Final Project

Strategic Options Valuation: Leveraging Advanced Predictive Models for Optimal Pricing

by

Diana Wu Chen - 6934129467

Manhui Zhu - 9852576503

Hsiao-Yun (Angel) Tsui - 8464673270

Marshall School of Business

DSO 530 Applied Modern Statistical Learning Methods

Paromita Dubey

May 2, 2024

Executive Summary

Our analysis first focused on predicting the 'Value' of options using a broad range of modeling techniques (11 different methods), assessing them based on Mean Square Error (MSE) and out-of-sample R-Squared. To enhance model accuracy, we standardized features to mitigate bias from scale differences, particularly crucial for models sensitive to feature distance. The standout models in terms of low MSE and high R-squared were Neural Networks, Gradient Boosting Regression, and Random Forest Regression. These models excel in capturing complex, non-linear relationships within the data, a critical advantage in financial markets influenced by dynamic, non-linear factors like market volatility and sentiment. The Neural Network, optimized with Adam optimizer and trained using five-fold cross-validation, proved particularly effective, highlighting its ability to learn and adapt from data intricacies.

In classification, Random Forest, Bagging, and Decision Tree Regression emerged as top performers due to their minimal classification errors. Parameter tuning identified optimal settings for Random Forest at 200 trees and 1 feature to consider in each split, balancing accuracy with robustness against overfitting. Random Forest's approach of using a random subset of features to decide per tree split decorrelates trees and reduces overfitting risks, which is crucial for reliable predictions in complex datasets. This model's effectiveness was further demonstrated through a detailed structure analysis, showing how each decision node and branch contributes to its high precision and efficiency. The optimization of Random Forest through parameter tuning by rigorous grid searches and cross-validation solidified its status as the preferred model for handling complex, high-dimensional datasets in fluctuating market conditions.

Dataset Processing and Split

The 'option_train' dataset is utilized for the project. Initially, we transformed the 'BS' column by assigning 0 for 'Under' and 1 for ‘Over’, and removed irrelevant predictor 'unnamed columns' and missing data. Next, the dataset was split into two training (80%) and testing sets (20%). We used 'Value' as the response variable for the regression models and 'BS' for the classification models. The *‘StandardScaler( )’* is applied to the dataset before we fit them into scaling-sensitive models.

Part I: Response variable: 'Value'

To predict option value, we tried eleven models and used Mean Square Error (MSE) and out-of-sample R-squared to evaluate the model performance [Figure 1]. Standardized data were fit into models that are sensitive to the scale of features or depending on distance matrix to predict value.

The three models with the smallest MSE and highest R-squared were Neural Networks, Gradient Boosting Regression, and Random Forest Regression. Tree-based models and models that can extract non-linear relationships performed better on our dataset than the linear regression models. Our optimal model, Neural Networks [Figure 2], has 4 layers in total: one input layer with input size equals to four (4 predictors); first hidden layer with 64 hidden neurons; second hidden layer with 32 hidden neurons; one output layers with output size equals to 1 (1 output: option value).

The two hidden layers are fully connected, meaning each neuron in these layers is connected to every neuron in the previous and subsequent layers. We used the *ReLU* function as the activation function. We employed the *Linear* activation function for the output layer since it suited our regression task for predicting option values. *MSE* was the loss function we used to

evaluate the model performance. Considering the learning rate and tuning hyper-parameters, we used *Adam (Adaptive Moment Estimation)*–an extension of stochastic gradient descent (SGD) optimization algorithm– as our optimizer. One advantage of Adam is that it can automatically decrease the learning rate for parameters with frequent updates or large gradients and increase it for parameters with infrequent updates or small gradients. This optimizer can train models more efficiently and converge faster than traditional SGD optimizer.

During the training process, we used 5-folds cross validation to improve the robustness and reduce overfitting and bias. Each iteration recorded the MSE and out-of-sample R-squared, with 5 values each, then we averaged them to get the final MSE and out-of-sample R-squared. We improved training efficiency and reduced computational power by updating the model parameters with only 32 samples (batch-size = 32) per iteration. With 3200 samples in the training set (800 in the validation set), we updated parameters 100 times per epoch. With 50 epochs in each iteration, the entire dataset was passed through the model 50 times.

The model performance implies that the relationship between the four predictors and the response variable `option value` is non-linear. Neural Networks excels at capturing non-linear relationships between input variables and the option values, and it can automatically learn relevant features representations and extract important features. Real-world option pricing is impacted by complex, non-linear factors like market sentiment, volatility, and non-stationarity. Neural networks' flexibility, adaptability, and scalability allow it to integrate diverse data sources, and learn intricate patterns in the financial markets.

Part II: Response Variable: 'BS'

Three models out of eleven models we tried– Random Forest, Bagging, and Decision Tree Regression – emerged as top performers due to their minimal classification errors. A closer

analysis showed that Random Forest surpassed others with the lowest error rate of 0.066, primarily due to its robustness against overfitting. [Figure 3].

In the Random Forest model, we conducted parameter tuning to further refine its accuracy. We implemented a grid search to identify the optimal number of trees and maximum feature parameters, critical to enhancing model performance. *'GridSearch'* was used to identify the option number of trees by trying [10, 25, 50, 75, 100, 150, 200] for *'n_estimators'* and identify best maximum number of features to consider in each split by trying [1, 2, 3, 4] for *'max_feaures*'. We confirmed that the best configuration was n_estimators equal to 200 trees, and max_features equal to 1 to balance the model's complexity and its ability to generalize.

During training, we used 5-fold cross validation to improve model robustness and generalization. Upon evaluating this model against our test set, we observed a classification error of approximately 0.066, reinforcing the effectiveness of our parameter tuning approach. This meticulous optimization process was essential in ensuring that our Random Forest model performs optimally with our dataset, solidifying its status as the leading choice among the tested classification algorithms.

Random Forest applied a random subset of features per tree, enhancing diversity and reducing overfitting risks. This feature randomness, coupled with the model's ability to capture complex data patterns and exhibit lower variance, enhances predictive accuracy and stability, especially in noisy environments. Additionally, despite incorporating more randomness, it maintains computational efficiency through the construction of relatively shallow trees, which simplifies the training and prediction process.

Figure 4 is a graphical representation of the detailed structure of a tree within the model. It highlights how each node acts as a decision point based on a feature value, with edges

depicting outcomes of these decisions. Key metrics displayed at each node included the threshold for splits, the count of samples reaching the node, and the Gini impurity –a measure of classification purity at that node, with lower values indicating higher purity and precision in classification. By dissecting these elements, we demonstrated how effectively Random Forest manages specific datasets, further cementing its status as the preferred model due to its accuracy, efficiency, and effectiveness in dealing with complex, high-dimensional datasets.

Conclusion

Machine learning models have the potential to outperform the Black-Scholes models. These models excel at capturing non-linear relationships and can incorporate diverse data sources, which enhances their predictive capabilities under various market conditions. However, high prediction accuracy in machine learning models often sacrifices interpretability. In financial analysis, where interpretability is crucial for compliance and accountability, tree-based models can serve as a good compromise by offering substantial interpretability and high accuracy.

Upon analyzing the four predictors in our dataset, we found no significant multicollinearity, indicating no redundancy and thus no need for feature selection. Considering the dimensionality of our dataset relative to the sample size, there is no need to rank feature importance or use regularization methods to identify the most informative predictors.

Nonetheless, we do not recommend using our trained model to directly predict option values for Tesla stock. Since our model was trained using the S&P 500 dataset, it does not account for the unique characteristics of Tesla stock and its high volatility. Careful considerations of data compatibility, model generalization, performance evaluation, and feature engineering are critical before deploying our model on Tesla stocks.

References

Wu Chen, Diana; Zhu Manhui; Tsui, Hsiao-Yun. Options Pricing: Python Code. Retrieved from https://colab.research.google.com/drive/1EjruJZAKSjqMsmRmEAG1OrHXZ_k83mDX?usp=sharing

Appendix

	model_name	R_squared
0	Linear Regression	0.924229
1	Best Subset Selection	0.927749
2	KNN Regression	0.986369
3	Decision Trees for Regression	0.992582
4	Random Forest Regression	0.996497
5	Lasso Regression	0.927727
6	Ridge Regression	0.927744
7	SVM for Regression	0.917220
8	Neural Networks for Regression	0.997777
9	Radial Basis Function Networks	0.922188
10	Gradient Boosting Regressor	0.995540

	model_name	mean_squared_error
0	Linear Regression	1153.748990
1	Best Subset Selection	1153.770471
2	KNN Regression	217.668541
3	Decision Tree Regression	118.462792
4	Random Forest Regression	55.937987
5	Lasso Regression	1154.115254
6	Ridge Regression	1153.842655
7	SVM for Regression	1321.900982
8	Neural Networks for Regression	34.766048
9	Radial Basis Function Networks	1242.562821
10	Gradient Boosting Regressor	71.223594

Figure 1. R_squared and Mean_Squared_Error of 11 models for option value prediction

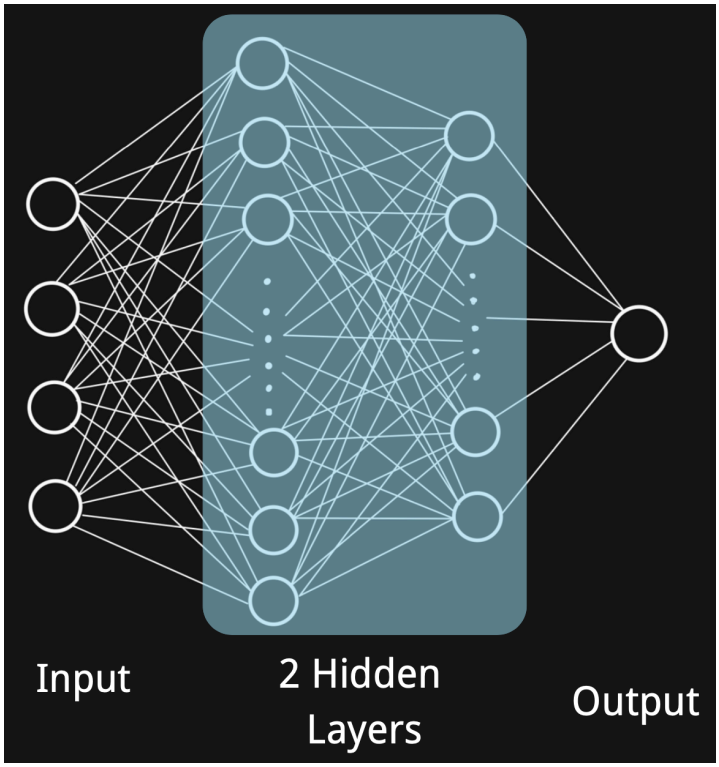

Figure 2. Architecture of Neural Networks for option value prediction

	model_name	classification_error
0	Logistic Regression	0.10600
1	Logistic Regression after Best Subset Selection	0.10600
2	Quadratic Discriminant Analysis	0.10800
3	KNN Classification	0.09000
4	Decision Trees	0.08000
5	Random Forest	0.06625
6	SVM	0.09900
7	Neural Networks	0.08800
8	Boosting	0.09600
9	Bagging	0.07000
10	Naive Bayes	0.12600

Figure 3. Classification Error of 11 models for BS prediction

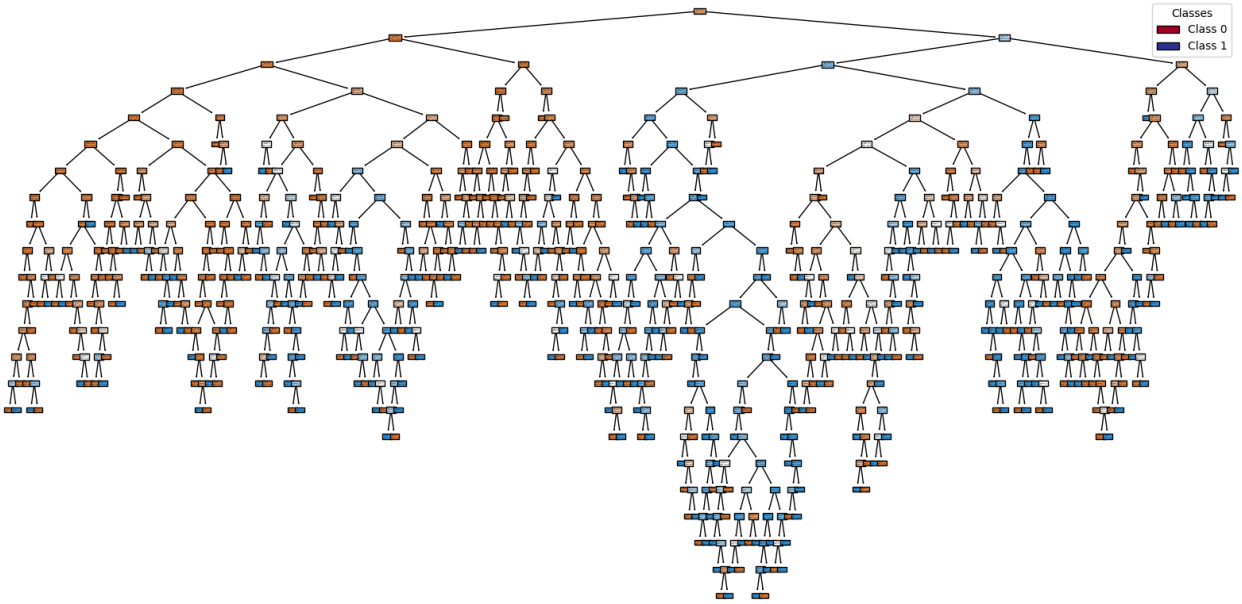

Figure 4. Single Tree Illustration of the Random Forest model

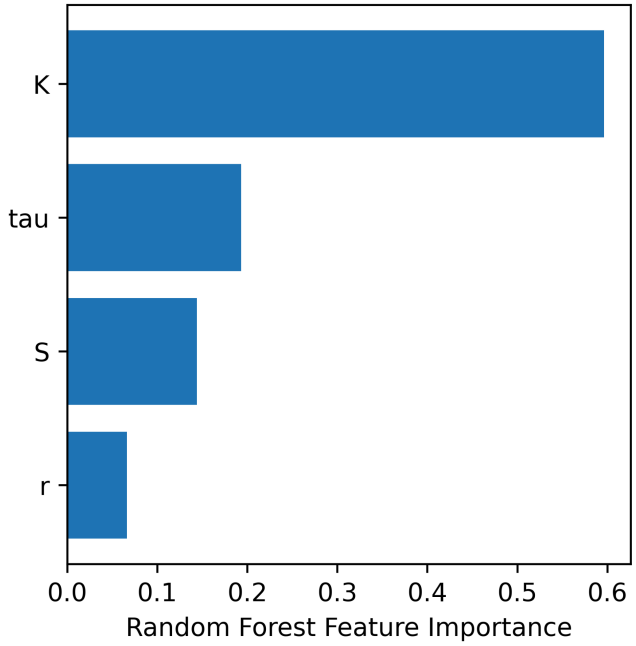

Figure 5. Random Forest Features Purity (Gini Index)