Machine Learning for Finance Training

Katherine Bridges, Alana Edwards, Adam Ezzaoudi,
Effie Ktendis, Danny Mohseni

April 2025

Link to code:
`https://drive.google.com/drive/folders/1QW6GWsWwWTfZ5CRhuYDuIN2gMyRO1maQ?usp=sharing`

Contents

List of Figures

List of Tables

1 Introduction

Predicting the stock market can be difficult, especially considering all the external factors that influence it. Using Machine Learning can help understand how the market behaves and achieve the seemingly unachievable.

In today's environment, information is in an influx; attention spans have shortened, and news doesn't stay as relevant or reach nearly as many people as its used to. It seems volume has surpassed quality, and a large majority of people are more likely to make economic decisions based on public opinion and controversy affecting a stock. Furthermore, in recent years, investors have moved beyond traditional fundamentals, such as earnings and revenue, and instead emphasized future outlook, growth potential, and industry popularity when making their financial decisions. This shift has been driven in part by the AI boom and related political controversies such as trade disputes. This paper seeks to explore how Machine Learning models can incorporate this information into a market prediction model.

2 Problem & Data

2.1 Problem Motivation

One can see that stock volatility depends on so much more than simple metrics, and this is where Machine Learning models can elevate volatility predictions. When using a machine learning model to predict market fluctuations, one can consider public opinions, news, politics, and more. However, there is still no true one-size-fits-all model to ensure someone's portfolio grows faster than current market indexes. This report dives into the complexities of predicting stock volatility and compares XGBoost, Random Forest, and RNN against the current baseline GARCH Model used in the market today. Conclusions will be drawn on which model is the most effective and which features were most important in hopes of identifying a modern way to predict stock volatilities.

2.2 Asset & Data Collection

Asset and Data Collection for our models was rigorous and time-consuming, as raw datasets were created for all of our models utilizing webscraping techniques from 3 different data sources. The data sources we used are as follows:

Source	Data & Access Method	Issues & Limitations
Hacker News (BigQuery)	Article titles, URLs, and comment metadata via Google BigQuery	Data quota caps; occasional missing fields; requires Google Cloud billing
Reddit (r/wallstreetbets, r/stocks, ...)	(Real-time) Comments fetched via Python Reddit API Wrapper (PRAW)	Pushshift shutdown; PRAW rate limits; ticker false positives (e.g. "V"); only top 1,000 posts accessible
Yahoo Finance (yfinance)	Historical returns and volume data via yfinance	Generally stable; limited to 5 years of free history without a paid plan

Table 1: Overview of Data Sources.

Due to Reddit's limitation of only allowing 1-year's worth of data to be downloaded, we decided to split our raw data into 2 datasets for modeling. A description of what data source, time periods, and stock tickers are being used in each modeling dataset can be seen in the figure below:

Model	Period	Data Sources	Tickers Used
One-Year Model	Feb 2024–Apr 2025	BigQuery, PRAW, `yfinance`	AAPL, AMZN, BRK-B, COST, GOOG, GOOGL, HD, KO, META, MSFT, NVDA, PG, PLTR, PM, TSLA
Three-Year Model	Jan 2022–Apr 2025	BigQuery, `yfinance`	PG, MCD, TSLA, JPM, GOOGL, AAPL, COST, KO, NFLX, PM, META, NVDA, BRK.B, WMT, AMZN, HD, GOOG, MSFT, PLTR, INTU, JNJ, ORCL, PEP, RTX, VZ, IBM, CRM, PGR

Table 2: Comparison of Modeling Setups

For ticker selection, we first started by choosing a list of the 50 top tickers by weight from the S & P index. After pulling raw data for each time period (refer to Table 2), we include only tickers for which there was comment data for more than 70% of days, resulting in 15 total tickers for the 1-year data. This was preferred, as having too many features over only a year long time period quickly resulted in extra noise being added to the models. For the 3-year data, a similar method was employed to remove tickers without much sentiment information; a total of 22 were removed.

2.3 Data Characteristics

The size of the 1-year model dataset is **4892** days (across 15 tickers) X **7** features and the size of the 3-year model dataset is **21978** days (across 28 tickers) X **3** features. The features for each of the modeling datasets can be seen below:

Model	Feature	Description / Calculation
One-YearOne-YearOne-Yearpt¡ -One-Yearpt¿	RealizedVol_3d	Realized volatility over the p computed as the standard devia turns.
	reddit_sentiment_lag1	Avg. sentiment score (VADER dit comments from 1 day prior.
	reddit_volume_lag1	Total number of Reddit comm stock on the previous day.
	news_sentiment_lag1	Avg. sentiment score of Hacke prior.
	news_volume_lag1	Count of Hacker News articles get day.
	reddit_sentiment_missing	Binary (1 / 0) denoting whethe reddit_sentiment_lag1 day.
	news_missing	Binary (1 / 0) denoting whethe sentiment data was unavailable.
Three-YearThree-YearThree-Yearpt¡ -Three-Yearpt¿	RealizedVol_3d	Same as above.
	news_sentiment_lag11	Same as above.
	news_volume_lag1	Same as above.

Table 3: Feature Definitions / Calculations for Each Modeling Dataset

The target for both the 1-year and 3-year model datasets is the next-day realized volatility.

2.4 Data Quality, Preprocessing & Potential Biases

Given the disparate and noisy nature of our sources, careful preprocessing was essential to ensure time matching, reduce missingness, and prepare the data for machine learning. We had to thoroughly clean the data, due to size differences, ticker misrepresentation, and webscraping limits. We handled missing values and missing data due to holidays by reindexing all time series to align with NYSE trading days, ensuring consistency and accounting for holidays and weekends. We filled missing sentiment or volume data for a given ticker and date with zeros, working under the

assumption that no significant news implies neutral or no sentiment, and dropped rows with NaNs resulting from lag operations. We cleaned the raw text data–removed HTML tags, lowercased text, etc.–and used the VADER model to compute sentiment. Using keyword-based heuristics, tickers were filtered to match NYSE trading days. We then aggregated sentiment and comment volume per ticker per day and aligned them with stock volatility metrics. To prevent domination by high-magnitude features, we applied feature scaling to sentiment and volume metrics where appropriate. However, despite our thorough preprocessing, our data still has limitations. Sentiment from social platforms is inherently noisy and may include sarcasm, spam, or other irrelevant content, which may lead to the persistence of mismatches or false positives. There may also be a delay between when a sentiment is posted and when the market reacts, which the lag features only partially address. Additionally, Hacker News and Reddit likely have demographic and topic biases, favoring tech-savvy and retail investor communities, which could skew sentiment and volume patterns towards certain tickers. Due to the nature of our project, our analysis ultimately excluded stocks that had no comment or sentiment data (i.e., were not discussed on social platforms), which introduces the possibility of survivorship bias in the data. This means that our findings may not be able to be generalized to stocks that are less-mentioned, less-volatile, or less-known.

3 Methodology

3.1 GARCH

GARCH is a traditional econometric model that forecasts volatility using only past price data. It captures volatility clustering by modeling variance based on lagged squared returns and past variances. [2] In our setup, the only input is daily log returns (percentage-scaled), and the target is next-day realized volatility. A rolling window of the previous three days is used, fitting the model per stock and day from 2022 to 2025. No sentiment or volume data is included. Instances with insufficient history are logged and skipped. This provides a clean, price-based benchmark for comparing against more complex, feature-rich machine learning models.

3.2 Random Forest

A Random Forest is an ensemble machine learning model, which we used for threshold binary classification of future volatility states. This method handles mixed data types, nonlinearities, and overfitting well, making it ideal for our blend of traditional and alternative features. We use a Random Forest classifier to predict next-day high volatility in stocks, defined as the top 30% of realized volatility. Features include realized volatility, lagged sentiment and volume, which

we reduced via PCA, missingness indicators, and one-hot encoded tickers. We stratify by date to preserve class balance and prevent data leakage by lagging all predictors. PCA was applied only after standardizing relevant features. Model parameters, like number of trees and depth, were tuned via grid search with cross-validation. Key challenges included missing alt-data and feature collinearity, addressed through missingness flags and PCA, respectively. Reproducibility was ensured via fixed seeds, documented preprocessing, and scikit-learn pipelines.

3.3 XGBoost Regression

Extreme Gradient Boosting (XGBoost) is a machine learning algorithm that is used for supervised learning tasks, and for our model we used regression. It is widely used for predictive models for structured data (like time sequential data). XGBoost focuses on sequentially training models, in our case, decision trees, in which each new model tries to correct the errors of the previous one. The final prediction from the model is the weighted sum of all of the previous tree predictions [4].

Some of the more prominent challenges we addressed included missing data, ticker heterogeneity, and overfitting on short sequences, which we mitigated via feature lagging, filtering for sufficient history, and model regularization.

3.4 RNN (LSTM)

A Recurrent Neural Network (RNN) is a type of deep learning model designed to process sequential data. Unlike feedforward neural networks, RNNs maintain a hidden state across time steps, allowing them to learn temporal dependencies. This makes RNNs especially effective for financial modeling, where market conditions evolve over time and today's volatility may depend on previous sentiment and volume trends.

- **Using lagged and smoothed time-series features related to sentiment, volume, and volatility. The feature set included:**
  - `RealizedVol_3d`: 3-day realized volatility
  - `combined_sentiment_lag1`: one-day lagged sentiment (Reddit + news)
  - `combined_volume_lag1`: one-day lagged comment volume
  - `sentiment_missing`: binary indicator for missing sentiment data
  - `Target_lag1`, `Target_lag2`: lagged target values for additional memory

  The prediction target was a smoothed version of volatility (`Target_smooth`) using a 3-day moving average to reduce noise. Features were selected based on their temporal alignment and potential predictive value for next-day volatility.

To avoid data leakage, all lagged features were created using `groupby('ticker').shift()` to ensure future data was not included in past predictions. Smoothing operations were causal, and the train/test split was done without shuffling to preserve the chronological order.

- **Experimental Setup & Implementation**
  - **Model inputs:** Combined sentiment, comment volume, lagged target, and realized volatility.
  - **Target:** `Target_smooth` (a continuous regression target representing smoothed next-day volatility).
  - **Time-series aware splits:** Used an 80/20 chronological split via `train_test_split(..., shuffle=False)` to avoid look-ahead bias.
  - **Toolkits:** PyTorch (model training), scikit-learn (scaling, metrics), pandas (preprocessing), matplotlib (visualization).
  - **Brief notes on tuning:** Hidden size (128), layers (3), dropout (0.3), and sequence length (10) were chosen based on prior experiments and adjusted manually. No grid search or early stopping was applied.
  - **Steps for reproducibility:** All random splitting was time-based; data was cleaned and scaled consistently. Future iterations could fix random seeds and log results.
  - **Challenges overcome; adaptations made:** Sentiment data often had missing values; we encoded this explicitly and included it as a binary feature. We also included lagged target values to improve temporal context. To evaluate directionality of predictions, we computed directional accuracy post-inference.

4 Results

4.1 Baseline: GARCH

4.1.1 Evaluation Metrics

For each trading day between 2022 and 2025, provided at least three past observations are available, a new GARCH(3,3) model is fit using the `arch` package. The model then forecasts volatility for the next day. No sentiment, volume, or future data are used, ensuring that the model relies solely on historical price-based volatility patterns. Each stock is modeled independently, and to avoid data leakage, only data available up to the current prediction date is used. Instances with

insufficient data were logged and skipped. This setup ensures a realistic and reproducible baseline for evaluating the effectiveness of more complex models.

4.1.2 Main Findings

We used RMSE to compare the GARCH predictions to the true volatility per ticker per day. We found that it had an average RMSE of about 0.0115. This is a good error rate, but we aim to see if machine learning methods could improve this error, resulting in a more accurate prediction.

4.2 Random Forest

4.2.1 Evaluation Metrics

In this model (1 YR MODEL), we utilized F1 Score for the Positive Class (1) as the primary evaluation metric to assess accuracy due to the prediction metric defined by the following:

$$\tau \;=\; Q_{0.7}\big(\{\hat{p}_i\}_{i=1}^{N}\big) \tag{1}$$

$$\hat{y}_i \;=\; \begin{cases} 1, & \hat{p}_i \geq \tau \\ 0, & \hat{p}_i < \tau \end{cases} \tag{2}$$

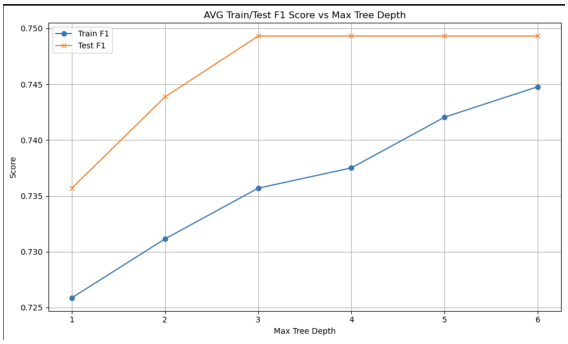
subsequented by ROC-AUC comparisons which represent the model's performance were it over a broad range of thresholds. The purpose is to classify on top 30% of days in a predicted time period that will have high volatility, allowing them to space trades, know the number of trades they will take in a time period, and when to increase/decrease the size of their positions.

4.2.2 Main Findings

Find Classification report for first attempt below. F1 for positive class was .75. As expected, RealizedVol3d dominated in terms of feature importance (at about .8), Because so much weight was placed on this feature, the trees likely split early on it and then used the sentiment features to filter out noise on small decisions, showing why 'F1-Test' score quickly flattens out at 3 trees. Also we tested the strategy to see an example of what the model determined a high-confidence trade. (including sentiment features + 3dVolatility from yfinance as features)

```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.89      0.89       856
           1       0.75      0.75      0.75       367

    accuracy                           0.85      1223
   macro avg       0.82      0.82      0.82      1223
weighted avg       0.85      0.85      0.85      1223

Confusion Matrix:
 [[764  92]
 [ 92 275]]
MACRO F1: 0.7493188010899183
Trade Signals Triggered: 367 out of 1223 samples
Precision of Strategy: 0.749

Top 5 high-confidence trades:
      vol_prob  true_label
4682  0.956791           1
4568  0.948520           1
4506  0.937026           1
3296  0.933987           1
4365  0.932920           1
```

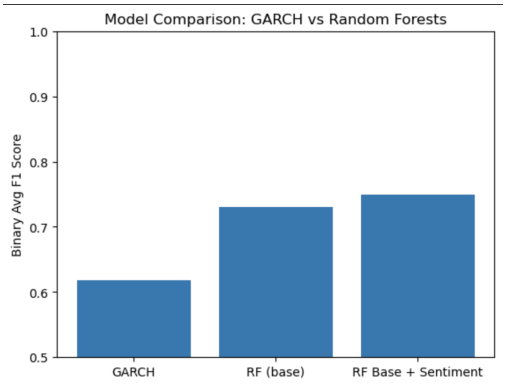
(a) Classification Report



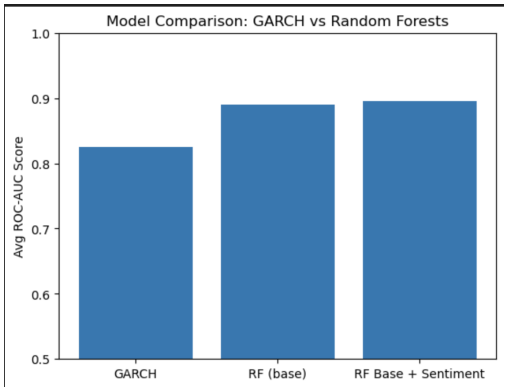
(b) F1/ROC-AUC vs. Tree Depth

Figure 1: Random Forest 1 Yr Results

Optimal parameters for the Random Forest Model were determined by grid search cross validation. Next we compared the model to Garch before applying any transformations. We compared three versions - the baseline (Garch), R.F. with just yahoo finance data, and RF with yfinance + Sentiment Data - in order to determine if adding sentiment was truly increasing accuracy. F1 and AUC-ROC were highest for the model with sentiment/yfinance data. Sentiment added .02 to the score for F1 and about .01 to AUC-ROC. While a small increase it was an increase nonetheless, however we needed to confirm our findings weren't just noise by employing cross validation (find more in section 4.2.3). GARCH Target volatility was binarized at same quartile threshold for consistency and comparison.



(a) F1 Comparison



(b) ROC-AUC comparison

Figure 2: Random Forest 1 Yr Vs. GARCH

4.2.3 Robustness & Overfitting

Because improvement was slim, we employed K-fold cross validation to confirm whether or not the 1 year of sentiment data was providing a meaningful signal. We found it actually did improve average F1 and ROC-AUC by about .01 when compared to a RF model built only using finance data. Because the six sentiment features combined had such little feature importance, we tested a model with 3 PCA components on the

sentiment features but performance declined as the PCA was forcing linear projection of likely non-linear relationships. Find results in figure below:

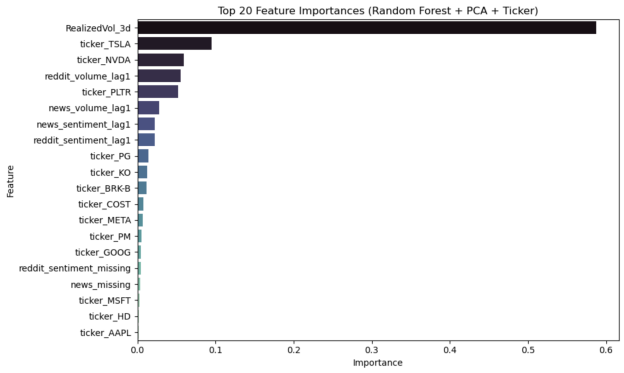
```
=== 3DVol Only RF (Base) ===
ROC-AUC per fold: [0.864 0.887 0.86  0.879 0.894]
Mean ROC-AUC: 0.877
F₁ per fold:    [0.685 0.726 0.702 0.716 0.744]
Mean F₁:    0.715

=== Rf sentiment + Base ===
ROC-AUC per fold: [0.878 0.886 0.868 0.883 0.897]
Mean ROC-AUC: 0.883
F₁ per fold:    [0.706 0.733 0.7   0.723 0.758]
Mean F₁:    0.724

=== PCA Sentiment + Base ===
ROC-AUC per fold: [0.874 0.888 0.861 0.88  0.901]
Mean ROC-AUC: 0.881
F₁ per fold:    [0.688 0.737 0.694 0.733 0.759]
Mean F₁:    0.722
```

Figure 3: K-Fold Cross Validation Scores RF 1 Yr

At this point, it was clear the model needed more features for it to learn more complex relationships. The next logical step was to employ one-hot-encoding on the tickers, in order to gain more information from the data. Find Cross validated f1/AUC scores and feature importance below:

```
Cross-validated AUC scores with ticker encoding: [0.89826208 0.90532052 0.88372985 0.89016716 0.90368282]
Mean AUC: 0.8962324864218312
Cross-validated f1 scores with ticker encoding: [0.74760383 0.75945537 0.71016692 0.74418605 0.75642965]
Mean F1:  0.7435683645290891
```

(a) cross-validation test scores



(b) Feature Importance

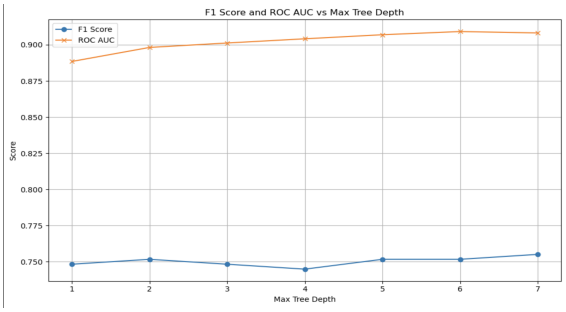
Figure 4: RF 1 Yr + One-Hot Encoding

4.2.4 Error Analysis

Grid search helped to identify the maximum tree depth to utilize for the ticker encoded model, helping reduce overfitting. A graph of the F1/ROC-AUC score showed it increased actually with more trees instead of stagnating early on like the model without one-hot encoding. Finally, walk forward testing was employed and in analysis, accuracy fluctuates throughout time, similar to market movement patterns. Importantly, days where ROC-AUC is greater than accuracy show us that the model is classifying correctly but the

chosen threshold may be incorrect. This helps show that the high order quantile-based threshold may be worse than a simple 50/50 split for prediction. Adaptive thresholding is recommended for future analysis.

(a) Walk Forward Testing



(b) F1/AUC vs. Tree Depth for Ticker Model

Figure 5: Error Analysis RF 1 Yr

4.3 XGBoost

4.3.1 Evaluation Metrics

In this model, we used Root Mean Squared Error (RMSE) as the primary evaluation metric to assess the accuracy of the XGBOOST regression model. RMSE measures the square root of the average of squared differences between predicted and actual values. By averaging RMSE across all rolling prediction steps, we obtain a robust estimate of the model's overall predictive performance across time.

4.3.2 Main Findings

Our model was trained separately for each ticker, not on the full dataset across all tickers to avoid data leakage between stocks and allow the model to learn stock specific patterns in volatility. The average RMSE across all tickers was about 0.0069. Below is a graph of the actual volatility mapped with our predicted volatility for each day for the ticker AAPL. We see that our model closely follows the trends of the true volatility where it does not predict the extreme highs and lows as accurately.

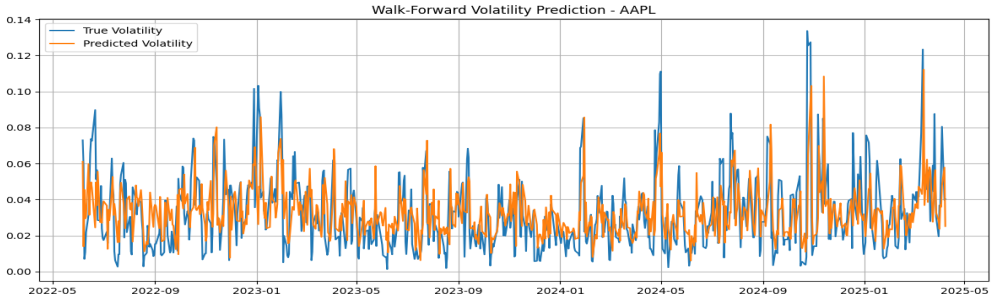


Figure 6: XGBoost: True vs. Predicted Volatility for AAPL.

We directly compared the XGBoost model's performance to the baseline GARCH model by aligning predictions on both date and ticker. From the previous GARCH model section above, we see that it has a RMSE of about 0.0115. We see that the XGBoost model performed better with an error of about 0.0069. This indicates that news sentiment does play a role when it comes to accurately predicting next day volatility. Below, in Figure 8, we see graphs for both the GARCH model and the XGBoost model, such that each data point corresponds to the volatility of a certain ticker on a certain day and is graphed based on it's true volatility vs it's predicted volatility.

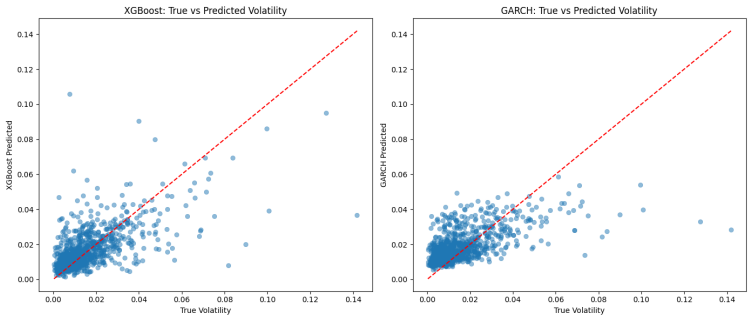


Figure 7: XGBoost vs GARCH: True vs. Predicted Volatility

4.3.3 Robustness & Overfitting

We conducted both walk-forward testing (per ticker) and time-aware 80/20 train/test splits on the one year data and the three year data to mimic real-world forecasting and avoid look-ahead bias. We found that the walk-forward testing for the three year model resulted in the lowest error, so that is the model discussed above. We used a `StandardScaler-XGBoost` pipeline and selected parameters: `n_estimators=200`, `max_depth=5`, and `learning_rate=0.05` to balance performance and runtime. By fixing random seeds we were able to maintain reproducibility.

4.3.4 Error Analysis

While we used average RMSE as our metric for general model performance, we also conducted a residual error analysis by computing and plotting the distribution of prediction errors for both the GARCH and XGBoost models. The histogram visualization provided insight into the spread and skew of errors, so we were able to to see how accurate each model was and how biased the errors were. Based on the histogram below, we see that GARCH is more likely to underestimate volatility whereas the XGBosst model has a more even distribution.

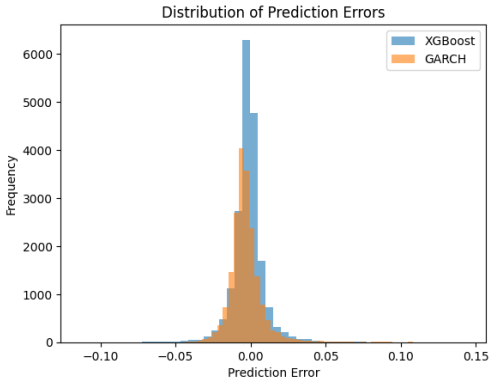


Figure 8: XGBoost vs GARCH: Error Analysis

4.4 RNN (LSTM)

4.4.1 Evaluation Metrics

For the LSTM model, we used multiple evaluation metrics to assess predictive accuracy, including Root Mean Squared Error (RMSE), R^2 score, Mean Absolute Percentage Error (MAPE), and Directional Accuracy. RMSE was the primary metric for comparability with other models. Our final LSTM model achieved an RMSE of approximately 0.0070 on the test set, significantly outperforming both the GARCH (0.0115) and XGBoost (0.0110) baselines. For this model, we found **RMSE:** 0.0070, **R^2 Score:** 0.7835, **MAPE:** 34.06%, **Directional Accuracy:** 67.46%

4.4.2 Main Findings

The LSTM model effectively captured temporal dependencies between lagged sentiment, volume, and historical volatility. As shown in Figure **??**, the predicted volatility closely tracks the true values over time. The model successfully learned to follow the trend and magnitude of volatility, although it showed a tendency to under-predict during extreme spikes. In the scatter plot (Figure **??**), the predicted volatility values align closely with the 45-degree line, indicating high correlation between predicted and actual values.

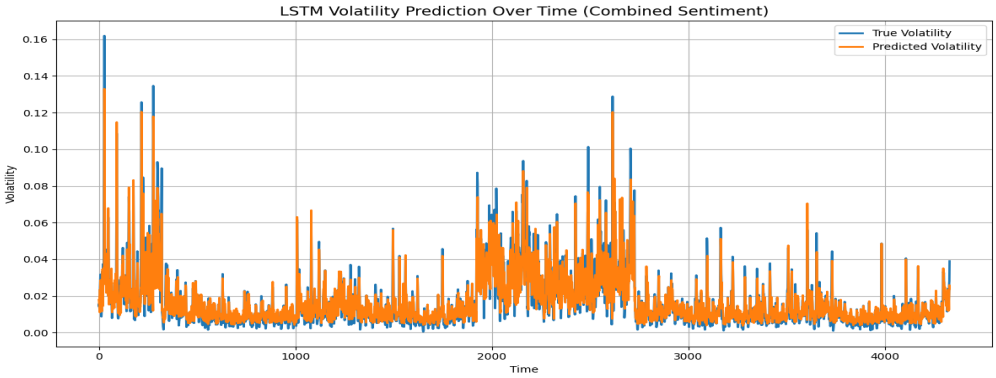
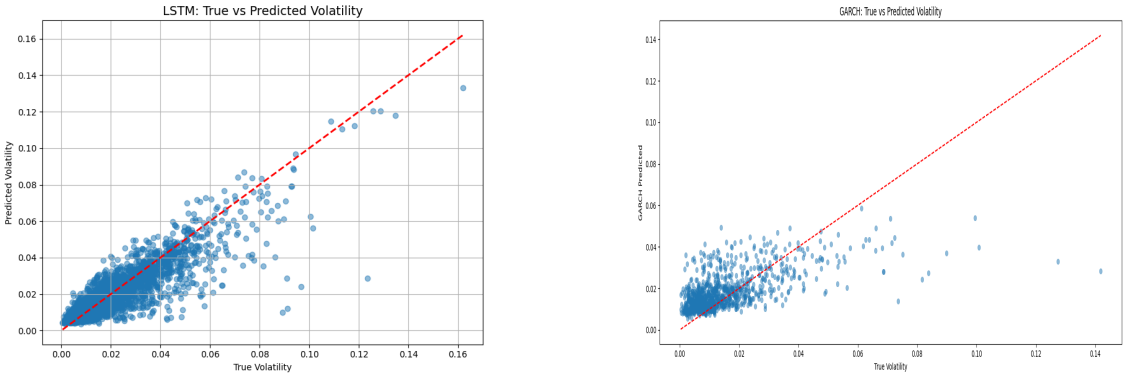


Figure 9: RNN (LSTM) Volatility Prediction Over Time (Combined Sentiment)



(a) LSTM (b) GARCH

Figure 10: True vs. Predicted Volatility: LSTM vs. GARCH

4.4.3 Robustness & Overfitting

The model was trained over **300** epochs using an LSTM with three hidden layers, each containing 128 units and dropout of **0.3**. The training loss consistently declined from **0.90** to **0.23**, with no signs of overfitting based on test RMSE and evaluation plots. We used a time-aware **80/20** train-test split (with `shuffle=False`) to ensure chronological order and avoid look-ahead bias. All features and target values were normalized using `MinMaxScaler` and `StandardScaler` to stabilize training.

Future improvements could involve early stopping based on test RMSE or implementing validation splits for model checkpointing during training. Additionally, experimenting with alternate architectures (GRUs or attention) could further boost robustness.

4.4.4 Error Analysis

Although the model performed strongly overall, MAPE still remained relatively high (**34.06%**), indicating that the model occasionally misestimated the absolute scale of volatility, particularly during volatile periods. This is consistent with what is observed in the scatter plot, where some points deviate significantly from the ideal prediction line. A residual analysis could help identify systematic error patterns, like consistent underprediction during market shocks or increased variance during low volume liquidity time periods. This would significantly enhance some performance.

5 Conclusion

5.1 Summary of Findings

Our comparative analysis of three models: Random Forest, XGBoost, and LSTM to the baseline GARCH model yielded many key insights. Our results show that machine learning models, especially LSTM, out-

performed the traditional GARCH model in predicting next-day stock volatility. The LSTM and XGBoost achieved the lowest RMSE with 0.0070 and 0.0069 respectively, both improvements from the baseline GARCH model (0.0115). This is also seen in the Random Forest model, where sentiment increased F1 and ROC-AUC scores. While, sentiment was not the most important feature, it helped to create more accurate predictions. Thus, we found that including news sentiment and volume into our models consistently improved performance. Combining financial and alternative data sources led to better generalization, but models still underestimated extreme volatility spikes. Overall, machine learning and news data enhanced next-day volatility prediction beyond traditional methods.

Interestingly, our Random Forest model performed best on the 1-year dataset with both Reddit and HackerNews sentiment, rather than the 3-year dataset. We predict this is because Random Forest benefits more from rich, relevant features than from large sample sizes. It also lacks mechanisms to adapt to temporal drift or capture sequential patterns, making it less effective on longer, noisier data. In contrast, XGBoost and LSTM handled the extended time horizon better by adaptively weighting features or modeling temporal dependencies. This ability to leverage longer histories and capture evolving patterns is likely why these models performed best overall.

5.2 Lessons Learned

Through the development and evaluation of multiple models, we learned that while sentiment data can meaningfully improve next-day volatility prediction, it requires careful preprocessing due to its noisy nature. Lag features were essential to avoid data leakage and ensure the models reflected realistic forecasting conditions since you cannot predict the next day until the end of the current day, but using lag features allows you to predict next day while in the current day. Overall, combining traditional financial metrics with sentiment data added predictive power, but it depends heavily on thoughtful feature engineering and robust model validation.

5.3 Future Work

Building on the findings of this paper, further analysis will be conducted to investigate why Random Forest performed better on the 1-year dataset and why XGBoost and LSTM outperformed Random Forest. This deeper analysis could guide the development of an even better model. To improve model performance, future work could incorporate intraday data to capture short-term volatility patterns and utilize options-based metrics like implied volatility for a forward-looking view [3]. Advanced LLMs such as FinLlama offer zero-shot sentiment analysis, enabling richer sentiment features without labeled data [5]. Additionally, order flow and options data can provide a more detailed ground truth for market dynamics [1]. Finally, deploying the model in a backtesting pipeline would help evaluate the practical effectiveness of volatility forecasts through simulated trading performance [6]. There are many ways to move forward with our findings to create an even better model.

References

[1] Cheddar Flow. (2024, January 23). *Options order flow: Everything you need to know*. https://www.cheddarflow.com/blog/options-order-flow-everything-you-need-to-know/

[2] Engle, R. (n.d.). *GARCH 101: An Introduction to the Use of ARCH/GARCH models in Applied Econometrics*. https://web-static.stern.nyu.edu/rengle/GARCH101.PDF/

[3] Northwestern. (n.d.-a). *Intraday Volatility Patterns from Short-Dated Options*. https://www.kellogg.northwestern.edu/faculty/todorov/htm/papers/odp.pdf

[4] NVIDIA. (n.d.). *XGBoost*. NVIDIA Glossary. https://www.nvidia.com/en-us/glossary/xgboost/

[5] Restackio. (n.d.). *Zero-shot learning applications in Finance*. https://www.restack.io/p/ai-in-finance-answer-zero-shot-learning-finance-cat-ai

[6] Stepanenko, A. (2024, October 24). *Step four- optimizing trading strategies: Stages of the backtesting pipeline*. Medium. https://medium.com/@inbox.artem/step-four-optimizing-trading-strategies-stages-of-the-backtesting-pipeline-9862