Advanced Programming 2025

Machine Learning Model Comparison for Financial Asset Return and Volatility Forecasting

Final Project Report

Diona Avdija
diona.avdija@unil.ch
Student ID: 22435333

January 4, 2026

Abstract

This project evaluates the predictability of equity returns and volatility using a reproducible forecasting pipeline. We focus on a high-volatility equity (Tesla, TSLA) as the empirical setting. We compare a random-walk benchmark with AR(1), auto-selected ARIMA, regularized linear models, and XGBoost across horizons from 10 days to 1 year. Model selection is performed strictly on a validation window to avoid leakage, then refit on train+validation and tested on out-of-sample price-level errors. Return forecasting remains difficult: most models fail to consistently beat the random walk once realistic time-series validation is enforced. By contrast, volatility forecasts are consistently more reliable. A GJR-GARCH(1,1) model with Student-t innovations outperforms EWMA across all horizons and provides reliable uncertainty bands for Monte Carlo price simulations. Overall, the results highlight the limits of return predictability and the practical value of volatility modeling for risk assessment.

Keywords: data science, Python, machine learning, time-series forecasting, equity returns, volatility modeling, ARIMA, GARCH, XGBoost, Monte Carlo simulation

Contents

1 Introduction

Forecasting asset prices and their volatility is a central problem in financial modeling and risk management. Accurate volatility forecasts are critical for derivative pricing, portfolio risk control, and Value at Risk (VaR) estimation, while reliable return forecasts would offer direct value for investment decisions. Despite extensive academic and practical effort, predicting equity returns remains notoriously difficult, largely due to the efficient market hypothesis, which asserts that asset prices incorporate available information and therefore follow a near-random walk (Fama 1970). In contrast, volatility often exhibits persistence and clustering, suggesting that second-moment dynamics may be forecastable even when mean returns are not.

This project assesses the relative performance of traditional econometric models and modern machine learning methods for forecasting equity returns and volatility across multiple horizons. Using Tesla (TSLA) stock data, we evaluate whether increased model flexibility translates into improved predictive accuracy, and we build a reproducible end-to-end pipeline that integrates model estimation, validation-based selection, and Monte Carlo simulation for uncertainty quantification. The evaluation explicitly follows instructor guidance: model selection is performed on a validation set, and regularized linear baselines are included to mitigate overfitting in noisy return data.

Tesla is an informative test case because it combines high volatility, sharp regime shifts, and heavy retail attention. These features amplify the difficulty of return prediction while providing a clear signal for volatility dynamics. Framing the project around a single asset allows for a focused, reproducible experiment that highlights what is and is not forecastable in practice.

The report is organized as follows. Section 2 states the research question and reviews related work. Section 3 describes the methodology and algorithmic framework. Section 4 discusses the implementation, and Section 5 explains how the codebase is maintained and updated. Section 6 presents the empirical results, and Section 8 concludes.

2 Research Question and Related Literature

Research question. Can standard econometric models and modern machine learning methods deliver return forecasts that beat a random walk benchmark, and can volatility models provide reliable risk estimates across multiple horizons for a high-volatility equity? We study this question using TSLA as a demanding empirical setting and evaluate all models under a strict time-series validation protocol.

The literature draws a clear distinction between return predictability and volatility predictability. Under market efficiency, excess returns are difficult to forecast, and the random walk with drift is a hard-to-beat benchmark (Fama 1970). Empirical studies show that linear predictability in daily returns is weak and unstable across time and assets (Campbell, Lo, and MacKinlay 1997; Hansen and Lunde 2005). Large-scale out-of-sample comparisons also find that many popular predictors fail to beat the historical mean when evaluated in real time, reinforcing the robustness of simple baselines (Welch and Goyal 2008).

Volatility, by contrast, exhibits strong persistence and clustering. The ARCH framework (Engle 1982) and its generalization to GARCH (Bollerslev 1986) model conditional variance directly and have become standard tools for forecasting risk. Asymmetric extensions such as GJR-GARCH (Glosten, Jagannathan, and Runkle 1993) capture leverage effects, where negative returns increase volatility more than positive returns of the same magnitude, and have demonstrated improved performance in equity data (Hansen and Lunde 2005). Survey evidence consistently ranks GARCH-type models among the most reliable practical tools for volatility prediction (Poon and Granger 2003), and realized-volatility frameworks further highlight the persistence of second-moment dynamics in equities (Andersen et al. 2003).

Machine learning methods capture nonlinearities in structured data. Gradient-boosted trees such as XGBoost (Chen and Guestrin 2016) are a common example. However, out-of-sample performance in finance is mixed, with many reported gains disappearing under strict time-series validation. Recent empirical studies on financial prediction emphasize that ML models can add value in some settings but are sensitive to sample size, nonstationarity, and validation design (Gu, Kelly, and Xiu 2020; Krauss, Do, and Huck 2017). This motivates careful evaluation protocols and strong benchmarks when comparing ML approaches to econometric baselines.

3 Methodology and Algorithmic Framework

3.1 Data Description

Source and collection. We use Tesla, Inc. (TSLA) as a representative high-volatility equity. Historical market data are collected from Yahoo Finance via `yfinance` and stored locally as a cached CSV snapshot (`data/raw/yfinance_cache_TSLA.csv`). The initial prototype supported arbitrary tickers via live API calls, but the final pipeline uses a fixed local cache to ensure reproducibility and avoid API rate-limit interruptions.

Sample size and characteristics. The cached dataset contains 1,256 daily observations spanning 2020-12-14 to 2025-12-12, with standard OHLCV fields. The EDA summary reports zero missing values and zero zero-volume days in the cached sample.

Target variables. Let P_t denote the adjusted close price at date t. Daily log-returns are computed as

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right). \tag{1}$$

Return forecasting targets r_t at each horizon, computed on resampled price series (monthly: last price in the month with month-start index). Volatility forecasting is modeled separately: the GARCH component targets conditional variance of daily returns, and evaluation uses realized variance/volatility derived from daily returns.

Summary statistics. Table 1 summarizes key properties from the cached sample (computed on daily log-returns).

Table 1: TSLA dataset summary (cached sample).

Item	Value
Date range	2020-12-14 to 2025-12-12
Trading days	1,256
Current price	452.41 USD
Mean daily return	0.1328%
Std. dev. daily return	3.8315%
Annualized volatility	60.82%
Min / Max daily return	-15.43% / 22.69%
Missing values	0

The EDA also reports skewness and kurtosis and runs normality tests (Jarque-Bera and Shapiro-Wilk when applicable). Both tests reject normality (p-values < 0.001), indicating heavy tails;

this motivates Student-t innovations in the GARCH model and cautions against strict normality assumptions in return forecasting.

3.2 Approach

3.2.1 Forecast horizons and resampling

We evaluate five horizons: 10 trading days, 1 month, 3 months, 6 months, and 1 year. Monthly horizons are interpreted in trading time using 21 business days per month, consistent with the implementation.

Table 2: Forecast horizons and trading-day convention.

Horizon	Unit	Steps	Trading days per step
10 days	daily	10	1
1 month	monthly	1	21
3 months	monthly	3	21
6 months	monthly	6	21
1 year	monthly	12	21

3.2.2 Feature construction

All predictors are computed causally (using information available at time t):

- Lagged returns $\{r_{t-1}, \dots, r_{t-5}\}$ for ML and regularized regressions.
- Normalized volume (z-score): proxy for liquidity and trading intensity.
- Momentum: rolling mean of simple returns over 5 and 10 days, capturing short-term trend effects.
- Historical volatility: 21-day rolling standard deviation of daily log-returns, annualized and aligned to the forecast index, capturing recent risk regime and volatility clustering (see Appendix A).

These variables are used as optional exogenous regressors for AR(1), ARIMA, and the ML baselines. Feature design is intentionally simple and transparent: the goal is to test whether widely used technical signals (volume, momentum, recent volatility) add incremental forecasting power without introducing look-ahead bias or complex data dependencies. All rolling statistics are computed with past information only and aligned to the forecast index to maintain causal integrity.

3.2.3 Return forecasting models

Random walk with drift. The benchmark assumes constant expected return equal to the in-sample mean:

$$\hat{r}_{t+h|t} = \bar{r}_{\text{train}} \quad \forall h = 1, \dots, H. \tag{2}$$

AR(1). Autoregressive dynamics are modeled as

$$r_t = \mu + \phi r_{t-1} + \varepsilon_t. \tag{3}$$

For daily horizons this is implemented as ARIMA$(1, 0, 0)$, with optional exogenous regressors.

ARIMA (auto-selected). For non-daily horizons (monthly in this project), model order (p, d, q) is selected by `auto_arima` using information criteria (non-seasonal).

Regularized linear models. Ridge, Lasso, and Elastic Net regressions are included to counteract low signal-to-noise ratios in returns. Models are trained on lagged returns and exogenous features with time-series cross-validation to select regularization strength.

XGBoost. Gradient-boosted trees are trained with a small deterministic grid and time-series CV when enough data are available. Forecasting is iterative: one-step predictions are fed back as lagged inputs for multi-step horizons.

3.2.4 Volatility forecasting

Volatility is modeled separately from returns. We fit a GJR-GARCH$(1, 1)$ model with Student-t innovations when possible, falling back to a symmetric GARCH$(1, 1)$ with Gaussian errors if optimization fails. The Student-t choice is motivated by the heavy tails observed in the return distribution.

$$\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \gamma\mathbb{I}_{\{\varepsilon_{t-1}<0\}}\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2. \tag{4}$$

To keep units consistent, GARCH is always fit on daily returns, and daily variance forecasts are aggregated to the target horizon:

$$\widehat{\mathrm{Var}}\left(\sum_{j=1}^{m} r_{t+j}\right) \approx \sum_{j=1}^{m} \hat{v}_{t+j|t}, \qquad \hat{\sigma}_{\text{step}} = \sqrt{\sum_{j=1}^{m} \hat{v}_{t+j|t}}. \tag{5}$$

An EWMA benchmark is included with $\lambda = 0.94$. This is the classic RiskMetrics decay factor and provides a strong, industry-standard baseline for volatility forecasting.

$$\sigma_t^2 = \lambda\sigma_{t-1}^2 + (1 - \lambda)r_{t-1}^2. \tag{6}$$

Volatility backtests compare forecasts to forward realized variance computed over a rolling window aligned to the horizon, and we evaluate the most recent year of usable points for stability and comparability across horizons.

3.2.5 Model selection and evaluation

We use a strict chronological split tailored to the horizon H: the final H observations form the test set, the preceding H observations form the validation set, and the remaining initial segment is used for training. Model selection is based on validation RMSE, and final metrics are computed on the test set after refitting on train+validation to avoid test leakage.

Selection flow: validation $\rightarrow$ refit on train+validation $\rightarrow$ final test evaluation

Forecasts are evaluated on price levels by compounding predicted log-returns:

$$\hat{P}_{t+h} = P_t \exp\left(\sum_{j=1}^{h} \hat{r}_{t+j|t}\right). \tag{7}$$

We report RMSE and MAE on prices because investment decisions and baseline comparisons are naturally framed in price levels rather than raw returns. MAPE is provided in Appendix A for interpretability. For AR(1) and XGBoost, results with MAPE at or above 100% are dropped to avoid extreme failures dominating selection. Volatility backtests additionally report the QLIKE loss on variance forecasts, computed as

$$\mathrm{QLIKE}(v, \hat{v}) = \log(\hat{v}) + \frac{v}{\hat{v}}, \tag{8}$$

which matches the implementation (constant terms are omitted since they do not affect model ranking).

3.2.6 Uncertainty quantification

Monte Carlo simulation combines point forecasts with volatility estimates. For each scenario s:

$$\tilde{r}^{(s)}_{t+h} = \hat{r}_{t+h|t} + \hat{\sigma}_{t+h|t} z^{(s)}_h, \quad z^{(s)}_h \sim \mathcal{N}(0,1), \tag{9}$$

$$\tilde{P}^{(s)}_{t+h} = P_t \exp\left(\sum_{j=1}^{h} \tilde{r}^{(s)}_{t+j} \right). \tag{10}$$

For long horizons, the point forecast for AR(1) and ARIMA is replaced by the historical drift to avoid unrealistic mean-reversion, and a mild 1.25 volatility scaling is applied to the Monte Carlo bands to avoid under-dispersed uncertainty.

4 Implementation

Languages and libraries. The project is implemented in Python 3.10.0 using `pandas`/`numpy` (data handling), `statsmodels` and `pmdarima` (AR/ARIMA), `scikit-learn` (regularized models and time-series CV), `xgboost` (ML model), and `arch` (GARCH volatility).

System architecture. The pipeline is modular and CLI-driven. `main.py` orchestrates EDA, backtests, forecasting, plotting, and result export. Core modules include:

- `data_loader.py`: cached data loading, resampling, and feature construction.
- `models.py`: AR(1), ARIMA helpers, regularized regressions, XGBoost training and forecasting.
- `backtests.py`: train/validation/test backtest logic and metric computation.
- `volatility.py`: GARCH and EWMA volatility backtests.
- `evaluation.py`: end-to-end forecasting and Monte Carlo simulation.
- `plots.py`, `results.py`: visualization and CSV artifact export.

Reproducibility controls. Determinism is enforced through fixed seeds, single-thread execution (OMP/BLAS environment variables are set to 1), and cached data. Running `python main.py` triggers the deterministic pipeline (EDA + all horizons) using the fixed TSLA snapshot in `data/raw/`.

```python
# main.py (top-level setup)
os.environ["PYTHONHASHSEED"] = "0"
os.environ["OMP_NUM_THREADS"] = "1"
os.environ["OPENBLAS_NUM_THREADS"] = "1"
os.environ["MKL_NUM_THREADS"] = "1"
os.environ["NUMEXPR_NUM_THREADS"] = "1"
os.environ["VECLIB_MAXIMUM_THREADS"] = "1"

np.random.seed(42)
args.cache_only = True
```

Listing 1: Reproducibility controls (excerpt).

5 Maintenance and Updating the Codebase

The repository is versioned with git, enabling traceable changes to data, code, and report text. Dependencies are pinned in `environment.yml` (conda) and `requirements.txt` (pip), and the README provides the canonical setup commands. To refresh the dataset, replace the cached CSV in `data/raw/yfinance_cache_TSLA.csv` with a new snapshot and rerun `python main.py`. All outputs are written to `results/` with timestamps to preserve run history. The project does not include a formal unit-test suite; instead, the end-to-end pipeline serves as a reproducible regression check. Future maintenance could add unit tests around data loading, horizon settings, and metric computations.

6 Results

6.1 Experimental Setup

The evaluation uses the cached TSLA dataset described in Section 3.1. Models are evaluated on price-level errors (USD) using RMSE and MAE; MAPE is reported in Appendix A. Monte Carlo uses 500 scenarios per horizon. All artifacts (plots and CSVs) are written under `results/`.

- Fixed horizon grid: 10 days, 1 month, 3 months, 6 months, 1 year.
- Monte Carlo scenarios: 500 per horizon.

6.2 Performance Evaluation

Tables 3 and 4 report the validation selection step and the final test evaluation.

Return forecasting results. Return forecasting is the most challenging part of the pipeline. The signal-to-noise ratio in daily equity returns is low, and small changes in training windows or horizons can alter model rankings. The validation-based protocol therefore plays a central role: it prevents selecting models that only look good in hindsight. In the code, model selection is based on the smallest validation RMSE (price-level errors), then the chosen model is refit on train+validation and evaluated once on the test set.

Table 3: Validation model selection by lowest RMSE.

Horizon	Lowest RMSE model	Val RMSE
10 days	ElasticNet	22.35
1 month	Random walk	6.73
3 months	ARIMA	28.56
6 months	ARIMA	59.25
1 year	ARIMA	49.70

For each horizon, we run all available models (random walk baseline, regularized linear models, AR/ARIMA, and XGBoost when available) and select the one with the lowest validation RMSE. Table 3 summarizes these selections: ElasticNet is chosen at 10 days, the random walk at 1 month, and ARIMA from 3 months onward. This pattern suggests that ARIMA is comparatively strong at longer horizons within the set of models that can be reliably estimated.

Table 4: Test set after refit: validation-selected model vs. random walk baseline (RMSE).

Horizon	Model	Model RMSE	RW RMSE	Beats RW?
10 days	ElasticNet	19.63	17.54	No
1 month	Random walk	31.09	31.09	Baseline
3 months	ARIMA	110.48	103.18	No
6 months	ARIMA	72.15	66.43	No
1 year	ARIMA	66.37	62.44	No

Although ARIMA appears strong in Table 3, Table 4 shows no horizon where the validation-selected model beats the random walk on test (the 1-month case is a tie). In fact, ARIMA's refit RMSE is higher than the random walk at 3, 6, and 12 months. This reinforces the idea that, once we enforce a realistic time-series protocol, returns are difficult to forecast with stable gains over simple benchmarks, consistent with the empirical literature and the instructor's guidance. Second, the diagnostic table in Appendix A shows that the random walk is not always the lowest-RMSE model on test: ARIMA is best at 1 month and XGBoost at 3 months (Table A.1). These results are not used for selection, but they suggest ML can add value at intermediate horizons. Because these gains are not reliably identified by validation, we keep them diagnostic to avoid test leakage.
Overall, the evidence points to the same conclusion: return forecasts should be treated as weak signals, while uncertainty, driven by volatility, carries most of the actionable information. Practically, return models are best used as inputs to scenario analysis rather than as standalone point-forecast engines.

Volatility forecasting results. Volatility forecasts are produced with daily GJR-GARCH dynamics and aggregated to each horizon for Monte Carlo uncertainty bands. Table 5 reports out-of-sample backtests for GARCH versus the EWMA baseline. We report RMSE and MAE to summarize absolute forecast errors, and QLIKE to assess variance calibration, which is the most relevant property for risk modeling. GARCH outperforms EWMA at every horizon on all three metrics, with the advantage widening at longer horizons; by 1 year, RMSE and MAE are roughly halved relative to the baseline. These results confirm that volatility is a reliably forecastable component even when mean returns are not, and they are practically important because volatility directly affects option pricing, structured product design, and risk management decisions.

Table 5: Volatility backtest (annualized, TSLA).

	RMSE		MAE		QLIKE	
Horizon	GARCH	Δ	GARCH	Δ	GARCH	Δ
10 days	21.73%	-0.72	17.63%	-0.20	6.06	-0.01
1 month	19.14%	-1.81	15.36%	-1.08	6.83	-0.03
3 months	18.29%	-3.07	15.43%	-2.47	8.08	-0.05
6 months	16.46%	-3.86	13.46%	-2.99	8.87	-0.06
1 year	9.84%	-7.67	8.59%	-6.85	9.50	-0.19

Δ = GARCH – EWMA.

Two patterns stand out. First, the performance gap widens with horizon length, which is consistent with volatility persistence: longer horizons average out short-term noise and allow mean reversion to dominate, so the asymmetric GARCH structure has more room to add value. Second, the QLIKE metric remains consistently better for GARCH, indicating improved variance calibration rather than just lower error in volatility units. This matters for risk management because it directly affects the width of Monte Carlo prediction bands and the plausibility of tail scenarios. In short, the backtests provide strong evidence that volatility is a reliably forecastable component in this dataset, even when mean returns are not.

6.3 Horizon-Specific Observations

The horizon analysis highlights different failure modes and strengths across models:

- **10 days:** The random walk baseline is hard to beat; even the best regularized model only marginally improves validation metrics and loses on the test set. Volatility accuracy is also tight here, with GARCH only slightly ahead of EWMA, reflecting short-horizon noise.
- **1 month:** ARIMA delivers the lowest test RMSE, but it is not selected in validation. This illustrates how sensitive selection is to short validation windows. Volatility gains become clearer, with GARCH improving RMSE and MAE over EWMA.
- **3 months:** XGBoost performs best on the test set, suggesting nonlinear signals can matter at intermediate horizons, yet validation does not reliably identify this gain. Volatility gaps widen further, consistent with stronger persistence at longer horizons.
- **6 months and 1 year:** Performance converges back to the random walk, and model differences shrink. This is also where data scarcity after resampling most limits ML methods. Volatility forecasting is strongest here, with GARCH delivering the largest error reductions relative to EWMA.

Overall, the horizon breakdown reinforces the core narrative: return predictability is fragile, while volatility predictability improves as the horizon increases.

6.4 Visualizations

Forecast and volatility plots are generated automatically under `results/`. Example figures are included below using copies stored alongside the report.

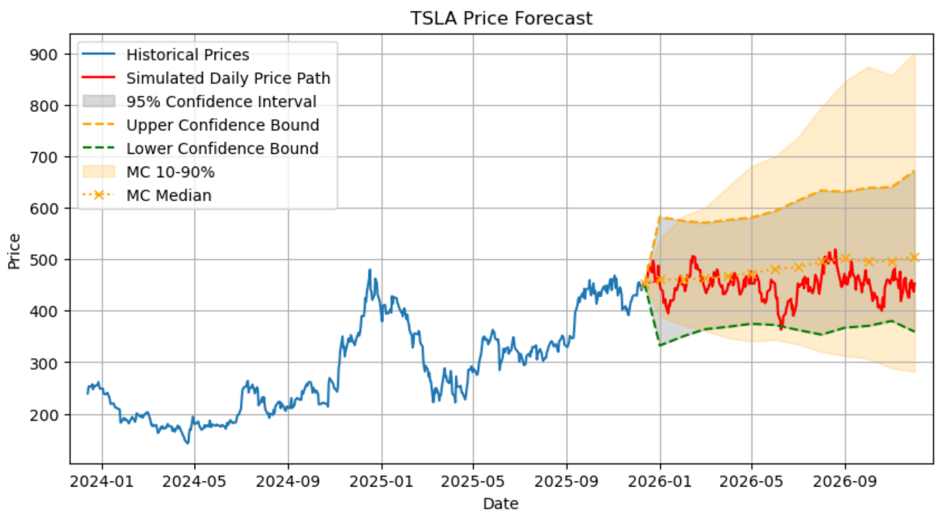

Figure 1: One-year price forecast with Monte Carlo bands (TSLA).

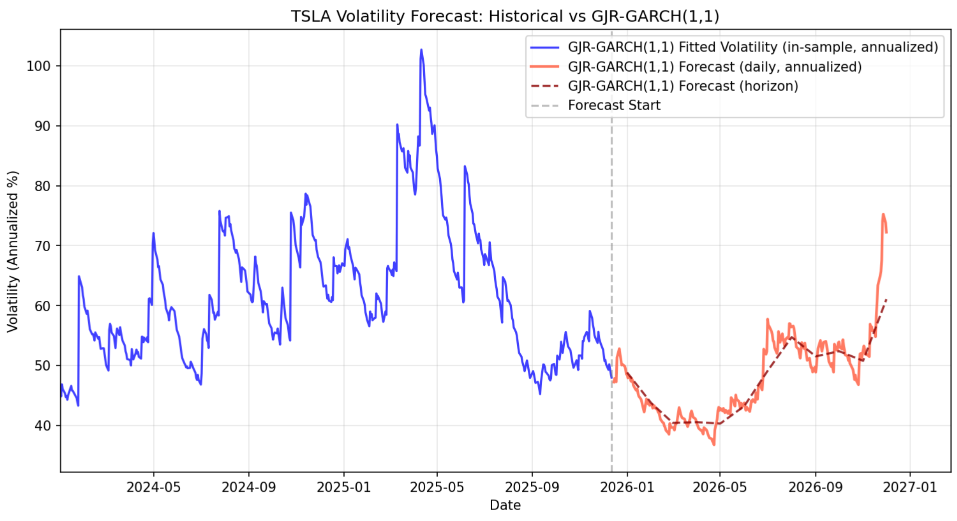

Figure 2: One-year volatility forecast (annualized, TSLA).

6.5 Case Study: One-Year Horizon

Figures 1–2 summarize the one-year case: expected returns stay close to a drift, while uncertainty expands rapidly as the volatility path evolves. This is the most demanding setting for any return model because it combines fewer training observations (due to monthly resampling) with the cumulative effect of forecast errors. In our pipeline, the expected return component is anchored to the historical drift, while uncertainty is driven by the GARCH volatility path. This design choice is deliberate: it avoids unrealistic mean-reversion in ARIMA-style forecasts and puts the emphasis on risk rather than on a fragile point estimate.

From a practical perspective, the one-year forecast should be read as a distribution rather than a single price target. The median path reflects the drift, while the 10th and 90th percentiles provide a plausible range for downside and upside scenarios. This framing is closer to how risk managers interpret long-horizon forecasts: the width of the fan conveys the degree of uncertainty, and the volatility model determines how quickly that fan expands.

This case study also illustrates why the volatility component is the strongest part of the project. Even if the mean return is hard to predict, the conditional variance exhibits persistence and provides a stable signal. As a result, the pipeline yields informative risk bands that can be used for stress testing, scenario analysis, and communication of uncertainty to end users.

7 Discussion

The results are consistent with established findings in the return-predictability literature. Short-horizon returns behave close to a random walk, and the baseline often dominates more complex models. At intermediate horizons, some models can deliver improvements on the test set, but these gains are not reliably identified by validation-based selection, underscoring the instability of predictive signals in financial data. This aligns with long-standing evidence that AR and ARIMA models rarely beat the random walk in returns forecasting (Fama 1970; Campbell, Lo, and MacKinlay 1997; Hansen and Lunde 2005). The 3-month horizon illustrates the risk: XGBoost performs well on the test set but is not selected by validation, suggesting sensitivity to noisy patterns rather than stable predictive structure.

The machine-learning results highlight a structural constraint: as horizons lengthen and data are resampled to monthly frequency, the effective training sample shrinks and recursive forecasting errors compound. In the current run, AR(1) and XGBoost drop out of the 6-month validation split and the 1-year test split because the pipeline excludes models with failed fits or extreme errors (MAPE $\geq$ 100%), which becomes more likely when the monthly resampling leaves too few points.

Even with regularization and time-series CV, tree-based models can latch onto transient patterns that do not persist. This is not a failure of the implementation; it is a realistic outcome of the data-generating process for equity returns.

To mitigate this, we considered keeping daily predictors while forecasting long-horizon returns to preserve a larger training sample. We ultimately rejected this option because it mixes time scales, complicates alignment, and would change the definition of the forecasting task, making comparisons across horizons less consistent.

Volatility modeling shows clearer structure. Even when return forecasts are weak, volatility forecasts provide coherent uncertainty estimates and help generate realistic prediction intervals for prices. This supports the practical value of modeling conditional variance separately from mean returns and is consistent with the evidence in the volatility-forecasting literature.

From an applied perspective, the pipeline is most valuable as a risk-forecasting tool rather than a return-forecasting engine. The practical output is not a single best price but a distribution of plausible paths, which is more aligned with real portfolio and risk management decisions.

Some plots show a visible gap between the end of the historical series and the start of the forecast. After inspection, this is expected: the forecast begins at the next valid period boundary (e.g., the next month-start index), so the first forecast point is not anchored to the last observed date. We keep this gap to avoid implying continuity or backfilling; forcing the lines to touch would misrepresent the timing of the forecast. An illustrative example is provided in Appendix Figure A.3.

- Return models are best interpreted as weak signals or scenario inputs, not as precise point forecasts.
- Volatility dynamics are stable enough to provide actionable risk bands, especially for medium and long horizons.
- Validation-based selection is essential; test-set selection would have favored models that did not generalize.

Limitations include the focus on a single asset (TSLA), a restricted set of engineered features, and reliance on a single train/validation/test split per horizon. Extending the study to multiple assets, adding macro or cross-sectional predictors, and using rolling-origin evaluation would strengthen robustness and generalizability.

8 Conclusion

8.1 Summary

This project delivers a reproducible forecasting pipeline and a transparent comparison of econometric and machine learning models for TSLA return and volatility forecasting. Across horizons, the random walk baseline remains a strong benchmark, and selection based on validation RMSE does not always generalize to the test set. The results therefore reinforce a core empirical message: return predictability is fragile, while volatility dynamics are more stable and forecastable.

Methodologically, the pipeline emphasizes time-series integrity, reproducibility, and probabilistic outputs. The Monte Carlo layer and volatility modeling make long-horizon forecasts usable as risk bands rather than single price targets.

In short, the project succeeds more as a risk-forecasting and uncertainty-quantification tool than as a return-forecasting engine.

8.2 Future Directions

- Expand to multiple tickers and market regimes.
- Add exogenous features (macro, sector indices, options-implied vol).
- Use rolling-origin backtests to characterize performance distributions.
- Explore alternative ML and probabilistic return models.

References

Andersen, Torben G. et al. (2003). "Modeling and Forecasting Realized Volatility". In: *Econometrica* 71.2, pp. 579–625.

Bollerslev, Tim (1986). "Generalized Autoregressive Conditional Heteroskedasticity". In: *Journal of Econometrics* 31.3, pp. 307–327.

Campbell, John Y., Andrew W. Lo, and A. Craig MacKinlay (1997). *The Econometrics of Financial Markets.* Princeton University Press.

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, pp. 785–794.

Engle, Robert F. (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation". In: *Econometrica* 50.4, pp. 987–1007.

Fama, Eugene F. (1970). "Efficient Capital Markets: A Review of Theory and Empirical Work". In: *Journal of Finance* 25.2, pp. 383–417.

Glosten, Lawrence R., Ravi Jagannathan, and David E. Runkle (1993). "On the Relation Between the Expected Value and the Volatility of the Nominal Excess Return on Stocks". In: *Journal of Finance* 48.5, pp. 1779–1801.

Gu, Shihao, Bryan Kelly, and Dacheng Xiu (2020). "Empirical Asset Pricing via Machine Learning". In: *Review of Financial Studies* 33.5, pp. 2223–2273.

Hansen, Peter R. and Asger Lunde (2005). "A Forecast Comparison of Volatility Models: Does Anything Beat a GARCH(1,1)?" In: *Journal of Applied Econometrics* 20.7, pp. 873–889.

Krauss, Christopher, Xuan Anh Do, and Nicolas Huck (2017). "Deep Neural Networks, Gradient-Boosted Trees, Random Forests: Statistical Arbitrage on the S and P 500". In: *European Journal of Operational Research* 259.2, pp. 689–702.

Poon, Ser-Huang and Clive W. J. Granger (2003). "Forecasting Volatility in Financial Markets: A Review". In: *Journal of Economic Literature* 41.2, pp. 478–539.

Welch, Ivo and Amit Goyal (2008). "A Comprehensive Look at the Empirical Performance of Equity Premium Prediction". In: *Review of Financial Studies* 21.4, pp. 1455–1508.

A Additional Figures

Supplementary figures and diagnostics can be placed here.

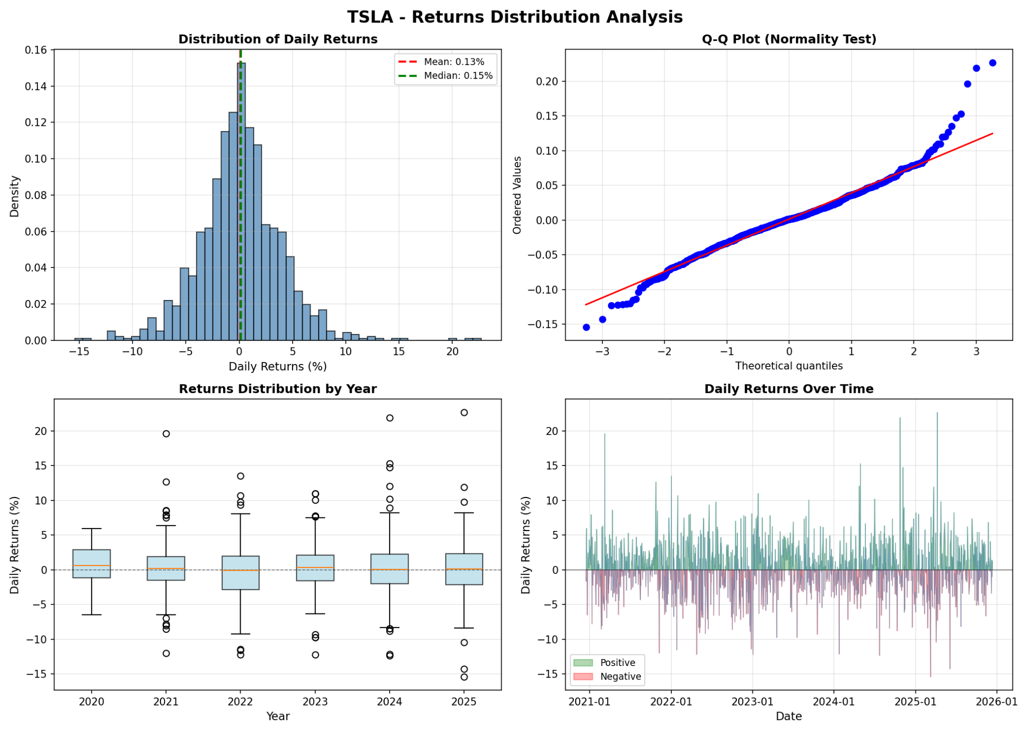

Figure A.1: EDA: return distribution and normality diagnostics (TSLA).

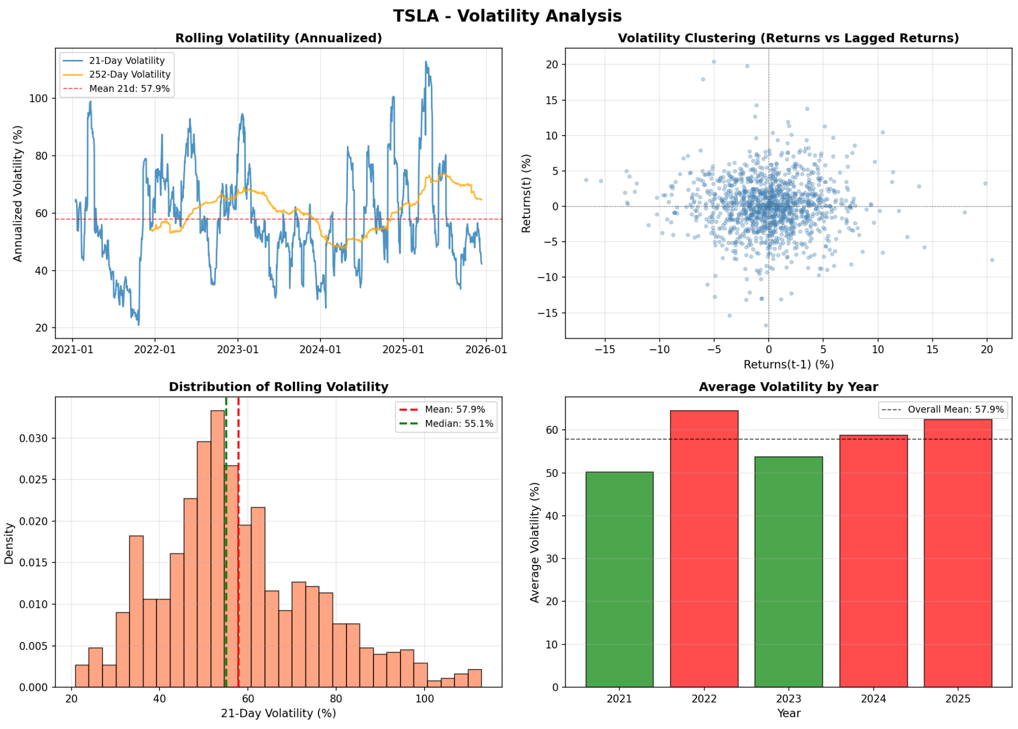

Figure A.2: EDA: 21-day rolling volatility showing clustering (TSLA).

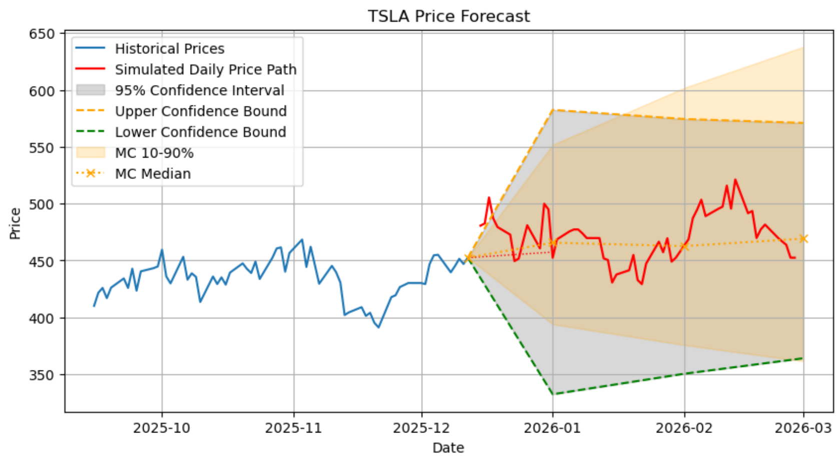

Figure A.3: Example forecast gap at the horizon boundary (3-month forecast).

Table A.1: Test selection model (diagnostic).

Horizon	Lowest RMSE model	Test RMSE
10 days	Random walk	17.54
1 month	ARIMA	26.39
3 months	XGBoost	45.65
6 months	Random walk	66.43
1 year	Random walk	62.44

Table A.2: Horizon summary for the validation-selected model (test MAPE after refit on train+validation).

Horizon	Best model	MAPE	Expected return	Risk level	Signal quality
10 days	ElasticNet	3.79%	+1.16%	HIGH	HIGH
1 month	Random walk	7.23%	+1.28%	HIGH	MEDIUM
3 months	ARIMA	24.73%	+3.71%	HIGH	LOW
6 months	ARIMA	15.15%	+8.56%	HIGH	LOW
1 year	ARIMA	16.12%	+11.62%	HIGH	LOW

B Code Repository

Repository: `https://github.com/Dionavdj/asset-forecasting.git`
The codebase is organized around `main.py` and the `src/` modules described in Section 4. Reproducibility is achieved via cached data and fixed random seeds. Follow the README to set up the environment; in brief:

```
conda env create -f environment.yml
conda activate stock-forecast
python main.py
```

These steps install dependencies, activate the project environment, and run the pipeline end to end.

C AI Usage

This project used AI tools for debugging, refactoring suggestions, documentation support, and editing of the report text for clarity and structure. All code, results, and written content were reviewed and verified by the author to comply with course guidelines. A usage log is provided in `AI_USAGE.md`.