Hierarchical Learning for Option IMPLIED Volatility Pricing

Henry Han
Fordham University
xhan9@fordham.edu

1. Introduction

The Black-Scholes-Merton (BSM) model has been ubiquitous in financial research since the 1980s notwithstanding critiques for its not all empirically valid assumptions such as log-normal distributions of stock prices [1-2]. It not only inspires rigorous explorations in option pricing but also continues to be a practical guide in trading for its useful approximation to reality [3]. The volatility in the BSM model is a type of forward volatility called implied volatility that measures investor’s confidence about the future risk of the stock. It is a future volatility of the stock but cannot be observed directly from the historical data. It indicates the current market expectation of future stock volatility and impacts market option prices seriously. Theoretically, implied volatility is the value that makes the theoretical price of an option under an option pricing model equal to its current market price [4]. Traders usually quote options by implied volatility rather than the price [5]. Implied volatility is also an important indicator of the financial market: it decreases in bullish markets and increases in bearish markets. The well-known VIX (Chicago Board Options Exchange (CBOE) volatility index) index is obtained by conducting implied volatility pricing for S&P 500 index options by tallying up and averaging relevant implied volatilities. The VIX index itself is not only a tradable asset, but also a daily market indicator, i.e. ‘investor fear index’, measuring option market risks followed by various investors and market participants. Moreover, high-accuracy implied volatility pricing plays an essential role in successful hedging and trading in the financial market, where trading is more and more dominated by high-frequency trading. The high-speed trading requires options to be priced accurately as much as possible. Therefore, implied volatility needs to be priced accurately for the sake of trading and market understanding.

Implied volatility pricing remains an important problem in finance in the era of big data where trading volumes and speed grow exponentially. It demands more accurate implied volatility pricing for the sake of option pricing that affects corresponding equity hedging procedures. In this study, we propose a novel hierarchical learning algorithm to enhance machine learning pricing to provide more accurate implied volatility prediction via a two-stage learning procedure. The proposed hierarchical learning works well especially for large amount of data or even big data. It boosts state-of-the-art machine learning models’ prediction at least 33.68% averagely on behalf of moneyness. Our study suggests that machine learning pricing performs better performance for OTM options, identifies Gradient boosting (GB) models outperform other peers in prediction, and provides an efficient procedure for high-accuracy option pricing, which has been challenging the implied volatility pricing community [4,5,6].

There are quite a lot of classic implied volatility pricing methods rooted from the BSM model. They can be classified as iteration and closed-form methods. The former solves a corresponding nonlinear equation numerically and the latter seeks a closed-form to
model implied volatility by using at-the-money (ATM) option price as an initial point [5].

An ATM option means its strike price is identical to the market price of the stock. Similarly, ITM and OTM both describes the moneyness of options. ITM means the stock’s market price of a call (put) option is above (below) its strike price. An ITM option has a positive intrinsic value, which is the difference between the market price and the strike price of the option. OTM refers to a call (put) with a strike price higher (lower) than the market price of the stock. Both OTM and ATM options only have time value rather than intrinsic value. OTM options need some time to be profitable for option buyers. OTM, ITM, and ATM can be well separated under knowledge-based visualization.

![Figure 1. The PCA visualization of OTM and ITM/ATM among the 6041 call options of the 2017 option dataset.](image)

Figure 1 visualizes OTM and ITM/ATM of 6041 calls of the 2017 option dataset used in this study into two well-separated groups under PCA. Since ATM options count only 21 samples, we group it with ITM that has 4328 samples into the ITM/ATM group. Generally, an ATM will appear as the boundary point between the ITM and OTM in PCA visualization.

The iteration methods include different types of root-finding methods that range from Newton-Raphson to Brent-Dekker method as well as their variants [6,7]. Some may suffer from a slow convergence to find implied volatilities, some cannot guarantee a convergence, and some improved ones have quite complicated implementations [6-9]. Furthermore, they cannot handle deep in-the-money (ITM) and deep out-of-the-money (OTM) option pricing well [4,6].

The deep ITM call/put options have strike prices much lower/higher than the market stock price. For example, a call whose strike price is at least $10 lower than the market stock price will be a typical deep ITM call. Their option prices are highly sensitive to the change of their stock prices instead of implied volatilities. The deep ITM options are usually preferred by long-term investors for its high intrinsic value. On the other hand, the deep OTM options have the strike prices significantly higher /lower for call/put than the stock price. Deep OTM options are considered as high-risk ones because they are more likely to bring high payoffs or losses.

The closed-form approximation methods seek an approximated formula to calculate implied volatility directly. They are extremely fast in implied volatility pricing compared to the iteration methods. They employ Taylor expansions on an at-the-money (ATM) option price point to obtain an approximated implied volatility formula [10]. Brenner and Subrahmanyan, Corrado and Miller, Chambers and Nawala made several successful attempts to optimize the approximation form of implied volatility [10-13]. However, the methods may perform poorly on out-of-the-money (OTM) options either [12].

The closed-form and iteration methods have well-established theoretical background. However, they are model-driven methods and cannot take advantage of large amount of available data in the market. Actually, a large option dataset can even be a nightmare for the slow-convergent iteration methods. Since different models have to be developed for different types of options, they face the challenges from the high modeling complexity and trade-off between the theoretical market assumptions and implied volatility pricing accuracy. More technically, both are not able to handle OTM option implied volatility pricing well.

During the recent two decades, the applications of machine learning have been developing rapidly and actively in predicting implied volatilities [14-17]. Machine learning implied volatility pricing is different from the model-driven methods for its data-driven. Almost all machine learning models do not rely on theoretical assumptions about markets and options. Instead, they dig knowledge and learn implied volatilities from input option data and construct an implied volatility prediction function.

Malliaris and Salchenberger apply neural networks to predict S&P 100 implied volatility by using past volatilities and other options market factors [14]. Gavrishchaka and Banerjee forecast stock market volatility using support vector machines (SVM), suggesting the efficiencies of working with high-dimensional inputs [15]. Yang and Lee predict the implied volatility distribution by using Bayesian kernel machines [16]. Zeng and Klabjan design an online adaptive primal support vector regression model to explore the implied volatility surface and realize dynamically updating support vectors to improve its efficiency [17].

The increasing data volumes in the financial market call for a data-driven way to exploit a large amount of data in implied volatility pricing. The machine learning approaches meet such an urgent demand. It makes it possible to derive an appropriate prediction model by listening to ‘data talks’ and achieve ‘more data, better prediction’. It can fully exploit the impacts of all
possible variables on implied volatilities in a learning procedure rather than a few ones specified by the model. It can do implied volatility pricing for almost any type of options given enough data. It brings a built-in pricing generalization for different types of options and avoids the modeling complexities to build different pricing models. On the other hand, it enables implied volatilities more interpretable for the real market. Implied volatilities can be easily interpreted as a function of a set of variables such as option price, stock price, strike price, and other related ones under a machine learning pricing.

However, different challenges remain though more and more machine learning methods are employed in implied volatility pricing [14-16]. Since the strong nonlinearity between implied volatility and its affecting factors from the market with a stochastic nature, implied volatility prediction is a ‘learning-hard’ problem that has a relatively low prediction accuracy for almost all machine learning models. How to enhance machine learning prediction accuracy remains an urgent challenge.

Besides, it remains unknown which machine learning models can ‘fit’ option data well though many have been employed [14,16,17]. It is also unknown how options in different moneyness status behave under the machine learning approaches. How OTM, ITM, and ATM options react differently in machine learning pricing? The answer to the query will unveil latent knowledge in implied volatility pricing on behalf of moneyness.

In this study, we propose a novel hierarchical learning algorithm to improve the accuracy of machine learning pricing. Unlike traditional learning, it presents a hierarchical learning scheme for data via a two-stage learning procedure. The potentially well-performed data will have a priority in learning. It boosts each machine learning model’s prediction by providing a selective mechanism to pick high-quality data involved in learning. It seeks a partial but better learning result as well as provides an adaptive learning scheme for the potentially poorly performed data. It is particularly suitable for solving the learning-hard problem such as implied volatility prediction. We apply the proposed algorithm to benchmark option datasets under state-of-the-art machine learning models. The proposed method boosts each model’s prediction at least 33.68% averagely on behalf of moneyness.

Furthermore, our study answers the unsolved queries in fintech: which machine learning models fit option implied volatility pricing better and How options with different moneyness statuses react differently in machine learning pricing? We identify gradient boosting (GB) as the best model for implied volatility pricing for its leading performance over its peers. Our study suggests that machine learning pricing performs best for OTM options though traditional models usually fail on the high-risk options [4,5,6].

Our study also presents a new option moneyness classification by grouping options as OTM, ITM, and NTM (near-the-money) for the sake of the robustness of machine learning pricing. The NTM options include ATM options and the shallow OTM and ITM options. An OTM/ITM option is an NTM if the absolute difference between its strike prices and the stock prices is less than $0.5. Such classification groups the boundary moneyness options into the same group so that ITM and OTM options are more representative in learning for their similar behavior in the market. Our result indicates that the implied volatilities of the ITM options can be more unpredictable than the OTM and NTM options though they have relatively small price move percentages in the market.

2. Hierarchical learning (HL) pricing

Hierarchical learning (HL) is a novel generic acceleration algorithm for any supervised machine learning. It consists of two stages: selective learning and adaptive learning. The selective learning stage is to eliminate ‘bad guys’ from training and test data. The ‘bad guys’ are the options with or potentially with poor performance under a given machine learning model. We name them as bad points, problematic points or potentially problematic points in our algorithm. Then the following learning is built upon the good-quality data. The adaptive learning stage builds customized training data for each ‘bad guy’ dropped in the selective learning stage to conduct learning adaptively to enhance learning performance. In our context, the bad guys are those options that negatively affect the prediction: they are options with or potentially with poor implied volatility pricing results. The proposed hierarchical learning has a specific algorithm to identify them even before learning.

2.1. Machine learning pricing evaluations

Machine learning implied volatility pricing is essentially a regression procedure where implied volatility is a response variable. Although $R^2$-based model evaluation theoretically works for general regression models, it cannot disclose real differences between two models involved in pricing, i.e. their $R^2$ values can be close or the same but their real performance can be quite different. Thus, we use mean square error (MSE) and prediction error to evaluate a learning model’s performance for their more concrete performance. The prediction error ($Err$) evaluates the
performance of the model on an individual data point (option).

\[ Err = |\bar{IV} - IV| \] (1)

where \( IV \) is the true implied volatility, and \( \bar{IV} \) is the predicted implied volatility. The mean square error (MSE) represents the average performance of the model for \( n \) options. The less MSE value signifies the better performance of the model.

\[ MSE = \frac{1}{n} \sum_{i=1}^{n} (IV_i - \bar{IV}_i)^2 \] (2)

2.2 Selective learning: the 1st stage of hierarchical learning

Hierarchical learning assumes we have enough training and test data in learning. It has two stages: selective learning and adaptive learning. It has three basic inputs: a machine learning model \( \theta \), training data \( X = \{u_i, v_i\}_1^n \), and test data \( X' = \{x_i, y_i\}_1^n \), where \( u_i \) and \( x_i \) are a training and test option, and \( v_i \) and \( y_i \) are their corresponding implied volatilities. But the implied volatilities in the test data are supposed to be unknown. We introduce selective learning as follows first.

Unlike traditional learning using the whole training and test data unselectively, selective learning eliminates the potentially poorly performed options by the proposed probing learning and nearest neighbor search to obtain good-quality training and test data that fit the model \( \theta \) better. Selective learning consists of probe learning, nearest neighbor search, and data clean three main components.

Probing learning is a warm-up learning procedure on the ‘whole training data’ to find initial ‘bad guys’ under the machine learning model \( \theta \). It views training data as an entirety to perform machine learning. To conduct probing learning, we first randomly split training data into two parts: train-train \( X_{t\text{train}} \) and train-test \( X_{t\text{test}} \) according to the threshold ratio specified for the training and test data size: \( \tau = \frac{\left|x\right|}{\left|x\right|+\left|x'\right|} \). For example, if \( \tau = 80\%:20\% \), then \( X_{t\text{train}} \) and \( X_{t\text{test}} \) will count 80% of training and 20% test samples respectively. The 80% data splitting will make sure there are enough training data available in the selective learning stage.

Problematic dataset collection. The machine learning model \( \theta \) is fitted by \( X_{t\text{train}} \) to predict the implied volatility for each entry of the train-test data \( X_{t\text{test}} \). Since all the true implied volatilities are known for \( X_{t\text{test}} \), we can identify the first group of ‘bad guys’: ‘bad points’, i.e. problematic points, whose absolute errors are top-ranked (e.g., 90th percentile) among the absolute errors of all entries in \( X_{t\text{test}} \), i.e.,

\[ S_p = \{x: Err(x) \geq Err_{90}, x \in X_{t\text{test}}\} \] (2)

where \( Err_\delta \) is the \( \delta \)th percentile of all train-test absolute errors (generally \( \delta = 90 \)). The problematic point set includes the options with poor prediction errors under the model \( \theta \). The 90th percentile choice can guarantee to obtain poorly-performed ‘bad points’ under a single ML model rather than using several ones under a relaxed cutoff.

Nearest neighbor search (NNS) finds the second group ‘bad guys’, i.e. potentially problematic points, which are close to the first group of ‘bad guys’ \( S_p \). They are defined as the union of the points close to each option \( x \in S_p \) in the whole training data \( X \) and test data \( X' \). For any point \( x \in S_p \), the nearest neighbor search is employed to find its \( k \) nearest neighbors in the training data \( X \) according to a distance measure,

\[ S_{x,k\text{train}} = \{a: D(x,a) < \tau_k, x \in S_p\} \] (3)

It answers the query: ‘which ones are the closest neighbors to a problematic point \( x \)’, where \( \tau_k \) is the distance to find the \( k \) nearest neighbors. Suppose an option \( x \) is marked as a ‘bad guy’ in the probing learning under a learning machine, say gradient boosting (GB), then the NNS calculates its distances to the other points in the training dataset, sorts the distances from the smallest to the largest, and identifies the top \( k \) (e.g. \( k = 5 \)) neighbors with the smallest distances to it. Similarly, its potentially problematic point set in the test dataset can be obtained by finding \( k' \) neighbors of \( x \),

\[ S_{x,k'\text{test}} = \{a: D(x,a) < \tau_{k'}, x \in S_p\} \] (4)

where \( k' \) is usually set as \( k \). Thus, the sets of potentially problematic points in the training and test are \( X_k = \bigcup_{x \in S_p} S_{x,k\text{train}} \) and \( X_{k'} = \bigcup_{x \in S_p} S_{x,k'\text{test}} \), respectively.

We employ the Manhattan distance in the nearest neighbor search for its advantage over other measures (e.g. the Euclidean distance) in identifying potentially problematic points for option data. The important reason why the Manhattan distance outperforms other peer distances is that it will not blur or offset some variables with small values (e.g. volatility, implied volatility, time to maturity, or even some option prices) and distinguish option similarity better. For example, the Euclidean distance will let the variables with the large values (e.g. strike price) dominate the distance calculation for their large values and shadow the contributions of the variables with small values in its squaring calculation. As a result, the ‘true differences’ between options may not be measured well and the quality of nearest neighbor search will be affected. The superior implied volatility pricing results under the Manhattan distance to those under the other distances in our study also support it.

Data clean cleans and updates training and test data by removing all the ‘bad points’: \( X_{\text{clean}} = X - X_k - S_p \)
and \( X'_{\text{clean}} = X' - X'_k \), where \( X_k \) and \( X_{k'} \) are the second group of potentially problematic points from the training and test dataset respectively and \( S_p \) is the first group of problematic points from the training dataset.

The cleaned training data \( X_{\text{clean}} \) will be used to predict the implied volatilities for the cleaned test data \( X'_{\text{clean}} \). Since the problematic points perform poorly under the machine learning model \( \theta \) and its associated points potentially to have a similar behavior under the model, it is reasonable to believe that removing them and their nearest neighbors in the training and test dataset will enhance the following machine learning pricing. The algorithm 1 describes the details of the proposed selective learning.

**Algorithm 1** Selective learning

**Input:**
- Training data \( X = \{u_j, v_j\}_j^N \)
- Test data \( X' = \{x_i, y_i\}_i^m \)
- Machine learning model \( \theta \)
- Training data partition threshold \( \tau \)
- The percentile cutoff \( \delta \)
- The number of nearest neighbors \( k, k' \)

**Output:**
- Cleaned training data \( X_{\text{clean}} \)
- Clean test data \( X'_{\text{clean}} \)

// Probing learning

1. // Partition training data as train-train and train-test under a threshold \( \tau \)
2. \( X^\text{train}_t, X^\text{test}_t \leftarrow \text{partitionTrainingData}(X, \tau) \)
3. // Training machine learning model \( \theta \) by train-train data
4. \( \theta \leftarrow \text{fit}(\theta, X^\text{train}_t) \)
5. // Predict implied volatilities for train-test data
6. \( \theta \leftarrow \text{predict}(X^\text{test}_t) \)
7. // Calculate absolute error for all entries in train-test data
8. \( \text{Err} = |\theta - v| \)
9. // Identify problematic points (bad points)
10. \( S_p \leftarrow \text{FindProblematicPoints}(\text{Err}, \delta) \)

// Nearest neighbor search

11. // Find potentially bad points in training and test data
12. \( X_k = X_k' = \{ \} \)
13. For \( x \in S_p \)
14. \( X_k, X_k' \leftarrow \text{NearestNeighborSearch}(x, k, X) \)
15. \( X'_k, X'_k' \leftarrow \text{NearestNeighborSearch}(x, k', X') \)
16. // Clean training and test data
17. \( X_{\text{clean}} \leftarrow X - X_k - S_p \)
18. \( X'_{\text{clean}} \leftarrow X' - X_k' \)

The selective learning stage gives better approximate learning by removing ‘bad data’ that does not match the learning machine before learning is made.

It can be used independently to handle the learning-hard problems by finding approximate solutions.

### 2.3 Adaptive learning: the 2nd stage of hierarchical learning

The selective learning stage brings a new learning mechanism by fitting a learning machine with high-quality data to find partial but better approximate solutions instead of the whole solutions. But sometimes we may not be able to simply disregard the ‘bad-guys’ in learning. They can be urgent options in pricing.

We propose the secondary stage of hierarchical learning: adaptive learning to handle this situation. The adaptive learning constructs a local individual training dataset for each ‘bad guy’ dropped from the test data and conducts learning adaptively.

The basic idea is to generate a local training dataset \( S_x \) for each bad point \( x \in X_k \) by searching its \( m \) (e.g. \( m = 20 \)) nearest neighbors in the training dataset \( X \) based on a distance (e.g. ‘manhattan’ distance). The local training dataset of \( x \) consists of its nearest neighbors, i.e. proximity points, which will have more advantages to predict its implied volatility than a genetic training dataset. The local training dataset \( S_x \) is employed to fit a machine learning model before predicting the implied volatility of \( x \). Such adaptive learning can enhance the prediction of each ‘bad guy’ because of a more correlated local training dataset construction. It is noted that the machine learning models in the adaptive stage should not be those models that require a large training dataset (e.g. deep neural networks) [18].

### 2.4 Model selection and extreme case handle

It is noted that hierarchical learning can reuse the known training results, which include the known ‘bad points’ and an updated better-quality training dataset for a given set of new options arriving. Hierarchical learning only needs to identify the ‘buddies’ of the known ‘bad points’ in the arriving test dataset and obtain a better-quality test dataset for the selective learning stage. The identified potentially bad points will enter the adaptive learning stage. However, training needs to be redone for the customized small training dataset for each potentially bad point in the test dataset that generally contains \( m = 20 \) samples in the adaptive learning stage.

It is recommended to use gradient boosting (GB) or similar extra-tree models in hierarchical learning for the sake of accuracy and computing speed [19-20]. Hierarchical learning does not require to use the same learning model in the selective and adaptive learning stages. But the adaptive learning stage needs to choose the models that work well for a small dataset (e.g.
support vector machines (SVM) or GB) [19,22]. Both DNN and SVM are not recommended in hierarchical learning because the former suffers from a long training time and the latter may encounter the scalar issue for large datasets in the selective learning stage.

The extreme case can be both training and test datasets can be quite small. The number of ‘bad-guys’ identified in probing learning can be few. It is suggested to only use the adaptive learning step under the situation for the sake of efficiency, where an individually customized training dataset will be built dynamically for each test entry.

2.5 ML models for hierarchical learning

As mentioned before, hierarchical learning is a generic method applied to any machine learning models. We employ six state-of-the-art models to evaluate its performance. They include k-nearest neighbor (k-NN), support vector machines (SVM), random forests (RF), gradient boosting (GB), extra tree (ET), and deep neural networks (DNN) [18-23]. We focus on the GB model for its importance in this study and the page limit.

Gradient Boosting (GB) seeks an implied volatility prediction function by optimizing the prediction functions of weak learners along with the negative gradient directions of the loss function. Unlike bagging ensemble learning that averages the prediction functions from the independent weak learners, the weak learners of GB are no longer independent [19]. On the other hand, each weak learner is added sequentially to the procedure to ‘boost’ learning results. GB learns its prediction function in an iteration model,

\[
\hat{f}_k = \hat{f}_{k-1} + r_k \frac{\partial L}{\partial \hat{f}_k}
\]

where \( L(\cdot) \) is the loss function of the learning model. The GB prediction function is initially formulated as the form of a weighted sum of decision functions \( h_i(\cdot) \) of the weak learners \( \hat{f}(\cdot) = \sum_{i=1}^{K} y_i h_i(\cdot) \), where the weights \( y_i \) grow in each step when a new weak learner is introduced. It is further optimized in gradient learning. The samples are no longer equally likely to be selected to train a weak learner in GB. Instead, those with larger predation errors are more likely to be selected for training, because GB learns from mistakes committed by the weak learners in the previous iterations. As a result, GB does not demonstrate overfitting robustness as RF [21]. There are different loss functions, but we choose the least square for its mathematical efficiency.

3. Results

3.1 Data acquisition and preprocessing

We developed python option acquisition software: OptionGlean for this study. It is designed specifically to retrieve option data from Yahoo Finance (https://finance.yahoo.com) and Nasdaq (https://www.nasdaq.com). The OptionGlean requires ticker names as input besides accepting user-specified input about options such as type, expiration, moneyness, and exchanges, etc. Three option datasets acquired by using OptionGlean include options in 2015, 2017, and 2018. We name the corresponding datasets as option data 2015, 2017, and 2018, which contain 25701, 14251, and 36646 options respectively. The options can be any type of options in the market. They are traded in Nasdaq and NYSE and are not high-frequently traded options.

Figure 2 compares the p.d.f.s of the four variables of the three datasets: option prices, and stock & strike price ratio (S/K), implied volatility, and time to maturity besides their boxplots of the 2017 option dataset. The p.d.f.s illustrate the 2015 and 2017 option datasets share more similarity between each other. We skip other variables such as option type, ask, bid, and volumes in the plots.
The reason to introduce a threshold in moneyness classification is to enhance machine learning pricing generalization. Traditional classification may not match the market reality very well. A very shallow OTM or ITM option can be equivalent to an ATM option because of a trade commission. Adding a threshold in option moneyness classification can make machine learning results in pricing close to the market reality. Furthermore, using NTM data rather than only ATM data can prevent the failure of machine learning because the small size of ATM data can lead to a learning failure.

The call and put distributions generally are balanced for the 2015 and 2017 option data. But they are imbalanced for the 2018 OTM and ITM data. The OTM data has 13849 and 6501 calls and puts, but the ITM data has 444 calls and 471 puts. Compared to 2015 OTM data, the NTM data has the least amount in each dataset. For example, the 2015 NTM data consists of 728 calls and 777 puts; the 2017 NTM data only consists of only 444 calls and 471 puts; the 2018 NTM data has 684 calls and 618 puts.

### 3.2 Selective learning pricing

![Image](https://example.com/image.png)

Figure 3. The MSE comparisons of the six machine learning models using and without using selective learning on different ITM, OTM and NTM options.

We employ the three benchmark datasets to evaluate the effectiveness of the proposed selective learning in implied volatility prediction under the six machine learning models in terms of MSE values. That is, we only evaluate the first stage of hierarchical learning. Each dataset is partitioned as 80% training and 20% test in learning. We choose the percentile cutoff $\delta = 90$ in the selective learning stage to identify the potential ‘bad points’ under the ‘manhattan’ distance. The neighborhood sizes in NNS are set as $k = k' = 10$.

Figure 3 compares the MSE values of selective learning under the six models for the OTM, ITM, and NTM datasets. The MSE values of all the models under selective learning are lower or even much lower than those of the original ones. This suggests the effectiveness of the proposed selective learning in implied volatility pricing. In particular, selective learning achieves impressive advantages on the 2015 OTM, ITM and NTM datasets, but almost all original models have a poor performance on them.

Given a dataset, we define an enhancement ratio $\eta$ to evaluate the impact of selective learning under a machine learning model $\theta$:

$$\eta = \frac{\text{mse} - \text{mse}_{\theta}(\eta)}{\text{mse}_{\theta}}$$

where $\text{mse}$ and $\text{mse}_{\theta}(\eta)$ represent the MSEs before and after selective learning. Except for only one negative enhancement ratio, the other ratios on the 2015 option dataset are all between 18.84% and 92.24%. DNN achieves 87.74%, 84.47%, and 80.56% enhancement ratios for the 2017 OTM, ITM, and NTM data respectively.

The ITM and OTM options have more favorable enhancement ratios than the NTM options. We validate it by calculating the harmonic mean of the enhancement ratios as follows.

An enhancement ratio matrix $\eta$ is generated by the six machine learning models on the OTM, ITM, and NTM datasets. For example, the 2015 option dataset will generate a 3x6 enhancement ratio matrix. We calculate the median of each row $\eta_{\text{med}} = \text{median}(\eta_i)$ and group them as a median enhancement ratio row vector $\eta_{\text{med}} = \{0.6317,0.8950,0.5768\}$. By doing the same calculations for the 2017 and 2018 option data, we have a 3x3 matrix by grouping all median enhancement ratio vectors.

$$\eta_{\text{med}} = \begin{bmatrix} 0.6317 & 0.8950 & 0.5768 \\ 0.6077 & 0.5585 & 0.2644 \\ 0.4378 & 0.2914 & 0.2949 \end{bmatrix}$$

Each column in the matrix represents the median enhancement ratio for the OTM, ITM, and NTM data respectively. We calculate the harmonic mean $\eta_h = \Sigma_{i=1}^n \frac{n}{1/\eta_{ij}}$ for each column to get the expected enhancement ratios on behalf of moneyness: $\eta_h = \{0.5442,0.4732,0.3368\}$. It indicates that the OTM, ITM, and NTM data achieve 54.42%, 47.32%, and 33.68% average enhancement ratios in selective learning. Therefore, the OTM and ITM options may be more suitable for machine learning implied volatility pricing than the NTM data. Such a result is also verified by each model’s median enhancement ratio, i.e. OTM and NTM datasets always have a better median enhancement ratio for each model. For example, DNN has 84.47% and
54.98% median ratios for the OTM and ITM data, but only 33.30% for the NTM data. Besides, GB outperforms its peers in pricing no matter in selective learning or original learning for its almost the lowest MSE values, though it does have a poor case on the 2015 OTM data. It is not the ML (machine learning) model with the highest enhancement ratio in selective learning. But it always attains lower MSEs than others no matter in original learning or selective learning. In summary, selective learning can improve performance of the machine learning models in pricing for different option data at different moneyness. It enhances pricing via machine learning more efficient by dropping possible bad guys in training and test by providing a better pricing by screening a portion of high-quality options.

Checking error distributions in pricing. It is desirable to check the changes in the prediction error Err = |P – IV| distribution from the models under selective learning compared to the original models. We compare the Q1, Q2, and Q3 values of the prediction errors to evaluate the impacts of selective learning on each model. All Q1, Q2, and Q3 values decrease after selective learning for each learning model. The only slight difference is probably that the decrease of Q3 can be more obvious than those of Q1 and Q2. It suggests that selective learning can help to enhance the dataset with large prediction errors for its ‘bad guys removing mechanism’.

![Figure 4](image_url) **Figure 4.** The comparisons of the median prediction errors (Q3) of the six machine learning models under selective learning and its original models on different option datasets.

Figure 4 compares the Q3 values of the original learning models with those of selective learning (SL) on behalf of the OTM, ITM, and NTM data. The selective learning Q1 values, denoted by ‘SL_error’ in Figure 4, decrease for almost all models compared to the original errors no matter the statuses of moneyness. All lines have obviously decreasing slope values. GB overperforms other peers in pricing as it does in the Q1, Q2 comparisons. It also indicates the ITM and OTM data can be more suitable for machine learning pricing though different models may be suitable for different options at different statuses of moneyness. For example, the 8-layer DNN model in our experiment may not work that well under selective learning for some 2017 NTM options in which the original model with small or median prediction errors. But it achieves good performance in the 2017 ITM and OTM datasets in which the original ML models with large prediction errors.

### 3.3 The adaptive learning pricing

We apply the 2nd stage adaptive learning to price those dropped the ‘bad guys’ of the test data in the first stage hierarchical learning. We set the neighbor size m=20 to build the local training dataset for each ‘bad guy’ in the test data. We drop DNN in the secondary stage of hierarchical learning (2nd HL) because it is not suitable for small datasets. Figure 5 compares the MSEs of original learning and selective learning for all potentially problematic points of each test dataset. Each model is colored by the color of the OTM, ITM and NTM data they are applied to. The points above the dashed-curve in each scatter plot indicate their original MSEs are lower than the hierarchical learning MSEs (HL-MSEs). It means the 2nd stage hierarchical learning fails on them. The points on the curve indicate the 2nd stage hierarchical learning has the same-level performance as the original one or only slight improvements. The points below the curve are those whose 2nd stage hierarchical learning results are better than their ones and save the ‘bad guys’ that are dropped in the first stage of selective learning.

![Figure 5](image_url) **Figure 5.** The comparisons of different MSEs of the potentially problematic points in the test data before and after
adaptive learning: the 2nd phrase of the proposed hierarchical learning on behalf of OTM, ITM, and NTM data.

The OTM options seem to be more favorable than the others in the 2nd stage hierarchical learning (2nd HL), though different data demonstrate quite different patterns. The 2nd HL still slightly improves the performance of the potential problematic points. For example, ET and SVM both achieve decent improvements for the ‘bad guys’ of the OTM and ITM options of the 2015 option datasets. However, only ET attains a good performance on the ‘bad guys’ of 2017 OTM data. Alternatively, ET and RF both have good pricing performance for the ‘bad guys’ of 2018 OTM, NTM and ITM data.

### 4. Discussion and conclusion

We propose a novel hierarchical learning approach to enhance machine learning implied volatility pricing. It overcomes the weakness of implied volatility pricing via machine learning by boosting each model’s prediction accuracy on different option data on behalf of moneyness. It somewhat challenges the existing learning philosophy by proposing a new learning concept, that is, fit a learning machine with the most suitable high-quality data to seek approximate and partial but better learning results. It is particularly good for solving the learning-hard problems like implied volatility pricing by machine learning.

This study finds that the OTM and ITM options have a favorable status in machine learning implied volatility pricing than the NTM options besides identifying GB as a robust pricing model. It is possible to apply our approach to price the deep OTM and ITM options that are challenging the traditional model-driven approaches [6-7]. Furthermore, we find the OTM options can be more suitable for the 2nd stage hierarchical learning than its ITM and NTM peers. Therefore, we can conclude that the OTM options are more favorable in machine learning pricing and gradient boosting (GB) can be the best machine learning models. Compared to the existing and previous work, the proposed hierarchical learning not only shows better pricing accuracy than the existing machine learning peers, but also demonstrates a leading advantage in pricing the OTM options over the traditional model-driven ways besides its powerful generalization [7,12, 14,17]. It is worthwhile to point out the hierarchical learning pricing results still may not be able to completely compete with some well-designed model-driven methods for the ITM or OTM options in terms of accuracy though they can’t take advantage of a large amount of available data [5,17]. Our study provides a high-performance machine learning tool for implied volatility pricing by overcoming the weakness of the existing machine learning pricing approaches. Although more deep learning models are not investigated in this study, the proposed hierarchical learning can also be applied to the models to handle the challenge from the big option data pricing form high-frequency trading [24,25].

An interesting concern can be about the possible overfitting risk of the proposed hierarchical learning. The generalization of the learning machine under hierarchical learning will increase or decrease? The answer is that hierarchical learning will contribute to decreasing the overfitting risk because of its hierarchical learning scheme. This is because the data quality in the training and test is improved in the selective learning stage and customized training dataset is built for each bad point individually in the adaptive learning stage. Compared to the original ‘ad-hoc’ training data, the customized training dataset is more targeted and high-quality one, which decreases the overfitting risk. Thus, the implied volatility prediction function generalization in hierarchical learning is optimized because of high-quality training and test data in the selective learning stage and customized training data in the adaptive learning stage.

The experiments in this study run on a computer under Mac OS with 16 GB RAM and quad-core CPU with 2.8 GHz. It takes about 20 minutes to complete the implied volatility pricing for the largest 2018 option dataset with 36646 options in this study under the DNN models. However, SVM can take more than an hour computing time for the large kernel matrix generated from the dataset. The GB and ET models demonstrate advantage over the two by using much less time (e.g. <5 minutes).

The complexity of the nearest neighbor search (NNS), which is the essential component in the proposed hierarchical learning also challenges the computing for a large input data. The complexity of the NNS is \(O(nl + ml)\) where \(n\), \(l\), and \(m\) are the numbers of training data points, bad points, and test data points respectively. If we follow the training/test and \(X_\text{train}, X_\text{test}\) partition ratios and the bad point selection criteria, where \(l = 0.02n\) and \(m = n/4\), in this study, the complexity of the NNS will be \(O(0.025n^2)\). Such a complexity can be large when \(n\sim10^4\).

Furthermore, it is worthwhile to point out that the second stage in the hierarchical learning that builds an individual customized training dataset for each bad point can be expensive also, especially when there are many of the bad points identified in the selective learning stage. The weakness can be fixed by doing a batch training dataset construction for a batch of bad points. For example, the customized training dataset \(T_s\)
for a batch of $l$ bad points will be union of each one’s customized training dataset: $T_i = \bigcup_{l=1}^{l} T_i$. Such a batch approach will lower the high computing complexity issue. At the same time, the k-d tree speedup can be applied to the NNS to lower its complexity.

Our on-going work is to optimize hierarchical learning pricing by identifying possible prototypes of the OTM, ITM, and NTM data through traditional model-driven approaches so that the potentially problematic point search can be augmented with more prior knowledge. It is likely to conduct hierarchical learning for some specific types of options such as deep OTM/ITM to further polish it and observe their behaviors to get higher accuracy in pricing. Furthermore, we are interested in employing state-of-the-art clustering algorithms (e.g. HDBSCAN) to enhance the efficiency in finding potentially problematic points [26]. At the same time, we plan to integrate hierarchical learning in genetic algorithm oriented implied volatility pricing besides other specifically designed deep learning models for implied volatility prediction and related analytics topics [27-28].

5. References