Expert Systems With Applications 163 (2021) 113799

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Option pricing using Machine Learning

Codruț-Florin Ivașcu

Bucharest University of Economic Studies, 5-7, Moxa Street, Bucharest, Romania

ARTICLE INFO

Keywords:
Machine Learning
Option pricing
Nonparametric methods
Implied parameters

ABSTRACT

This paper examines the option pricing performance of the most popular Machine Learning algorithms. The classic parametrical models suffer from several limitations in term of computational power required for parametric calibration and unrealistic economical and statistical assumptions. Therefore, a data driven approach based on non-parametric models is are well justified. Most of the previous researchers focus especially on the neural networks method (NN), the other algorithms being unexplored. Beside NN, this paper also analyses the performance of the Support Vector Regressions and Genetic Algorithms and propose three other Decision Tree methods, respectively Random Forest, XGBoost and LightGMB. In order to emphasize the power of this algorithms, a comparison with classical methods like Black-Scholes and Corrado-Su with both historical and implied parameters have been conducted. The analyzes were performed on European call options who have as underlying asset the WTI crude oil future contracts. Machine Learning algorithms outperform by a great margin the classical approaches regardless of the moneyness and the maturity of the contracts.

1. Introduction

The massive losses registered by the traders on the financial derivatives market have become recurring topics in economic news. Reports of financial institutions with losses of hundreds of millions or even billions of dollars have shocked the financial world. As a result of these disasters, many reputed institutions have gone bankrupt. These stories, though important, fascinating and occasionally outrageous, have led to a negative connotation regarding the use of derivatives. However, properly used, they are key tools for proper risk management, so a good evaluation model is required.

Among the most used derivatives are options. Their huge popularity can be attributed to the work of Black and Scholes (1973) who proposed a stochastic model for calculating their market value. After almost 50 years since publishing, the model is still extensively used by practitioners all over the world. Despite its success, the model fails to capture empirical phenomena like a leptokurtic distribution of returns (not a normal one) or the existence of a volatility smile that suggests that volatility is not constant as the model assumes.

Many financial engineering models have tried to relax the Black-Scholes model restrictions and improve the empirical results. Some of the most popular approaches are (i) statistical series expansion, such as the models suggested by Corrado and Su (1996) or Jarrow and Rudd (1982), (ii) local volatility models, such as the Schroder model (1989), (iii) stochastic volatility models such as Hull and White (1987), Heston (1993) or Schöbel and Zhu (1999) and (iv) models with jumps like Merton (1976) and Bates (1996). However, although these models have proved to be more efficient in terms of valuation, they are much more computationally costly, requiring many implicit parameters to be calibrated. Also, they do not present a closed form solution of the valuation equation, meaning that numerous optimization proceedings are needed. In addition, they are still restricted by some economic and statistical assumptions, like no-arbitrage or market completeness assumptions.

Therefore, in order to efficiently price financial derivatives in a way that is both fast and accurate, another line of research based on data-driven models have been developed. If options are considered a functional mapping between the contracted terms (inputs) and the premium (outputs), then the Machine Learning algorithms become very handy. There is substantial literature in this domain, starting with the seminal study of Hutchinson et al. (1994). They argued that a data-driven approach is adaptive and responds to structural changes in the data in ways that parametric models cannot. Also, since the models do not rely on restrictive parametric assumptions, they are robust to the specification errors that limit classical models. In addition, a non-parametric approach is flexible enough to be used in the valuation of a wide variety of derivatives. The authors demonstrated that a Neural Network (NN) is an excellent vehicle to approximate the option pricing function, being more accurate and computationally more efficient than the Black-Scholes model. Later, Anders et al. (1998), Garcia and Gençay (2000), Yao et al. (2000), Amilon (2003), Yang and Lee (2011) or Liu et al.

E-mail address: ivascu_codrut@yahoo.com.

https://doi.org/10.1016/j.eswa.2020.113799
Received 23 March 2020; Received in revised form 9 July 2020; Accepted 26 July 2020
Available online 31 July 2020

(2019) among others confirm that an artificial neural network is very performant in terms of option valuation.

Other papers focus also on different algorithms. Grace (2000) suggests that Genetic Algorithms are also fitted to valuate options. Park, Kim and Lee (2014) realize a comprehensive study that compares the performance of both state-of-arts parametric no-arbitrage models such as Merton and Heston and non-parametric machine learning models such as a Support Vector Regression (SVR) model (also checked by Huang (2008) and Liang et al. (2009)) and Gaussian Process (GP). They found that SVR, GP and NN have comparable results, all of them outperforming parametrical approaches. Of course, these advantages come with some costs, respectively a large quantity of historical data required for the training set which may not be available for less traded derivatives. Also, if the dynamic of the asset is very well modeled, then the parametric models will be better. However, these conditions occur rarely enough so a non-parametric approach can add a great practical value in option pricing.

This paper brings the following contributions to the literature. Firstly, continuing the direction propose by Park et al. (2014), more models have been used. Besides Neural Networks, Support Vector Regression and Genetic Algorithms, I have proposed three different Decision Trees algorithms, respectively Random Forest, XGBoost and LightGBM which was not tested before in option pricing literature. However, all of them have been successfully used in other financial fields such as stock prediction (Basak et al., 2019) and risk management (Chang et al., 2018). To emphasize the power of these newer models, a comparative analysis between non-parametric and parametric models have been tested. From author knowledge, this is the most comprehensive study conducted.

Secondly, compared to other papers that test their performance on options on indices (S&P500 index in the case of Hutchinson et al. (1994). Gençay and Qi (2001) or Gradojevic et al. (2009), Swedish Stock Index in the case of Amilon (2003), KOSPI 200 ELW in the case of Han and Lee (2008), Yang and Lee (2011) and Park et al. (2014)), I preferred to use call options on crude oil futures. The options on this asset have one of the highest liquidity on the Chicago Mercantile Exchange market, being used both for speculation and especially, for hedging. Oil has a significant impact on the real economy, oil market reports being followed by both domestic and private consumers but also banks and state institutions.

Robustness and homogeneity tests have been conducted. Machine Learning algorithms outperformed by a great margin the classical stochastic models taken as benchmarks in term of prediction power, additive boosting algorithms having the best performance.

The paper is structured as follows: Chapter II presents a briefly introduction of the models, Chapter III the data and the methodology and Chapter IV and V empirical results and conclusions.

2. Models description

Options are financial instruments that give the holder the right (but not the obligation) to buy or sell an underlying asset. The stochastic nature of financial assets requires modeling using nonlinear and multivariate functions, making option valuation extremely challenging. Classical approaches have numerous statistical assumptions regarding the behavior of the market or data. On the other hand, Machine Learning algorithms are highly effective in modeling nonlinear data without this kind of limitations. This section presents briefly the non-parametric models used in the analyses.

2.1. Support Vector Regression

The Support Vector Machine algorithm, with the extension of Support Vector Regression (SVR), has been developed to a large extent at AT&T Bell Laboratories by Boser et al., 1992, Guyon et al., 1993, Vapnik, 1995, Vapnik et al., 1997.

The model produced by Support Vector Regression depends only on a subset of the learning data because the cost function ignores all training data appropriate to the model prediction. Suppose that the training data $\{(x_1, y_1), \cdots, (x_l, y_l)\} \subset \mathscr{X} \times \mathbb{R}$ are known, where $\mathscr{X}$ represents the multidimensional space determined by the input parameters (e.g. $\mathscr{X} = \mathbb{R}^d$). In our case, these can be the price of the underlying asset and the exercise price. In ϵ-SV regression, (Vapnik, 1995) the goal is to find a function $f(x)$ that has at most a deviation of ϵ from the target prices y_i for the entire training set, provided that $f(x)$ has a slope as small as possible. In other words, we are not interested in errors if they are smaller than ϵ, but we will not accept a deviation greater than that.

Of course, in practice, these constraints are very restrictive, often wishing to allow for prediction errors. Analogous to the “soft margin” from the cost function of the Support Vector Machine model, two new variables, namely ξ_i, ξ_i^*, were introduced, in order to relax the optimization conditions. Therefore, we will arrive at the formulation presented in Vapnik (1995):

$$\begin{aligned} & minimize \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ & subject\ to \begin{cases} y_i - \langle\omega, x_i\rangle - b \le \epsilon + \xi_i \\ \langle\omega, x_i\rangle + b - y_i \le \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \ \ge 0 \end{cases} \end{aligned} \tag{1}$$

The constant $C > 0$ represents the trade-off between the reduced slope of the function f and the magnitude of the deviation over ϵ that will be tolerated. In the case of nonlinearity, Kernel function will be applied. One of the most common function is radial basis function (RBF), respectively $k(x_i, x) = \exp(-\gamma\|x_i - x\|^2)$. Using the derivation steps presented by Smola and Schölkopf (2004), the new Lagrange function will take the form:

$$L := \frac{1}{2}(\|\omega\| + b) + \frac{C}{2}\sum_{i=1}^{l} p_\epsilon^2\left(k(x_i, x_j)\omega + b - y_i, a\right), \tag{2}$$

where $p_\epsilon^2 = p(x - \epsilon, a)^2 + p(-x - \epsilon, a)^2$ and $p(x, a) = x + \frac{1}{a}log(1 + \exp(-ax)), a > 0$.

2.2. Random Forest

The Random Forest model, developed by Breiman (2001), is one of the most successful models available today. The supervised learning procedure operates according to the “divide and conquer” principle: the input data will be divided into groups, from each group a decision tree will be created, and the results will be aggregated for a single prediction.

What contributed greatly to the popularity of the model was that it can be applied to a very large range of predictions, requiring very little calibration parameters, both for regression problems and for classifications. The method is generally recognized for predictive accuracy and for the ability to work with numerous data in a multi-dimensional space.

An important role in understanding the model is played by two main components, namely begging (Breiman, 1996) and Classification and Regression Trees (CART) – split criterion (Breiman et al., 1984). Begging is an aggregation scheme that generates a set of data by bootstrapping from the original set, makes a prediction for each set and decides the final result by average. The CART-split criterion is used in the construction of a single tree to find the best “cut” perpendicular to the axes. At each node in each tree, the best cut is selected by optimizing this informative criterion, based on Gini impurity (on classifications) or square errors of prediction (on regressions).

2.3. XGBoost

Since its introduction in 2014, XGBoost (short for Extreme Gradient Boosting) has become the most widely used technique in machine learning competitions such as hackathons or Keagle. According to Chen (2016), the success of the model is its huge adaptability and convergence speed up to 10 times higher than the other popular solutions. Also, it can be easily parallelized, offering a huge advantage in data mining issues. In the presentation of the model I will use the notations used by Chen and Guestrin (2016), the creators of the algorithm.

Given a dataset with n observations and m explanatory variables, the tree $D = \{(x_i, y_i)\}cu(|D| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R})$will use K additive functions to predict the output:

$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), where f_k \in \mathscr{F}$. Here q represents the structure of each tree that links the example with the corresponding leaf index. T is the number of leaves on the tree. Each f_k corresponds to an independent tree structure q and a weight for the leaf ω. Unlike a decision tree, each regression tree contains a score expressed as a continuous function in each leaf, where ω_i represents the score in leaf i. For a given example, we will use the decision rules in the tree (given by q). to classify it into leaves and calculate the final prediction by summing up the score obtained in the corresponding leaves (given by ω). To train the model, we will minimize the following objective function:

$$\mathscr{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{3}$$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2$. Here l is the cost differentiable function that measures the difference between the prediction $\hat{y}_i$ and the target value y_i. The second term, Ω, is a function of penalizing the complexity of the model (of the regression tree). The additional regularization term helps reduce overfitting. Intuitively, the objective function will choose the model with the best prediction and the lowest complexity.

The model is then trained in an additive manner. Formally, we will note $\hat{y}_i^{(t)}$ the prediction$i$ from the iteration of t. We will add a function f_t in order to minimize the objective function:

$$\mathscr{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{4}$$

2.4. Light Gradient boosting Machine (LGBM)

LightGBM is another gradient boosting framework that is based on decision tree algorithms. It was introduced in the Machine Learning community by a team of Microsoft's data scientists, respectively Ke et al. (2017). The authors tried to increase the efficiency and scalability of other boosting algorithms when the feature dimension is high and data size is large. Their method grows the trees by applying the leaf-wise strategy, while other ensemble learning algorithms (like XGBoost) use the level-wise (or depth-wise) strategy. LightGBM benefits from two novel techniques, called Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

The computation cost is reduced due to these techniques. GOSS excludes a significant proportion of data instances with small gradients, just the rest being used to estimate the information gain. Thus, GOSS can obtain a quite accurate estimation with a much smaller data size. EFB will bundle mutually exclusive features to reduce their dimension.

2.5. Neural networks

A neural network is a set of interconnected processing nodes whose functionality is based on an animal's neural network. The processing power of the network is determined by the weights given to each node. The weights are obtained from a learning process (or adaptation) from a training dataset. Neural networks are applied in statistical analysis and data modeling being used as a substitute for non-linear regressions or classification algorithms.

Artificial neural networks were first introduced in an article by McCulloch and Pitts (1943) where the authors presented a model of how biological neurons are interconnected and work together. Although there are numerous artificial networks with different properties, one of the most popular models is the multilayer perceptron which is based on the work of Fukushima (1980). As the name suggests, multilayer perceptron (MLP) is made up of several layers of neurons (also called perceptron). The independent variables are entered in an input layer, and the variables to be estimated in the output layer. Among them, there are one or more subjectively chosen hidden layers.

The functionality of an individual neuron is simple and direct. Each neuron summates all the signals sent to it, adds a bias term and performs a non-linear transformation. The activation function (transfer) is an increasing monotonic function, most often a logistic function, hyperbolic tangent or ReLu type. A linear function could also be used, but it could not explain the nonlinear behavior of the output data. The signal transformed into a neuron is forwarded by a certain weight to another neuron in another layer, and the process is repeated. The value of the output neurons is compared to the actual value of the data (target value) and a cost or error function is calculated. The error is propagated back into the network, and the weights are adjusted so that the cost function is minimized. This procedure is iteratively performed until the network estimates the output data with acceptable accuracy. The learning algorithm by weight adjustment is called back propagation and was initially derived by Rumelhart et al. (1986) using gradient descent.

2.6. Genetic algorithms

Genetic algorithms represent a metaheuristic inspired by the natural selection process and belong to a more general class of evolutionary algorithms. Genetic algorithms are used to generate optimization solutions based on genetically inspired operators, such as mutation (the genome of an individual will undergo a random modification), crossover (a new individual is created by recombining chosen parts from different individuals) and selection (an individual program is copied in the next population). They were introduced by John Holland (1960) based on concepts of Darwin's theory of evolution.

A genetic algorithm works on individuals who are solutions to problems in an artificial environment. Each individual is described by a constant number of chromosomes (usually one) represented by bits or alphanumerical characters. Each chromosome is composed of genes (features) arranged in a linear succession.

Each individual has a fitness value associated with it. The fitness value describes the individual's ability to survive in the environment. Operations that aim to mirror the Darwinian principles of reproduction and survival of the fittest are used on a set of individuals which forms the population.

In the beginning, the population with individuals will be randomly generated. Based on their fitness value and using the genetic operators, a new generation of individuals will be created. The process is iterative and it stops when some terminal conditions (declared a priori) are met. For more details, an interested reader can check Nordin et al., 1998.

3. Data and methodology

The data were represented by the price of all European call options that had as support the WTI oil futures contracts (Reuters Index Code: CL) traded on the Chicago Mercantile Exchange between 03.01.2017 and 14.11.2018. The paper consisted of analyzing 1465 options, representing 121,488 daily prices, after filtration. All data was provided by Thomson Reuters Tick History DataStream.

The data were filtered so that in the analysis were used only options that had a maturity of less than 1 year, a moneyness (S / K) less than 1.7 and a premium greater than 1. I have also eliminated the options that did not satisfy the lower bound condition according to Anders (1996):

$$C_t \geq S_t - Ke^{-r_t(T-t)} \tag{5}$$

In the case of the interest rate, yield term structure from American T-bills was used. Through linear interpolation the risk-free rate for each maturity of each option was determined.

The data used was divided into two distinct sets, respectively an in-sample set called in the Machine Learning terminology train set, and an out-of-sample set, called a test set. The proportion of sets is 80% train set and 20% test set. To avoid overfitting on a particular period, data have been shuffled.

Pricing performance was calculated using the root mean square error (RMSE) criterion. The errors were calculated both according to the duration of the contract, respectively short term (<3 months), medium term (3–9 months) and long term (>9 months), as well as according to moneyness, respectively Deep In The Money ($S/K < 0.8$), In The Money ($S/K \in (0.8, 0.96)$), At The Money ($S/K \in (0.96, 1.04)$), Out of The Money ($S/K \in (1.04, 1.2)$) și Deep In The Money ($S/K > 1.2$).

As benchmark, two parametric models have been used, respectively Black-Scholes model (1973) and Corrado-Su model (1996). To the best of the author's knowledge, no other paper has compared a statistical series expansion model with Machine Learning models. For a comparison with a stochastic volatility model or a model with jumps, an interested reader can check Park et al. (2014). The methodology for quantifying the pricing performance of parametric models is based on determining the implied parameters by nonlinear OLS. The theoretical price of the option is defined as $C_t^{th*} = C_t^{th}(\Psi_t^*)$. Ψ_t^* is a vector defined as a solution of the following minimization program:

$$\Psi_t^* := Arg\left\{\min_{\Psi_t}\left[\sum_{j=1}^{n}\left[C_j^{th}(\Psi_t) - C_j^{obs}\right]^2\right]\right\} \tag{6}$$

where Ψ_t is the vector of the estimated values of the implied parameters. Parameter n represents the number of prices observed on the market on the trading day t. Therefore, a vector with the implicit parameters obtained for each trading day is estimated. The result will be a single set of parameters that will correspond to all the prices observed on that day. The Black-Scholes model has a single implied parameter, respectively the volatility σ, and the Corrado-Su model, three parameters, respectively the volatility, skewness and kurtosis.

To highlight the difference between using implicit parameters and using historical parameters, I estimated the theoretical value of a call using historical volatility. To calculate it, I used the approach described by Hutchinson et al. (1994) where$\hat{\sigma} = s/\sqrt{60}$, where s is the standard deviation for the most recent 60 daily compound yields. Of course, volatility was annualized.

Support Vector Regression was calibrated using the Kernel radial bias function to give it nonlinearity. I used the parameter $\gamma = (numberofdimensions)^{-1}$. The tolerance level for the stop criterion is set to 10^{-4}. Penalty term C was considered 1. The value of ϵ which determines the corridor under which the error associated with the training set is not penalized was set to 0.1.

Random Forest was estimated using 100 decision trees with a maximum depth of 20 levels. The criterion for measuring the quality of the division of a tree is given by the function of mean squared error. XGBoost and LightGBM uses as a gbtree booster with a learning rate of 0.3. Tree depth is 20, as in Random Forest. The objective function is to minimize the smallest squares.

The best configuration for neural networks will be found by trial. Thus, the in-sample data series will be divided into two subsets, namely the training set and the validation set (80% / 20%). The networks will be trained using the training data set and will be tested on the validation data set. The network configuration with the best result will be chosen to be tested on the same set as the rest of the models. Over 80 different combinations of models will be checked, changing both the network depth (the number of hidden layers), the number of neurons, the type of the optimization function and the activation function of each layer.

Table 1
Pricing Error (RMSE) for 5 different models.

	M1	M2	M3	M4	M5
Input	S/K,τ	S, K,τ	S, K, τ ,r	S, K, τ ,r,σ_{60}	S, K, τ ,r,$\sigma_{60}, \sigma_{30}, \sigma_{10}$
Output	C/K	C	C	C	C
$BS_{implied}$	1.40	1.40	1.40	1.40	1.40
BS_{60}	1.99	1.99	1.99	1.99	1.99
CS	**0.91***	0.91	0.91	0.91	0.91
SVR	2.02	0.59	0.59	0.59	0.58
RF	1.19	0.44	0.42	0.39	0.39
XGB	1.06	0.34	0.33	**0.31***	**0.31***
LGBM	1.08	**0.33***	**0.32***	**0.31***	0.32
NN	1.25	0.62	0.57	0.54	0.56
GA	1.36	0.81	0.80	0.77	0.79

Note: The results of machine learning algorithms are on out of sample data and the results for benchmark models (Black-Scholes and Corrado-Su) are on in sample data. Analyzed data represents 20% of the total data. Stared numbers represents the lowest pricing error for each model.

In the case of genetic algorithms, the first implementation of genetic programming for the evaluation of options was made by Keber (1999). He used the procedure to determine a formula for evaluating American put options. For the terminal values he used the variables $S, K, r, \tau, \sigma$and the constants π and e. The mathematical functions used were $+$, $-$, $*$, $/$, $\sqrt{x}$, $\ln(x)$, x^2, x^y, the distribution function of the standard normal $\phi(x)$, the logical operators $<$, $\leq$, $=$, $>$, $\geq$ and the conditional operators IF, THEN and ELSE. However, in order to demonstrate the predictive power of the algorithm, I did not give as input the constants π and e and the functions $\sqrt{x}$ and x^2as the algorithm should be able to find the values from the other functions and/or random inputs. To keep the model as simple as possible, I did not use either logical or conditional operators. However, the functions of minimum and maximum, as well as sinus and cosine were added.

The population size was set at 2000 individuals, with reproduction, crossover and mutation having the probabilities 0.8, 0.9 and 0.4 respectively. The mutation was used also on the structure and on the leaves. The number of generations has not been fixed, the stopping condition being only exceeding an error threshold. The fitness function was defined as:$f(c) = \sum_{i=1}^{n} \Delta\left(C_i, C_i^o\right)^2$, where C_i refers to the value calculated by the genetic program, and C_i^o to the value of the call observed in the market. The error threshold was considered the Black-Scholes error.

4. Results

This section discusses the empirical results of the models presented in section II. All the results presented are on out of sample data because I consider ineffective showing the in sample errors. It is known that Machine Learning models will overfit the data in the train set, so the errors will be even lower. To emphasize the prediction power of these algorithms, the comparison will be made with the in-sample error of parametric models (Black-Scholes and Corrado-Su).

To find optimal features space, 5 models were estimated, each with different inputs and outputs. The results can be seen in table 1. Black-Scholes has been estimated using both implied volatility and historical volatility (60 days standard deviation). As expected, the model with implied volatility has lower error. All parameters of Corrado-Su model were implied. Here, S represents the WTI oil price, K is the strike price, τ is the time expressed in years until the maturity of the option contract, r is the risk free rate and σ_n is the standard deviation of the most recent n continuously compounded daily returns of crude oil.

Model 1 configuration is the same as the one proposed by Hutchinson et al. (1994). In order to reduce dimensionality, he assumes that the evaluation function is homogeneous of degree one in S and K, respectively $f(S_t, K, \cdots)/K = f(S_t/K, 1, \cdots)$. This approach has been intensively

Table 2
Out of sample RMSE for Model 4 in term of moneyness and time to maturity.

	Model	All	DOTM	OTM	ATM	ITM	DITM
All	BSimplied	1.40	0.72	0.92	1.29	0.84	1.93
	BS60	1.99	1.20	2.11	2.55	1.54	1.94
	CS	0.91	0.32	1.11	1.17	0.80	0.71
	SVR	0.59	0.12	0.23	0.36	0.53	0.81
	RF	0.39	0.07	0.13	0.20	0.40	0.54
	XGB	**0.31***	**0.04***	**0.09***	**0.12***	0.33	0.43
	LGBM	**0.31***	0.06	0.10	0.13	**0.30***	**0.42***
	NN	0.64	0.13	0.30	0.43	0.59	0.87
	GA	0.53	0.11	0.39	0.52	0.43	0.66
Short term	BSimplied	0.90	–	1.03	1.14	0.57	0.98
	BS60	0.93	–	1.16	1.23	0.60	0.98
	CS	0.78	–	1.70	1.12	0.79	**0.42***
	SVR	0.61	–	0.17	0.36	0.50	0.76
	RF	0.48	–	0.10	0.19	0.46	0.58
	XGB	0.39	–	**0.05***	**0.11***	0.41	0.46
	LGBM	**0.37***	–	0.08	0.11	**0.34***	0.46
	NN	0.58	–	0.34	0.39	0.49	0.71
	GA	0.44	–	0.44	0.41	0.37	0.51
Medium term	BSimplied	1.55	0.51	0.98	1.42	0.91	2.15
	BS60	2.05	1.06	1.89	2.61	1.58	2.15
	CS	1.00	0.25	1.24	1.29	0.82	0.78
	SVR	0.60	0.07	0.22	0.35	0.54	0.84
	RF	0.38	0.07	0.13	0.20	0.37	0.53
	XGB	0.31	**0.03***	**0.09***	**0.13***	0.30	0.43
	LGBM	**0.30***	0.08	0.11	0.14	**0.29***	**0.42***
	NN	0.68	0.12	0.28	0.41	0.62	0.95
	GA	0.57	0.10	0.40	0.55	0.44	0.72
Long term	BSimplied	1.40	0.76	0.74	0.93	1.02	2.60
	BS60	2.79	1.23	2.62	3.67	2.61	2.68
	CS	0.68	0.33	0.56	0.65	0.69	0.91
	SVR	0.47	0.12	0.26	0.42	0.56	0.72
	RF	0.28	0.07	0.14	0.22	0.40	0.40
	XGB	0.19	**0.04***	**0.10***	**0.11***	0.28	0.28
	LGBM	**0.18***	0.06	0.10	0.12	**0.26***	**0.27***
	NN	0.56	0.13	0.35	0.57	0.64	0.79
	GA	0.48	0.12	0.35	0.52	0.49	0.68

Note: Stared numbers represents the lowest pricing error for each moneyness with respect to each contract's time to maturity.

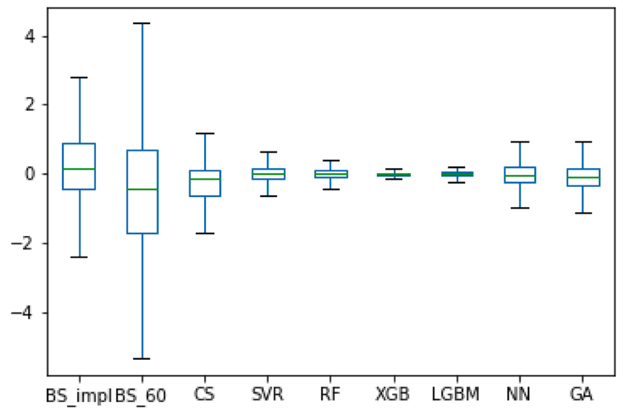

Fig. 1. Boxplot of pricing error for out of sample period in terms of USD. Note: The body of the candle represents the distribution of errors between 25% and 75% of the data and the wick represents the maximum and minimum recorded error. The results are shown for Model 4.

used in literature (Hutchinson et al., 1994; Amilon, 2003; Yang and Lee, 2011 etc.) with good reported results. In this case, all non-parametric algorithms excepting SVR have indeed outperformed Black-Scholes with implied volatility, but not Corrado-Su.

Model 2 uses the same parameters as Model 1, but not the homogeneity assumption. Surprisingly, all algorithms outperformed both parametric models by a great margin. Decision trees algorithms (RF, XGB, LGBM) have the best performance by far. SVR outperformed NN, in accordance with the results obtained by Park et al. (2014).

Table 3
Pricing errors for 7 non-overlapping periods.

	$BS_{implied}$	CS	SVR	RF	XGB	LGBM	NN	GA
1	1.79	1.27	1.73	0.84	**0.77***	0.78	0.78	0.93
2	1.61	1.23	1.65	1.25	**0.96***	1.18	1.00	1.12
3	1.94	1.87	0.88	0.82	0.84	**0.80***	0.92	0.87
4	1.90	1.68	1.04	0.89	**0.78***	0.79	0.87	1.14
5	1.33	1.25	0.94	0.93	**0.84***	0.89	0.75	1.01
6	1.58	1.12	**0.78***	1.07	0.85	0.79	0.99	0.99
7	1.43	1.03	**0.47***	0.64	0.58	0.51	0.73	0.69

Note: Train set is represented by 1 year of data and test set by the next 1 month. Stared numbers represents the lowest pricing error in each period.

For the next models, more features have been considered. Model 4 uses the same inputs as Black-Scholes with historical volatility. In Model 5 different timeframes for volatility have also been added to allow algorithms to learn a volatility structure (also done by Amilon, 2003). Best results have been obtained using Model 4 feature space. Later results will be shown on this model.

Table 2 presents a much granular pricing error, in term of moneyness and time until maturity. In all cases, Machine Learning algorithms have systematically outperformed the benchmark. Best performances have been recorded by additive boosting algorithms, respectively by XGBoost and Light Gradient Boosting with a predictive error up to 17 times lower than Black Scholes with implied volatility. As expected, Black-Scholes with historical volatility has the worst performance.

Fig. 1 presents the boxplot of pricing errors in term of USD. The body of the candle represents the errors distribution between the first and third quantile. The wick represents the maximum and minimum

Table 4
Heterogeneity analysis.

	DOTM	OTM	ATM	ITM	DITM
SVR	0.10	0.22	0.36	0.54	0.67
RF	0.07	0.13	0.22	0.38	0.47
XGB	**0.05***	**0.08***	0.16	0.32	0.40
LGBM	**0.05***	0.08	**0.15***	**0.31***	**0.39***
NN	0.18	0.24	0.42	1.12	0.80
GA	0.10	0.32	0.41	0.40	0.62

Note: Train and test set are represented by the options with the same moneyness. The proportion is 80% train with 20% test. The star numbers represent the lowest pricing error in term of RMSE for each moneyness.

recorded error. As you can see, all non-parametrical models have the distribution of error centered on 0 with the tails almost symmetrical. XGBoost seems to slightly overprice the options.

For robustness, a different prediction approach has been conducted. Train set was represented by 1 year of data and test set by the next 1 month of data. Using a rolling window of 1 month, seven non-overlapping periods have been tested. The results can be seen in Table 3. Again, non-parametric models have superior prediction capabilities. XGBoost performed the best in 4 of 7 cases. Surprisingly are the results of the SVR surpass Decision Trees algorithms in the last 2 periods.

However, the differences between non-parametric and parametric models have been diminished. An explication could be the smaller train set compared with the first analysis, one of the biggest drawbacks of these models being the necessity of huge data set.

A heterogeneity analysis has also been conducted. So far, additive boosting models (XGBoost and LGBM) perform best on full data training. This analysis compares the predictive errors on the same types of options. In other words, the models have been trained on one type of moneyness and tested on the same moneyness. Train-test sets have the same 80%-20% report. The results can be seen in Table 4. Again, additive boosting algorithms have had the best performance on each type of option. Surprisingly enough, the overall results haven't improved drastically compared with the full sample analysis. Furthermore, there are cases on which the models perform worse. For example, XGBoost, in the case of at the money moneyness, recorded a RMSE of 0.12 on full data train and 0.16 on only at the money train set. These results emphasize once more the need of bigger training set. Additionally, this could also suggest that the differences in moneyness are not influencing the learning procedure.

5. Conclusion

This paper provides an overview of the most popular regression models in the Machine Learning sphere. The models were applied for pricing European call options on WTI crude oil, financial instruments with a very high degree of nonlinearity due mainly to the stochastic nature of the underlying asset.

The methods used were Support Vector Regression, Random Forest, XGBoost, Light Gradient Boosting Machine, Neural Networks and Genetic Algorithms. Two parametric models widely used in practice, namely Black-Scholes (with historical and implied volatility) and an extension of it, Corrado-Su, were considered as benchmark. The underlying asset on which the analysis was performed was represented by crude oil future contracts.

Machine Learning models outperformed by a great margin the parametric models, on both shuffle data analysis and on forecasting analysis. A heterogeneity test has also been conducted. By far, the best models are represented by additive boosting models, namely XGBoost and LightGBM, the newest developed algorithms.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Amilon, H. (2003). A neural network versus Black–Scholes: A comparison of pricing and hedging performances. *Journal of Forecasting, 22*(4), 317–335.

Anders, U., Korn, O., & Schmitt, C. (1998). Improving the pricing of options: A neural network approach. *Journal of forecasting, 17*(5–6), 369–388.

Basak, S., Kar, S., Saha, S., Khaidem, L., & Dey, S. R. (2019). Predicting the direction of stock market prices using tree-based classifiers. *The North American Journal of Economics and Finance, 47*, 552–567.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies, 9*(1), 69–107.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144–152).

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy, 81*(3), 637–654.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and regression trees. *Wadsworth Int. Group, 37*(15), 237–251.

Breiman, L. (1996). *Bagging predictors. Machine learning, 24*(2), 123–140.

Breiman, L. (2001). *Random forests. Machine learning, 45*(1), 5–32.

Chang, Y. C., Chang, K. H., & Wu, G. J. (2018). Application of eXtreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. *Applied Soft Computing, 73*, 914–920.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Corrado, C. J., & Su, T. (1996). Skewness and kurtosis in S&P 500 index returns implied by option prices. *Journal of Financial research, 19*(2), 175–192.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics, 36*(4), 193–202.

Garcia, R., & Gençay, R. (2000). Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics, 94*(1–2), 93–115.

Gençay, R., & Qi, M. (2001). Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks, 12*(4), 726–734.

Grace, B. K. (2000). Black-Scholes option pricing via genetic algorithms. *Applied Economics Letters, 7*(2), 129–132.

Gradojevic, N., Gençay, R., & Kukolj, D. (2009). Option pricing with modular neural networks. *IEEE transactions on neural networks, 20*(4), 626–637.

Guyon, I., Boser, B., & Vapnik, V. (1993). Automatic capacity tuning of very large VC-dimension classifiers. In *Advances in neural information processing systems* (pp. 147–155).

Han, G. S., & Lee, J. (2008). Prediction of pricing and hedging errors for equity linked warrants with Gaussian process models. *Expert Systems with Applications, 35*(1–2), 515–523.

Heston, S. 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.

Holland, J. G. (1960). Teaching machines: An application of principles from the laboratory. *Journal of the Experimental Analysis of Behavior, 3*(4), 275.

Huang, S. C. (2008). Online option price forecasting by using unscented Kalman filters and support vector machines. *Expert Systems with Applications, 34*(4), 2819–2825.

Hull, J., & White, A. (1987). The pricing of options on assets with stochastic volatilities. *The journal of finance, 42*(2), 281–300.

Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance, 49*(3), 851–889.

Jarrow, R., & Rudd, A. (1982). Approximate option valuation for arbitrary stochastic processes. *Journal of financial Economics, 10*(3), 347–369.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., … Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems* (pp. 3146–3154).

Keber, C. (1999). Option pricing with the genetic programming approach. *Journal of Computational Intelligence in Finance, 7*(6), 26–36.

Liang, X., Zhang, H., Xiao, J., & Chen, Y. (2009). Improving option price forecasts with neural networks and support vector regressions. *Neurocomputing, 72*(13–15), 3055–3065.

Liu, S., Oosterlee, C. W., & Bohte, S. M. (2019). Pricing options and computing implied volatilities using neural networks. *Risks, 7*(1), 16.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics, 5*(4), 115–133.

Merton, R. C. (1976). Option pricing when the underlying stock returns are discontinuous. *Journal of Financial Economics, 3*, 125–144.

Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic programming.* Springer.

Park, H., Kim, N., & Lee, J. (2014). Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over KOSPI 200 Index options. *Expert Systems with Applications, 41*(11), 5227–5237.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning representations by back-propagating errors. nature, 323*(6088), 533–536.

Schöbel, R., & Zhu, J. (1999). Stochastic volatility with an Ornstein-Uhlenbeck process: An extension. *Review of Finance, 3*(1), 23–46.

Schroder, M. (1989). Computing the constant elasticity of variance option pricing formula. *the. Journal of Finance, 44*(1), 211–219.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing, 14*(3), 199–222.

Vapnik, V. (1995). *The nature of statistical learning theory*. Berlin: Springer.

Vapnik, V., Golowich, S. E., & Smola, A. J. (1997). Support vector method for function approximation, regression estimation and signal processing. In *Advances in neural information processing systems* (pp. 281–287).

Yao, J., Li, Y., & Tan, C. L. (2000). Option price forecasting using neural networks. *Omega, 28*(4), 455–466.

Yang, S. H., & Lee, J. (2011). Predicting a distribution of implied volatilities for option pricing. *Expert Systems with Applications, 38*(3), 1702–1708.