JOIM
www.joim.com

JOURNAL OF INVESTMENT MANAGEMENT, Vol. 18, No. 2, (2020), pp. 1–16

USING MACHINE LEARNING TO PREDICT REALIZED VARIANCE

Peter Carr[*,a], *Liuren Wu*[†,b] *and Zhibai Zhang*[‡,a]

Volatility index is a portfolio of options and represents market expectation of the underlying security's future realized volatility/variance. Traditionally the index weighting is based on a variance swap pricing formula. In this paper we propose a new method for building volatility index by formulating a variance prediction problem using machine learning. We test algorithms including Ridge regression, Feedforward Neural Networks and Random Forest on S&P 500 Index option data. By conducting a time series validation we show that the new weighting method can achieve higher predictability to future return variance and require fewer options. It is also shown that the weighting method combining the traditional and the machine learning approaches performs the best.

1 Introduction

Estimating future return variance is an essential part for investing. Similar to future return, a security's return variance often exhibits stochastic behavior which makes it challenging to forecast. For many derivative instruments, pricing is predominantly determined by modeling the underlying asset's volatility in the risk neutral measure. There has been a broad literature on derivative pricing which involves various volatility models. These range from the simple setup with constant volatility (Black and Scholes, 1973) to the ones that use deterministic functions (Dupire, 1994), and to the complex models that treat volatility as stochastic processes (Heston, 1993). On the other hand, the market price for options and other derivative instruments on a security reflects a universal expectation of the underlying's future return variance. Therefore, it is reasonable to use option market price and the implied volatility to project future return variance even without a specific volatility model.[1]

Since there are numerous options written on a security and each of them has different implied

[a]Department of Finance and Risk Engineering, NYU Tandon School of Engineering, New York, USA.
[b]Zicklin School of Business, Baruch College, City University of New York, New York, USA.
[*]petercarr@nyu.edu
[†]liuren.wu@baruch.cuny.edu
[‡]z.zhibai@gmail.com, corresponding author.

volatility, aggregating the options to build a single variance estimate is non-trivial. This can be viewed as a classic indexing problem that consists of security-selection rules and a weighting scheme. The most well-known volatility index in this category is the CBOE Volatility Index (VIX), which is composed of options with close to 30-day maturity on S&P 500 index (SPX). It is one of the most commonly used estimators for 30-day future variance of SPX. There is an increasing number of volatility indices on various equity indices and major instruments in fixed-income, currency and commodity in recent years. For VIX-styled indices, the option weights are determined by a variance swap pricing formula such that the index's squared payoff replicates the underlying's variance in the risk-neutral measure. Albeit its popularity, the VIX-styled indexing possesses some caveats. First, as the weights are set in the risk-neutral measure, it is not certain if the weighting scheme has the optimal forecastability to future volatility, especially out-of-sample (OOS) in the market measure. Additionally, it involves a large number of out-of-the money options with ascending illiquidity. This makes it expensive and impractical for hedging any tradable products associated with the index, as it requires one to hold many thinly traded options.

In this paper we propose a new volatility indexing method to improve predictability and liquidity. To do so, we formulate a regression problem to predict realized variance by using option price as features and construct a weighting scheme from the loadings of the regression. We experiment with algorithms including linear and machine learning techniques such as Ridge, Feedforward Neural Networks (FNN) and Random Forest, and impose constraints on model selection to make sure that the prediction can be replicated by an option portfolio. We test the algorithms with a time series validation approach on SPX and its option data.

We discover that by combing the prediction model and the VIX-styled weighting scheme, one can achieve an index that has improved predictability and liquidity. The best performing approach is to use machine learning regression to forecast the deviation between the realized volatility and the VIX-styled index's prediction, which is a proxy of the variance risk premium. Intuitively, this approach can be interpreted as applying a machine learning algorithm to minimize the deviation between a human model's prediction and the actual outcome. Therefore, our results represent a successful combination of human learning and machine learning. We also discuss suitability of different regression algorithms for volatility indexing. As we will show, the tradability condition in fact imposes a strong constraint on algorithm selection, which most models do not satisfy except for piece-wise linear ones such as FNN with an ReLU activation function. Additionally, we employ a machine learning feature importance method to test every option's contribution to the prediction. We find that the the options' out-of-the-moneyness is proportional to their predictability to realized variance, and calls on average have higher predictability than puts.

This paper joins a large number of literature on variance forecasting. In the past, there has been great progress on time series-based models such as ARCH/GARCH (Engle, 1982; Bollerslev, 1986) and HAR (~~Fulvio,~~ 2009). It has been shown that historic volatility measures exhibit predictability at future realized volatility. To this end, we also experiment with historic volatility features as alternative tests to the main model and we find that their contribution is limited in our framework. There has also been abundant

study on the predictability of option price and implied volatility at realized volatility (Federico *et al.*, 2008; Andersen *et al.*, 2007; Jeff, 1998; Busch *et al.*, 2011). More recently, application of machine learning techniques in volatility forecasting have emerged (Luong *et al.*, 2018; Hamid, 2004). To the best of our knowledge, this paper is the first attempt to apply machine learning to predict volatility and build a volatility index at the same time.

The rest of this paper is organized as follows: In Section 2 we review the details of VIX-styled volatility indices' construction and caveats. In Section 3 we show how to process option data to formulate a realized variance prediction problem. We also discuss model selection and evaluation. This is followed by Section 4, where we present the main result on prediction performance. Section 5 concludes and presents directions for future research.

2 Volatility index and variance prediction

Established in 1993, the CBOE Volatility Index (VIX) is one of the first equity volatility indices and it has been broadly used as the volatility/variance benchmark for the entire US equity market. Initially, VIX is built from only the at-the-money implied volatility of the underlying. In 2003, CBOE changed the pricing method to incorporate a variance swap pricing formula which is based on the price of a large number of out-of-the-money options. This new method proved to be more robust as it covers a much wider range of the implied volatility surface. Since then, VIX and its tradable derivatives (i.e., options, futures and exchange-traded products) have grown considerably more popular. In recent years, CBOE has carried out a series of VIX-styled volatility indices on other assets, including major equity indices, interest rates, FX and commodities. In this section, we review the details of the variance swap pricing-based index weighting method. Schematically, the index level is calculated as the squared root of the value of a portfolio of out-of-the-money (OTM) SPX options with weights inversely proportional to the squared strike prices:

$$\text{VIX}^2 \sim \sum \frac{2\Delta K}{K^2} O(K, \tau), \tag{1}$$

where $O(K, \tau)$ is the price of an option with strike price K and time-to-maturity τ, and ΔK the constant spacing between strikes. The $\frac{2\Delta K}{K^2}$ weighting scheme comes from the variance swap pricing formula in (Carr and Madan, 2001; Carr and Wu, 2009). The weights are determined such that the portfolio's payoff perfectly replicates the variance of the underlying in the risk-neutral measure.

As for the same underlying security there are numerous active options with different strikes and time-to-maturity's, it is important to have selection criteria for the options that go into Equation (1). Since VIX is designed for a 30-day horizon, ideally it is best to choose options expiring in exact 30 days. However, this is not realistic as most of the time these options are not available on the market. CBOE applies a more general two-step procedure to select option tenors. Namely, one selects 'near-term' options with maturity τ_1 the closest to yet less than 30 days, and 'next-term' options with maturity τ_2 the closest to yet greater than 30 days. These two tenors are then linearly interpolated to formulate an exact 30-day horizon. Secondly, for each term, one selects as many out-of-the-money options as possible, with ascending (for calls) or descending (for puts) strikes K. For both terms, all the strikes are included until two consecutive strikes are missing a quote. As such, the number of strikes varies from time to time, highly depending on the liquidity of the option market.

Once the options are selected, one computes the implied variance for each term:

$$\begin{aligned}\sigma_1^2 &= \frac{2}{\tau_1}\sum_h \frac{\Delta K}{K_h^2} e^{R_1\tau_1} O(K_h,\tau_1) \\ &\quad - \frac{1}{\tau_1}\left(\frac{F}{K_0}-1\right)^2, \\ \sigma_2^2 &= \frac{2}{\tau_2}\sum_h \frac{\Delta K}{K_h^2} e^{R_2\tau_2} O(K_h,\tau_2) \\ &\quad - \frac{1}{\tau_2}\left(\frac{F}{K_0}-1\right)^2,\end{aligned} \tag{2}$$

where $\Delta K = K_{h+1} - K_h$ is the spacing in strike prices, R_i is the interest rate for each term and F is the forward index value. The second term on the r-h-s in each equation is considerably smaller than the first term. Finally VIX is computed as the square root of a weighted sum of the r-h-s in the above equations up to scaling by 100:

$$\frac{\text{VIX}}{100} = \sqrt{\tau_1\sigma_1^2\left(\frac{\tau_2-30}{\tau_2-\tau_1}\right) + \tau_2\sigma_2^2\left(\frac{30-\tau_1}{\tau_2-\tau_1}\right)} \tag{3}$$

When there is options with 30-day time-to-maturity, one can simply compute the variance term in Equation (2) for them without further interpolation. For more details, see (VIX White Paper, xxxx). It is worth noting that by its nature, this pricing formula-based weighting scheme overweights OTM put options and underweights OTM call options, as shown in Figure 1. ~~This leads to puts overweighted than calls.~~

As mentioned above, the normalized[2] level of VIX is commonly considered a benchmark for S&P's 30-day realized volatility forecast. We will show that this measure indeed exhibits predictability to realized volatility, measured by positive out-of-sample R^2. However, the weights determined in the risk-neutral measure may not have the most optimal predictability. Furthermore, the option selection criteria normally produce a large number of options. For instance, in the example in (VIX White Paper, xxxx), with the spot price at $1960, the lowest put strike is at $1370 while the highest call strike is at $2125, the entire universe contains 149 options in total. Holding that many OTM options is extremely costly as the liquidity is very low for deep OTM options in general. This makes hedging challenging for VIX derivatives sellers. For these reasons, we explore machine learning algorithms

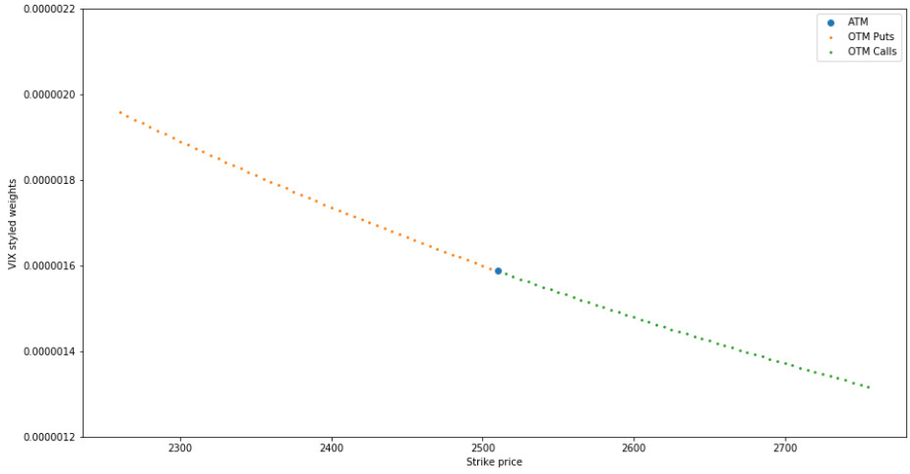

Figure 1 VIX weights as of January 2, 2019.

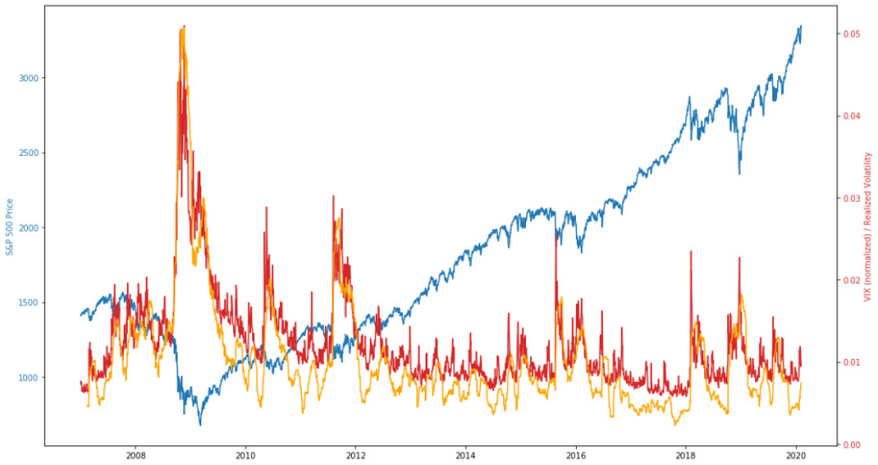

Figure 2 S&P 500 price (blue), its 30-day realized return volatility (orange) and VIX (red).

to improve the VIX weighting scheme's predictability and liquidity.

3 Formulate a machine learning regression problem

In this section we propose a new method to build volatility indices using a machine learning approach. More specifically, we set up a supervised machine learning regression problem that uses option price to forecast future return volatility. By taking the regression loadings as weights, we then construct an option portfolio as the volatility index.

To begin with, we briefly review the basis of the supervised machine learning paradigm. Without loss of generality, a supervised ML algorithm can be summarized as a function approximation problem aimed to find a function $f(\cdot)$ such that

$$\begin{aligned} y &= \hat{f}(\vec{x}), \\ \hat{f} &= \arg\min\{\mathrm{Err}(f(\vec{x}), y)\} \end{aligned} \tag{4}$$

where $\mathrm{Err}(f(\vec{x}), y)$ is a pre-defined object function defined on the sample data set $(y, \vec{x})$. For many ML algorithms, $f(\cdot)$ is either semi-parametric (with a large number of parameters) or non-parametric (can only be carried out operationally and does not have a closed-form expression). y is usually referred to as the target value and entries in $\vec{x}$ are referred to as features. Next, let us specify the features and target value that is suitable for a volatility index.

3.1 Data processing and feature generation

We use daily data of options written on SPX with a time span from 1996 to 2016, which includes more than 5,000 ~~unique~~ trading days. The option market data is obtained from OptionMetrics. On each trading day, the data contains midquotes of options with multiple maturities and strikes. A sample data set is shown in Figure 3. Furthermore, interest rate and SPX spot and forward price data are also used.

Generating features from option price time series proves to be a non-trivial task. There are two aspects of this data set that raise problems for a machine learning formulation. First, the actively traded options are those whose strikes are centered around the spot, which varies all the time. As a result, the set of OTM options needs to be re-selected daily. Moreover, the options' maturities continue to decrease until expiry. In general,

Date	expiry	ID	Days	F	S	R	Div	K	CP	IV	Bid	Ask	Volume	OptionIntests
734627.0	735589.0	48499593.0	962.0	1308.651293	1347.319946	0.010787	0.021835	1900.0	1.0	0.144413	7.100000	9.900000	0.0	2222.0
734627.0	735589.0	48499643.0	962.0	1308.651293	1347.319946	0.010787	0.021835	1900.0	2.0	0.142685	579.200012	586.400024	3.0	2.0
734627.0	735589.0	48499595.0	962.0	1308.651293	1347.319946	0.010787	0.021835	2000.0	1.0	0.138912	3.500000	4.700000	0.0	957.0
734627.0	735589.0	48499645.0	962.0	1308.651293	1347.319946	0.010787	0.021835	2000.0	2.0	0.137613	672.500000	679.200012	1.0	92.0
734627.0	735589.0	49769975.0	962.0	1308.651293	1347.319946	0.010787	0.021835	2250.0	1.0	0.141170	1.000000	1.300000	100.0	14211.0
734627.0	735589.0	49288567.0	962.0	1308.651293	1347.319946	0.010787	0.021835	2250.0	2.0	0.140184	913.000000	919.099976	0.0	160.0
734627.0	735589.0	49288568.0	962.0	1308.651293	1347.319946	0.010787	0.021835	2500.0	1.0	0.153048	0.500000	0.750000	0.0	146.0
734627.0	735589.0	48499648.0	962.0	1308.651293	1347.319946	0.010787	0.021835	2500.0	2.0	0.154798	1155.900024	1161.400024	2.0	507.0
734627.0	735589.0	49288570.0	962.0	1308.651293	1347.319946	0.010787	0.021835	3000.0	1.0	0.171956	0.000000	0.450000	0.0	1.0
734627.0	735589.0	48657884.0	962.0	1308.651293	1347.319946	0.010787	0.021835	3000.0	2.0	0.187250	1642.199951	1646.800049	4.0	633.0

Figure 3 A sample of the OptionMetrics' option market data. The fields relevant to the analysis are trading date (Date), expiration date (expiry), spot price of SPX (S), short-term interest rate (R), call/put type (CP), bid price (Bid) and ask price (Ask).

most machine learning algorithms require the features to be stationary. Since our goal is to use option price to forecast the realized volatility with a fixed horizon, it is also important to have the options' maturities in sync with the forecast horizon. Fortunately, the construction of VIX provides a solution to the varying maturity issue. Namely, we can linearly interpolate the option with the closest maturities to the forecast horizon as in Equation (3). Since regression problems require a fixed number of features, we need to select a constant number OTM options with strikes centered around the spot price. Put together, we generate raw option price features as follows:

$$\tilde{O}_t(K_h, T) = O_t(K_h, \tau_1)\left(\frac{\tau_2 - T}{\tau_2 - \tau_1}\right) + O_t(K_h, \tau_2)\left(\frac{T - \tau_1}{\tau_2 - \tau_1}\right), \quad (5)$$

where T is the forecast horizon (in the case of VIX this is 30 day), τ_1 and τ_2 are the two closest existing maturities to T of all available options at day t. For each day, we select $N = 2n + 1$ of strikes K_h centered around the at-the-money strike K_0:

$$\{K_{-n}, K_{-n+1}, \ldots, K_{-1}, K_0, K_1, \ldots, K_{n-1}, K_n\}, \quad (6)$$

where $\Delta K = K_i - K_{i-1}$ (for SPX options, $\Delta K = \$5$). It needs to be understood that the at-the-money strike K_0 changes every day following the spot price S_t. Therefore, the strike set Equation (6) is determined on a daily basis. Notice that as n becomes larger, the corresponding strike K_n is more out-of-the-money and the option tends to be less frequently traded. If for a specific K_h, the option does not have a quote, we linearly interpolate the midquote using midquotes of the options with the two closest strikes to K_h. This way, Equation (5) gives rise to a consistent feature generation once the number of features N and forecast horizon T are chosen for an option time series. The detailed feature generating procedure is shown in the code snippets at the end of this subsection.

In machine learning for most algorithms, it is important to normalize features to meet stationarity conditions. For each feature, a common practice is to subtract its sample mean and divide it by its sample standard deviation. We apply this technique and normalize the features using the training sets. In addition, there is subtlety related to financial time series, which is that a security's price is a non-stationary process as otherwise there is arbitrage. Therefore, option price features in Equation (5) contain this specific nonstationarity that cannot be removed by the standard machine learning normalization

procedure. A rationale is that, an option on SPX worth $5 in 2010 is not the same as an option worth the same thing in 2015, as the level of SPX has grown considerably over that period. For this reason, we apply the following pre-processing before the standard ML normalization to the option price features in Equation (5):

$$\bar{O}_t(K_h, T) = \frac{\tilde{O}_t(K_h, T)}{K_h^2}. \tag{7}$$

In addition to the main option price features, we also experiment with returns-based idiosyncratic features of SPX as alternative tests. These include realized returns and variance with different look-back windows.

- Realized returns, $r_{t,l} = \frac{p_t - p_{t-l}}{p_{t-l}}$ with $l \in \{1, 5, 15, 30, 60, 90\}$
- Realized variance, $var_{t,l} = \frac{1}{l}\sum_{i=1}^{l}(r_{t-i} - \bar{r})^2$, with $l \in \{15, 30, 60, 90\}$

We only consider these features in combination with the option features, but not their predictability individually. This is mainly because that once these features are added, the prediction is no longer tradable as they cannot be replicated by any portfolio.

3.2 Target value and two regression approaches

Though volatility and variance are virtually the same variable, we are predominantly focused on forecasting future realized return variance. This is to avoid the extra step of taking square root in Equation (1). The realized return variance between time t and $t + T$ is given by

$$Var_t^T = \frac{1}{T}\sum_{i=1}^{T}(r_{t+i} - \bar{r})^2, \tag{8}$$

where $r_j = \frac{p_j - p_{j-1}}{p_j}$ is the security's daily return at time j and $\bar{r}$ is the average return between $t+1$ and $t+T$. Sometimes the mean return $\bar{r}$ is omitted as it is close to zero in most cases. Volatility is the square root of the variance and it is more often quoted in the derivative markets. In this paper, we use the terms volatility and variance interchangeably.

Our first approach is to directly model future realized variance as a function of option price. Mathematically, this is

$$Var_t^T = f(\{O_t(K_i, T_j')\}) + \epsilon_t, \tag{9}$$

where $\{O_t(K_i, T_j')\}$ is all the options across selected strikes and tenors and ϵ_t is a zero-mean

Algorithm 1. Strike selection

Input: $\{1, 2, \ldots, D\}$ = timestamps; ATM_t= ATM strike on day-t; n=# of selected OTM put/call
Output: KS = selected strikes
$KS \leftarrow \varnothing$
for Each day $t \in \{1, 2, \ldots, D\}$ **do**
  $KS_p \leftarrow \varnothing$, $KS_c \leftarrow \varnothing$
  for $i \in \{1, \ldots, n\}$ **do**
    Append $(ATM_t - i \times \Delta K)$ to KS_p
    Append $(ATM_t + i \times \Delta K)$ to KS_c
  end
  $KS_t \leftarrow KS_p \bigcup KS_c \bigcup \{ATM_t\}$
  Append KS_t to KS
end

Algorithm 2. Data processing and feature engineering

Input: $\{1, 2, \ldots, D\}$ = timestamps; KS = selected strikes; T = forecast horizon
Output: $\{\tilde{O}\}$ = option features; VIX^* = synthetic VIX index
for Each day $t \in \{1, 2, \ldots, D\}$ **do**
  if On day-t there exists options with exact 30-day maturity **then**
    Select all options with tenor $\tau = 30$ and strike $K \in KS_t$
    if There exists missing price for selected options **then**
      Fill missing price by liner interpolation in the selected strike set
    end
    Collect selected options' price $\{\tilde{O}_t\}$; Compute VIX_t^* with $\tau = 30$
  end
  else
    Find near-term $\tau_1 \leftarrow argmin_{\tau<30}|\tau - 30|$
    Find next-term $\tau_2 \leftarrow argmin_{\tau>30}|\tau - 30|$
    for $\tau \in \{\tau_1, \tau_2\}$ **do**
      if There exists missing price for selected options **then**
        Fill missing price by liner interpolation in the selected strike set
      end
      Collect selected options' price $\{\tilde{O}_t\}$; Compute VIX_t^* with τ's using Equation (3)
    end
    Generate features using options' price with τ_1 and τ_2 from Equation (5)
  end
end

noise term. As mentioned above, the function f will be determined by fitting the algorithm on the training data set. We call this approach Regression I.

An ML-based function approximation such as Equation (9) can be useful in estimating functions with high non-linearity and non-parametricity. This makes Equation (9) desirable as realized variance is expected to be strongly non-linear. However, the large number of parameters and complex optimization involved in ML algorithms may also be problematic, as they can lead to overfitting due to outliers in the training set (e.g. extremely high volatility events). Additionally, as VIX-styled weighting scheme already contains some predictive power, in principle we would like to incorporate it with the regression approach too. This ~~lead~~ to our second regression approach:

$$Var_t = VIX^*(\{O_t(K_i, T_j')\})^2 + f(\{O_t(K_i, T_j')\}) + \epsilon_t, \quad (10)$$

where VIX^* is the synthetic VIX index using the selected options. The only difference between the two is that VIX^* contains a fixed number of options throughout time. We call approach Equation (10) Regression II.

Note that if one switches the VIX^{*2} to the l-h-s, then it is equivalent to regress a variance risk premium (VRP) proxy, which is

$$VRP_t = VIX^{*2}_t - Var_t. \quad (11)$$

So this approach is to train the algorithm to forecast VRP_t and combine it with VIX^* to forecast realized volatility. Though Equation (10) is a special case of Equation (9), the two approaches in general produce different results in non-linear cases. Regressing the difference between Var_t^T and VIX^{*2} rather than directly regressing Var_t^T can be considered as a normalization procedure. In non-linear models, proper normalization can often substantially improve an algorithm's fitting and performance. We will show that Regression II indeed outperforms Regression I.

3.3 Algorithms

One advantage of machine learning is that the algorithm can be highly adaptive. In the form of Equation (4), this means that $\hat{f}$ usually does not have a close form expression. Since we want to be focused on an indexing problem, this may cause trouble. In both of the two regression approaches Equations (9) and (10), if we require that the forecast can be replicated by an option portfolio, we need $f(\cdot)$ to be at least a piece-wise linear function. This is actually a strong constraint that renders many ML regression methods not suitable. For instance, for K-nearest neighbors regression the forecast is made on the sub-sample mean of the training set and is clearly not piece-wise linear. In that case, obviously one cannot build an option portfolio whose payout is the forecast.

In this paper, we consider four regression algorithms: linear regression, ridge regression, feed-forward neural network regression with ReLU activation function and random forest regression. We start off with linear regression for its simplicity. Compared with ML algorithms, linear regression has the advantage that it does not involve tunable parameters which makes it the least prone to overfitting. However, linear regression also has a few shortcomings, such as its lacking of non-linearity. Another potential issue is that, since our main feature set is a basket of options that have strong correlation, it may cause trouble for the ordinary least square fitting. For this reason, we consider ridge regression, which has the same functional form as linear regression but the error function is $L2$ regularized.

Feedforward neural network (FNN) is one of the simplest neural network models. The graphic representation of a neural network is composed of a number of layers, including an input layer corresponding to the features and an output layer corresponding to the labels. The rest is referred to as hidden layers. Each layer contains several neurons and different layers are connected by a certain topology. Mathematically, both neurons and connections between neurons correspond to variables of the prediction functions. A feedforward neural network with one hidden layers has the following functional form

$$y(\boldsymbol{x}, \boldsymbol{w}) = O\left(\sum_j w_j^{(2)} \cdot h\left(\sum_i w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_0^{(2)}\right), \tag{12}$$

where O and H are so-called activation functions on the output and hidden layers, w_{ji}^p is connection weight that connects the i-th neuron on the $(p-1)$-th layer to the j-th neuron on the p-th layer. Activation functions are functions that filter the input information, determine the amount of output information, and are usually non-linear. For our purposes, we select rectified linear unit (ReLU) as the activation function for both the hidden layer and the output layer. This is because among the most commonly used activation functions, only ReLU is both non-linear and piece-wise linear. ReLU is defined as

$$ReLU(x) = max\{0, x\}. \tag{13}$$

With an ReLU FNN, y is piece-wise linear:

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{i,j} 1_{(2),j} w^{(2)}_j w^{(1)}_{ji} x_i + const. \qquad (14)$$

where $1_{(p),k}$ is an indicator function that corresponds to the ReLU activation function on the k-th neuron on the p-th layer. Now it is evident that ReLU FNN is piece-wise linear and thus suitable for our tradability constraint. For model simplicity, we construct FNN with only one hidden layer, whose number of neurons is the same as the input layer.

On the contrary, some ML algorithms are not piece-wise linear. Random forest is a very popular algorithm in this category. Random forest is an ensemble algorithm that consists of multiple base algorithms called decision trees. For decision tree regression, the algorithm iteratively splits the input data on the features such that the information gain from splitting a parent sample to children samples is optimized. After the algorithm is trained, each sample (regardless of the training or the test set) can be run through the tree and land on one of the many sub-samples after splitting the original training sample. The prediction is then given as the mean value of the dependent variable in the specific sub-sample. In random forest regression, multiple decision tree regressions are fitted, each on a randomly sampled training set to increase robustness, and the final prediction is an average of the prediction of all decision trees. So schematically, the forecast from random forest regression is

$$y(x) = \frac{1}{N_I} \sum_{i \in I} y_i, \qquad (15)$$

where I is the subset the (x, y) belongs to, N_I is the total number of samples in this subset and y_i is the value of the dependent variable of the i-th sample in the subset. Relating to our setting, this means that when using random forest, the predicted variance is a function of the realized variances of ~~at~~ selected past timestamps, which clearly cannot be the payoff of an option portfolio. More straightforwardly, Equation (15) is not a piece-wise linear function. Nonetheless, we will keep random forest in the test for predictability comparison.

3.4 Validation, evaluation and model calibration

To evaluate all algorithms and feature combinations, we formulate an out-of-sample (OOS) test on the data set in a time series fashion. To do so, we reserve the first 1,000 observations for both hyperparameter optimization and initial training, and we make OOS prediction on the first 30 observations on the remaining set. We re-train the model after every 30 observations with all the available observations on a rolling basis. Every time the training set is purged to get rid of the observations that has an informational overlap with the test set (see (Lopez de Prado, 2018) for similar techniques). Obviously a different combination of the size of each training set and the frequency of model retraining may vary the performance of each model. We do not test other validation settings as this may cause overfitting due to the limited amount of data. A graphic representation of the OOS test is shown in Figure 4.

For prediction performance metrics, we use OOS R^2

$$R^2 = 1 - \frac{\sum_i (y_i - p_i)^2}{\sum_i (y_i - \bar{y})^2}, \qquad (16)$$

where y_i is the actual realized variance for sample i, $\bar{y}$ is the mean realized variance of all samples and p_i is the model predicted realized variance for sample i. For Regression II (Equation (10)), even though the direct prediction is the negative variance risk premium, we convert it back

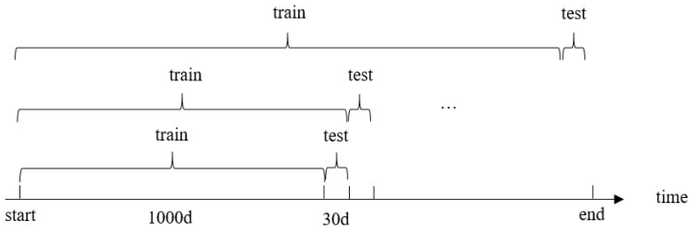

Figure 4 An illustration of the rolling validation process.

to realized variance by adding VIX^{*2} when computing the R^2. This way the R^2 for both regression approaches is comparable.

Most if not all ML algorithms contain multiple hyperparameters that need to be tuned. A common practice is to optimize a large number of hyperparameter combinations using cross-validation and choose the combination that performs the best. It is worth noting that tuning hyperparameters unavoidably raises the likelihood of overfitting as a trade-off. This is especially true with financial time series data, which exhibit very low signal-to-noise ratio. Therefore, we need to keep the number and kind of hyperparameters as low as possible. To achieve this, for each of the three selected algorithms, we only tune a hyperparameter associated with the regularization strength. More specifically:

- for Ridge, the $L2$ regularization strength $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^2, 10^3, 10^4\}$
- for ReLU FNN, the $L2$ regularization strength $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10^2, 10^3, 10^4\}$
- for Random Forest, maximum tree depth $\in \{3, 5, 10, \infty\}$.

We note that, the $L2$ regularization strength for both Ridge and ReLU FNN in Regression II Equation (10) has a fairly clear interpretation: It represents how much the new weights deviated from the old VIX's weights. For each algorithm, we optimize the hyperparameter in the initial 1,000 sample training set by taking the first 900 samples as training and the rest 100 samples as test. For FNN, we use Adaptive Moment Estimation (Adam) as the solver. All the algorithms are implemented in Python using the Scikit-learn package (Pedregosa *et al.*, 2011).

4 Main results

In this section we present our main finding. First, we show the OOS R^2 for both regression methods with all four algorithms. We conduct the experiment with a varying number of consecutive strike prices and two different forecast horizons. The performance is compared across different algorithms. Then we report the results of some alternative tests, including using non-consecutive strike prices and adding historic volatility and returns as extra features. Last but not least, we apply a robust machine learning feature importance analysis on the option price features to test the contribution of each option to forecast future realized volatility. To our surprise, it turns out that calls have higher importance than puts, in contrast to VIX's weighting scheme.

4.1 $T = 30$ *days*

The first forecast horizon we test is 30 days, which is the designated horizon for VIX. For each algorithm, we use $2k + 1$ consecutive options as features (k OTM put, k OTM call and 1 ATM), with $k \in \{10, 20, 30, 40\}$. We present the performance for the two regression approaches, and highlight the best performing algorithm with a specific number of options.

- OOS R^2 for Regression I,

# of options	VIX^{*2}	Linear	Ridge	RF	FNN
21	0.169	**0.313**	0.276	−0.018	0.154
41	0.313	0.191	**0.329**	−0.021	0.114
61	**0.366**	0.163	0.339	−0.023	0.251
81	**0.389**	0.153	0.334	−0.026	0.339

- OOS R^2 for Regression II,

# of options	VIX^{*2}	Linear	Ridge	RF	FNN
21	0.169	**0.313**	0.290	0.227	0.227
41	0.313	0.191	**0.329**	0.327	0.326
61	0.366	0.163	0.326	**0.371**	0.368
81	0.389	0.153	0.312	0.392	**0.394**

Combining the two regression methods' results and best performance of each algorithm for a specific number of options, the overall model comparison is shown in Figure 5. First, it is apparent that Regression II produces greater performance than Regression I, as the OOS R^2 for the former is higher than that of the latter for all non-linear algorithms (for VIX^* and linear regression, I and II are equivalent). It is also obvious that including more options enhances OOS R^2, except for linear regression. This is because adding more options whose prices are highly correlated deteriorates linear regression's fitting. Nonetheless, when the number of options is small ~~(Equation (21)),~~ linear regression actually outperforms the others. As the number of options increases, FNN with Regression II becomes the best one, though the difference to other methods is only incremental[3].

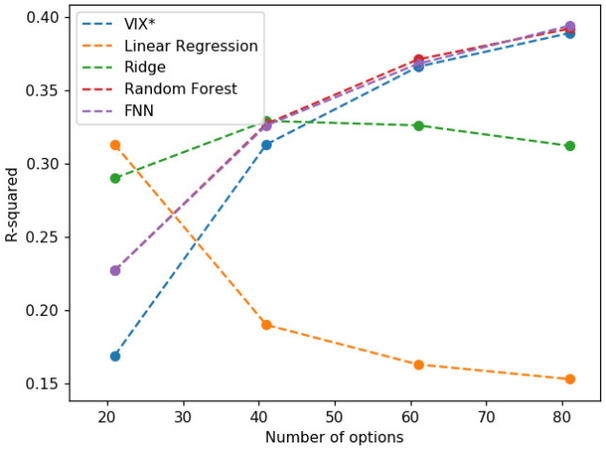

Figure 5 $T = 30$ days. OOS R^2 for different algorithms, optimal performance combining Regressions I and II.

4.2 $T = 60$ *days*

Next we conduct a similar test with $T = 60$ days. We present the performance for the two regression approaches, and highlight the best performing algorithm with a specific number of options.

- OOS R^2 for Regression I,

# of options	VIX^{*2}	Linear	Ridge	RF	FNN
21	**0.300**	0.264	0.226	−0.015	0.094
41	0.206	0.269	**0.276**	−0.023	0.099
61	0.014	0.247	**0.303**	−0.022	0.140
81	−0.155	0.216	**0.313**	−0.025	0.146

- OOS R^2 for Regression II,

# of options	VIX^{*2}	Linear	Ridge	RF	FNN
21	**0.300**	0.264	0.247	0.296	0.297
41	0.206	0.269	0.288	**0.339**	**0.339**
61	0.014	0.247	**0.299**	**0.299**	0.297
81	−0.155	0.216	**0.294**	0.243	0.244

Combining the two regression methods' results and best performance of each algorithm for a specific number of options, the overall model comparison is shown in Figure 6. Interestingly, as the forecast horizon goes up the benefit of including more options diminishes. In most cases, the performance becomes worse after the number of options is above a certain value. For 60-day horizon the optimal number of options is 41, with FNN, Ridge and Random Forest being quite close.

4.3 *Alternative tests*

To further verify our models' predictability, we run a couple of alternative tests in addition to the

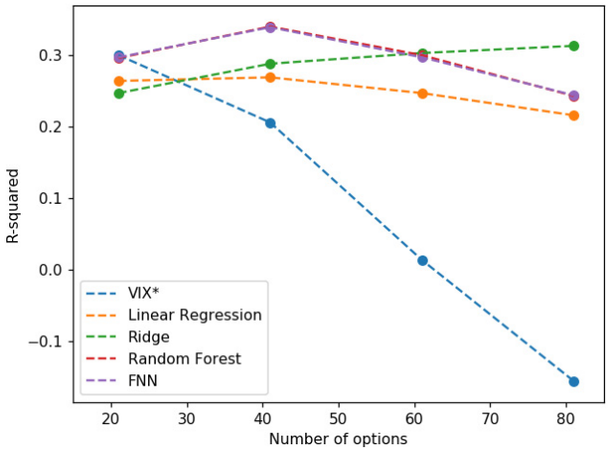

Figure 6 $T = 60$ days. OOS R^2 for different algorithms, optimal performance combining Regressions I and II.

results reported above. First, as mentioned in Section (3.1), we add returns-based features to the main option-based features. Since a key of our approach is to maintain both predictability and tradability, it is important to note that these features are not tradable and cannot be included in an index. Nonetheless, we add these features to see how much predictability they may contribute to the main option features. To do this, we re-run the OOS test using Regression II for $T = 30$ days, with 41 consecutive options and the 10 realized returns and variance features. The OOS R^2's are:

algorithms	Options	Options and realized returns/variance
Linear	0.191	0.124
Ridge	0.329	0.335
Random Forest	0.327	0.331
FNN	0.326	0.330

It shows that except for linear regression, including realized returns and variance features can improve the performance, even though the change is incremental.

Since the options of the same underlying tend to have strong correlations, it is interesting to see if one can increase the spacing in strikes when selecting options. By selecting options with larger distance in their strikes, one gets the benefit of using fewer options, which corresponds to greater liquidity. So far we have only tested the algorithms with consecutive strikes, namely, the spacing is \$5. Here we re-run FNN using regression II with 21 and 41 options and increased spacing of \$10. The results are quite interesting:

# of options	$\Delta K = 5$	$\Delta K = 10$
21	0.227	0.253
41	0.326	0.268

It shows that having consecutive strikes is in fact necessary to maintain the predictability, as when the number of options is 41 the performances with $\Delta K = 5$ and $\Delta K = 10$ diverge and the larger spacing set performs less well.

4.4 Feature importance for calls and puts

To better understand the mechanism of the new approach, we investigate how the regression methods weight the options with strikes. To check this, a straightforward way is to compare the new weights in Equations (9) and (10) with VIX's weights in Equation (3). However, this has a drawback: as the new weights are determined by regression loadings that are fitted on training data, the larger weights do not necessarily correspond to higher importance to OOS prediction. For this reason, we apply a feature importance measure called mean-decreased-accuracy (MDA), a method extensively used in the machine learning literature.

The MDA procedure computes the importance of each feature as follows: (a) one trains an algorithm on the training set using all the original features; (b) predictions are made on the test set, and a

performance measure (e.g. R^2) is recorded as p_0; (c) values of one of the features, i, in the test set are randomly shuffled and predictions are re-made on the test set; and (d) the performance associated with the shuffled feature i is recorded as p_i. The MDA feature importance for i is then

$$MDA(i) = \frac{p_0 - p_i}{p_0}. \tag{17}$$

We apply MDA to the data set with these specifications: we take Regression II with $T = 30$ days and 61 options using Ridge regression. The feature importance is shown in Figure 7. We observe two interesting patterns: (a) for both puts and calls, the more OTM, the higher the feature importance is; (b) calls are on average more important than puts. This is surprising as the importances are somewhat different from that of VIX's weighting scheme. It furthers shows that the distinction between results drawn in the risk-neutral measure and OOS predictions.

5 Conclusion

In this paper we focused on a machine learning-based realized variance prediction and indexing problem. Inspired by the predictability of VIX as a volatility index to SPX's 30-day realized variance, we proposed a new indexing method. More specifically, we formulated a regression problem to predict realized variance by using option price as features using SPX and its option data. We tested algorithms including linear and machine learning regression methods such as Ridge, Feed-forward Neural Networks (FNN) and Random Forest. It was demonstrated that when the algorithm is piece-wise linear, the prediction can be replicated by an option portfolio, which gives rise to a new volatility index.

We discovered that by combining the prediction model and the VIX-styled weighting scheme, the new index can achieve greater predictability and liquidity. In this approach the machine learning

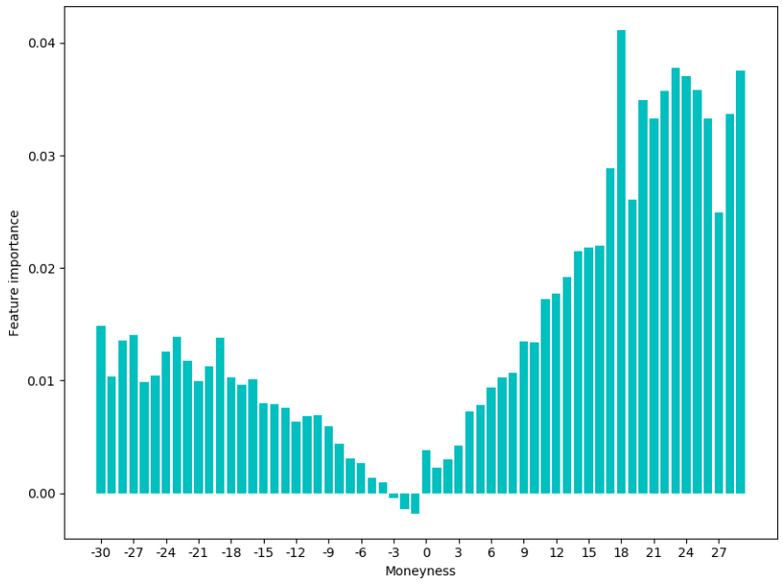

Figure 7 Feature importance across all options. The x-axis is the options' moneyness, e.g. -30 stands for the OTM put struck $-30 \cdot \Delta K$ away from ATM. Hence the left half is for OTM puts and the right half is for OTM calls. The y-axis is the feature importance of each option measure by the percentage improvement to the OOS volatility prediction.

algorithms are applied to correct the deviation between VIX's prediction and the actual realized volatility. It was shown that this approach outperformed VIX-styled weighting scheme and machine learning individually. Therefore, we argue that it represents a successful combination of human learning and machine learning. In addition, we employed a machine learning feature importance method to test every option's contribution to the prediction. While in VIX's weighting scheme the weights monotonically decrease with the strike price, we found that in the new approach the more out-of-the-money options have greater predictability and calls on average have higher importance than puts.

Before closing, we highlight a few future directions along the line. First it is worth looking into the predictability of other predictors such as macro and cross-asset factors. If these features are tradable, namely, they can be replicated by tradable securities, then the method can be generalized to a broader volatility index consisting of options and other instruments. Secondly, it will be useful to test the framework with higher frequency data. This way, with potentially much larger data sets, the power of machine learning may be enhanced. It will also be interesting to generalize the Regression II approach in this paper to other forecast problems and investigate whether ML can improve existing parametric models. As shown in this paper, the human + machine learning approach can outperform each of them individually. It will be intriguing to investigate if this works in other areas.

Notes

[1] Throughout this paper, we will use the terms volatility and variance interchangeably. For most cases, they are equivalent as volatility is just the square root of variance. Nonetheless, our prediction is formulated on variance instead of volatility as discussed in Section 3.

[2] Normalization is to divide the level by $100 \cdot \sqrt{252}$.

[3] In Gu *et al.* (2018), the authors report a similar incremental enhancement from ML in the case of asset pricing.

Acknowledgment

We would like thank Vasant Dhar, Rajesh Krishnamachari, Dacheng Xiu and Haoxiang Zhu for their valuable comments and suggestions. They are not responsible for any errors.

References

Andersen, Torben G., Per Frederiksen., and Arne D. Staal. (2007). "The Information Content of Realized Volatility Forecasts," Northwestern University, Nordea Bank, and Lehman Brothers.

Black, F. and Scholes, M. (1973). "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy* **81**(3), 637–654.

Busch, Thomas, Bent Jesper Christensen., and Morten Ørregaard Nielsen. (2011). "The Role of Implied Volatility in Forecasting Future Realized Volatility and Jumps in Foreign Exchange, Stock, and Bond Markets," *Journal of Econometrics* **160**(1), 48–57.

Bollerslev, Tim. (1986). "Generalized Autoregressive Conditional Heteroskedasticity," *Journal of Econometrics* **31**(3), 307–327.

Carr, Peter. and Dilip, Madan. (2001). "Towards a Theory of Volatility Trading," *Option Pricing, Interest Rates and Risk Management, Handbooks in Mathematical Finance*, 458–476.

Carr, Peter. and Liuren, Wu. (2009). "Variance Risk Premiums," *The Review of Financial Studies* **22**(3), 1311–1341.

Dupire, Bruno. (1994). "Pricing with a Smile," *Risk* 7.1, 18–20.

Engle, Robert F. (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation," *Econometrica* **50**(4), 987–1007.

Fulvio, Corsi. (2009). "A Simple Approximate Long-Memory Model of Realized Volatility," *Journal of Financial Econometrics* **7**(2), 174–196.

Federico, M. Bandi, Jerey R., and Russell, Chen Yang. (2008). "Realized Volatility Forecasting and Option Pricing," *Journal of Econometrics* **147**(1), 34–46.

Gu, S., Kelly, B., and Xiu, D. (2018). "Empirical Asset Pricing via Machine Learning,"

Hamid, Shaikh A. (2004). "Primer on Using Neural Networks for Forecasting Market Variables,"

Heston, Steven L. (1993). "A Closed-Form Solution for Options with Stochastic Volatility with applications to Bond and Currency Options," *The Review of Financial Studies* 6.2, 327–343.

Jeff, Fleming. (1998). "The Quality of Market Volatility Forecasts Implied by S&P 100 Index Option Prices," *Journal of Empirical Finance* **5**(4), 317–345.

Luong, Chuong., and Nikolai, Dokuchaev. (2018). "Forecasting of Realised Volatility with the Random Forests Algorithm," *Journal of Risk and Financial Management* **11**(4), 61.

Lopez de Prado, M. (2018). *Advances in Financial Machine Learning*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). 117 "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research* **12**, 2825–2830.

"VIX White Paper," CBOE.

Keywords: