Volatility Modeling

Ayush Singh

Abstract

This document outlines a comprehensive framework for predicting financial market volatility, a critical task for trading firms like Optiver. We address the project objective of building increasingly sophisticated models, starting with the Heterogeneous Autoregressive model for Realized Volatility (HAR-RV) as a baseline, progressing to the ARMA-GARCH framework, and culminating in an advanced Transformer-based deep learning model. We delve into the necessary financial theory, mathematical formulations, machine learning concepts, feature engineering strategies, and evaluation metrics required for implementation and assessment. The goal is to develop a predictor with high accuracy, leveraging high-frequency data and exploring models capable of capturing complex market dynamics.

Contents

1 Introduction

Volatility, typically measured as the standard deviation or variance of asset returns, quantifies the magnitude of price fluctuations. It is a cornerstone concept in finance, essential for risk management, option pricing (via models like Black-Scholes-Merton), and portfolio allocation. High volatility often signals market uncertainty and potential for large gains or losses, while low volatility suggests stability. Accurately forecasting volatility allows trading firms to adjust strategies, manage risk exposure effectively, and price derivatives more precisely. This project aims to build and evaluate models predicting future realized volatility using high-frequency data provided by Optiver for over 100 stocks.

2 Data and Realized Volatility

2.1 High-Frequency Data (HFD)

The project utilizes ultra-high-frequency data (UHFD), likely containing timestamped order book snapshots and executed trades. Key information includes:

- Bid prices and associated volumes at multiple levels.
- Ask prices and associated volumes at multiple levels.
- Timestamps (often to milliseconds or microseconds).
- Last traded price and volume.

Processing HFD requires careful handling due to its volume and potential for microstructure noise (e.g., bid-ask bounce, discrete price movements).

2.2 Realized Volatility (RV)

Unlike latent conditional volatility estimated by GARCH models, Realized Volatility (RV) is a model-free, observable measure of ex-post variation calculated from high-frequency intraday returns.

Definition: For a given trading day t, divided into N high-frequency intervals, let $p_{t,i}$ be the log-price at the end of the i-th interval within day t. The i-th intraday log-return is $r_{t,i} = p_{t,i} - p_{t,i-1}$. The daily Realized Volatility is typically calculated as the sum of squared intraday returns:

$$RV_t = \sum_{i=1}^{N} r_{t,i}^2 \tag{1}$$

Often, this is annualized by multiplying by the number of trading days in a year (e.g., 252).

Microstructure Noise Consideration: Simple summation of squared returns at the highest frequencies can be biased by market microstructure effects. Common adjustments include:

- **Sampling Frequency:** Using lower frequency returns (e.g., 5-minute or 1-minute intervals instead of tick-by-tick).
- **Realized Kernels:** Using methods like the Two-Scales Realized Volatility (TSRV) or realized kernel estimators to mitigate noise while using more data. For simplicity in initial modeling, fixed sampling (e.g., 1-minute or 5-minute) is common. Let's assume we use 1-minute returns for $N = 390$ (for a standard US equity market day).

The target variable for our prediction models will be the future RV, typically over the next day (RV_{t+1}) or a longer horizon. For this project, we aim to predict RV_{t+1} given information up to day t.

3 Model 1 (Baseline): HAR-RV Model

The Heterogeneous Autoregressive model for Realized Volatility (Corsi, 2009) is a simple yet effective time series model inspired by the Heterogeneous Market Hypothesis, which posits that market participants operate on different time horizons (daily, weekly, monthly).

3.1 Financial & ML Theory

The model assumes that volatility forecasts are influenced by past volatility observed over different time scales. It captures the long-memory property often observed in volatility (i.e., shocks decay slowly) using a simple autoregressive structure with lagged RV components averaged over different periods. It's essentially a linear regression model applied to time series data.

3.2 Mathematical Formulation

The standard HAR-RV model predicts the logarithm of RV (to ensure positivity and stabilize variance) at horizon h (here $h = 1$ day) using lagged daily, weekly (5-day average), and monthly (22-day average) log RVs:

$$\log(RV_{t+1}) = \beta_0 + \beta_d \log(RV_t) + \beta_w \log(RV_t^{(w)}) + \beta_m \log(RV_t^{(m)}) + \epsilon_{t+1} \quad (2)$$

where:

- RV_t is the realized volatility on day t.
- $RV_t^{(w)} = \frac{1}{5}\sum_{i=1}^{5} RV_{t-i+1}$ is the average RV over the past week.
- $RV_t^{(m)} = \frac{1}{22}\sum_{i=1}^{22} RV_{t-i+1}$ is the average RV over the past month.
- $\beta_0, \beta_d, \beta_w, \beta_m$ are coefficients estimated via Ordinary Least Squares (OLS).
- ϵ_{t+1} is the error term, assumed i.i.d. $N(0, \sigma_\epsilon^2)$.

The model can be extended to include other predictors like jumps (detected from HFD) or leverage effects (correlation between returns and volatility).

3.3 Implementation & Evaluation

Implementation involves calculating daily RV, computing the weekly and monthly moving averages, taking logarithms, and running OLS regression. Evaluation uses standard regression metrics on the predicted $\log(RV_{t+1})$ or transforms predictions back to RV levels for metrics like RMSE or MAE.

4 Model 2 (Intermediate): ARMA-GARCH Framework

This approach models the underlying asset returns directly and extracts the conditional volatility forecast. It combines an ARMA model for the conditional mean of returns and a GARCH model for the conditional variance.

4.1 Financial & ML Theory

Financial returns often exhibit stylized facts:

- **Fat Tails:** Returns distributions have heavier tails than a normal distribution.
- **Volatility Clustering:** Periods of high volatility tend to be followed by high volatility, and low by low.
- **Leverage Effect (Optional):** Negative returns tend to increase volatility more than positive returns of the same magnitude (captured by extensions like EGARCH or GJR-GARCH).

ARMA models capture linear autocorrelation in returns (often weak for daily returns). GARCH models directly address volatility clustering by modeling the conditional variance as dependent on past squared errors and past conditional variances.

4.2 Mathematical Formulation

Let $R_t = \log(P_t) - \log(P_{t-1})$ be the daily log-return. The ARMA(p,q)-GARCH(P,Q) model is specified as:

$$R_t = \mu_t + \epsilon_t \tag{3}$$

$$\mu_t = c + \sum_{i=1}^{p} \phi_i R_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \quad \text{(Conditional Mean - ARMA(p,q))} \tag{4}$$

$$\epsilon_t = \sigma_t z_t, \quad z_t \sim \text{i.i.d.} D(0,1) \tag{5}$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{Q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{P} \beta_j \sigma_{t-j}^2 \quad \text{(Conditional Variance - GARCH(P,Q))} \tag{6}$$

where:

- μ_t is the conditional mean of R_t.
- ϵ_t is the innovation (error term) at time t.
- σ_t^2 is the conditional variance of ϵ_t (and thus of R_t given μ_t). This is our one-step-ahead volatility forecast.

- z_t is an i.i.d. random variable with zero mean and unit variance, often assumed Normal or Student's t (to capture fat tails). $D(0, 1)$ denotes this distribution.
- c, ϕ_i, θ_j are ARMA parameters.
- $\omega > 0, \alpha_i \geq 0, \beta_j \geq 0$ are GARCH parameters ensuring non-negative variance.
- For stationarity of variance: $\sum \alpha_i + \sum \beta_j < 1$.

The orders (p, q, P, Q) are typically selected using information criteria (AIC, BIC) or by examining ACF/PACF of returns and squared returns. Model parameters are estimated using Maximum Likelihood Estimation (MLE).

4.3 Prediction & Relation to RV

The GARCH model provides a forecast for the *conditional variance* σ_{t+1}^2 based on information up to time t. This is conceptually different from the *realized variance* RV_{t+1}. However, σ_{t+1}^2 is the expected value of the integrated variance over day $t+1$ under certain assumptions. We can use $\hat{\sigma}_{t+1}^2$ from the GARCH model as a prediction for RV_{t+1}. Performance comparison against HAR-RV should ideally use RV_{t+1} as the ground truth.

5 Model 3 (Advanced): Transformer for Volatility

To potentially capture more complex, non-linear dynamics and longer-range dependencies than HAR or GARCH, we propose using a Transformer model, adapted for time series forecasting. LSTMs are common for sequence modeling but can struggle with very long sequences and their sequential nature limits parallelization. Transformers, with their self-attention mechanism, can model dependencies regardless of distance in the sequence and are highly parallelizable.

5.1 ML Theory: Attention Mechanism

The core of the Transformer is the self-attention mechanism. It allows the model to weigh the importance of different past time steps when predicting the future.

Scaled Dot-Product Attention: Given a query Q, key K, and value V matrices (derived from the input sequence), the attention output is calculated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{7}$$

where d_k is the dimension of the keys. The softmax ensures the weights sum to 1. In self-attention, Q, K, and V are derived from the same input sequence, allowing each position to attend to all positions in the previous layer.

Multi-Head Attention: This runs the attention mechanism multiple times in parallel with different, learned linear projections of Q, K, V. This allows the model to jointly attend to information from different representation subspaces at different positions. The outputs are concatenated and linearly projected.

Positional Encoding: Since Transformers don't process data sequentially, they require information about the position of each element in the sequence. This is added via positional encodings (e.g., sine and cosine functions of different frequencies).

5.2 Model Architecture for Volatility Prediction

We can use an Encoder-only Transformer architecture (similar to BERT but for regression) or an Encoder-Decoder architecture (standard for sequence-to-sequence tasks).

Input Features: The input sequence $X = (x_1, x_2, ..., x_T)$ consists of feature vectors at each time step t. Crucially, these features should include more than just lagged log RV:

- Lagged log RV: $\log(RV_t), \log(RV_{t-1}), ..., \log(RV_{t-T+1})$.
- Other engineered features (see Section 6).
- Potentially time features (day of week, month).

Encoder Architecture:

1. **Input Embedding:** Project the input features x_t into a higher dimensional space d_{model}. Add positional encodings.
2. **Multi-Head Self-Attention Layer:** Apply multi-head self-attention to the embedded sequence.

3. **Add & Norm:** Add the input of the attention layer (residual connection) and apply Layer Normalization.

4. **Feed-Forward Network:** Apply a position-wise fully connected feed-forward network (typically two linear layers with a ReLU or GELU activation).

5. **Add & Norm:** Another residual connection and layer normalization.

6. Repeat steps 2-5 for multiple layers (e.g., 2-6 layers).

Output Layer: Take the output corresponding to the last time step (or use pooling over all outputs) and pass it through a linear layer to predict the target $\log(RV_{t+1})$.

5.3 Mathematical Reasoning & Advantages

- **Long-Range Dependencies:** Self-attention directly models pairwise interactions between all input time steps, overcoming the limitations of RNNs in capturing very long-term effects prevalent in financial data.
- **Non-linearity:** The feed-forward networks and the attention mechanism itself introduce strong non-linear modeling capabilities.
- **Feature Interaction:** Attention can capture complex interactions between different input features (e.g., how past RV interacts with past order book imbalance).
- **Parallelization:** Computations within each layer are highly parallelizable, leading to faster training on GPUs compared to RNNs.

6 Feature Engineering

The performance of any ML model, especially deep learning models like Transformers, heavily depends on the quality of input features derived from the raw HFD. Beyond lagged RV ($RV_t, RV_t^{(w)}, RV_t^{(m)}$), crucial features include:

- **Order Book Statistics:**
  - **Weighted Average Price (WAP):** WAP $= \frac{P_{bid}V_{ask}+P_{ask}V_{bid}}{V_{ask}+V_{bid}}$. Used for calculating high-frequency returns robust to bid-ask bounce.

  - **Bid-Ask Spread:** $Spread = P_{ask} - P_{bid}$. Wider spreads often correlate with higher volatility.
  - **Order Book Imbalance (OBI):** OBI $= \frac{V_{bid} - V_{ask}}{V_{bid} + V_{ask}}$. Measures buying vs. selling pressure.
  - **Book Depth:** Total volume available within certain price levels from the midpoint.

- **Trade Statistics:**
  - **Trade Volume/Frequency:** Higher activity can indicate volatility.
  - **Trade Sign Imbalance:** Difference between buyer-initiated and seller-initiated trades.

- **Derived Volatility Measures:**
  - **Realized Bipower Variation (BPV):** Robust to jumps in prices. $BPV_t = \frac{\pi}{2} \sum_{i=2}^{N} |r_{t,i}||r_{t,i-1}|$.
  - **Jump Component:** $RV_t - BPV_t$ can estimate the contribution of price jumps to volatility.

- **Time Features:** Day of the week, time to market open/close.

These features should be calculated at high frequency (e.g., every minute or 5 minutes) and then aggregated (e.g., mean, std dev, sum) over the day to create daily features for predicting RV_{t+1}. For the Transformer, sequences of these features over past days (T lags) would be used.

7 Model Evaluation

To assess prediction accuracy and reliability, especially for volatility which is non-negative and has a skewed distribution, multiple metrics are needed:

- **Root Mean Squared Error (RMSE):** $RMSE = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (y_i - \hat{y}_i)^2}$, where y_i is the true value (e.g., RV_{t+i}) and $\hat{y}_i$ is the prediction. Penalizes large errors. Best calculated on RV levels.

- **Mean Absolute Error (MAE):** $MAE = \frac{1}{M} \sum_{i=1}^{M} |y_i - \hat{y}_i|$. Less sensitive to outliers than RMSE. Best calculated on RV levels.

- **Mean Absolute Percentage Error (MAPE):** $MAPE = \frac{100\%}{M}\sum_{i=1}^{M}|\frac{y_i - \hat{y}_i}{y_i}|$. Useful for relative comparison but undefined if $y_i = 0$ and unstable for y_i near zero.
- **QLIKE (Quasi-Likelihood):** $QLIKE = \frac{1}{M}\sum_{i=1}^{M}(\log(\hat{y}_i) + \frac{y_i}{\hat{y}_i})$. Suitable for variance forecasts, penalizes underprediction more than overprediction. Use $y_i = RV_{t+i}, \hat{y}_i = \hat{RV}_{t+i}$.
- R^2 **(Coefficient of Determination):** $R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$. Measures the proportion of variance explained by the model. Often calculated on $\log(RV)$ for HAR-RV.
- **Diebold-Mariano Test:** A formal statistical test to compare the predictive accuracy of two competing forecast models.

Evaluation should be performed using out-of-sample data, employing a rolling forecast origin or expanding window approach appropriate for time series data.

8 Implementation Considerations

- **Software:** Python with libraries like ‘pandas‘, ‘numpy‘, ‘statsmodels‘ (for HAR, ARMA-GARCH), ‘tensorflow‘ or ‘pytorch‘ (for Transformer), ‘scikit-learn‘ (for evaluation metrics, potentially feature scaling).
- **Hardware:** Training deep learning models like Transformers benefits significantly from GPUs.
- **Data Preprocessing:** Handling large HFD requires efficient data storage and processing pipelines. Cleaning (filtering outliers, handling missing data) is critical. Aligning timestamps across different data sources (order book, trades) is necessary.
- **Hyperparameter Tuning:** All models have hyperparameters (e.g., lags in HAR, orders in ARMA-GARCH, layers/heads/dimensions in Transformer, learning rate, batch size). Use techniques like grid search, random search, or Bayesian optimization with cross-validation (time-series aware).
- **Interpretability:** While HAR-RV and GARCH offer some interpretability through coefficients, Transformers are black boxes. Techniques like SHAP

(SHapley Additive exPlanations) or attention map visualization might be explored to understand feature importance and model behavior, crucial for communication with traders.

- **Stationarity:** Ensure time series inputs (especially for ARMA/GARCH) are stationary, potentially requiring differencing. Test using ADF or KPSS tests.

9 Conclusion

This document outlines a structured approach to predicting financial volatility, progressing from the interpretable HAR-RV baseline, through the statistically grounded ARMA-GARCH, to a powerful Transformer model capable of learning complex patterns from high-frequency data features. Success hinges on careful data processing, thoughtful feature engineering, rigorous model selection and tuning, and appropriate evaluation. By comparing these models, we can determine the trade-offs between complexity, accuracy, and interpretability, ultimately aiming for a robust predictor suitable for practical use in trading. The advanced Transformer model, while complex, holds the potential for state-of-the-art accuracy by leveraging the full richness of the high-frequency data landscape.