Volatility Forecasting

Week 5 — Financial Management: Volatility, Risk, and AI

Yi-Hao Lai

2026 Spring

Today's Roadmap

1

The Story: The Horse Race

The Forecasting Arena

Scene: Priya arrives at VolTech with an LSTM model.

"Let me pit my LSTM against your GARCH."

The challenge: Which model best *predicts* tomorrow's volatility?

In-sample fit $\neq$ out-of-sample prediction

The Core Question

Can your model predict volatility it has **never seen**, or is it just memorizing the past?

Today's Learning Objectives

By the end of this session, you will:

1. Distinguish in-sample fit from out-of-sample forecasting
2. Implement rolling-window one-step-ahead forecasts
3. Evaluate forecasts with RMSE, MAE, and QLIKE
4. Run the Mincer–Zarnowitz regression
5. Compare models using the Diebold–Mariano test

2

The Forecasting Framework

In-Sample vs. Out-of-Sample

In-sample (estimation):

- Model "sees" all data
- Optimizes parameters to fit
- Can memorize noise
- Like a practice exam

Out-of-sample (evaluation):

- Model has never seen this data
- Forecasts into the unknown
- Tests genuine predictive power
- Like the real exam

⚠ Key Principle

A model that fits the past perfectly might be **overfitting**. The only honest test is out-of-sample.

Rolling Window Forecasting

The Procedure

1. Fix estimation window of W observations
2. Estimate model on days $1, 2, \ldots, W$
3. Forecast $\hat{\sigma}^2_{W+1}$ (one step ahead)
4. Roll forward: estimate on $2, 3, \ldots, W+1$
5. Forecast $\hat{\sigma}^2_{W+2}$
6. Repeat until end of data

Key: Each forecast uses *only* past information. No future data leakage.

Rolling vs. Expanding Window

Feature	Rolling	Expanding
Window size	Fixed (W)	Growing
Old data	Dropped	Kept forever
Structural breaks	Adapts	May be contaminated
Parameter stability	Less stable	More stable
Best for	Changing regimes	Stable environments

For equity markets: rolling windows are generally preferred because market dynamics evolve over time.

3

The Volatility Proxy Problem

What Are We Forecasting Against?

The Fundamental Challenge

We never observe true volatility σ_t^2. We need a **proxy** to evaluate forecasts.

Proxy	Precision	Data Required
Squared return r_t^2	Very low	Daily only
Range-based (Parkinson)	Moderate	OHLC
Realized variance RV_t	High	Intraday (5-min)

Using r_t^2: all models look bad (low R^2), but **rankings are preserved**.

4

Loss Functions

RMSE: Root Mean Squared Error

RMSE

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\sigma_t^{2*} - \hat{\sigma}_t^2)^2}$$

- Penalizes **large errors** heavily (squared)
- Good for detecting models with occasional terrible forecasts
- Sensitive to outliers in the proxy

MAE: Mean Absolute Error

MAE

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T} |\sigma_t^{2*} - \hat{\sigma}_t^2|$$

- Treats all errors **proportionally**
- More robust to outliers than RMSE
- Easier to interpret: "average forecast miss"

QLIKE: The Gold Standard

QLIKE

$$\text{QLIKE} = \frac{1}{T}\sum_{t=1}^{T}\left(\ln(\hat{\sigma}_t^2) + \frac{\sigma_t^{2*}}{\hat{\sigma}_t^2}\right)$$

- Penalizes **relative** errors (scale-independent)
- Robust to the choice of volatility proxy (Patton, 2011)
- **Preferred by practitioners** when rankings differ across metrics

Rule of Thumb

When RMSE and QLIKE disagree, trust QLIKE.

Why Multiple Loss Functions?

Property	RMSE	MAE	QLIKE
Large error penalty	Heavy	Linear	Relative
Outlier robustness	Low	Medium	High
Proxy robustness	Low	Low	**High**
Interpretability	High	High	Medium

Best practice: Report all three. If a model wins on *all* metrics, the conclusion is clear.

5

Mincer–Zarnowitz Regression

Testing Forecast Efficiency

Mincer–Zarnowitz Regression

$$\sigma_t^{2*} = a + b\,\hat{\sigma}_t^2 + u_t$$

Perfect forecast: $a = 0$, $b = 1$

Interpreting the coefficients:

- $b < 1$: Forecasts **overreact** (too variable)
- $b > 1$: Forecasts **underreact** (too smooth)
- $a \neq 0$: Systematic **bias**
- R^2: Proportion of variance explained

Test $H_0 : a = 0,\ b = 1$ jointly with an F-test.

Understanding R^2 in the MZ Regression

Don't Panic About Low R^2

Against r_t^2 proxy: $R^2 \approx 5$–15% is normal.

Against realized variance RV_t: $R^2 \approx 30$–50%.

The difference is entirely due to **proxy noise**, not model quality.

What matters: the *relative* ranking of models, not the absolute R^2.

6

Python Implementation

Step 1: Forecast Setup

```python
import numpy as np, pandas as pd
import yfinance as yf
from arch import arch_model

sp500 = yf.download("^GSPC",
    start="2015-01-01", end="2024-12-31")
prices = sp500["Close"].squeeze()
returns = 100 * np.log(
    prices / prices.shift(1)).dropna()

window = 1000  # rolling estimation window
n_oos = len(returns) - window
print(f"Out-of-sample: {n_oos} days")
```

Step 2: Rolling Forecast Loop

```
model_specs = {
    "GARCH":     {"vol":"Garch","p":1,"o":0,"q":1},
    "GJR-GARCH": {"vol":"Garch","p":1,"o":1,"q":1},
    "EGARCH":    {"vol":"EGARCH","p":1,"o":1,"q":1},
}
forecasts = {}
for name, spec in model_specs.items():
    fc = np.full(n_oos, np.nan)
    for i in range(n_oos):
        train = returns.iloc[i:i + window]
        model = arch_model(train, dist="t", **spec)
        res = model.fit(disp="off")
        fv = res.forecast(horizon=1)
        fc[i] = fv.variance.values[-1, 0]
    forecasts[name] = pd.Series(
        fc, index=returns.index[window:])
```

Step 3: Evaluation Metrics

```python
realized = returns.iloc[window:] ** 2

def evaluate(realized, forecasts):
    results = {}
    for name, fc in forecasts.items():
        r, f = realized.values, fc.values
        f = np.maximum(f, 1e-10)
        rmse = np.sqrt(np.mean((r - f)**2))
        mae  = np.mean(np.abs(r - f))
        qlike = np.mean(np.log(f) + r / f)
        results[name] = {"RMSE": rmse,
            "MAE": mae, "QLIKE": qlike}
    return pd.DataFrame(results).T

print(evaluate(realized, forecasts).round(4))
```

Step 4: Mincer–Zarnowitz Regression

```python
import statsmodels.api as sm

for name, fc in forecasts.items():
    r = realized.values
    f = fc.values
    X = sm.add_constant(f)
    ols = sm.OLS(r, X).fit()
    a, b = ols.params
    r2 = ols.rsquared
    f_test = ols.f_test("(const = 0), (x1 = 1)")
    print(f"{name:<14} a={a:.4f} b={b:.4f} "
          f"R2={r2:.4f} p={float(f_test.pvalue):.4f}")
```

Expected: GJR-GARCH has b closest to 1 and highest R^2.

7

The Diebold–Mariano Test

Comparing Two Models Formally

Diebold–Mariano Test

$$d_t = L(\hat{\sigma}^2_{1,t}) - L(\hat{\sigma}^2_{2,t})$$

$$DM = \frac{\bar{d}}{\hat{\sigma}_{\bar{d}}} \xrightarrow{d} N(0,1)$$

- H_0: Both models have equal predictive accuracy
- $|DM| > 1.96$: one model is significantly better (5%)
- Uses **HAC standard errors** (Newey–West) because forecast errors are serially correlated

Step 6: DM Test in Python

```python
from scipy import stats

def dm_test(realized, fc1, fc2):
    r, f1, f2 = realized.values, fc1.values, fc2.values
    L1 = np.log(f1) + r / f1  # QLIKE losses
    L2 = np.log(f2) + r / f2
    d = L1 - L2
    T = len(d)
    lag = int(T ** (1/3))
    gamma_0 = np.var(d, ddof=1)
    gamma_sum = sum((1-k/(lag+1)) *
        np.cov(d[k:], d[:-k])[0,1]
        for k in range(1, lag+1))
    var_d = (gamma_0 + 2*gamma_sum) / T
    dm = d.mean() / np.sqrt(max(var_d, 1e-20))
    p = 2*(1 - stats.norm.cdf(abs(dm)))
    return dm, p
```

8

The Horse Race Results

Who Wins?

Model	RMSE	MAE	QLIKE
GARCH(1,1)	5.82	2.14	3.72
GJR-GARCH(1,1)	**5.61**	**2.03**	**3.65**
EGARCH(1,1)	5.65	2.10	3.66

GJR-GARCH wins on all three metrics.

The leverage effect is not just a better in-sample fit — it provides **genuine predictive power**.

Key Findings

1. **GJR-GARCH consistently outperforms GARCH** across all loss functions (DM test confirms: $p < 0.01$)
2. **EGARCH performs similarly to GJR-GARCH** — both asymmetric models capture the leverage effect
3. **Improvements are modest but consistent** — don't expect huge gains; volatility is hard to predict
4. **All models struggle during crisis onset** — the first days of COVID were poorly forecast by every model

9

AI Spotlight: GARCH vs. Deep Learning

The GARCH vs. ML Debate

Horizon	GARCH	ML/DL
1–5 days	**Competitive**	Marginal gains
1–4 weeks	Good baseline	**Advantage**
1+ months	Struggles	**Clear advantage**

Why GARCH is hard to beat at short horizons:

- Parsimonious (3–4 parameters): hard to overfit
- The β term dominates 1-day forecasts
- Captures key stylized facts with minimal assumptions

Best approach: Hybrid models — GARCH output as a feature in an ML model, combined with VIX, volume, and sentiment.

Priya's LSTM Results

Honest Assessment

- LSTM beat GJR-GARCH on RMSE by $\sim$2%
- LSTM was *worse* on QLIKE
- Training time: $100\times$ longer
- Interpretability: black box

Take-home: GARCH is not obsolete. It is a **strong baseline** that any AI model must beat to justify its complexity.

10

Key Takeaways

Six Things to Remember

1. **Out-of-sample is the true test**: In-sample fit can be misleading due to overfitting
2. **Rolling window**: Re-estimate daily on a fixed-length window — the gold standard
3. **Three loss functions**: RMSE (large errors), MAE (average miss), QLIKE (relative, proxy-robust)
4. **Mincer–Zarnowitz**: Tests forecast efficiency ($a = 0$, $b = 1$)
5. **Diebold–Mariano**: Tests whether the difference is statistically significant
6. **GJR-GARCH beats GARCH**: The leverage effect provides genuine predictive power for equities

Mission 5: The Forecasting Horse Race

Deliverables

1. Rolling forecasts for GARCH and GJR-GARCH on Nikkei 225 and S&P 500 ($W = 1000$)
2. Compute RMSE, MAE, QLIKE for each model
3. MZ regressions: report $\hat{a}$, $\hat{b}$, R^2, F-test p-value
4. Forecast comparison plot for each index
5. 200-word report: which model should VolTech deploy?

Bonus: Repeat with expanding window and compare results.

Next Week Preview

Week 6: Value-at-Risk

Kenji asks: "What is the most we can lose tomorrow with 99% confidence?"

We translate volatility forecasts into the number every bank calculates every single day.

Topics: Parametric VaR, Historical simulation, Expected Shortfall, backtesting