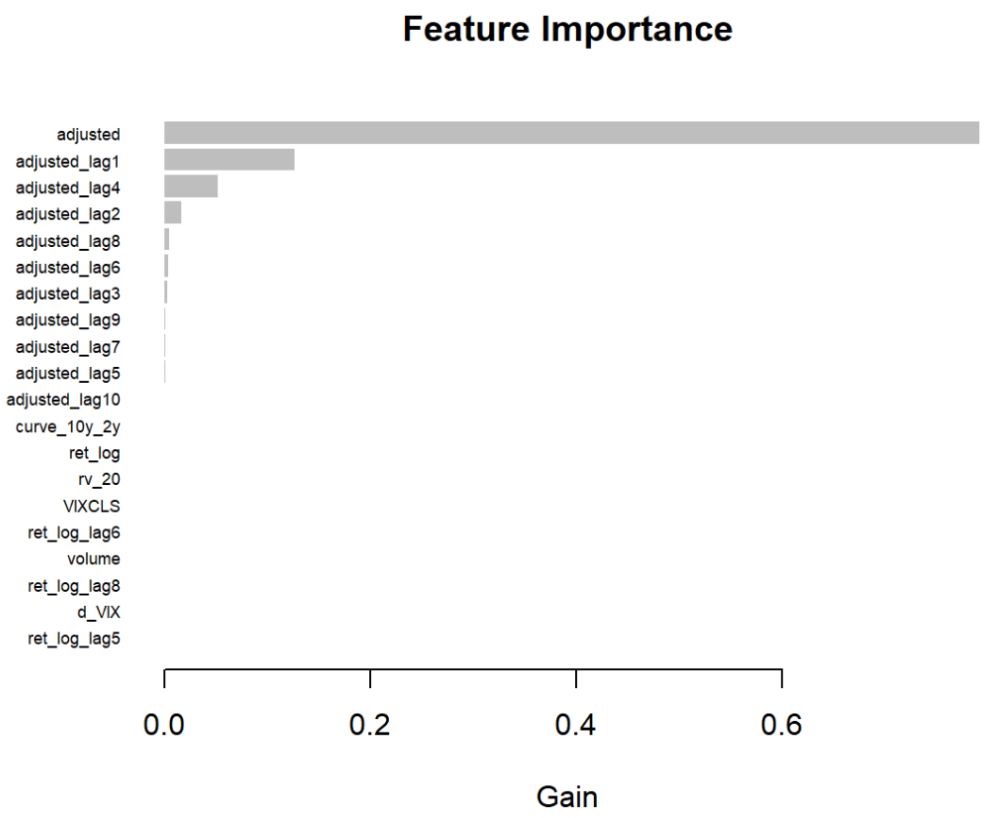
I added Prophet because EDA (and ARIMA limitations) suggested a strong trend and possible changing behavior over time, and Prophet is built to flexibly fit trends/changepoints (plus seasonality if it exists). I fit Prophet with weekly and yearly seasonality enabled and forecasted the same test window. In this SPY price-level setup, Prophet performed the worst of the four models: RMSE ≈ 67.74, MAE ≈ 61.38, MAPE ≈ 9.75%. I also ran Prophet's built-in time-series cross-validation using initial ≈ 2000 days, period ≈ 365 days, and horizon ≈ 90 days. The CV results showed the same general pattern you'd expect: error grows as the horizon increases and coverage (how often the actual value falls inside the interval) can drop as you go farther out. Overall, Prophet tended to underpredict the level in the 2025 test period, which is why its errors ended up large.

```
prophet::plot_cross_validation_metric(prophet_cv, metric = "rmse")
```



Model 4 (final machine learning): LightGBM regression.

My machine learning model was LightGBM, framed as predicting tomorrow's adjusted price from today's predictors. I built a feature set with current adjusted price, volume, return/volatility features, macro variables, and lags of both price and returns. I tuned it using a simple expanding-window time-series cross-validation approach with horizon = 60 and cutoffs inside the training set. In my mini-grid, the best-performing setting was num_leaves = 63, learning_rate = 0.1, min_data_in_leaf = 20, with mean CV RMSE ≈ 13.05. On the held-out test period, LightGBM landed in the middle: RMSE ≈ 48.74, MAE ≈ 35.72, MAPE ≈ 5.51%. Feature importance made the story pretty clear: the current adjusted price and its lags dominated the model, while macro features contributed only a small amount of gain.



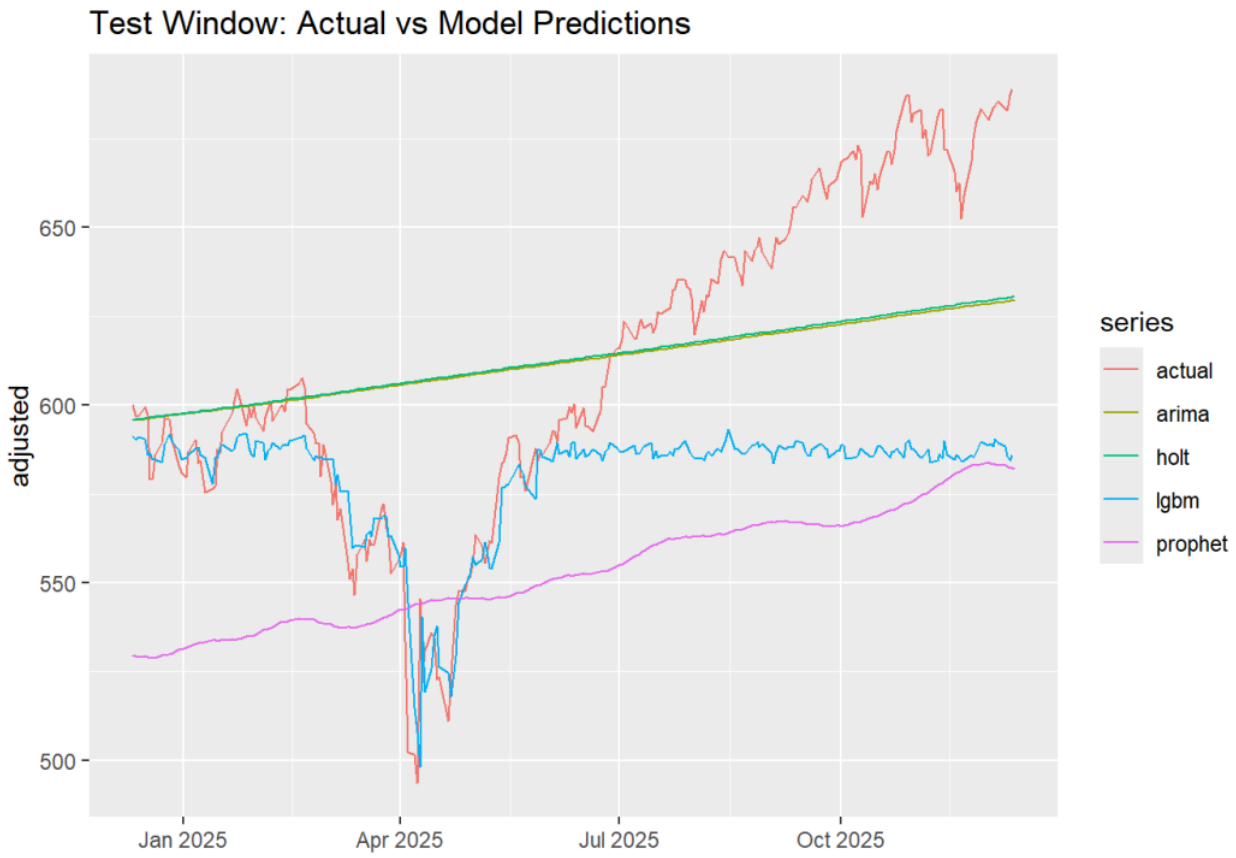
Model comparison (main result).

All four models were evaluated on the same holdout year, and the ranking was clear: Holt and ARIMA were best and basically tied (RMSE ≈ 37.6–37.8), LightGBM was next (RMSE ≈ 48.7), and Prophet was last (RMSE ≈ 67.7). Looking at the end of the test window shows why the errors happened: actual SPY was around 685–689, Holt and ARIMA were around 629–630, Prophet was around 582–583, and LightGBM was around 584–590. In other words, everyone underpredicted the strength of the late-period rally, but Prophet lagged the most and Holt/ARIMA lagged the least.

Forecasting beyond the dataset (next ~20 trading days).

To make the output more forward looking, I fit Holt, ARIMA, and Prophet on the full series and forecasted the next ~20 trading days. Holt's next-20 forecast was almost flat: it starts around

689.19 with a 95% interval about [682.77, 695.61], and rises to about 692.11 by step 20 with a 95% interval about [677.74, 706.47]. ARIMA's point forecast rose faster (from about 691.37 up to about 716.86 by step 20), but the uncertainty blew up quickly: by step 20 the 95% interval was roughly [589.79, 843.93], which is a good example of ARIMA intervals widen a lot at longer horizons. Prophet's next-20 weekday forecast was flatter and lower (around 669–675), with 95% bands in roughly the mid-640s to high-690s.



This figure compares the test-window predictions from all four models against the actual SPY adjusted price. Holt and ARIMA track the level most closely through the year, while Prophet sits much lower for most of the test period, and LightGBM sits in between. This picture matches the error metrics table, where Holt and ARIMA are essentially tied for best performance, LightGBM is third, and Prophet is last.

Final conclusion.

My direct answer is that, for forecasting the daily SPY adjusted price level in this project, Holt and ARIMA were the most reliable models on the one-year holdout test window (both around RMSE ≈ 38 and MAPE ≈ 5%). If I had to pick one default model for this exact target and horizon, I'd pick Holt because it slightly beat ARIMA on RMSE and it's very easy to explain and communicate (smoothed level + trend). The main limitation is that forecasting raw price levels for a major ETF is inherently tough, prices behave a lot like today plus noise, and big regime moves can leave every model behind. If I wanted to push this further, I'd try forecasting returns/volatility (often more stable), tune ML models more aggressively with broader time-series CV, and/or use stronger exogenous/regime signals instead of expecting simple macro levels to add much incremental predictive power.

Final project questions (answering 4 of 5).

(1a) With respect to an ARIMA model: Why is maximum likelihood estimation (MLE) desirable?

I like MLE because it gives me one clean, consistent way to estimate all the ARIMA parameters under the probability story I'm assuming for the errors. In plain language, it tries to find the parameter values that make the observed data most likely under the model. That matters because it usually produces stable estimates, it comes with standard errors (so I can check if terms are actually meaningful instead of just noise), and it connects naturally to AIC/BIC for comparing models. So MLE is just a practical way to fit the model and quantify uncertainty in the coefficients.

(1b) Why do prediction intervals get wider farther into the future?

Prediction intervals widen because uncertainty stacks up as the horizon grows. One step ahead,

the big unknown is basically just tomorrow's random shock. But 10 or 20 steps ahead, there are many future shocks we haven't seen yet, and they can push the series in lots of different directions. Also, if the model's drift (or trend) is a little off, that small mismatch has more time to pull the forecast away from reality. So the farther out I go, the more honest the model has to be about the many possible futures, and that shows up as wider bands.

(2) Are Prophet and machine learning "black boxes"?

They can feel like black boxes, but they're not just random guessing. Prophet is fairly interpretable because it's built as trend + seasonality + noise. Even though it automatically handles things like changepoints, I can still look at what trend it learned and whether the seasonal patterns make any sense. LightGBM (my machine learning model) is more complicated than ARIMA, but it's still a defined method: it builds lots of small decision trees and combines them. And I can peek inside using feature importance (which predictors mattered most) and more advanced tools like SHAP values if I want a deeper explanation. So I'd call them less transparent than ARIMA, but not unexplainable.

(3) Why is cross-validation important for time series, and what's different about it?

Cross-validation matters because it's the best quick reality check on how well the model predicts data it didn't train on. The big difference in time series is that I can't shuffle the data into random folds (that would leak future information and make the model look better than it really is). Instead I have to use rolling or expanding windows, where the training chunk always comes before the validation chunk. That mirrors real forecasting: I learn from the past and predict the future. It's a simple idea, but it ensures we don't accidentally cheat in evaluation.

(5) What would I need to make the conclusions stronger in a real job?

First, I'd ask what stronger means for the person making the decision. Do they care about next-

day accuracy, next-month accuracy, or about capturing big downside risk? Do they want tight intervals, or do they just want the direction of the trend? Do they need a model they can explain to stakeholders? Once that's clear, if they gave me resources to improve accuracy, I'd do more careful tuning (bigger hyperparameter searches plus more time-series CV), and I'd seriously consider changing the target to something more forecastable (like returns or volatility rather than raw price level). If they cared about generalizability, I'd test across multiple time periods and stress regimes (not just one holdout year), and I'd try stronger exogenous/regime information (or models explicitly designed to use it) instead of expecting today's price lags to do all the work.

References

- Dancho, M., & Vaughan, D. (2025). *tidyquant: Tidy Quantitative Financial Analysis* (R package version 1.0.11) [Computer software]. Comprehensive R Archive Network (CRAN). CRAN
- Federal Reserve Bank of St. Louis. (n.d.). *FRED, Federal Reserve Economic Data*. Retrieved December 15, 2025, from https://fred.stlouisfed.org/ FRED
- Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software, 27*(3), 1–22. https://doi.org/10.18637/jss.v027.i03 Journal of Statistical Software
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (NeurIPS 2017). https://proceedings.neurips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf NeurIPS Proceedings

- Taylor, S. J., & Letham, B. (2017). *Forecasting at scale* (technical report). https://facebook.github.io/prophet/static/prophet_paper_20170113.pdf Facebook
- SPDR S&P 500 ETF Trust. (2018). *Prospectus* (Prospectus dated January 18, 2018). U.S. Securities and Exchange Commission. https://www.sec.gov/Archives/edgar/data/884394/000119312518014611/d469252d497.htm sec.gov