Call Options Pricing Report

Group 6: Linh Chu, Teresa Dang, Linh Pham, Cong Minh Tran

DSO 530: Applied Modern Statistical Learning Methods

Dr. Xin Tong and Dr. Paromita Dubey

May 2, 2024

Executive Summary

The primary objective of this project is to optimize the valuation and pricing of stock options, using the training data to build statistical/Machine Learning models with Value as the response (i.e., a regression problem) and Black-Schole as the response (i.e., a classification problem). We first conducted an Explanatory Descriptive Analysis (EDA) on the dataset to ensure that the dataset was thoroughly cleaned and understood. Then, we performed feature engineering based on the 4 main features (S, K, tau, r). We built 5 other additional features, which are S/K, |S-K|, intrinsic value, K*S, and tau_days. With these 9 features, we conducted regression and classification approaches to find the best model with the highest performance. Regarding regression, we compared the mean CV scores from all approaches that we tried to find the best model with the highest R squared value. For classification, we aimed to find the best model with the smallest mean CV classification errors. As a result, XGBoost was recommended as the optimal model for both regression and classification tasks due to its superior performance and adaptability. The findings suggest that machine learning models are the preferred approach for option value prediction compared to traditional methods like Black-Scholes, as they offer greater flexibility and the ability to incorporate new features, thereby enhancing predictive accuracy.

1. Explanatory Descriptive Analysis (EDA)

We initiated our analysis by performing exploratory data analysis (EDA) on the dataset and found no missing values or extreme outliers. Consequently, we decided to retain all 5,000 data points and standardized all features using the StandardScaler. We investigated feature correlations with the target variable "Value" (current option value), the strike price K exhibited a negative correlation with Value (Figure 1), while no clear linear correlations existed between Value and other features like S, r, and tau.

To examine the Black-Scholes classification, we overlaid the box plots of each feature to visualize the separation between classes based on that feature. More separation between the box plot distributions indicates that the feature may have greater discriminative power. From these graphs, we observed that the strike price K appeared to separate the two classes fairly well (Figure 2).

By conducting a thorough EDA and visualizing the feature distributions, we gained insights into the potential influence of each feature on the target variable and identified promising candidates for further analysis and model development.

2. Feature Engineering

Based on the exploratory analysis and options pricing theory, we engineered several new features that could potentially improve the predictive performance of our models. One set of features related to option pricing included moneyness (S/K), which indicates whether an option is in-the-money, at-the-money, or out-of-the-money, and intrinsic value, which equals max(S-K,0).

This intrinsic value feature directly captures the intrinsic option value and enhances interpretability. According to the graph (Figure 1), intrinsic value exhibited a strong linear relationship with the option value. Another feature incorporated was K*S, which represents the monetary value of the underlying asset relative to the strike price. The box plot demonstrated that this feature could partially separate the Black-Scholes values (Figure 2). For the time variable tau, we introduced tau_days to provide additional granularity and capture more nuanced temporal information, particularly in financial contexts where time plays a critical role in option pricing and market dynamics.

By engineering these new theory-driven features, we aimed to enhance the predictive power and interpretability of our machine learning models, leveraging insights from options pricing theory to improve model performance.

3. Regression

3.1. Exploration

For the regression task, our initial overall approach was to evaluate seven models ranging from linear to tree-based models, namely: Lasso, Ridge, Linear Regression with Best Subset Selection, Decision Tree, Gradient Boosting, XGBoost, and Random Forest. However, we decided not to employ regularization techniques like Ridge and Lasso since our dataset did not have a large number of features, and we aimed to prioritize model interpretability for linear regression.

Subsequently, our approach involved running each model with a 10-fold cross-validation (CV) to obtain the mean score across the entire training dataset. We began with the default parameter settings and then tuned the hyperparameters to further improve the model's performance. Finally, we compared the best mean scores to select the final model with the highest R-squared value.

Linear Regression with Best Subset Selection

We implemented linear regression with the Best Subset Selection method, utilizing seven features: S, K, tau, r, S/K, |S-K|, and S-K. We observed that the adjusted R-squared was highest at 99.2563% with six predictors, which was not significantly higher than the adjusted R-squared of 99.2519% with four predictors. Consequently, we decided to retain four predictors: tau, S/K, |S-K|, and intrinsic value, to enhance model interpretability. This approach yielded a mean CV R-squared score of 99.25%.

Decision Tree

We started with the Decision Tree model as the first tree-based approach due to its ability to handle complex and non-linear relationships while retaining interpretability. Using the default parameters, the Decision Tree model yielded an improved mean R-squared of 99.58%, albeit at the cost of producing a highly complex and bushy tree structure.

Random Forest

To mitigate the variance inherent in the Decision Tree model and improve overall robustness, we implemented the Random Forest algorithm. This approach achieved a mean CV R-squared of 99.77%, with the number of estimators selected as 300.

Gradient Boosting & XGBoost

Regarding boosting algorithms, we evaluated both Gradient Boosting and XGBoost to determine which approach led to better prediction performance on the stochastic option features. XGBoost slightly outperformed Gradient Boosting, achieving mean CV R-squared values of 99.87% and 99.81%, respectively.

	LR	**DT**	**RF**	**GB**	**XGB**
R^2	99.25	99.58	99.77	99.81	99.87

Table 1 Mean cross-validated R^2 for 5 different regression models

3.2. Final Model

Among all models tested, XGBoost with 300 estimators provided the best prediction on cross-validated 10-fold score. Its ability to explain an average of 99.87% of the variance in the option value is impressive, along with its built-in regularization to prevent overfitting. Notably, the model attributed 'intrinsic_value', 'S/K', |S-K| and 'tau' as 4 features with the highest importance scores, similar to the best subset selection from the linear model (Figure 3). By interpreting these important features in the context of domain knowledge, practitioners can gain valuable insights into the factors driving option value and make informed decisions in their trading strategies or risk management practices.

4. Classification

4.1. Exploration

Logistic regression

Initially, Logistic Regression was chosen due to its simplicity and capacity to offer probabilistic interpretations of predictions. The Logistic Regression algorithm was applied using all nine features as a baseline model, resulting in a mean cross-validation classification error of 10.64%. However, despite subsequent feature selection and hyperparameter tuning efforts, the classification error saw minimal reduction, persisting at 10.54%. This outcome positioned it as our worst-performing model.

Support Vector Machine (SVM)

We subsequently evaluated SVM for classification, given their adeptness in managing intricate, high-dimensional datasets and delineating non-linear decision boundaries. Following a meticulous feature selection process, the model retained the features 'S', 'K', 'tau', 'r', 'moneyness', and 'abs_S-K'. Notably, the SVM model exhibited the lowest classification score, achieving 8.50% error, while employing optimal parameters: 'C': 100, 'gamma': 'scale', and 'kernel': 'rbf'. This parameter configuration demonstrated superior performance in delineating the decision boundaries within the dataset.

K-Nearest Neighbor (KNN)

We considered the K-Nearest Neighbors (KNN) algorithm as we expected some stocks to exhibit similar behaviors helpful for classification. The baseline model with all nine features yielded a mean cross-validated classification error of 8.56%. In comparison, the KNN model with feature selection and hyperparameter tuning achieved a classification error of 8.12%, which was a slightly better performance than the baseline model. This outcome suggested that while feature engineering can enhance model performance, alternative algorithms or ensemble techniques might be required to achieve superior predictive accuracy for this particular stock option prediction task.

Decision Tree

Although decision trees offer interpretability, we anticipated challenges applying them to the complex call option pricing problem. The baseline decision tree model yielded a high 8.64% mean classification error. Analyzing the tree revealed 'K' (strike price) and 'tau' (time to expiration) as important features. A new model with hyperparameters tuning on a subset of features including ‘S’, ‘K’, ‘tau’, ‘r’, ‘intrinsic_value’, and ‘k*s’ led to a performance decrease to 9.46%. We expected the lower performance to be due to limiting the maximum tree depth to maintain some interpretability.

Random Forest

To improve the robustness of classification, we introduced a random forest model. The base model achieved a promising 6.6% classification error. However, feature engineering did not yield substantial improvements, with the engineered model's error at 6.6% (Figure 4). Visualizing feature importances revealed 'K' (strike price), 'moneyness', and ‘k*s’ as the most influential predictors. Tuning the number of estimators and maximum number of features hyperparameters improved the classification error slightly, down to 6.54%.

XGBoost

XGBoost, grounded in an ensemble of decision trees, incorporates inherent regularization mechanisms that effectively mitigate overfitting, consequently enhancing generalization and

robustness across diverse datasets. XGBoost demonstrated exemplary performance, attaining the lowest classification error rate of 5.56%.

	Logistic Regression	Decision Tree	SVM	KNN	Random Forest	XGB
Classification error	0.10539	0.0946	0.0849	0.0812	0.0654	0.05579

Table 2 Mean cross-validated classification error for 6 different classification models

4.2 Final Model

Among all models tested, XGBoost with 300 estimators provided the best prediction on a cross-validated 10-fold score of 5.58%. Some notable hyperparameters considered were gamma, which served to regulate model complexity, and learning rate, which gradually adjusted towards minimizing the loss function (Figure 6). Our objective is to strike a balance between accuracy and model simplicity. It is also worth noting the Black-Scholes formula's tendency to overvalue options with higher strike prices.

Conclusions

Based on the results, we recommend the XGBoost model for both predictions as it exhibits the highest performance with a mean CV R^2 of 99.87% for value prediction (regression) and a mean CV classification error of 5.58% for BS prediction (classification). We also advocate for machine learning (ML) models over the Black-Scholes (BS) model due to their flexibility, non-linearity, ability to learn from data, and adaptability across industries and features.

However, ML models require balancing complexity and interpretability. While hyperparameter tuning and feature selection generally improved performance, they increased model complexity. On the other hand, linear regression showcased predictor-outcome relationships effectively, albeit with slightly worse performance. From a business perspective, discarding certain features may be reasonable to reduce costs and achieve acceptable predictions.

Despite promising training set performance, generalizing predictions to other stocks such as Tesla warrants caution due to potential industry-specific factors and variations not captured in the training data.

In conclusion, we emphasize the importance of interpretability over pure prediction accuracy. Understanding the underlying factors driving predictions builds trust, identifies potential biases, and enables informed decision-making, particularly in high-stakes applications.

Appendix

1. Explanatory Data Analysis (EDA) + Feature Engineering

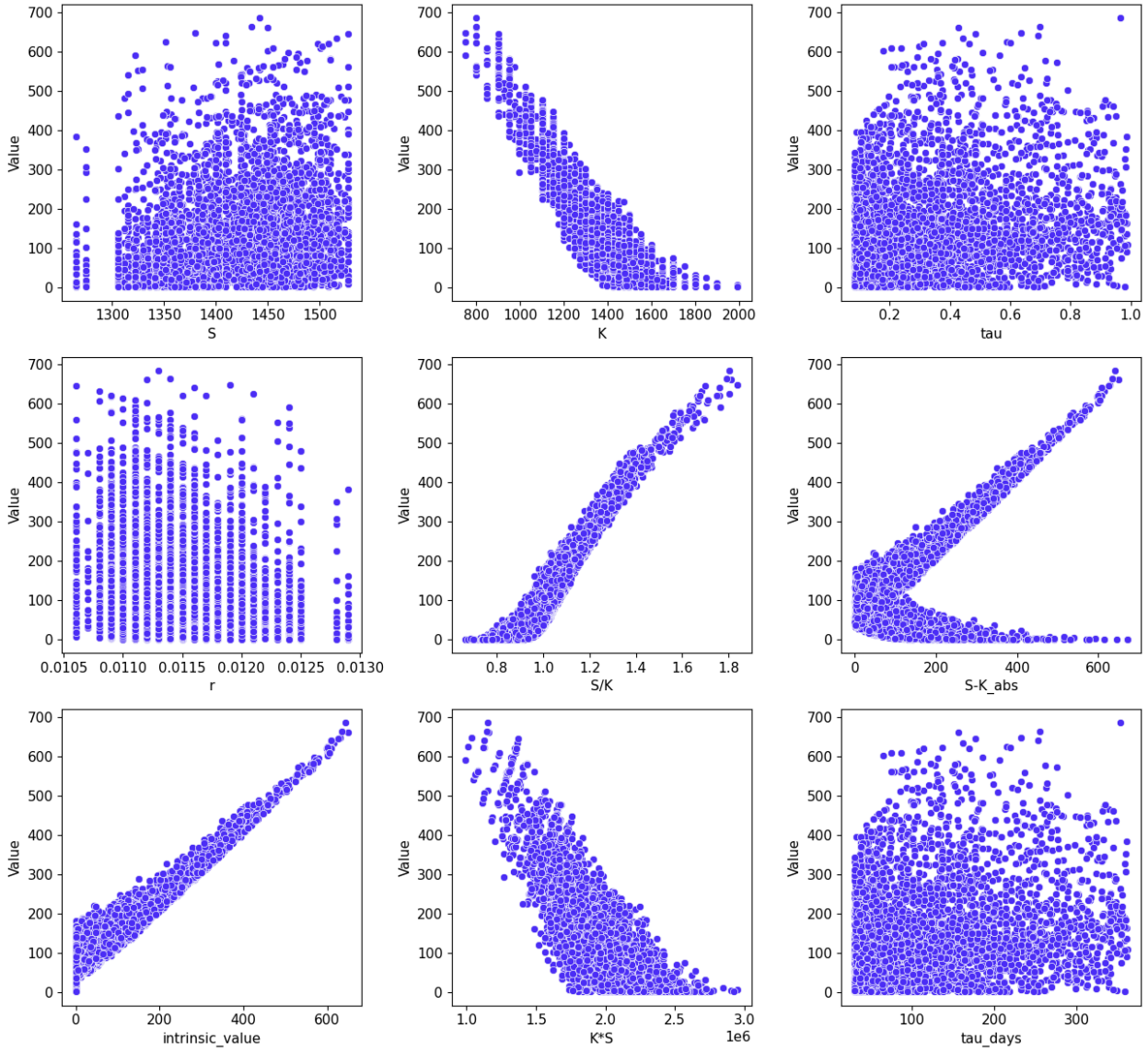

Figure 1: Scatterplots of Value Response Variable against Predictor Variables

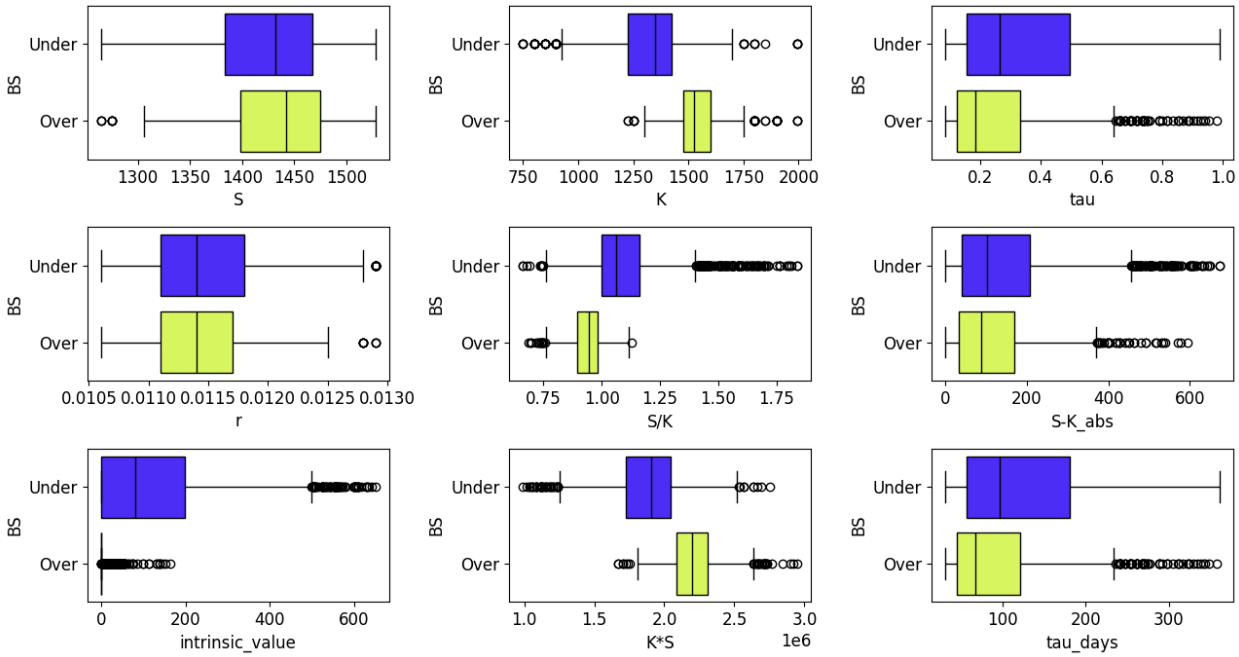

Figure 2: Boxplots of BS Response Variable and Predictor Variables

2. Regression

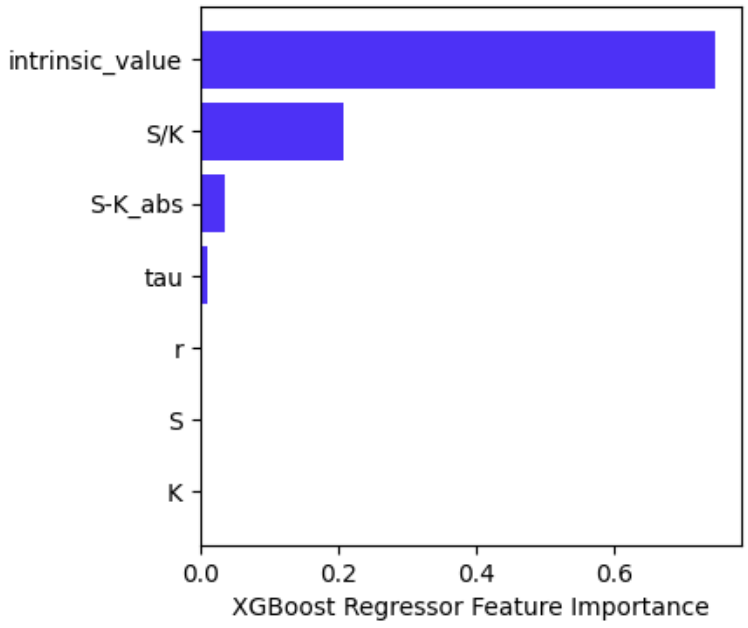

Figure 3: Feature importance from XGBoost Regressor

3. Classification

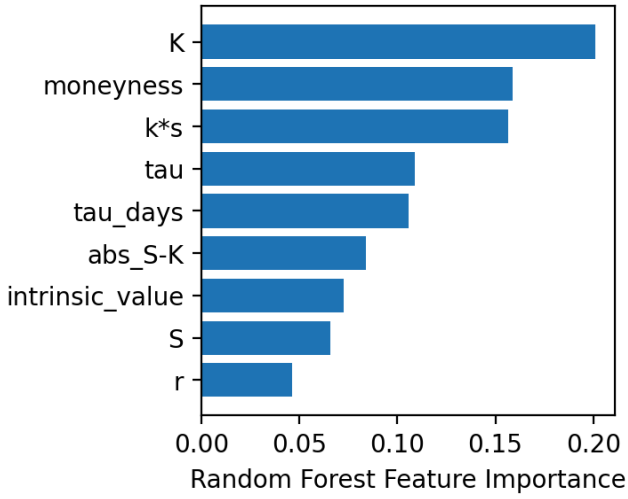

Figure 4: Feature importance from Random Forest Classifier

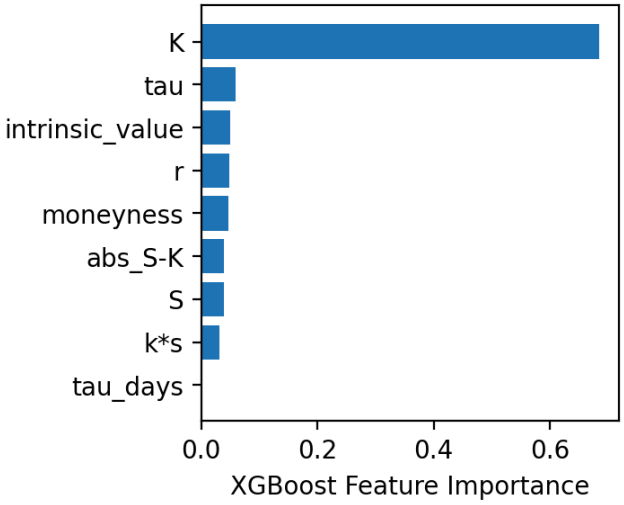

Figure 5: Feature importance from XGBoost Classifier

Hyperparameter	Value
gamma	0.1
learning_rate	0.1
max_depth	7
n_estimators	300
subsample	0.8

Figure 6: Hyperparameter tuning of XGBoost Classification