Forecasting Realized Volatility for Crude Oil Futures with High-Frequency Data

Soren Dunn[1*] **Noah Fischer**[1*]
[1]University of Chicago
sorendunn@uchicago.edu
nfischer@uchicago.edu

Abstract

This paper investigates approaches for nonparametric forecasting of realized volatility as measured by n-period historical standard deviation of log returns for oil futures contracts on 5-minute resolution data. Kernel Ridge Regression, Gradient Boosting, Random Forest, K-Neighbors, and Nadaraya-Watson regressors were used for prediction. Our best-performing model achieved 13.37% MAPE on volatility predictions up to 8 hours in the future. Inputs used for these regressors included futures contract returns as well as lagged versions of various variability and jump tests for the futures' returns drawn from literature.

1 Introduction

The problem of volatility estimation in financial markets is crucial. Accurate estimation of volatility is a requirement for the pricing of derivatives such as options as well as crucial for speculative investment. The ability to forecast future volatility can lead to tremendous advantages in almost all markets, as derivatives can be priced faster than the market, underlying assets can be traded with a knowledge of future price action, and portfolios can be adjusted to maintain uniform risk.

Volatility is often considered to be an unobserved latent variable. Theoretically, volatility is an instantaneous variable. The theoretically correct way to find true volatility would be to consider integrated volatility. However, actual financial data has a discrete timescale.

One approach to deriving volatility from real financial data is by using the notion of Implied Volatility. To derive Implied Volatility, volatility is backwards calculated from an option pricing model (such as the Black-Scholes model). This method of calculating volatility underlies volatility indices such as the Chicago Board Options Exchange's Volatility Index (VIX). Since we were unable to acquire the requisite data to feed into an options pricing model we did not use Implied Volatility as our notion of volatility.

A different approach to deriving a notion of volatility from real financial data is by approximating integrated volatility by the sum of log returns from time t_1 to t_2. This measure of volatility is termed Realized Volatility and given by:

$$\widehat{\sigma}_{RV} = \sqrt{\sum_{t=t_1}^{t_2} log\left(\frac{P_t}{P_{t-1}}\right)}$$

where P_t is the price at time t, as discussed in Ge et al. [2022].

Common methods of forecasting realized volatility, or the square of realized volatility termed realized variance, are generalized autoregressive heteroscedastic (GARCH) models as well as newer neural network or deep-learning based approaches. GARCH models are models of the form:

Preprint. Under review.

$$\sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

where α_0 is the mean, α_1 and β_1 are fitted coefficients, y_{t-1} is the value being predicted at time t - 1 and σ_{t-1}^2 is the realized variance of the model at time t - 1. The specific equation given is for the simpliest form of the GARCH model termed GARCH(1,1). Variations of GARCH models were the standard for volatility prediction for many years. However, GARCH-type models make fundamentally linear assumptions about auto-regressive patterns which Lanne and Saikkonen [2005] and others have questioned under concerns that they adapt more slowly to quickly changing market conditions (especially in the aftermath of shocks). Instead, we trained several non-parametric methods as well linear model comparisons on both lagged price and realized volatility values as well as derived features. Using this approach we were able to achieve mean absolute percent error (MAPE) of 13.37% which compares favorably with related work.

2 Related Work

As financial prediction is a highly lucrative field, there is a vast amount of work that has gone into volatility prediction both inside and outside academia. Unfortunately, incentives do not favor sharing of industry results, so we looked to academic literature for related work.

Work on futures volatility prediction vary along several dimensions: the granularity of data used, the horizon of prediction, the sources of data used, the volatility measure being predicted, and the performance criteria utilized. Tables 1 and 2 detail the setup and performance of nine notable academic papers on this subject (see Ge et al. [2022]). Asterisk indicate that the corresponding column could not be determined from the paper.

Table 1: Previous Work: Methods

Authors	Year	Method	Data set	Period
Doering et al. [2017]	2017	CNN	London Stock Exchange	2007-2008
Nikolaev et al. [2013]	2001	MM, RNN, GARCH	Call/put options	1991-1998
Xiong et al. [2015]	2016	LSTM	S&P500	2004-2015
Zhou et al. [2019]	2019	LSTM	Baidu search data	2006-2017
Kim and Won [2018]	2017	LSTM + GARCH	Interest rates	2001-2011
Nikolaev et al. [2013]	2012	RMDN-GARCH	DEM/GBP exchange rate	*
Psaradellis and Sermpinis [2016]	2016	HAR-GASVR	VIX, VXN, VXD	2002-2014
Kang et al. [2009]	2008	GARCH-type models	Crude oil spot prices	1992-2006
Sadorsky [2006]	2006	GARCH-type models	Future prices	1988-2003

Table 2: Previous Work: Results

Authors	Lag	Horizon	Performance Criteria	Best Performance
Doering et al. [2017]	*	14 d	Accuracy, kappa	0.223 kappa
Nikolaev et al. [2013]	10 obs	*	Daily profit mean	Daily profit mean 0.1
Xiong et al. [2015]	*	1 d	MAPE, RMSE	MAPE 24.2%
Zhou et al. [2019]	5 d	5 d	MSE, MAPE	MAPE of 17%
Kim and Won [2018]	22 d	1 d	MAE, MSE	MAE of 0.01069
Nikolaev et al. [2013]	*	*	NMSE, NMAE	NMSE 0.7562
Psaradellis and Sermpinis [2016]	5 d	1 d, 22 d	MAE, RMSE	MAE 0.03
Kang et al. [2009]	1 d	1 d, 5 d	MSE, MAE, DM	1 d MAE 2.88
Sadorsky [2006]	20 d, 60 d	1 d	MSE, MAD	150 MSE

Research on this topic largely varies along two general axes. First, earlier work tended to use GARCH-derived models due to lower computation costs and newer work tends to utilize largely neural-network or deep learning based approaches. A break from this trend is Nikolaev et al. [2013] which used daily price and option data to predict volatility changes with Markov Models (MM) and Recursive Neural Networks (RNN) as early as 2001. An example of a more recent paper is Doering et al. [2017] who used complete message history from 2007 and 2008 for the London Stock exchange along with convolutional neural networks to achieve kappa performance of 0.223 on the next 20 bid/sell events, demonstrating fair ability to predict market volatility microstructure.

On the other end of the spectrum are models (largely GARCH derived) which use and predict data with a daily resolution. A typical example of this is Kang et al. [2009] who used previous data price data from crude oil spot prices to predict next day and next d day volatility. Their models achieved mean absolute error of 2.88 on one day out prediction.

We chose to use 5-minute data as it was the highest frequency reasonably clean data we had access to. Even though many papers used daily data, our goal was shorter-term predictions so higher resolution data made more sense. Similar to GARCH-type models we used lagged realized volatility along with lagged returns as major inputs into our models, but looked to increase the flexibility of the methods by using nonparametric instead of linear approaches.

Since most of these papers either have different prediction targets or data granularities than we use, few are suitable for direct comparison with our results. The most suitable for comparison are Zhou et al. [2019] and Xiong et al. [2015]. Zhou uses price information in conjunction with Baidu search data to train short-term memory networks (LSTM's) to predict 5 day out volatility with MAPE of 17%. Xiong et al. 2016 also uses LSTM's but combines price and Google Trends data to achieve 1-day-out predictive volatility MAPE 24.2%. These papers are relevant since they mirror our approach in how they showcase newer models, predict MAPE (which is comparable even when predicting on volatility of underlying data with different magnitudes), and aim to predict volatility on the order of days. Best-performing models by both authors significantly outperformed GARCH-type models (in the case of Zhou et al. [2019] achieving one-seventh of the GARCH-type model's MAPE). We train on 900-point chunks of data and test performance on the next 100 points. Due to the 5-minute data resolution, this results in use predicting out around 8 trading hours in the future which makes the closest comparisons in the literature papers that predicted one day out volatility, as done by Xiong et al. [2015].

3 Data

We use 5-minute resolution futures data from Dukascopy [1] from 2/11/2020 until 2/11/2023 and derive features from this time series. Our principal time series is the front month WTI Light US Crude Oil futures contract but we also obtain the Natural Gas, Bitcoin, and 10-Year US Treasury Bill contracts at the same resolution and date range. Below is a plot of the oil price time series, which we observe displays high variability in volatility:

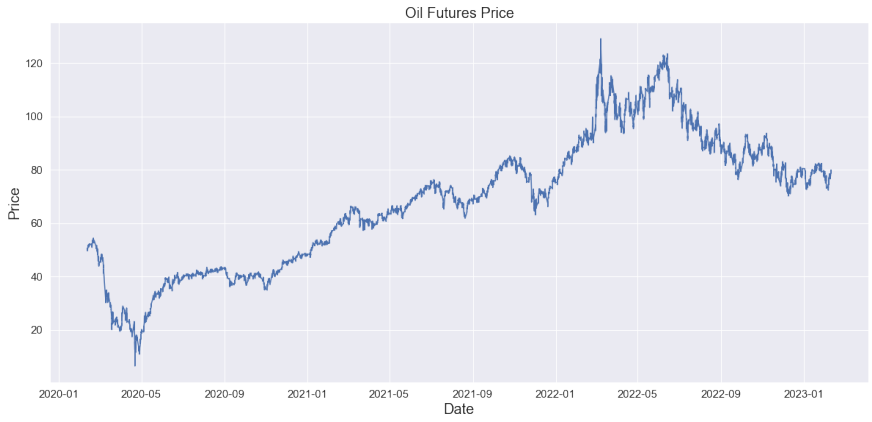

Figure 1: Oil Futures Price

3.1 Pre-Processing

In order to get the data into a format that would allow for the rolling predictions we intend to make, we used two different methods. The first was considering each day as a separate dataset, then training

[1] `https://www.dukascopy.com/swiss/english/marketwatch/historical/`

the suite of models on some proportion p of the day and predicting the remaining $1 - p$. We will call this the “day-by-day method". This method had a major problem in that each day did not contain sufficient data to perform a train-test split nor was it sufficiently long to construct meaningful outcome data. This method is however interpretable and forces predictions to occur at the same times each day. Considering the below plots of average log return and average realized historical volatility, we see that time of day has a significant effect on our features and outcome.

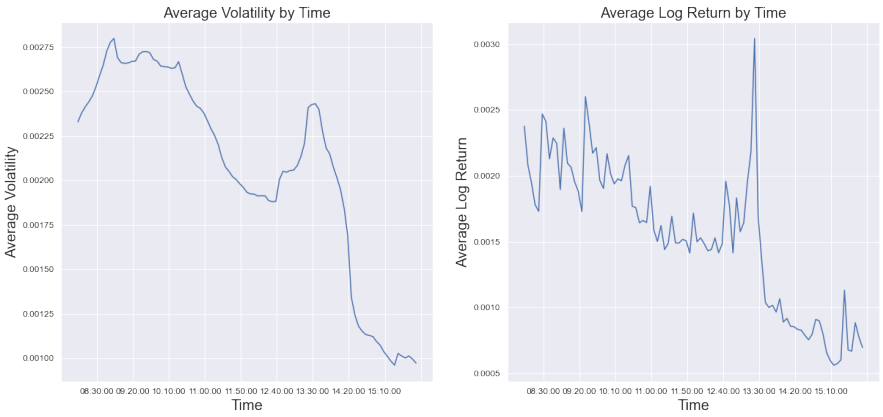

Figure 2: Time Dependency of Data

These temporal effects cannot be ignored, and later we will compare the “jumpiness" of this data processing method and see that it will not suffice for our analysis.

Our second method attempted to alleviate the above problems. The “chunk-by-chunk method" partitions the data into 1000-length chunks instead of daily ones. This allows for more control of the train and test fraction sizes and provides sufficient data on which we can train models. This also removes the day-by-day method’s inability to be generalized across prediction times. As the above plot shows, our outcome is dependent on time. Further, time series must be split linearly into training and testing sets. Therefore, the testing set under the day-by-day method will always comprise the same times resulting in models that cannot make good predictions for times outside of the testing set range since our outcome varies with time. We can visualize the difference in the methods’ creation of our outcome measure.

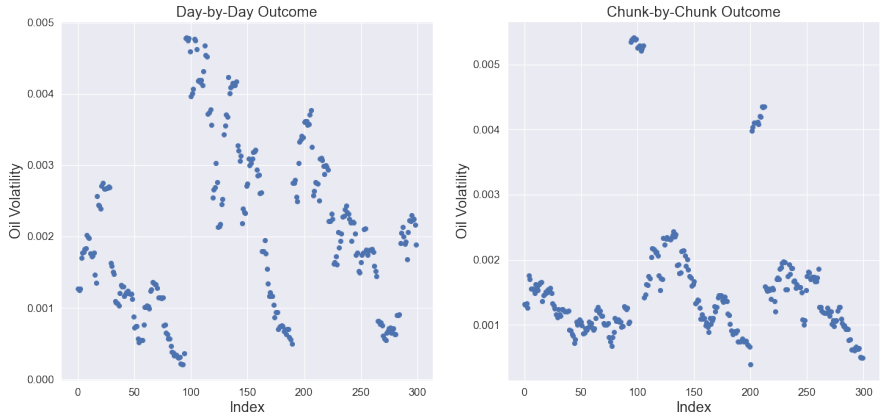

Figure 3: Preprocessing Method Comparison

While there are still jumps at the beginning of every day using the chunk-by-chunk method, the result is on average much more smooth than the day-by-day method. Thus, we elect to use the chunk-by-chunk method in our analysis.

3.2 Feature Creation

Numerous papers have been written on useful measures for predicting volatility. Some of these are tests for jumps, which are major sources of error in both our and many other financial datasets, and several have merely been empirically found to be useful in volatility forecasting. In order to leverage this research, we used several measures from the literature as features for our models. The most important of these predictors was the one-period lagged realized volatility.. Since tests for auto-correlation detected heavy levels of auto-correlation in the realized volatility having an equivalent lagged predictor was essential for ensuring model accuracy. Since Forsberg and Ghysels [2006] referenced several other papers finding that absolute returns empirically perform extremely well in predicting returns we also included absolute returns and standardized tri-power quarticity as features.

Barndorff-Nielsen and Shephard [2006] provided asymptotic distribution theory and finite sample tests of several nonparametric tests based on the use of bipower variation so it was included as feature in hopes of improving the models' performance at jump prediction. In a similar vein, Huang and Tauchen [2005] examined tests for jumps based on asymptotic results using Monte Carlo simulation and found that the daily ratio z-statistic has good power, size, and jump detection capabilities. Since we were not predicting by day we adapted their ratio z-statistic to our specific grouped setting.

As an additional measure of risk we also calculated the realized semivariance based on Barndorff-Nielsen et al. [2008]. They proposed realized semivariance as a measure of risk based on downward moves and demonstrated that it provided useful properties for predicting market volatility. Patton and Sheppard [2015] argued that future volatility was more strongly related to negative returns than positive returns, resulting in their proposed signed jump test which we used as an additional feature in hopes of further improving the produced models' predictions of jumps. The test can be formulated to detect either negative or positive variation and since we thought the different signs might exhibit different predictive behavior used both as model features. Lastly, Amaya et al. [2015] found a complicated relationship between realized skewness and kurtosis and future volatility. Their results indicated that stocks with negative skewness were more likely to exhibit higher future volatility and returns but as skewness increased, returns began to have a negative relation to volatility. Since our data included logarithmic returns we predicted that including skewness and kurtosis in our feature set might improve volatility predictions if our models picked up on the relationship Amaya et al. [2015] found between returns and skewness.

Feature creation is done within each chunk, including the creation of our outcome measure which is 12_oil_std or the rolling 12 data point (60 minute) realized standard deviation of oil future log returns. The $n = 12$ was chosen to smooth the outcome, however after testing several values for n the effect appeared negligible. Each of the above features is created by using the previous 12 datapoints. Since our data comes in 5-minute intervals, this usually represents the aggregate value for the feature from the last hour. Many of the features used were originally proposed for daily aggregation, but we hoped they would still be useful in the intraday setting.

We checked to ensure that there was no strong multicollinearity among our features. While many of the variance-derived features were correlated, we noted no concerningly high correlations and our results validated these claims. Our models did not exhibit strong feature lean due to overfitting. Additionally, all linear models used included penalization terms which helped reduce their variance in the presence of correlations among features.

4 Methodology

We chose to approach this problem in a relatively model-agnostic manner. Rather than selecting a single model, we use a wide array of nonparametric models (and parametric benchmarks). Hyperparemeter tuning for each model is done via grid search cross validation. This method leaves room for improvement in terms of finding the truly optimal hyperparemeter set, however it is computationally

efficient and allows for a wide breadth of hyperparemeters to be tested. To tackle this problem we performed the following algorithm which is provided in pseudocode below:

Chunk-by-Chunk Model Fitting and Prediction

```
procedure MODEL SELECTION
    for each data chunk do
        Train test split
        for each model in our set of models do
            Tune hyperparameters by grid search CV
            Take the best model and predict on the test data
            Save the best model and OOS accuracy statistics
```

4.1 Models

We chose 9 models to test. Our focus was on the performance of nonparametric models, however we included parametric models in order to benchmark our results. We will detail the expectations of each model. We believe that these nonparametric models are the proper tool for this forecasting problem since we have no assumption on the distribution of log returns or volatility, and further we have no assumption of linearity of the trends.

4.1.1 AdaBoost

$$G(\underline{x}) = sign(\sum \beta_m G_m(\underline{x}))$$

Adaptive boosting (**AdaBoost**) is a naive boosting model and a specific case of Gradient Boosting that optimizes exponential loss. As shown above, in the basic binary classification case AdaBoost is the sign of the sum of m weak learners G_m weighted in sequence by β_m which adaptively increases the weight of misclassified points. This model is computationally fast as it requires little hyperparemeter tuning. We do not expect AdaBoost to perform well on our data, since it is highly susceptible to outliers since its loss function is defined as an exponential and is therefore not robust. The distribution of volatility is very kurtotic, containing many outliers. Thus, we do not anticipate this model to perform well in terms of out of sample accuracy.

4.1.2 Gradient Boosting

Gradient Boosting has risen to stardom in most applied settings for its generally superior ability to predict high dimensional data out of sample. It takes the same basic form as AdaBoost, however Gradient Boosting allows for an arbitrary loss function to be used and optimized via gradient descent. The choice of learning rate here is important, and gradient descent is very computationally expensive. However, this model is robust and excellent at dealing with high dimensional and noisy data. While the model can overfit if given a poorly chosen learning rate, we account for this by allowing a wide range of learning rates in our grid search.

4.1.3 Random Forest

Random Forest models are excellent at dealing with outliers due to their large number of weak learners and both feature and data boostrapping. These models are extremely computationally intensive, leading us to limit the maximum number of trees to 2000. These models are also excellent at handling large datasets and are robust. We expect Random Forests and Gradient Boosting models to perform best in our task, and our later results will confirm this.

4.1.4 Lasso, Ridge, and Elastic Net

$$\hat{\beta} = \text{argmin}_\beta \|y - X\beta\|^2 + \lambda_2 \|\beta\|_2^2 + \lambda_1 \|\beta\|_2$$

Elastic Net regression is a parametric regression combining the two most popular regularization terms. L_2 or **Ridge** regularization is a shrinkage technique that shrinks the magnitude of the β coefficients. L_1 or **Lasso** regularization shrinks the L_1 norm of the coefficients to zero. Armed with both regularization techniques, this method is a more robust linear model, however it has the pitfalls of Lasso that often suffers from poor handling of high dimensional but small data. Note that Lasso

and Ridge regressions are special cases of the Elastic Net where the opposite regularization term is zero. As noted earlier, the penalizing terms present in these models help reduce the variance induced by correlated features.

4.1.5 Kernel Ridge Regression

$$\hat{f} = \text{argmin}_{f \in \mathcal{H}} \frac{1}{2} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2$$

Kernel Ridge Regression combines the Ridge regularization in the previous method with the kernel trick from Reproducing Kernel Hilbert Spaces. This model has excellent computational advantages due to the representation of these models as the sum of the product of some α weight and the kernelized data, thus allowing us to work in the Hilbert space of kernelized data rather than the raw data. We see that this model has shockingly good results for our problem.

4.1.6 Nadarya-Watson

$$\hat{m}_h(x) = \frac{\sum_{i=1}^{n} y_i \cdot K(\frac{x - x_i}{h})}{\sum_{i=1}^{n} K(\frac{x - x_i}{h})}$$

The **Nadarya-Watson** estimator is a local averaging estimator that takes the kernel density approximation K as a weighting scheme for the data within each bin of size h. The estimator depends heavily on the bandwidth h, and since we conduct only small cross validation via grid searching, we do not expect the best results from this estimator. The advantages with the Nadarya-Watson estimator are in the proper treatment of outliers and the locality, which creates an independence between far away outliers and noise on close data. Other than our choice of h, the Nadarya-Watson estimate in time series has a steep drop off in accuracy since out-of-sample data must be predicted from a bin that has no data overlap. In other words, out of sample data is being predicted from pre-trained kernel density estimates and therefore as we test further away from the last training data, the outdated kernels are still used for prediction

4.1.7 *k*-Neighbors Regression

$$\hat{f}(x) = \frac{1}{k} \sum_{j=1}^{n} y_j 1_{j \in \mathcal{N}(\rangle)}$$

The k**-Neighbors Regression** takes a uniform average over the k nearest (by norm distance) points to x. This neighborhood is represented by $\mathcal{N}(i)$ as the k closest points to point i. This model suffers heavily from the curse of dimensionality. That is, as the problem increases in dimension, the distances get much larger and the L_2 norm is no longer useful in distinguishing neighbors since the points are all far apart. We do not expect good performance from this estimator due to lack of standardization of features (so features with high magnitudes will be weighted disproportionally heavily) and due to the large set of features.

4.2 Training and Testing

As our algorithm states, we train test split within each chunk. We train on 90% of each chunk and test on the remaining 10%. This ensures 900 data points are used to train and 100 are predicted which is sufficient training data in the case of each our models, while also ensuring that we do not attempt to predict data that is unreasonably far out-of-sample. We tried to maximize our testing size while still ensuring good results. This ratio of $9:1$ train to test seems to perform well while still predicting a sizable amount of data.

5 Results

We trained, tuned, and then predicted using all 9 models. For each model we trained on 900 5-minute intervals and predicted 100, which is roughly 8 hours or a normal trading day. Then, we took the

average error for each error metric across all chunks for each model to determine which model performed the best. We believe this estimate to be a reasonable representation of the true total time interval prediction error. Of course, predictions at interval 901 are better than at interval 999. However, we believe our predictions to be quite good for some models despite the relatively large prediction horizon. Our key metrics are $MAE, MAPE, MSE,$ and $RMSE$ for each model and we show them in the table below. The MAPE for Kernel Ridge Regression and Gradient Boosting compare favourably with the best-performing models of Xiong and Zhou with best MAPE of 24.2% and 17%, respectively. These comparisons are particularly impressive when comparing our models with Xiong since we both had attempted prediction horizons of around a day and Xiong incorporated additional Google trends data into their models.

Table 3: Model Results

Model	MAE	MAPE	MSE	RMSE
Kenel Ridge Regression	0.000317	13.368425	2e-06	0.000665
Gradient Boosting	0.000354	14.62726	4e-06	0.000804
Random Forest	0.000365	14.635831	4e-06	0.000787
Ridge	0.000367	15.543471	2e-06	0.000709
Elastic Net	0.000375	15.822203	2e-06	0.000721
Lasso	0.000396	17.265219	2e-06	0.000747
AdaBoost	0.000455	17.91219	4e-06	0.000865
k-Neighbors Regression	0.000905	42.037707	5e-06	0.001403
Nadarya-Watson	0.001017	47.012259	5e-06	0.001468

We use MAPE as our primary metric to compare between models. We see a surprising result in the effectiveness of Kernel Ridge Regression outpredicting both Gradient Boosting and Random Forest. We also observe that k-neighbors and Nadarya-Watson regression did tend to perform poorly, likely due to reasons discussed above.

Now, let us consider the feature importances across models for which such a notion exists. As our procedure states, for each chunk, we have a tuned model of each type that is trained on p portion of that chunk. We have 83 chunks and therefore 83 models of each type. To construct an overall notion of feature importance, we take the mean feature importance for each feature across all 83 models. Below is a plot of these feature importances for each model for which this data is available.

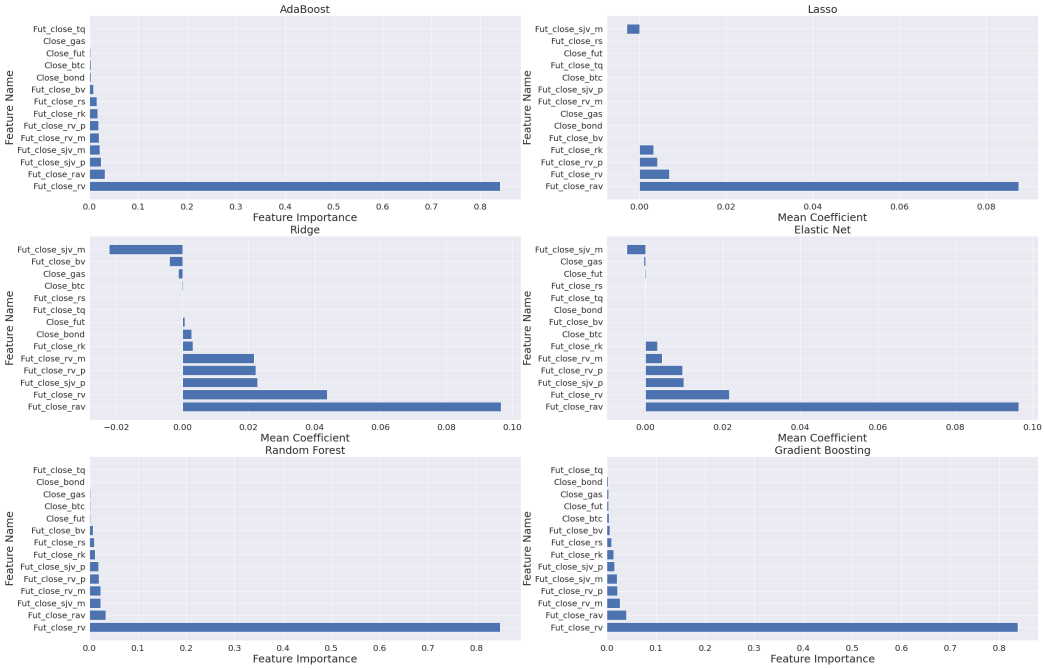

Figure 4: Feature Importances by Model

We see that the strongest feature is the lagged volatility. This makes sense, since financial data tends to be highly autocorrelated. All 6 models place very high feature importance on this feature. Other than lagged volatility, the other most highly weighted features by the four most accurate methods (Kernel Ridge Regression, Gradient Boosting, Random Forest, and Ridge) were the absolute variation, realised semivariance, and signed jump variations. The high importance of absolute variation, realised semivariance, and pure realized volatility all reinforce the notion that lagged volatility is the most important predictor of future volatility. The high importance of signed jump variation also validates the idea that correctly specified jump predictors can serve as useful predictors for volatility. On the other hand, none of the models put significant weight on any of the pure returns features, which suggests returns do not play an important role in predicting volatility. While these plots can indicate data snooping, we thoroughly checked and discounted this possibility.

6 Conclusion

We conclude that nonparametric models perform better than linear models in the prediction of realized volatility on crude oil futures and perform competitively with LSTM-based approaches. There can of course be improvements on our methodology. Choosing quadratic variation, which is a known unbiased estimator of realized volatility, as a feature or outcome measure could bolster results. Further, more robust hyperparameter tuning would certainly improve our results albeit at a high computational cost. Our current training and predicting loop takes roughly 1 hour to run on 35 cores, so any more sophisticated solution would certainly require large compute resources. Further, features created from options market implied volatility would be extremely useful and most likely would be a very important feature to our models. Higher quality data and higher resolution data could both improve our predictions, as well as more sophisticated tick-aggregation based preprocessing methods. We are pleased with our predictive ability based on our best models' errors.

References

Diego Amaya, Peter Christoffersen, Kris Jacobs, and Aurelio Vasquez. Does realized skewness predict the cross-section of equity returns? *Journal of Financial Economics*, 118(1):135–167, 2015. ISSN 0304-405X. doi: https://doi.org/10.1016/j.jfineco.2015.02.009. URL https://www.sciencedirect.com/science/article/pii/S0304405X15001257.

Ole E Barndorff-Nielsen and Neil Shephard. Econometrics of testing for jumps in financial economics using bipower variation. *Journal of financial Econometrics*, 4(1):1–30, 2006.

Ole E. Barndorff-Nielsen, Silja Kinnebrock, and Neil Shephard. Measuring downside risk - realised semivariance. *SSRN Electronic Journal*, 2008. doi: 10.2139/ssrn.1262194. URL https://doi.org/10.2139/ssrn.1262194.

Jonathan Doering, Michael Fairbank, and Sheri Markose. Convolutional neural networks applied to high-frequency market microstructure forecasting. In *2017 9th Computer Science and Electronic Engineering (CEEC)*, pages 31–36, 2017. doi: 10.1109/CEEC.2017.8101595.

L. Forsberg and E. Ghysels. Why do absolute returns predict volatility so well? *Journal of Financial Econometrics*, 5(1):31–67, November 2006. doi: 10.1093/jjfinec/nbl010. URL https://doi.org/10.1093/jjfinec/nbl010.

Wenbo Ge, Pooia Lalbakhsh, Leigh Isai, Artem Lenskiy, and Hanna Suominen. Neural network–based financial volatility forecasting: A systematic review. *ACM Comput. Surv.*, 55(1), jan 2022. ISSN 0360-0300. doi: 10.1145/3483596. URL https://doi.org/10.1145/3483596.

Xin Huang and George Tauchen. The Relative Contribution of Jumps to Total Price Variance. *Journal of Financial Econometrics*, 3(4):456–499, 08 2005. ISSN 1479-8409. doi: 10.1093/jjfinec/nbi025. URL https://doi.org/10.1093/jjfinec/nbi025.

Sang Hoon Kang, Sang-Mok Kang, and Seong-Min Yoon. Forecasting volatility of crude oil markets. *Energy Economics*, 31(1):119–125, 2009.

Ha Young Kim and Chang Hyun Won. Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, 103:25–37, 2018.

Markku Lanne and Pentti Saikkonen. Non-linear GARCH models for highly persistent volatility. *The Econometrics Journal*, 8(2):251–276, 07 2005. ISSN 1368-4221. doi: 10.1111/j.1368-423X.2005.00163.x. URL https://doi.org/10.1111/j.1368-423X.2005.00163.x.

Nikolay Nikolaev, Peter Tino, and Evgueni Smirnov. Time-dependent series variance learning with recurrent mixture density networks. *Neurocomputing*, 122:501–512, 2013.

Andrew J. Patton and Kevin Sheppard. Good Volatility, Bad Volatility: Signed Jumps and The Persistence of Volatility. *The Review of Economics and Statistics*, 97(3):683–697, 07 2015. ISSN 0034-6535. doi: 10.1162/REST_a_00503. URL https://doi.org/10.1162/REST_a_00503.

Ioannis Psaradellis and Georgios Sermpinis. Modelling and trading the us implied volatility indices. evidence from the vix, vxn and vxd indices. *International Journal of Forecasting*, 32(4):1268–1283, 2016.

Perry Sadorsky. Modeling and forecasting petroleum futures volatility. *Energy economics*, 28(4): 467–488, 2006.

Ruoxuan Xiong, Eric P Nichols, and Yuan Shen. Deep learning stock volatility with google domestic trends. *arXiv preprint arXiv:1512.04916*, 2015.

Yu-Long Zhou, Ren-Jie Han, Qian Xu, Qi-Jie Jiang, and Wei-Ke Zhang. Long short-term memory networks for csi300 volatility prediction with baidu search volume. *Concurrency and Computation: Practice and Experience*, 31(10):e4721, 2019.