DOI 10.48614/bara.7.2022.6047

Real Options Valuation using Machine Learning Methods

Levan Gachechiladze
Business School
Georgian-American University

Abstract

This paper follows work on establishing comprehensive framework for investment projects valuation discussed in [3], [8], [12], and [20]. Previous work is focused on capturing strategic value of investment projects while also incorporating strategic decisions of competitors. Two main methodologies that comprise such valuation framework are Real Options Analysis and Game Theory. In this paper, it is attempted to price real options using Machine Learning (ML) methods. First, selected machine learning models are trained to predict option prices as given by The Black-Scholes formula. Having shown some promise by work discussed in [22] and [23], real-world data has been selected for pricing options and then training machine learning models on them. Finally, various investment projects have been simulated to price option to expand using Cox-Ross-Rubinstein binomial model discussed in [4] and then train machine learning models to predict it. This, in turn, has potential to incorporate market competition implicitly in the value of the strategic option during training process. Hence, machine learning approach can become real options pricing method that is valid not only for monopolistic markets. With this aim, section 1 of the paper gives brief introduction of option pricing methods, section 2 uses Nasdaq Futures historical prices for training ML models to price financial options, and section 3 uses simulated investment projects for training ML models and pricing options to expand. Complete code is available at github.com/leongache/Real-Options-Valuation-using-Machine-Learning-Methods.

1. Overview of classical option pricing methods

This section briefly defines most commonly used methods for option valuation. The purpose of this section is to identify what makes each method usable in the first place and then what makes it less attractive in some practical use cases.

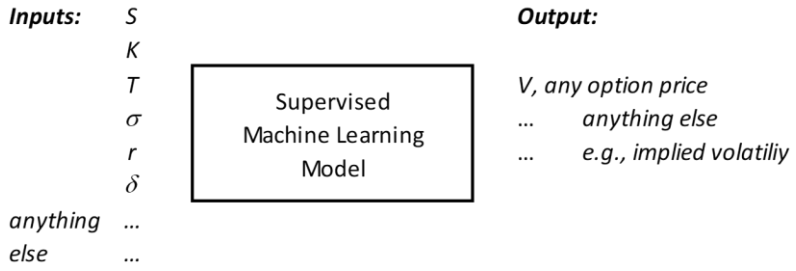
Let's start with The Black-Scholes model, also known as the Black-Scholes-Merton model. Most attractive feature of the model may be its closed form solution, it's simplicity despite complicated mathematics behind it. On the other hand, the Black-Scholes model makes certain assumptions that in certain market conditions may result in prices that deviate from real-world results. E.g., constant risk-free rate and volatility of underlying asset become less evident in extreme/turbulent markets with unpredictable high volatility. Though, improvements of the Black-Scholes model that account for some of its disadvantages exist, its unrealistic assumptions make it hard to use the plain formula for accurate option pricing in all market conditions.

Most flexible option pricing model that can improve on many of the Black-Scholes model limitations is Binomial Option Pricing model. It's commonly used to price American-style options that can be exercised before expiration and have flexibility to price options with any payoff formula as well as incorporate variable inputs for risk-free rate, volatility, etc. in dynamic fashion. Although, it's very hard to predict what those inputs will be equal to in future dates. Even if accurately predicted, incorporating variable inputs increases models' complexity to the point when it may become hard to formulate.

Lastly, commonly used option pricing method is Monte Carlo simulation where numerous random paths for the price of an underlying asset are generated, each having an associated payoff. Then present value of payoffs is computed, and their average becomes an option price that values in all simulated scenarios. Just like Binomial Option Pricing model, this method can incorporate any option payoff and dynamically introduced inputs. Moreover, this method is not restricted to a single or any distribution of underlying asset unlike models with closed-form solutions as given by the Black-Scholes. All that gives Monte Carlo simulation substantial number of use-case in real-life applications. Main disadvantage of the method stays to be heavy computational load as it requires a large number of simulations to improve average accuracy.

Next section in this paper introduces more recent approach to option pricing using artificial intelligence, mainly, machine learning (ML) methods. Using same or more number of inputs as in classical options pricing methodologies ML methods can be trained from both simulated and historical data to "learn" either observed option price or theoretical one given by option pricing method of our choice. Success of ML model will depend on quality of training data and its properties for generalization among other things. In case of creating successful ML model that accurately predicts option prices on out-of-sample data, one can conclude that disadvantages of classical option pricing models will no longer be a concern for both academia and practitioners. At its simplest form, machine learning approach is illustrated in figure below.

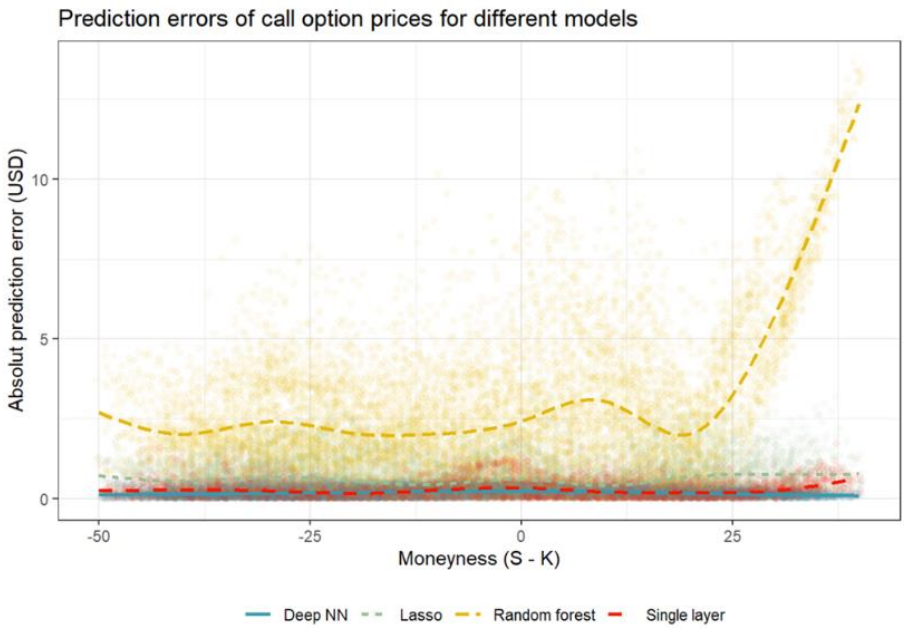
Figure 1 input-output structure of supervise machine learning models



2. Training ML models to learn The Black-Scholes formula

This section carries an experiment of training most popular Machine Learning models to predict call and put option prices as given by The Black-Scholes formula. Building up on work in [22] this example uses real-world prices of continuous Nasdaq Futures contracts publicly available on yahoo finance webpage. Figure 2 shows results from [22] where more than 1.5 million random parameter constellations were used to simulate options prices and train models on them. Figure clearly identifies that all but one ML model under consideration seem to be able to price call options with different moneyness levels.

Figure 2 Prediction error of call options prices for different ML models, source in ref. [22]



Above figure suggests that ML models can be used to price options, and in this section similar experiment is carried out now using actual data instead of simulated one for illustrative purposes.

Using daily prices for Nasdaq futures starting from earliest available date as of 19 Sep 2001, at-the-money call and put prices are computed using following input parameters:

- annual standard deviation of continuously compounded returns as a volatility input,
- annual risk-free rate of 3%, and
- time to expiration of 20 trading days.

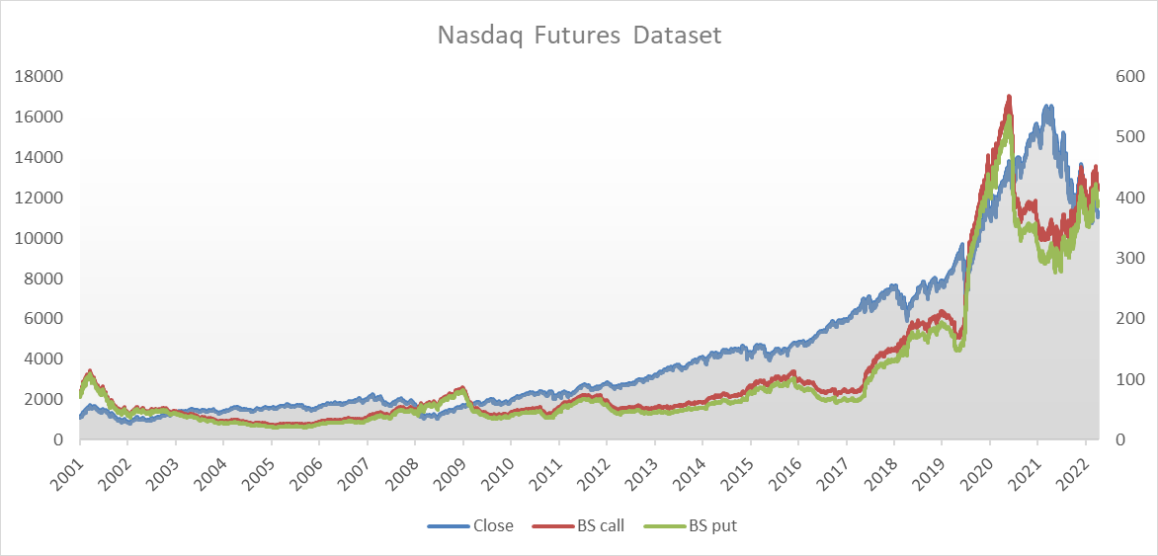
Code snippet below shows functions used to compute option prices for Nasdaq Futures data till most recent date as of time of writing, 23 Dec 2022.

Code Snippet 1 The Black-Scholes call and put prices, source https://www.codearmo.com/python-tutorial/options-trading-black-scholes-model

```
# functions for option prices
N = norm.cdf
def BS_CALL(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + sigma**2/2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S * N(d1) - K * np.exp(-r*T)* N(d2)


def BS_PUT(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + sigma**2/2)*T) / (sigma*np.sqrt(T))
    d2 = d1 - sigma* np.sqrt(T)
    return K*np.exp(-r*T)*N(-d2) - S*N(-d1)
```

Figure below shows complete dataset used for training and testing of ML models; this includes historical data of Nasdaq Futures as well as calculated option prices using formulas in Code Snippet 1.

Figure 3 Nasdaq futures historical prices and option prices by The Black-Scholes (BS) model



Despite high volatility in times series, most recent 5% of complete data was selected as out-of-sample for testing purposes. Out of most common and fundamentally different ML models, the following 4 were selected:

- K-nearest neighbors (KNN)
- Multi-layer Perceptron (MLP)
- Gradient Boosting Trees, specifically, LightGBM implementation of it
- Support Vector Machines (SVM)

In addition, standard scaler was used to standardize training and test data to have standard normal distributions before training any of above models. Moreover, a small sample of parameter distributions were selected in advance to run parameter optimization per each model, separately. For that purposes, Randomized Grid Search algorithm was selected. Specific ML configurations used are shown and described below.

For KNN:

- *leaf_size*: starting from 5, till 100, with steps of 5
- *n-neighbors*: starting from 5, till 100, with steps of 5

For MLP, fixed random state, 500 max iterations, lbfgs[1] solver, tolerance of 1e-8 and:

- *alpha*: [0.01,0.001,0.0001],
- *hidden_layer_sizes*: [(5,5,5,), (5,), (5,5,)]

For LightGBM, subsample and colsample_bytree of .8 and:

- *n_estimators*: starting from 50, till 1000, with steps of 50
- *max-depth*: starting from 3, till 10, with steps of 1

For SVMs:

- *C*: starting from .01, till 1, with steps of .01

[1] lbfgs - an optimizer in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) using a limited amount of computer memory.

- *gamma*: starting from .01, till 1, with steps of .01

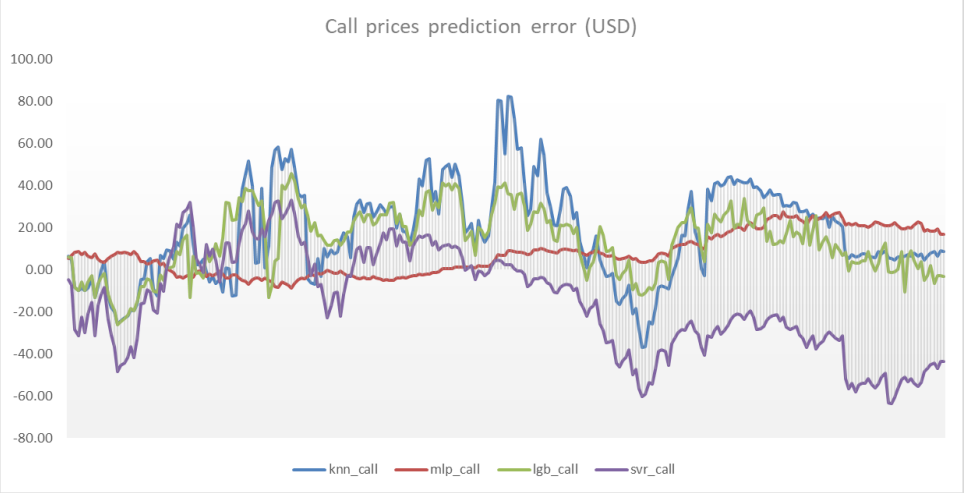
It's worth noting that default configuration for Randomized Grid Search has maximum number of iterations set to 10 and k-fold cross-validation set to 5 folds. Consequently, all models above will get 5-fold cross validation and 10 randomly chosen parameter combinations. Only exception is MLP as it only has 9 possible parameter combinations to search from, in which case exhaustive search will be implemented. It's important to note that there is no right architecture choice for neural networks in general, not so even for such simple networks as MLPs. For that reason, an arbitrary number of nodes and 3 hidden layer variations is chosen without any particular reason. Same is true for distribution ranges of other ML model parameters. Code snippet below summarizes final modules used for an experiment.

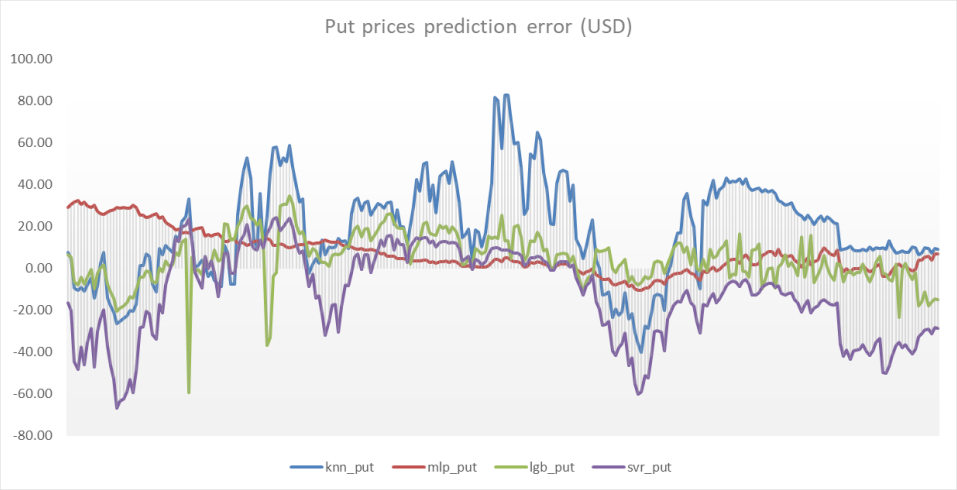
Code Snippet 2 Configuration of machine learning models used for option pricing

```
knn_search = RandomizedSearchCV(KNeighborsRegressor(), {'leaf_size':np.arange(5,100,5),'n_neighbors':np.arange(5,100,5)})

mlp_search = RandomizedSearchCV(MLPRegressor(random_state=0,max_iter=500,solver='lbfgs',tol=1e-8),
                                {'alpha':[0.01,0.001,0.0001],'hidden_layer_sizes':[(5,5,5,),(5,),(5,5,)]})

lgb_search = RandomizedSearchCV(LGBMRegressor(boosting_type='goss',subsample=0.8,colsample_bytree=0.8),
                                {'n_estimators':np.arange(50,1000,50),'max_depth':np.arange(3,10,1)})

svr_search = RandomizedSearchCV(SVR(),{'C':np.arange(.01,1,.01),'gamma':np.arange(.01,1,.01)})
```

Using total of 5380 days of real data, and having selected only 5% for testing purposes, models were trained on 5111 days and tested on next 269 days. Separate training for call and put option prices generated following dollar value errors per model class.

Figure 4 Prediction errors on call and put prices from Nasdaq futures





Considering lack of training data, prediction errors look satisfactory with MLP having smallest errors for both call and put option prices and, hence, can be considered most suitable ML model for the task. Table below summarizes average error and min max ranges for all models individually.

Table 1 Prediction error metrics per ML method

metric	knn_call	mlp_call	lgb_call	svr_call	knn_put	mlp_put	lgb_put	svr_put
average error	22.66	9.61	17.04	23.64	24.06	8.74	9.74	19.33
min error	0.20	0.04	0.00	0.29	0.25	0.00	0.09	0.14
max error	82.39	27.39	45.59	63.58	82.94	32.32	59.41	66.93
max neg error	-37.02	-8.64	-25.99	-63.58	-40.25	-10.54	-59.41	-66.93
max pos error	82.39	27.39	45.59	33.11	82.94	32.32	34.79	24.17

Table above identifies MLP and LGB as better models then KNN and SVR given prediction performance on test dataset under consideration.

Based on promising results from above, next section attempts to train selected ML models to price not financial but Real Options for investment projects valuation.

3. Pricing real options, namely, option to expand using ML methods

Most common examples of Real Options used for strategic investment valuations are:

- Option to Expand
- Option to Contract
- Option to Defer
- Option to Abandon
- Option to Choose
- Option to Switch Resources

If investment project has a strategic value in it, at least one from above or some other advance real option is built into the project and its valuation becomes inevitable part of comprehensive analysis of value. Unfortunately, it's widely known that Real Option valuation doesn't account for the effect of competitors, and for those reasons is only valid for a monopolistic environment. Variety of techniques to account for competition within real option valuation framework has been developed and one example is to incorporate Game Theory discussed in [3]. While plausible, market competition may not follow equilibrium strategy

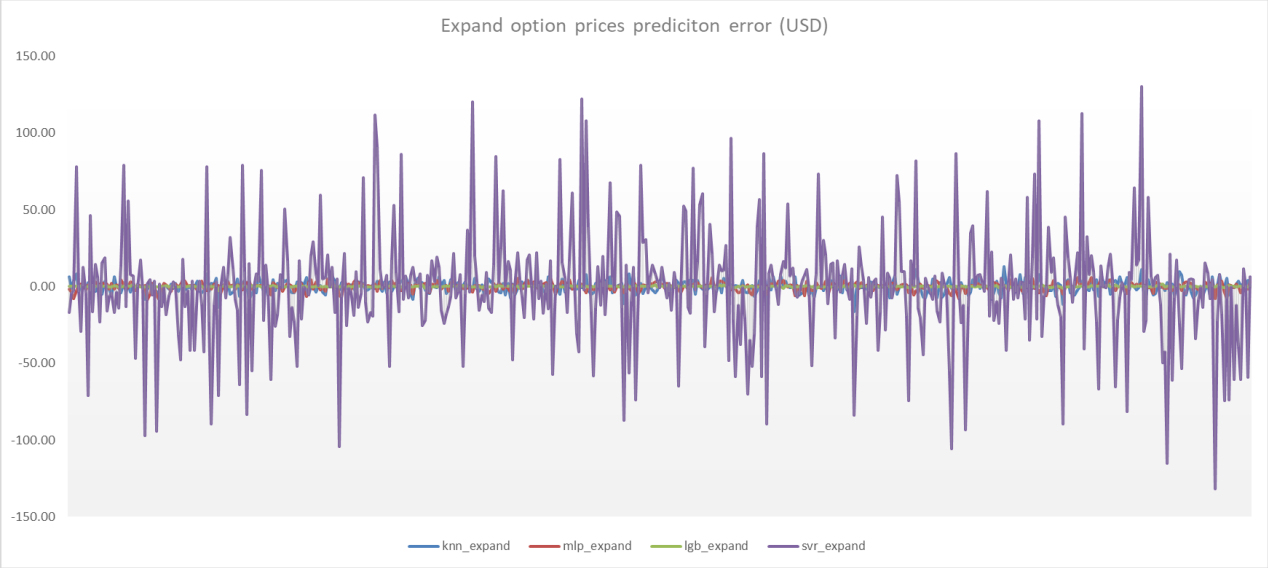
derived by Game Theory models and then biases valuation even further. In this section, option to expand is computed for simulated investment projects data and machine learning models configured in previous section are trained to predict expansion option prices on unseen data. If concept of pricing real options with ML methods is proven, then training ML models on actual data is likely to solve monopolistic nature of real options pricing without need for explicit consideration of market competition effects.

Let's define formula for pricing real option using Cox-Ross-Rubinstein method of constructing binomial tree. Cone snippet for that is show next.

```
# function for option to expand
def option_to_expand(T,S,sig,r,N,expand,cost):

    dt=T/N
    dxu=math.exp(sig*math.sqrt(dt))
    dxd=1/dxu
    pu=((math.exp(r*dt))-dxd)/(dxu-dxd)
    pd=1-pu
    disc=math.exp(-r*dt)

    St = [0] * (N+1)
    V = [0] * (N+1)

    St[0]=S*dxd**N

    for j in range(1, N+1):
        St[j] = St[j-1] * dxu/dxd

    for j in range(1, N+1):
        V[j] = max(St[j]*expand-cost,St[j])

    for i in range(N, 0, -1):
        for j in range(0, i):
            V[j] = disc*(pu*V[j+1]+pd*V[j])

    return V[0]
```

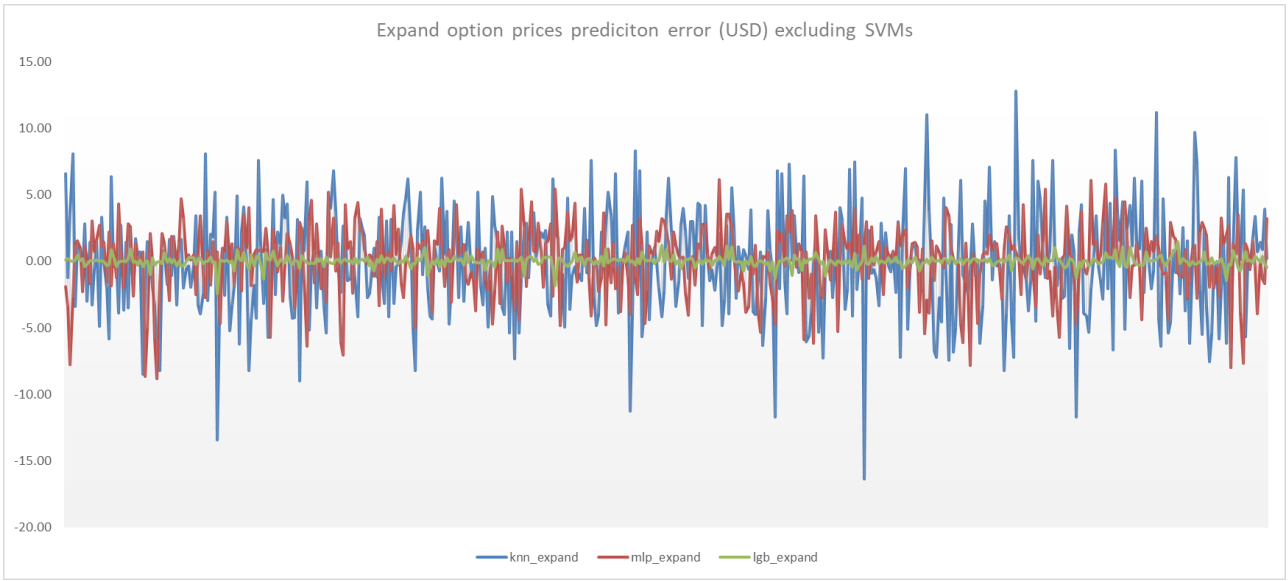
Next fixed interest rate of 5%, investment horizon of 3 years, and 1-year binomial steps are used to price option to expand with expansion factor of 1.5 at 50% cost of current project value. One thousand simulations of random numbers for projects' starting present value are drawn from range of values starting at 10 till 1000 with steps of 10, while volatility measures varying from 20 to 50% using 1% increments. As before, 5% of data is set aside for testing, and per each model prediction errors are graphed.

Figure 5 Prediction error for ML models on price of option to expand



Obviously, SVMs didn't lend themselves to accurately predict real option prices in the example but looking at the graph below excluding SVMs, it's clear that all other ML models seem to predict option prices very closely.

Figure 6 Prediction error for ML models on price of option to expand, excluding SVMs from the mix



Finally, let's look at average error and other prediction metrics in the table below.

Table 2 Options to expand pricing, prediction errors

metric	knn_expand	mlp_expand	lgb_expand	svr_expand
average error	3.09	1.93	0.26	24.03
min error	0.00	0.00	0.00	0.00
max error	16.36	8.80	2.45	131.65
max neg error	-16.36	-8.80	-2.45	-131.65
max pos error	12.81	6.12	1.48	130.13

Prediction error metrics suggest LGB as the best model for real options valuation.

Conclusion

This paper to some extent answered a question of whether Machine Learning models can be used to price financial options first and then real options. The former used real world data from historical prices Nasdaq Futures to compute option prices by The Black-Scholes model and then train ML models on them. The latter used simulated values of investment projects and their volatility estimates to compute predefined expansion option and then train ML models on them. Both experiments showed promising results with further room for improvements. If real option valuation using machine learning models can be statistically proven, strategic investment projects valuation formula could change from composite 1:

$$Project\ Value = Discounted\ Cash\ Flow + Real\ Options + Game\ Theory$$

To composite 2:

$$Project\ Value = Discounted\ Cash\ Flow + Machine\ Learning\ Option\ Price$$

Note that in composite 2, ML option price can already be trained to account for market competition.

References

[1] Robust Mean-Variance Hedging And Pricing of Contingent Claims In A One Period Model (2011) / *R. Tevzadze and T. UzunaShvili* / International Journal of Theoretical and Applied Finance / World Scientific Publishing Company

[2] Real Options Analysis: tools and techniques for valuing strategic investments and decisions / *Johnathan Mun* /2nd edition, published by John Wiley & Sons, Inc., 2006

[3] Investment Projects Valuation Using Real Options and Game Theory (2013) / *Levan Gachechiladze and Irakli Chelidze* / Academic-analytical Journal "Economics and Banking" 2013 Vol. I, N3

[4] Cox et al. (1979) / *Coxx, Ross, Rubinstein*

[5] Robustness (2008) / *Lars Peter Hansen, Thomas J. Sargent* / Published by Princeton University Press

[6] Hedging By Sequential Regression: An Introduction to the Mathematics of Option Trading (1989) / *H. FÄollmer and M. Schweizer* / ETH ZÄurich

[7] Financial Modeling, 3rd edition (2008) / *Simon Benninga* / MIT press

[8] Investment under Uncertainty (1994) / *Avinash K. Dixit and Robert S. Pindyck* / Princeton University Press

[9] Monte Carlo Methods in Financial Engineering/Paul Glasserman/published by Springer Science + Business Media, Inc., 2004

[10] Derivatives Market / Robert L. McDonald / published by Addison-Wesley, 2006

[11] Strategic Investment: real options and games / Han T.J. Smit and Lenos Trigeorgis / published by Princeton University Press, 2004

[12] Investment Projects Robust Valuation / Levan Gachechiladze, Revaz Tevzadze / MBA thesis, Georgian-American University, 2014

[13] Benninga, S., and M. Blume. 1985. On the Optimality of Portfolio Insurance. Journal of Finance, 40.

[14] Investment Projects Robust Valuation / Levan Gachechiladze, Tamaz Uzunashvili / Business Research Center of Georgian-American University, Applications of Random Processes and Mathematical Statistics in Financial Economics and Social Sciences II, Tbilisi Scientists and Innovations Festival 2017

[15] Application of Real Options in Valuation of Hedging Strategies / Levan Gachechiladze, Tamaz Uzunashvili, Teimuraz Toronjadze / Business Research Center of Georgian- American University, Applications of Random Processes and Mathematical Statistics in Financial Economics and Social Sciences III, Tbilisi Scientists and Innovations Festival 2018

[16] The Elements of Statistical Learning: Data Mining, Inference, and Prediction / Trevor Hastie, Robert Tibshirani, Jerome Friedman / Second Edition, Corrected 12th printing – Jan 13, 2017

[17] An introduction to Statistical Learning with Applications in R / Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani / Springer Science + Business Media New York, 2014

[18] Machine Learning for Algorithmic Trading, second edition / Stefan Jansen / Packt Publishing 2020

[19] Advances in Financial Machine Learning / Marcos Lopez de Prado / John Wiley and Sons, Inc. 2018

[20] Incorporating Machine Learning Models in Valuation of Hedging Strategies / Levan Gachechiladze, Tamaz Uzunashvili, Teimuraz Toronjadze / Business Research Center of Georgian- American University, Applications of Random Processes and Mathematical Statistics in Financial Economics and Social Sciences IV, Tbilisi Scientists and Innovations Festival 2021

[21] Application of Machine Learning in Option Pricing: A Review / Wends Li / Proceedings of the 2022 7th International Conference on Social Sciences and Economic Development (ICSSED 2022)

[22] Option pricing via machine learning / Tidy Finance with R, chapter 15 / Christoph Scheuch, Stefan Voigt, and Patrick Weiss / 2022.4-1

[23] Option pricing using Machine Learning / Codruţ-Florin Ivaşcu / Expert Systems with ApplicationsVolume 163, January 2021, 113799

[24] Option Pricing with Deep Learning / Alexander Ke, Andrew Yang / Stanford, SC230

[25] American option pricing with machine learning: An extension of the Longstaff-Schwartz method/ Jingying Lin, Caio Almeida / Brazilian Review of Finance, Vol. 19 No. 3 (2021)