AI in Finance

Week 8 — Financial Management: Volatility, Risk, and AI

Yi-Hao Lai

2026 Spring

Today's Roadmap

1. The Story: The Final Showdown
2. ML vs. Econometrics: Two Philosophies
3. Random Forest for Volatility
4. LSTM Neural Networks
5. Python Implementation
6. The Verdict: GARCH vs. ML
7. The FinTech Frontier
8. Key Takeaways

1

The Story: The Final Showdown

The VolTech Challenge

Scenario: QuantStar claims their ML system beats traditional models by **50%**.

Priya sets up the arena: GARCH on the left screen, neural network dashboard on the right.

Can we verify — or debunk — this claim?

The Core Question

Under what conditions does ML beat GARCH, and how can we combine them intelligently?

Today's Learning Objectives

By the end of this session, you will:

1. Explain the bias-variance tradeoff in model selection
2. Build a Random Forest volatility model with engineered features
3. Understand LSTM neural network architecture and gates
4. Compare ML vs. GARCH using RMSE, MAE, and QLIKE
5. Design a hybrid GARCH + ML model

2

ML vs. Econometrics: Two Philosophies

Two Approaches to Volatility

GARCH (Econometric)

- Start from **theory**
- 5–7 interpretable parameters
- Strong assumptions (parametric)
- Works well with limited data
- Auditable and Basel-compliant

ML (Data-Driven)

- Start from **data**
- Thousands of parameters
- Flexible (nonparametric)
- Needs large datasets
- Often a "black box"

⚠ Key Insight

Neither is universally superior. The choice depends on data regime, prediction horizon, and interpretability requirements.

The Bias-Variance Tradeoff

Prediction Error Decomposition

$$\text{MSE} = \underbrace{\text{Bias}^2}_{\text{model rigidity}} + \underbrace{\text{Variance}}_{\text{model instability}} + \underbrace{\text{Noise}}_{\text{irreducible}}$$

	Bias	Variance
GARCH	High (rigid assumptions)	Low (stable)
Neural Net	Low (flexible)	High (unstable)

In noisy financial data: high-bias/low-variance models often win at short horizons. ML advantage emerges with richer features and longer horizons.

3

Random Forest for Volatility

Random Forest: The Ensemble Approach

A **Random Forest** builds many decision trees, each on a random subset of data and features.

Why it works:

- Captures nonlinear patterns
- Robust to outliers
- Built-in feature importance
- No distributional assumptions

Prediction

$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(\mathbf{x})$

Average of B trees reduces variance without increasing bias.

Feature Engineering for Volatility

Feature Type	Examples
Lagged returns	$r_{t-1}, r_{t-2}, \ldots, r_{t-5}$
Lagged squared returns	$r^2_{t-1}, \ldots, r^2_{t-5}$ ($\approx$ ARCH term)
Rolling volatility	5-day, 10-day, 22-day rolling σ ($\approx \beta$ term)
Absolute returns	$\lvert r_{t-1}\rvert, \ldots, \lvert r_{t-5}\rvert$
GARCH variance	$\hat{\sigma}^2_t$ from GJR-GARCH (hybrid feature!)

Key insight: The most important ML features mirror GARCH's structure. The RF is rediscovering volatility clustering from data.

4

LSTM Neural Networks

LSTM: Memory for Time Series

Long Short-Term Memory processes sequences one step at a time, maintaining a "memory cell" of past patterns.

Three Gates:

1. **Forget gate**: "Should I discard old volatility information?" (regime changes)
2. **Input gate**: "Is today's return important enough to remember?" (shock detection)
3. **Output gate**: "What's my volatility forecast based on what I remember?"

The Catch

LSTM can learn GARCH-like behavior on its own, but GARCH achieves this with **5 parameters**. LSTM needs **thousands** of parameters and much more data.

LSTM Gate Equations

Gate Mechanism (Simplified)

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{Forget gate}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{Input gate}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{Output gate}$$

- σ = sigmoid function (output between 0 and 1)
- h_{t-1} = previous hidden state (past summary)
- x_t = current input (today's return)
- Each gate learns *when* to act from data

5

Python Implementation

Step 1: Feature Engineering

```python
import numpy as np, pandas as pd
import yfinance as yf
from sklearn.ensemble import (
    RandomForestRegressor)
from arch import arch_model

sp500 = yf.download("^GSPC",
    start="2010-01-01", end="2024-12-31")
returns = 100 * np.log(
    sp500["Close"] / sp500["Close"].shift(1)
).dropna().squeeze()
target = returns.shift(-1) ** 2
```

Step 2: Build Feature Matrix

```python
features = pd.DataFrame(index=returns.index)
for lag in range(1, 6):
    features[f"ret_lag{lag}"] = (
        returns.shift(lag))
    features[f"ret2_lag{lag}"] = (
        returns.shift(lag) ** 2)
for w in [5, 10, 22]:
    features[f"rvol_{w}d"] = (
        returns.rolling(w).std())
for lag in range(1, 4):
    features[f"abs_ret_lag{lag}"] = (
        returns.shift(lag).abs())
```

Result: 18 engineered features capturing volatility dynamics

Step 3: Time-Series Split

```python
data = pd.concat([features,
    target.rename("target")], axis=1).dropna()
split = "2020-01-01"
train = data[data.index < split]
test = data[data.index >= split]

X_train = train.drop("target", axis=1)
y_train = train["target"]
X_test = test.drop("target", axis=1)
y_test = test["target"]
```

Critical

Never shuffle time-series data! Always split by date to avoid look-ahead bias.

Step 4: Random Forest Model

```
rf = RandomForestRegressor(
    n_estimators=500, max_depth=10,
    min_samples_leaf=20,
    max_features="sqrt", random_state=42)
rf.fit(X_train, y_train)
rf_pred = pd.Series(rf.predict(X_test),
    index=X_test.index)

importance = pd.Series(
    rf.feature_importances_,
    index=X_train.columns
).sort_values(ascending=False)
print("Top 5:", importance.head(5))
```

Step 5: Hybrid Model

```python
# Add GARCH variance as a feature
model = arch_model(returns, vol="Garch",
    p=1, o=1, q=1, dist="t")
res = model.fit(last_obs=split, disp="off")
garch_var = res.conditional_volatility ** 2

data_h = data.copy()
data_h["garch_var"] = garch_var
# ... re-split and train hybrid RF ...
```

The hybrid secret: Let GARCH summarize its structural knowledge into one feature, then let ML learn on top of it.

Step 6: Head-to-Head Comparison

```python
def evaluate(actual, forecast, name):
    a, f = actual.values, np.maximum(
        forecast.values, 1e-10)
    rmse = np.sqrt(np.mean((a - f) ** 2))
    mae = np.mean(np.abs(a - f))
    qlike = np.mean(np.log(f) + a / f)
    return {"Model": name, "RMSE": rmse,
            "MAE": mae, "QLIKE": qlike}
```

Expected results: Hybrid $>$ RF $>$ GARCH on most metrics. But GARCH may win on QLIKE due to better calibration during calm periods.

6

The Verdict: GARCH vs. ML

Head-to-Head Results

Criterion	GJR-GARCH	Random Forest	Hybrid
1-day RMSE	5.61	5.52	**5.21**
1-day QLIKE	3.65	3.82	**3.51**
Interpretability	High	Low	Medium
Training speed	Fast	Moderate	Moderate
Regulatory fit	Strong	Weak	Medium
Crisis response	Good	Variable	Good

QuantStar's "50% improvement" was based on in-sample results with look-ahead bias.

Three Key Findings

1. **GARCH is remarkably competitive at daily horizons**
   With 5–7 parameters, it captures the dominant volatility dynamics. ML barely improves on it alone.

2. **The hybrid model wins consistently**
   GARCH as a feature + ML flexibility = best forecasts. The whole is greater than the sum of its parts.

3. **Interpretability has economic value**
   A model regulators can audit may produce better *business* outcomes than a more accurate black box.

Alex's Recommendation to Kenji

The Verdict

Deploy GJR-GARCH as the **primary risk engine** for daily VaR and ES reporting — it's fast, interpretable, and Basel-compliant.

Layer the hybrid Random Forest as a **supplementary monitoring tool** that flags when GARCH may be underestimating risk.

Best of both worlds: regulatory compliance *and* cutting-edge analytics.

7

The FinTech Frontier

What's Next in AI + Finance

1. **Foundation models**: LLMs for sentiment analysis, earnings call summarization, risk narratives
2. **Explainable AI (XAI)**: SHAP values, attention maps making black boxes transparent
3. **Real-time risk**: Streaming ML pipelines processing tick-by-tick data + news + social media
4. **Democratization**: Open-source tools (`arch`, `scikit-learn`, `pytorch`) make quant analytics accessible
5. **RegTech**: AI automating compliance reporting, anomaly detection, audit trails

The professionals who thrive will understand *both* financial theory *and* AI tools.

8

Key Takeaways

Six Things to Remember

1. **ML and GARCH are complementary**: GARCH provides structure; ML adds flexibility
2. **GARCH is hard to beat at daily horizons**: 5–7 parameters capture the dominant dynamics
3. **Feature engineering is king**: Top ML features mirror GARCH's own structure
4. **Hybrid models win**: GARCH-as-a-feature consistently produces the best forecasts
5. **Interpretability has value**: Regulatory compliance requires model explainability
6. **The future is integration**: Econometric structure + ML flexibility + human judgment

Mission 8: The VolTech AI Risk Report

Deliverables

1. Build RF model for Nikkei 225 and S&P 500 (15+ features)
2. Build hybrid model with GJR-GARCH feature
3. Compare GARCH, RF, and Hybrid using RMSE, MAE, QLIKE
4. Create forecast comparison visualization
5. Write 500-word final memo to Kenji's committee

Bonus: Implement a simple LSTM and compare with RF

The Journey: 8 Weeks in Review

Week	Key Milestone
1	Discovered fat tails and non-normality
2	Learned to measure time-varying volatility
3	Built first GARCH model
4	Captured asymmetry with GJR-GARCH
5	Developed rigorous forecast evaluation
6	Computed VaR and Expected Shortfall
7	Created regulatory-compliant reports
8	Integrated AI with traditional models

From fat tails to AI: You've built a complete risk management system from scratch.

The End — and the Beginning

Epilogue

Three months later, Alex sat in his new corner office — **Senior Quantitative Analyst, VolTech Analytics**.

The hybrid GARCH-ML system was live, serving four institutional clients across Asia.

A new challenge arrived: jump risk, market contagion, tick-by-tick data.

The journey in quantitative finance never truly ends.

Welcome to the frontier.