Forecasting Stock Volatility

A Model Comparison for Louis, Quantitative Trader

Presenter: Yun Chang 520397297

Project Overview & Data Description

Goal: Predict realized stock volatility from high-frequency data.
Data Used: First 500 time_id sessions from stock_1.csv
Key Features Created:

- **WAP (Weighted Average Price):** Combines bid/ask prices & sizes.
- **Bid-Ask Spread:** Relative measure of liquidity.
- **Log Returns:** Calculated per second from WAP.

Target Variable:

- **Realized Volatility:**
  Square root of the sum of squared log returns, computed in **20-second buckets**.

Data Split for Model Training

- Each time series (per `time_id`) was split **chronologically** into:
  - **First 80%** → used for training
  - **Last 20%** → held out for validation
- Ensures models are evaluated on **unseen future data**, simulating real-world forecasting.
- Prevents information leakage by maintaining the **temporal order** of volatility data.

Candidate Models

1. **Linear Regression**
2. **HAV-RV**
3. **eXtreme Gradient Boosting (XGBoost Tree-Based)**

Model 1 – Linear Regression

```
Call:
lm(formula = volatility ~ price + order + BidAskSpread, data = list.reg[[i]],
    weights = 0.8^(((len.train - 2):0)/2))
```

Objective:

Predict **realized volatility** using:

- **WAP (Weighted Average Price)** – proxy for execution price
- **Order Size** – total liquidity
- **Bid-Ask Spread** – market friction indicator

Uses **exponential weighting** to emphasize recent time buckets.
Built one model per time_id to capture local patterns.

Performance Summary (time_id 162):

- **Adjusted R^2:** 0.497

Model 2 – HAV-RV Model

- Designed specifically for **high-frequency financial data**, the HAV-RV model forecasts volatility using past volatility patterns.
- Predictors:
    - vol_1: volatility at time *t-1*
    - mean_vol_5: average volatility over past 5 buckets
- Equation:

$$\hat{v}_t = \beta_0 + \beta_1 \cdot v_{t-1} + \beta_2 \cdot \overline{v}_{t-5:t-1}$$

- Also apply **Weighted Least Squares (WLS)** using quarticity to adjust for heteroskedasticity — i.e., varying levels of volatility uncertainty.
- At *time_id 162*:
    a. **WLS Adjusted R^2**: 0.4002
    b. **Unweighted Adjusted R^2**: 0.0293

```
Call:
lm(formula = vol ~ vol_1 + mean_vol_5, data = list.HAV[[i]])
```

```
Call:
lm(formula = vol ~ vol_1 + mean_vol_5, data = list.HAV[[i]],
    weights = list.HAV[[i]]$vol_1/sqrt(quar[[i]]$quarticity[5:(len.train -
        1)]))
```

Model 3 – XGBoost Model: Non-Linear Tree-Based Regression

- Tree-based regression model
- Predicts volatility using market microstructure features
- Captures nonlinear patterns, outliers, and interactions
- No assumptions of stationarity/normality

Feature Engineering

- Computed log_return = log(WAP / lag(WAP)) to measure price movement.
- Aggregated features per time_id:
  - WAP_mean, WAP_sd
  - BidAskSpread_mean, BidAskSpread_sd
  - log_return_sd

Target Variable

- Realized volatility = sqrt(sum(log_return^2))

Model Evaluation

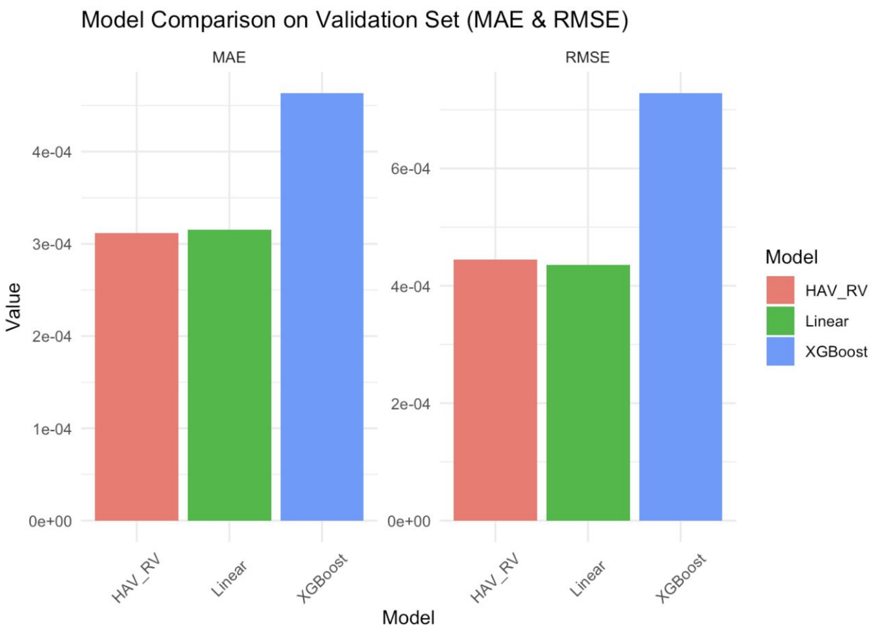
Evaluation Framework:

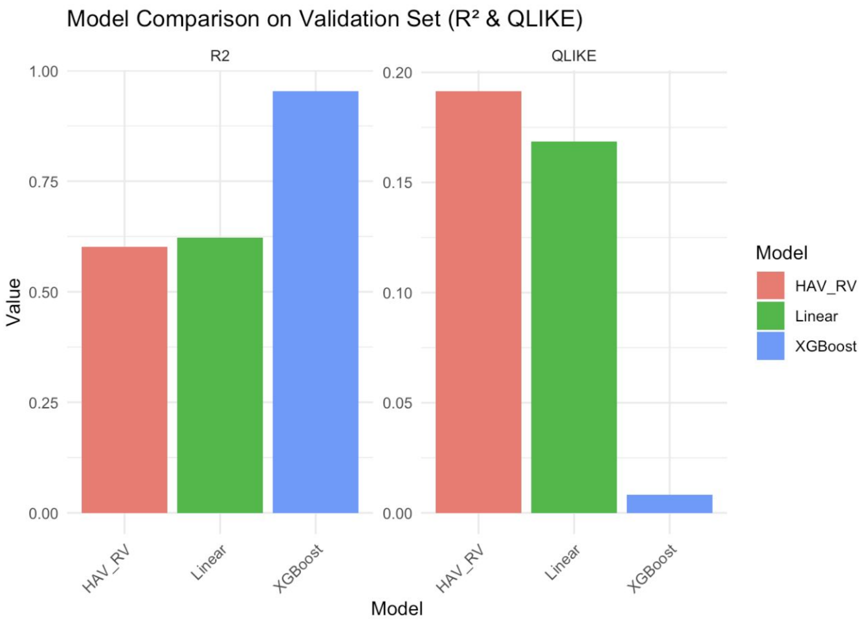
- **Data Split:** 80% training, 20% testing on time_id-level features.
- **Validation:** First 100 time_ids used for out-of-sample predictions for Linear, HAV-RV, and ARMA-GARCH. XGBoost evaluated on test set.
- **Metrics:**
  - **MAE / RMSE:** Measure average and squared error.
  - **R^2:** Captures goodness-of-fit.
  - **QLIKE:** Financially motivated loss for volatility accuracy.

Performance Results & Selection

	MAE	RMSE	R2	QLIKE
Linear	0.000315	0.000435	0.622229	0.168665
HAV_RV	0.000312	0.000444	0.601385	0.191400
XGBoost	0.000463	0.000728	0.953548	0.008318

Model Results & Selection





Final Model Selection: XGBoost

Why XGBoost was selected:

Best overall performance

- $R^2 = 0.95$, QLIKE = 0.0083

Strengths:

- Captures non-linear patterns, no distributional assumptions
- Robust to outliers and noise
- log_return_sd = most important feature

Limitations

- It can overfit on noisy features if not regularized well
- Less interpretable than linear models

Model Design:

- 80/20 split by time_id
- Engineered per time_id:
  - WAP, BidAskSpread, log_return (mean + sd)
- Target:
  - Realized volatility = $\sqrt{\sum \text{log\_return}^2}$
- Trained using:
  - 100 boosting rounds
  - reg:squarederror objective
  - Feature selection via xgb.importance()

Thank You For Listening
Any Questions?