Statistical/Machine Learning Models for Stock Options:

Project Report

Gabe Owens, Yuan-Cheng Tsai

Master of Science in Business Analytics - USC Marshall

DSO 530: Applied Modern Statistical Learning Methods

Professor Xin Tong

5 May 2022

EXECUTIVE SUMMARY

Overview

This report documents the entirety of our options pricing group project – from data cleaning to model conclusions.

Problem

Using European call option pricing data on the S&P 500, we will find the best regression model (in terms of maximum out-of-sample r squared) and best classification model (in terms of minimum classification error) to predict "Value" and "BS" respectively. Our dataset had information on 1,680 individual stock options with the following recorded variables:

- Value (C): Current option value
- S: Current asset value
- K: Strike price of option
- r: Annual interest rate
- tau: Time to maturity (in years)
- BS: The Black-Scholes formula was applied to this data (using some σ) to get C_pred. If an option has C_pred – C > 0, i.e., the prediction overestimated the option value

Our best-selected models would then be used to make predictions on an unseen test dataset of 1,120 options.

Solution

We underwent data cleaning, data analysis, and interpretation on the given training data set. We explored a variety of regression methods and classification methods. Our best method for "Value" was a linear regression model created with transformed variables: (Value = K + S + tau + K*S + K^2 + log(tau) + e^K + 1/r + tau^K). Our best model for "BS" was the XGBoost Classifier with hyper-tuned parameters: XGBClassifier(learning_rate= 1, max_depth= 4, eta=1).

Conclusions

While we have confidence both of our models will perform well on the unseen test dataset, we would not suggest using our model to make predictions on anything other than European S&P 500 options data.

REPORT

Quality Checks

Before analysis, we performed all applicable methods of data cleaning. Using the training data, we converted the binary "BS" variable to 1 if the prediction was “Over” and 0 if the prediction was "Under". We then dropped two rows that contained null values and four rows that contained extreme outliers – our final dataset shape was (1675, 6). We performed a train/test split on the training data with a test size of 30%. Finally, we normalized the predictor variables and added a very small indiscernible number to all values – this made sure there were no values of exactly 0 in the dataset which eliminated variable transformation restrictions.

Regression (“Value”)

We explored six regression analysis techniques with the goal of maximizing r-squared. We used GridSearchCV, an sklearn library function, to find the best combination of predefined, tuned hyperparameters to fit our model to the training set. For each model, we used repeated k-fold cross-validation (with 5 folds and 100 repeats) and recorded the average cross-validated r-squared score. Below is a table of all the regression techniques we used and their corresponding r-squared scores in ascending order.

Regression Technique	**R-Squared Score**
Ridge Regression	0.90915
Lasso Regression	0.91054
Elastic Net Regression	0.91055

Linear Regression	0.91074
Decision Tree Regressor	0.99095
Transformed Linear Regression Equation: (Value = K + S + tau + K*S + K^2 + log(tau) + e^(K)+ 1/r + tau^K)	0.99383

Out of all the regression techniques, our best performer was the Decision Tree Regressor (with an r-squared score of .99095). However, we decided to explore the effectiveness of transforming variables within a linear regression equation (as transformation preserves the linear relationships between variables while also making them stronger). We compared the adjusted r-squared values of different combinations of various transformations on all predictor variables. We cross-validated our results with the test data to determine the out-of-sample r-squared. Our final equation was: Value = K + S + tau + K*S + K^2 + log(tau) + e^(K)+ 1/r + tau^K. The r-squared score was .99383, making it the highest on our list.

Classification ("BS")

We explored 15 classification techniques with the goal of minimizing classification error. As in our regression analysis, we used both GridSearchCV and repeated k-fold cross-validation. Below is a table of all the classification techniques we tested and their corresponding classification errors in descending order.

Classification Technique	**Classification Error**
Gaussian Naive Bayes Classifier	.1242
MLP Classifier	.0984
KNN Classifier	.0948

Support Vector Classifier	.0912
Logistic Regression	.0902
Ridge Classifier	.0889
Quadratic Discriminant Analysis	.0881
Linear Discriminant Analysis	.0877
Decision Tree Classifier	.0838
AdaBoost Classifier	.0778
Random Forest Classifier	.0707
Gradient Boosting Classifier	.0707
Cat Boost Classifier	.0687
Extra Trees Classifier	.0680
XGBoost Classifier: XGBClassifier(learning_rate= 1, max_depth= 4, eta=1)	.0666

Our best performing classifier was the XGBoost Classifier. It is a highly efficient, scalable, flexible, and accurate distributed gradient boosting classifier that provides parallel tree boosting. Our final model with tuned input parameters was: XGBClassifier(learning_rate= 1, max_depth= 4, eta=1). The mean classification error was .0666, making it the lowest on our list.

Predictions

To make our predictions on the unseen test dataset we normalized the data in the same manner as the original training set – adding the same indistinguishable constant value. Using both our best

classification model and our best regression model, we made predictions on the final test dataset for both "Value" and "BS" respectively.

Discussion

We believe both of our models will make accurate predictions on the unseen test data and we are confident there was no overfitting nor data leakage during our analysis. We do think machine learning models will outperform Black-Scholes because they prioritize accuracy over interpretability, however, we would not trust our models on other real-world data. Given our models were trained exclusively on European S&P 500 call option data, we do not suggest their usage on external data (like Tesla call option predictions). Even if we discovered new models from analysis on samples of Tesla data, we would not trust them in making real-world predictions with personal money on the line. There is extreme volatility in the stock market caused by external influences that are both uncontrollable and unpredictable. A call option's value is extremely complex and an accurate prediction of it would certainly require more than the four simple predictor variables we used throughout our analysis.

Appendix

The organized Jupyter Notebook code output below provides a detailed account of our entire process, in order, from start to finish: