University of Southern California

Option Price Prediction

Janicia Chang (2596883539, janiciac@usc.edu)
Mengqi Tan (7494935350)
Shihkai Chen (1481484730)
Yue Tan (2430762469)
Zijing Wu (8545914435)
Ningchuan Peng (6971437507)

DSO 530 Applied Modern Statistical Learning Methods

Xin Tong

2021.05.05

Executive Summary

- Problem Statement

For this project, our goal is to find out the most accurate approach and use it to make predictions for current option value and BS (Over / Under) on the 1,120 options in the test data set. To find out the best model, we have tried 6 types of regression models and 10 classification models.

Regression - Linear / KNN / Decision Tree / Random Forest / Gradient Boosting / SVM
Classification - Logistic / LDA / KNN / Decision Tree / Random Forest / GPC / Gradient Boosting / SVM / AB / QDA

We have separately applied normalization and standardization to the training dataset and eventually selected the scalar with better performance. These techniques can lower the variance with the data and provide a more accurate data. To evaluate the performance and select an appropriate level of flexibility to our model, we decide to utilize cross-validation for all models since the R-squared of each 10 folds CV never goes beyond the maximum or minimum R-squared when using train and test split methods.

- Findings

After applying all of the regression methods mentioned above, the XGBoost performs the best (R-squared: 0.9990) among them. We believe that the advantage of this approach is that the new models being added are focused on correcting the mistakes which were caused by other models. In a standard ensemble method where models are trained in isolation, all of the models might simply end up making the same mistakes.
As for the classification method, voting has the lowest classification error (6.30%). The voting classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output based on their highest probability of chosen class as the output.

- Recommendations

In order to have a better performance on regression / classification models, below are the three methods that we can focus on:

1. Hyperparameter adjustment: the process of determining the right combination of hyperparameters that allows the model to maximize model performance. Setting the correct combination of hyperparameters is the only way to extract the maximum performance out of models.
2. Feature engineering: the process of using domain knowledge to extract features from raw data.
3. Feature selection: the process of reducing the number of input variables when developing a predictive model.

Dataset Overview and Pre-processing

This project contains the data of 1680 European call options with the following attributes:

- 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 applied to the data to get predicted current value. If the predicted value is higher than the actual, BS records it as Over; otherwise, it records Under.

Aligned with the classification requirement, we firstly converted the BS into a binary variable: Over into 1 and Under into 0. By comparing the minimum and maximum R-squared of executing train and test split and each R-squared using cross-validation with 10 folds, we found that the latter approach had less fluctuation, so we chose cross-validation for both regression and classification models to split our training dataset. Besides, we assigned 0 to the random state and 10 to the k-folds. We then separately applied normalization and standardization to our training data of four features to compare each model's R-squared or classification error using different scalers and hence choose the better one.

Regression

Because the four predictors have little linear relationships with the response, we were encouraged to explore non-linear algorithms for training. We have introduced five non-linear models, and the R-squared evaluated by the 10-fold cross-validation of each model is listed in the table below:

Model	R-squared
XGBoosting (XGB)	0.9990
Random Forest (RF)	0.9964
Decision Tree (DT)	0.9925
Support Vector Machines (SVR)	0.9777
K-nearest Neighbors (KNN)	0.9712

All of the 5 non-linear models have the R-squared that are higher than 0.9700. Among them, the 3 methods from the tree family have the R-squared close to 1, and XGBoost achieved the highest R-squared with 0.9986. Therefore, we decide to further investigate the XGBoost algorithm to improve the performance of our prediction model.

- XGBoost

XGBoost is an open-source library providing an implementation of gradient boosted decision trees. With the decision tree model, we simply trained one tree on our dataset and used that for prediction. With the random forest model, we built an ensemble but all of the trees were still trained and applied to our data separately. Boosting, on the other hand, trains models in succession, with each new model being trained to correct the errors made by the previous ones.

The performance of XGBoost is generally really good and that's also the case for this dataset. Even without any hyperparameter adjustments, our XGBoost regression model was able to achieve a mean R-squared of 0.9982 using 10-folds cross-validation. This result already beats all the models we tried previously. However, the power of XGBoost does not stop here. Another advantage of XGBoost is its flexibility, there are many hyperparameter tuning options available to make our model fit even better. After playing around with some parameters as well as running through a grid search process. We were able to increase the mean R-squared value to 0.9990 by changing the following hyperparameters.

n_estimators	max_depth	learning rate	subsample	R-squared
1000	5	0.2	0.8	0.9990

Note: n_estimators specifies the number of trees in the model and max_depth indicates the depth of each tree. Learning rate and subsample are parameters to avoid overfitting.

Classification

After we concluded the best model for the regression problem, we switched our focus on the second part of the project -- classification. As mentioned before, we converted the response variable BS to a binary variable to better make the predictions. We also applied the similar assumptions with regression to remove the randomness issue:

- Random state = 0
- Cross-validation with 10-folds

After running all possible classification models, below is the results we got:

Model	Classification Error	Model	Classification Error
XGBoosting (XGB)	6.49%	K-nearest Neighbors (KNN)	8.27%
Gaussian Process Classification (GPC)	6.61%	Logistic Regression (LR)	8.57%
Support Vector Machines (SVC)	6.84%	Ada Boosting (AB)	8.57%
Random Forest (RF)	7.38%	Linear Discriminant Analysis (LDA)	8.69%

Quartic Discriminant Analysis (QDA)	7.98%	Decision Tree (DT)	8.99%

- Voting Classifier

From the above classification errors, we find out that there are 5 models that give back classification error less than 8% and the XGBoost classifier gives back the least classification error of 6.49%. But there are still problems:

1. The standard deviation is high in each cross-validation error. Take Random Forest classifier, Gaussian Process classifier and SVC for example. All three of them give a mean classification error less than 7.5%, however the standard deviation is from 2.5% to 2.9%, which means the maximum classification error is more than 10%, even though the minimum classification error is less than 5%.
2. Although in general, we could use the XGBoost classifier as our best model with 6.49% classification error, could we make progress to lower the classification error?

To solve the above two questions, we introduce the Voting classifier. The Voting classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output based on their highest probability of chosen class as the output. In reality, some models do a good job in this part of data, and others do a good job in that part of data. With the Voting classifier, we can take the advantage of different models and get a better and more stable result.

For the Voting classifier, it is important to find the most suitable models to vote. Actually we can use the feature selecting method. We now have 10 models in total and we can run a loop to test all the combinations and use them as the models for the Voting classifier. Since there are 2^10-1=1023 models in total, it runs for a very long time. The final result shows that the Random Forest classifier, the Gaussian Process classifier and the XGBoost classifier is the best combination with the lowest mean classification error of 6.30% and lowest standard deviation of classification error of 2.5%.

Conclusion

To summarize, we find out the following models give back the best result for this regression and classification problem:

Regression	**Classification**
XGBoost	Voting Classifier (Random Forest, Gaussian Process, XGBoost)
R-squared: 0.9990	Classification error: 6.30%

To improve our model performance, we can execute hyperparameter adjustment, feature engineering and feature selection in the further research.