The Challenges of Volatility Forecasting

John Wu

Abstract

Volatility forecasting is crucial for financial decision-making, influencing investment strategies and risk management. Traditional models like Black-Scholes assume constant volatility, which fails to capture real market dynamics. Our project aims to explore volatility prediction by analyzing traditional models like GARCH with advanced machine learning techniques such as LSTMs and Transformers.

Using two decades of S&P 500 data, we explore various volatility measures, including Close-to-Close, Parkinson, and Yang-Zhang, to predict 3-day rolling volatility. Despite challenges with model performance and data biases, our approach tries to integrate autoregressive data, regularization methods, and k-fold cross-validation to enhance prediction accuracy.

Analyzing the performances of these models and the dataset underlying volatility modeling reveals challenges unique to this problem that traditional models may struggle to solve. Thus this project also sets the stage and proposes a path towards more sophisticated volatility forecasting models.

1 Goals and Objectives

Volatility forecasting plays a pivotal role in financial decision-making, influencing everything from investment strategies to risk management. In our Investments class, we're taught the Black-Scholes model, which assumes constant volatility across all strikes and maturities. However, this assumption is a significant limitation since real-world volatility is far from constant. To make financial models more practical and accurate, we need a dynamic and reliable volatility forecasting model.

Predicting market volatility is inherently challenging. Markets are driven by human behavior, which is often irrational and unpredictable. Volatility itself is non-stationary, meaning its patterns change over time due to various factors like economic events, market sentiment, and geopolitical developments. This makes accurate forecasting difficult. Our prediction strategy is autoregressive, meaning it uses past volatility indicators to predict future ones, but this can lead to multi-collinearity problems where predictor variables are highly correlated, complicating the model's accuracy.

The primary objective of this capstone project is to implement and evaluate several traditional and modern volatility modeling techniques. We'll explore models like Generalized Autoregressive Conditional Heteroskedasticity (GARCH). These models are widely used in financial forecasting and offer various methods for capturing and predicting volatility patterns.

In addition to these traditional techniques, we aim to investigate advanced machine learning approaches. These include Long Short-Term Memory networks (LSTMs) and Transformers, which are well-suited for time-series data due to their ability to capture long-term dependencies. By combining traditional econometric models with modern deep learning techniques, we hope to develop a more robust and accurate volatility forecasting model.

Furthermore, we plan to incorporate regularization methods such as Ridge and Lasso regression to address overfitting issues commonly encountered with volatility data. This will help ensure that our models generalize well to unseen data. Finally, we will evaluate the performance of our models using a variety of metrics, including Mean Squared Error (MSE) and R-squared, to determine their predictive accuracy and reliability.

2 The General Setting

2.1 Data Collection and Overview

In our research, we leveraged two decades of financial data from Yahoo Finance, focusing on the S&P 500 index. This dataset was selected for its comprehensive coverage and accessibility, providing detailed information on stock prices, including Open, Close, Low, High, and Adjusted Close values. The dataset's extensive nature allows for a thorough examination of volatility patterns over a significant period, enhancing the reliability and relevance of our findings. While the dataset's breadth is a significant advantage, it is crucial to acknowledge and address inherent biases:

1. **Stock Tenure Variation Bias:**
   A notable bias arises from the varying lengths of time different stocks have been listed in the S&P 500. Some stocks, particularly newer entrants, have shorter historical data compared to their longer-standing counterparts. This variation can potentially skew analyses, particularly in time-sensitive volatility modeling.

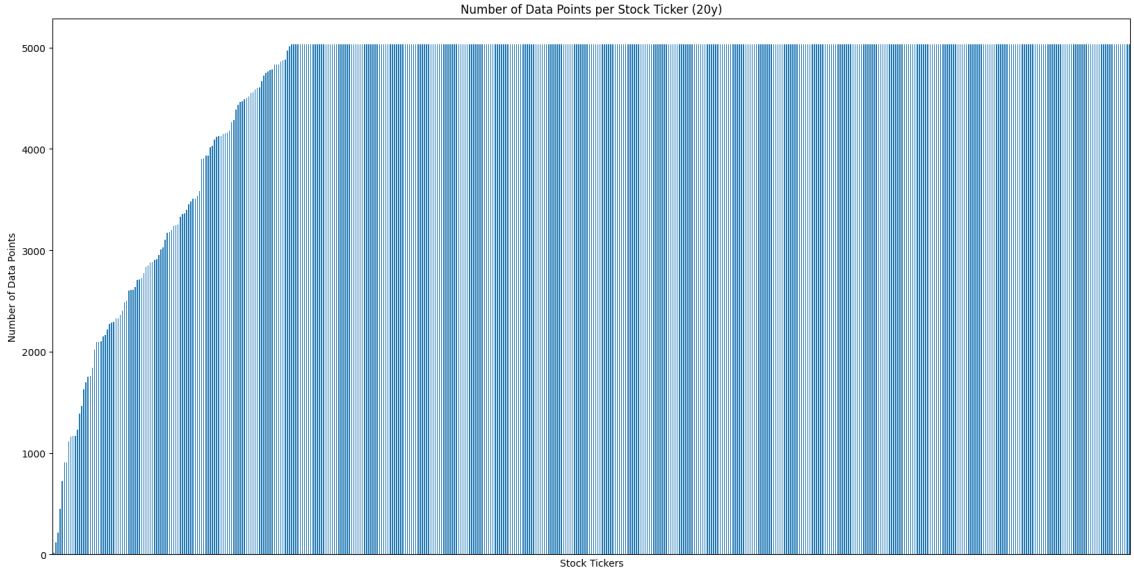


Figure 1: A plot displaying the amount of datapoints per stock ticker for the last 20 years. As you can see there is a good chunk of stock tickers which will have small representation in the dataset

2. **Sector Distribution Bias:** Another layer of bias stems from the asset class distribution within the S&P 500. To quantify and visualize this bias, we constructed a pie chart illustrating the proportion of data corresponding to each sector. This representation aids in understanding how sectoral dominance might influence overall volatility modeling results.

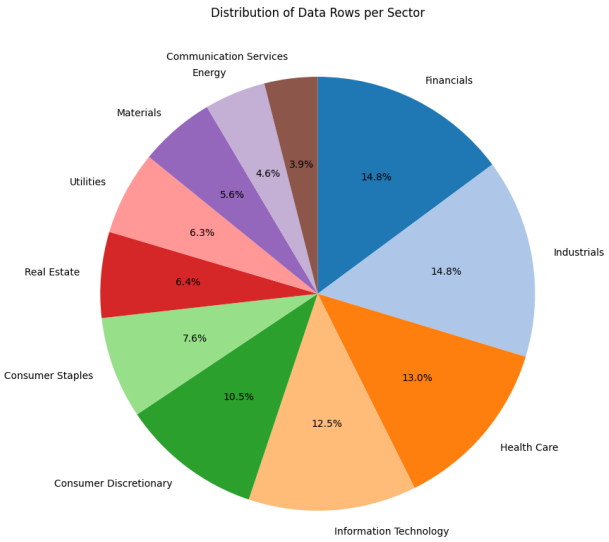


Figure 2: A plot displaying the distribution of stock data corresponding to each overall sector according to Wikipedia

2.2 Challenges and Volatility Calculations

When working with this dataset, I encountered several challenges, particularly due to the extensive training times required by its large scale when including every stock in the S&P 500. To address this, I streamlined the dataset to focus on the S&P 500 index as a whole, rather than individual stocks. However, I was able to generate the graphs shown above before making this switch.

Another challenge was determining the true volatility from the data. To tackle this, I used a few historical volatility indicators as benchmark values, utilizing OHLC (Open, High, Low, Close)

data over three days. The objective of our model is to predict the 3-day rolling volatility based on the following volatility measurements:

1. **Close-to-Close**: This model estimates volatility using the closing prices of the stocks. It calculates the daily returns based solely on the closing prices from one day to the next, providing a straightforward measure of price fluctuations over time.

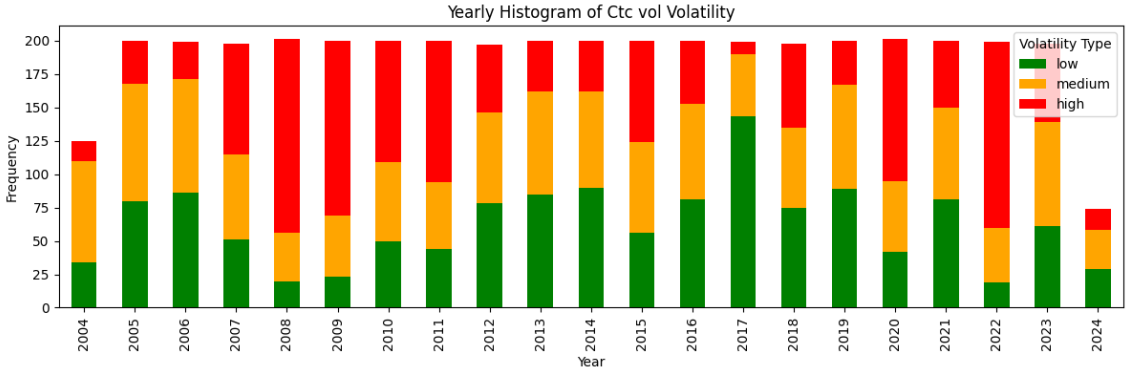
2. **Parkinson**: Utilizes the high and low prices of the day to estimate volatility. By considering the range between the highest and lowest prices each day, this model offers a broader perspective on price movement compared to the Close-to-Close model, capturing intraday volatility more effectively.

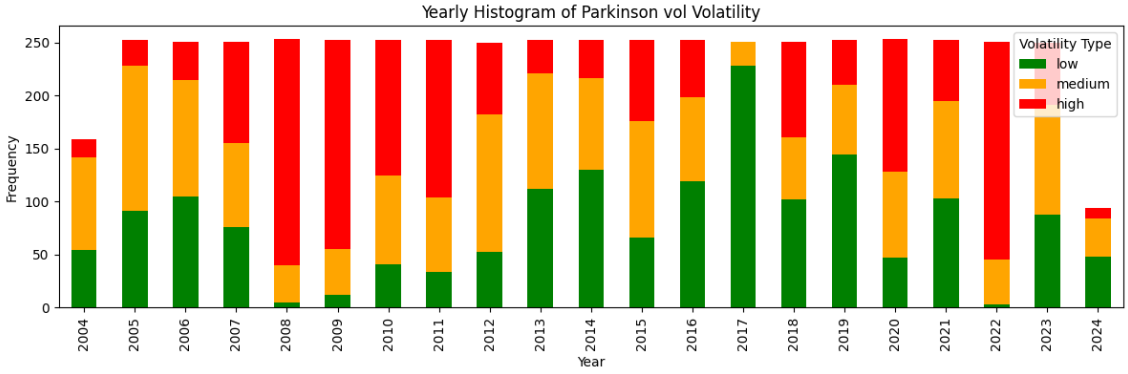
3. **Garman-Klass**: A more sophisticated approach that combines opening, closing, high, and low prices for a comprehensive volatility estimation. This method incorporates both intraday and interday price variations, enhancing the accuracy of volatility predictions by leveraging a wider set of price data points.

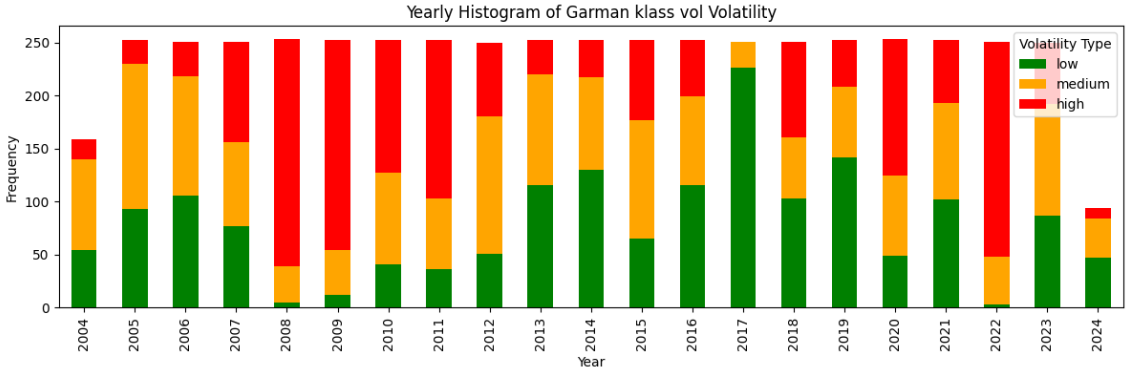
4. **Roger-Satchell**: Considers the opening, closing, high, and low prices, emphasizing the relationships between these price points. This model is particularly useful for capturing the directional movements and volatility patterns within each trading day, providing a nuanced view of price dynamics.

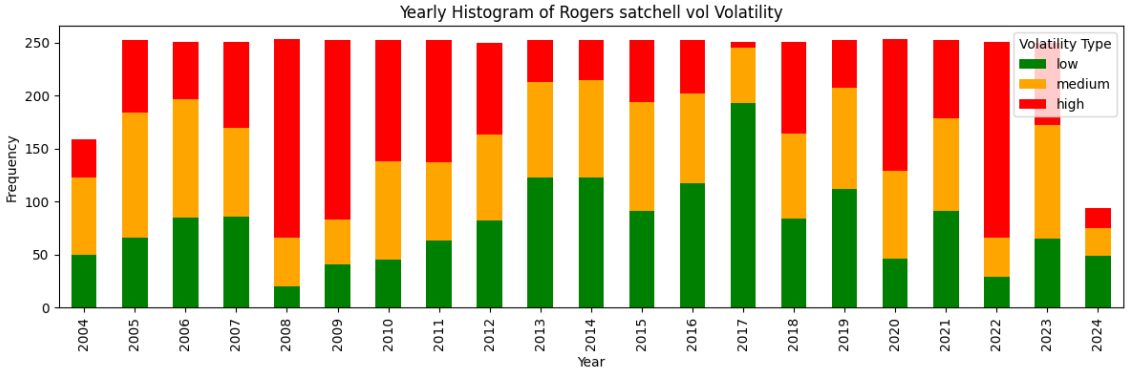
5. **Yang-Zhang**: An advanced model that incorporates overnight price moves, making it highly relevant for markets with significant overnight price variations. By accounting for both the close-to-open and open-to-close returns, this method offers a robust volatility measure that is sensitive to overnight information and price adjustments.

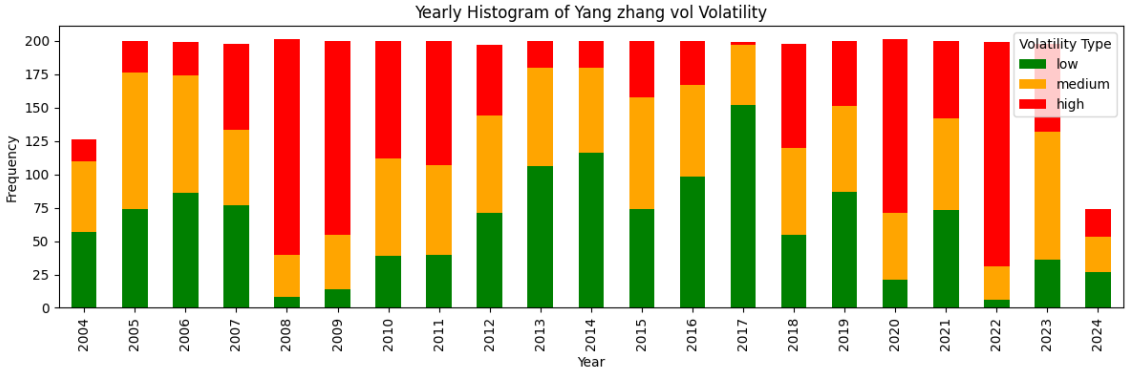
Notably, no matter what measurement of volatility that we use the dataset itself seems to contain a couple of regime shifts do to unprecedented events happening within the last 20 years. Specifically when we analyze periods of low, medium and high volatiltiy based on the different quartiles of volatility in the data we see that the years around 2008 are characterized by unprecedented high volatility and similarly in 2022 there was also higher volatility after interest rates rose given the effects of Covid19. Thus these regime-shifts and outlier events could pose a challenge for any machine learning approach that we throw at the problem due to inconsistency in patterns in the data.


Yearly Histogram of Ctc vol Volatility
Volatility Type
low
medium
high
Frequency
Year


Yearly Histogram of Parkinson vol Volatility
Volatility Type
low
medium
high
Frequency
Year


Yearly Histogram of Garman klass vol Volatility
Volatility Type
low
medium
high
Frequency
Year


Yearly Histogram of Rogers satchell vol Volatility
Volatility Type
low
medium
high
Frequency
Year


Yearly Histogram of Yang zhang vol Volatility
Volatility Type
low
medium
high
Frequency
Year

3 Methods

To deal with autoregression in the data we generate training folds with 20 datapoints of past volatility and data to train on and we try to predict the "3-day" volatility of next 3 datapoints (aka the measured volatility 3 days after the last data point in our training fold). Then to prevent any data leakage and look ahead bias we add a buffer of 10 days to the dataset and then take the gathered time-ids and shuffle them before applying k-fold cross validation on these shuffled time-id folds.

3.1 The Kitchen Sink

Unfortunately, all of this analysis in the previous sections didn't bode well for the actual attempts to solve the problem itself when using previous auto-regressive data to try and predict future data and most of them were done after the fact to try and figure out what was going on.

I experimented with various traditional autoregressive models such as HIST (Historical Average), EWMA (Exponentially Weighted Moving Average), and GARCH (Generalized AutoRegressive Conditional Heteroskedasticity), which typically perform well on autoregressive data. Additionally, I tested traditional machine learning models like Decision Trees and Linear Regression, as well as more advanced models such as LSTMs and Transformers, which are often used for autoregressive data. To address overfitting, I incorporated Ridge and Lasso regression models. However, I found that regression was not effective due to reasons outlined later.

3.2 Results

Unfortunately these models performed fairly horribly within the time-id fold testing framework that I set up likely due to many of the challenges highlighted earlier and also perhaps due to a lack of data and features to use during prediction. By reducing the training window of auto regressive data to 20 days we were able to collect more datapoints but the number of features that we could use for predictions was perhaps under-specified. Thus with the under specification of features, it's likely that the models were necessarily over fitting to the training data in a way that cannot be regularized. This over fitting is evident even in the performance of the Linear Regressor and Decision Tree models.

Here is a table and plot showing the results of each of these models on the given volatility modeling task.

Table 1: Model Performance Metrics

Model	Train Errors	Test Errors	Train R^2 Scores	Test R^2 Scores
ctc_vol				
Linear	8.2224e-34	8.4238e-05	1.0	-1.1201
Decision Tree	0.0	6.2686e-05	1.0	-0.8236
Ridge	9.2307e-05	4.4274e-05	0.1103	-0.1066
Lasso	1.0480e-04	4.6463e-05	0.0	-0.1726
LSTM	9.8192e-09	8.3637e-05	0.9999	-1.1021
Transformer	2.0848e-04	1.8912e-04	-1.6743	-3.7394
Historical Average	4.9390e-05	3.8268e-05	0.5084	-0.0352
EWMA	3.6993e-05	4.2993e-05	0.6455	-0.1405
GARCH	1.6921e-04	1.1443e-04	-0.6908	-2.3277
parkinson_vol				
Linear	1.2099e-33	1.9814e-05	1.0	-0.0962
Decision Tree	0.0	1.1463e-05	1.0	0.1512
Ridge	3.1827e-05	2.0901e-05	0.0806	-0.1760
Lasso	3.4971e-05	2.2227e-05	0.0	-0.2566
LSTM	3.7831e-08	1.9627e-05	0.9990	-0.0827
Transformer	7.4303e-04	7.1192e-04	-18.9865	-46.8897
Historical Average	1.3663e-05	1.1643e-05	0.5563	0.2297
EWMA	7.3200e-06	1.3198e-05	0.7592	0.1620
GARCH	152.6572	151.8628	-6909886.7541	-22898837.1820
garman_klass_vol				
Linear	3.4033e-33	2.5939e-05	1.0	-0.1359
Decision Tree	0.0	1.3461e-05	1.0	0.2102
Ridge	3.9322e-05	2.4434e-05	0.0960	-0.1672
Lasso	4.4030e-05	2.6343e-05	0.0	-0.2644
LSTM	1.0757e-07	2.4023e-05	0.9956	-0.0840
Transformer	7.7879e-05	5.5521e-05	-0.7678	-2.6309
Historical Average	1.7442e-05	1.5589e-05	0.5480	0.1847
EWMA	9.3304e-06	1.7659e-05	0.7554	0.1180
GARCH	0.0231	0.0229	-843.5670	-2661.2849
rogers_satchell_vol				
Linear	4.3659e-34	3.4610e-05	1.0	-0.8399
Decision Tree	0.0	2.0674e-05	1.0	-0.2977
Ridge	2.4528e-05	1.7212e-05	0.0428	-0.1022
Lasso	2.5737e-05	1.7589e-05	0.0	-0.1275
LSTM	8.2222e-09	3.4713e-05	0.9997	-0.8426
Transformer	7.0405e-05	7.0890e-05	-2.5844	-3.4086
Historical Average	1.1739e-05	1.9596e-05	0.5055	-0.1570
EWMA	5.9355e-06	2.3254e-05	0.7431	-0.3157
GARCH	0.2586	0.2575	-13050.0405	-19461.2907
yang_zhang_vol				
Linear	2.4212e-34	3.1781e-05	1.0	-0.9163
Decision Tree	0.0	1.8532e-05	1.0	-0.1900
Ridge	3.2615e-05	1.8430e-05	0.0433	-0.2371
Lasso	3.4211e-05	1.8892e-05	0.0	-0.2732
LSTM	1.3263e-06	2.6666e-05	0.9458	-0.6448
Transformer	1.1475e-04	9.6840e-05	-2.1163	-4.7156
Historical Average	1.7297e-05	1.6250e-05	0.4557	-0.0069
EWMA	1.0380e-05	1.7079e-05	0.6685	-0.0706
GARCH	0.0559	0.0523	-2113.8051	-2476.4569

Unfortunately, given these results, I did not attempt to ensemble the learners with a meta-model. I believe each individual model performed so poorly that the meta-model would not have been able to learn much from them. Instead I attempted to work on refining and trying out more models that may have given better results and refining my pipeline infrastructure.

4 Future Work and Challenges

Unfortunately, I ran out of time before I could achieve notable results on this project. The primary obstacle was the infrastructure. Many of the more successful approaches in Kaggle competitions use different infrastructures for setting up their time-IDs, collecting and refining features, and selecting and summarizing features for training. When trying to compile and analyze these different results, my greatest challenge—and perhaps failing—was initially trying to design a pipeline and infrastructure that could support every different model in a standardized way. This approach was overly ambitious and ultimately hindered progress due to scope creep. Instead, creating a simple pipeline (like I attempted to do in my second attempt) for each model separately, allowing for unique engineering of features for each model, would have been more effective.

Another approach I could have taken was to use the data provided and prepared in a Kaggle competition. However, in the interest of using "real data" and exploring this problem naturally, I focused on building infrastructure. Unfortunately, this approach limited the exploration of more advanced techniques that could have been more interesting and productive.

For future work, it would be beneficial to collect as many relevant signals as possible and then apply feature engineering and feature selection techniques commonly used in Kaggle competitions. This approach could help refine a model to perform better on the data, as our analysis has shown that auto-regressive data alone is likely too complex to be solved using traditional techniques. Once we have a collection of weak learners with at least positive R-squared values, we can explore using ensemble and boosting techniques as effective meta-models to combine information from each of these models, as originally proposed in the project. However, this highlights the infrastructural challenges and suggests that unfortunately more time might be needed for this project than what I managed to allocate within one semester, alongside other projects.

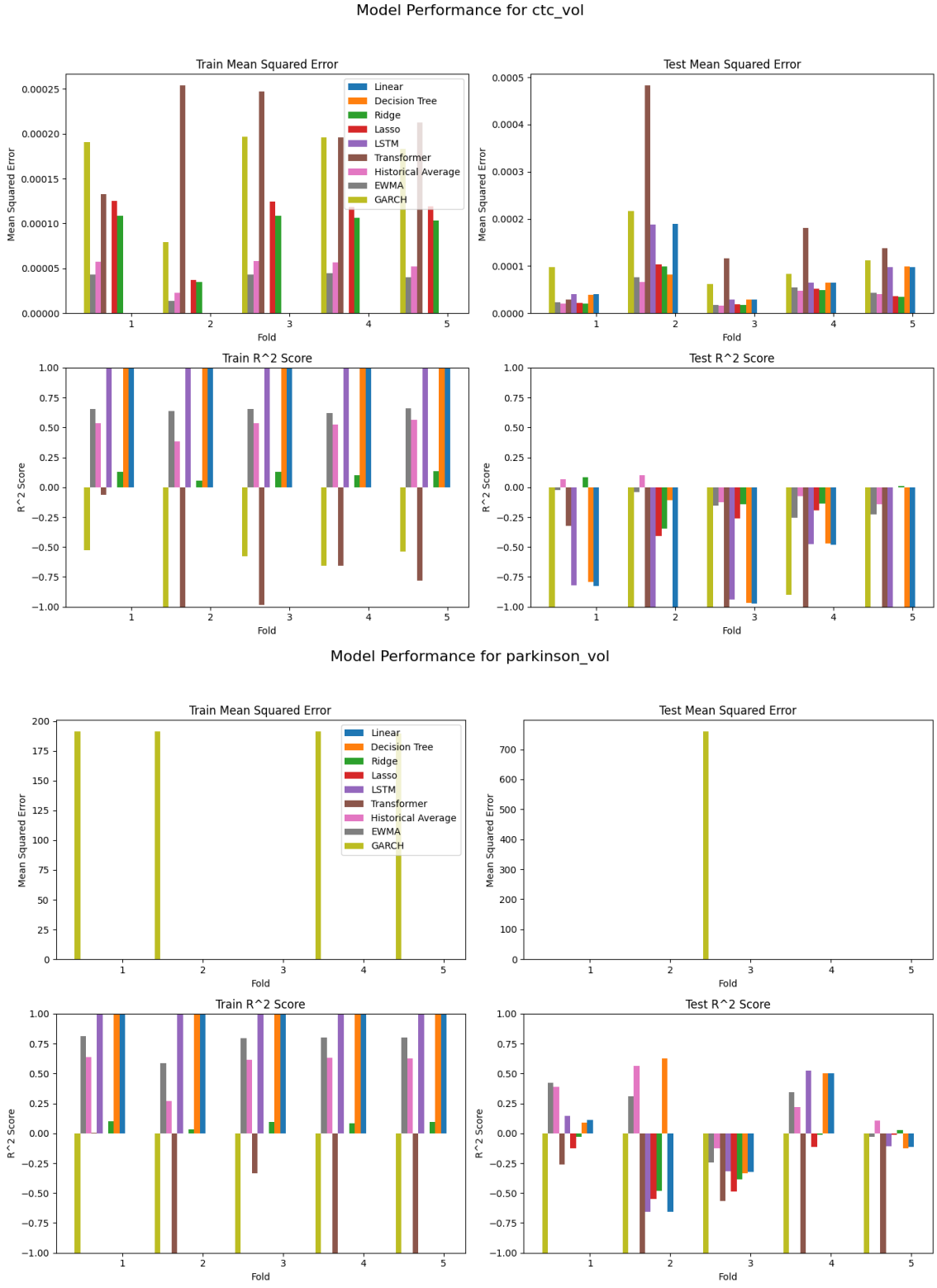


Figure 3: Performances of all the different models in the Kitchen sink on volatility modeling (Part 1)

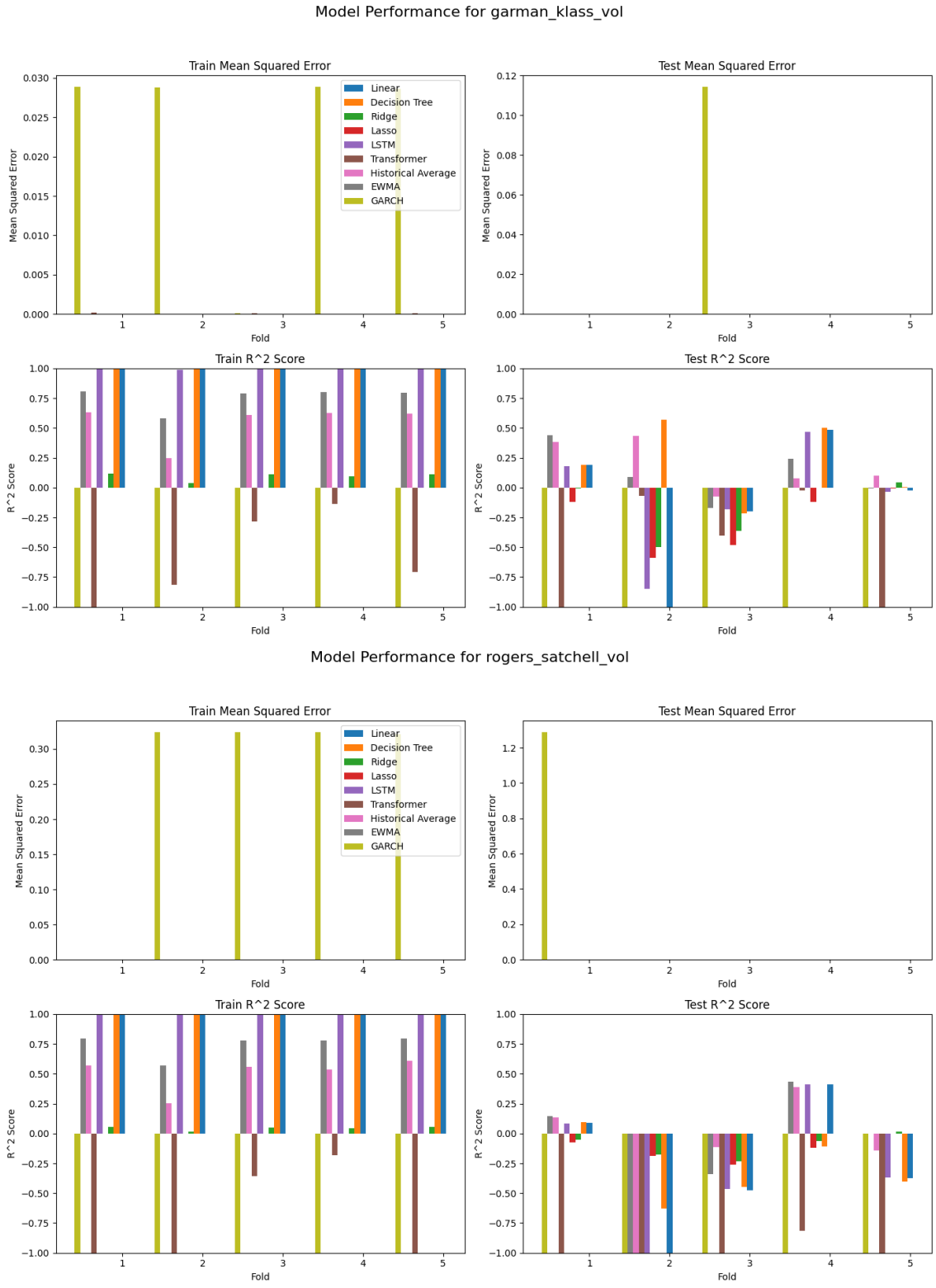


Figure 4: Performances of all the different models in the Kitchen sink on volatility modeling (Part 2)

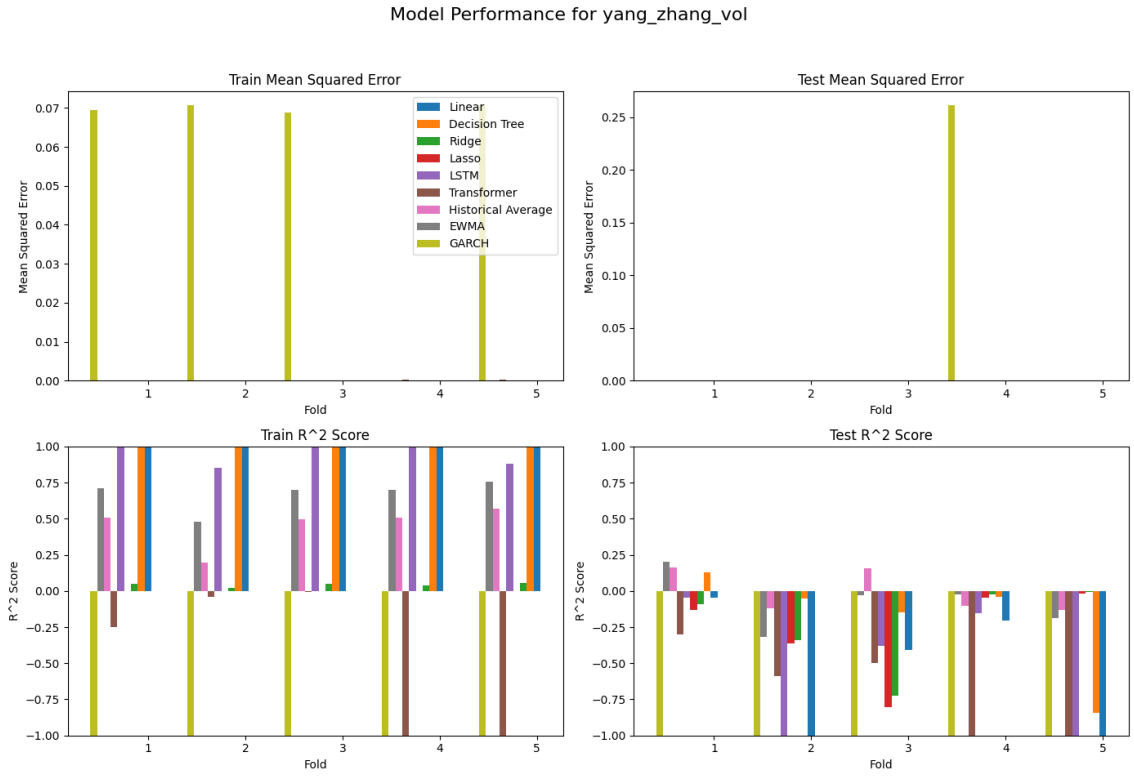


Figure 5: Performances of all the different models in the Kitchen sink on volatility modeling (Part 3)