UIUC FE Club – ML for American Option Pricing

Nick Clarisse, Ben Kazinsky

Summary

A first step towards using a Random Forest Machine Learning model to predict American option pricing. American options can be executed at any time between purchase and expiry, so this can add a level of complexity to the data. For simpler data sets, like a deterministic BSM data set with random inputs, the random forest regression model worked well, achieving a mean squared errors of 0.568 and 0.0686 for calls and puts, respectively. However, for historical data the as we implemented model performed quite poorly, achieving mean squared errors of 135.8 and 132.4 for calls and puts, respectively. Random Forest Regression was performed on the call or put option price after training it on 70% of the data. The data was composed of features and targets, where the features were input parameters to the BSM model and the target was the actual option price. The size of data sets for the synthetic case was 100,000 and for the historical SPY Jan 2023 data it was >78,000.

Workflow

First, we had to obtain the data. This was easy in the synthetic case since we just used the BSM formula and a set of randomly generated parameters with reasonable bounds on the values (e.g. T was limited between 0 and 2 years). For the historical case, we were able to obtain S, K, v, T, and the option price from optionDX. The dividend yield, q, for SPY in Jan 2023 was prorated, taking the Q1 value and dividing it by three (averaged) 0.1314%. The fed rate for January 2023 was 4.33%.

Synthetic data was already generated in this data structure form, but we had to perform some basic cleaning and reformatting to get the optionDX data into the python data frame with the following column structure, using the data collection description above. Features included everything but the option price, where that was the target.

Stock price (S)	Strike Price (K)	Dividend Yield (q)	Risk Free Rate (r)	Implied Vol (v)	Time to Expiry (T)	Option Price

A random forest model is an ensemble of many decision trees. Each decisions tree sees different subsets of the data via bagging and grows uniquely based on random feature sub-setting. Each tree grows to near full depth, becoming a strong (many levels deep),low-bias learner. This entire process controls overfitting at the forest level even if individual trees overfit bootstrap samples. This particular RF model was implemented by training it on 70% of the data, using 100 random trees, and regression on the option price target. The predictive performance was assessed using mean squared error.

The data was visualized using a scatter plot of various call option prices against their corresponding expiration times.

Data Collection
- Synthetic (BSM, random)
- Historical OptionDX (Jan ‘23)
- Fed rate (federalreserve.gov)
- Yahoo Finance SPY Dividend Yield (Q1 2023)

↓

Basic Data Cleaning
- Organize optionDX data into the S, K, q, r, v, T data structure

↓

Model Development
- Random Forest Model
- Python scikit-learn

↓

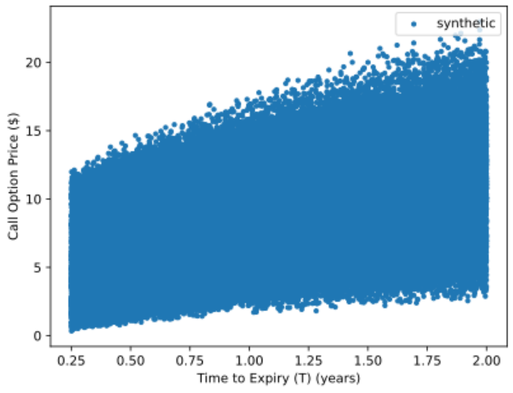
Data Visualization
- Scatter plot
- Option Price vs Expiry Time

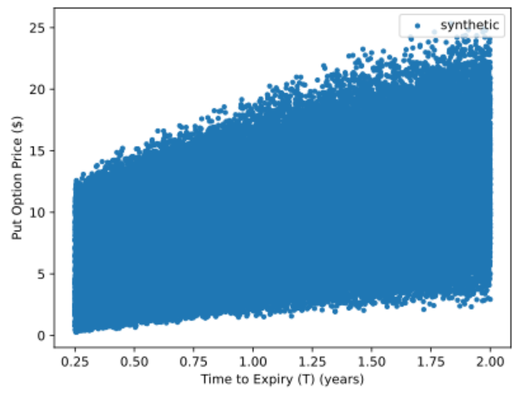
Synthetic Data

Synthetic option data was generated form the Black-Scholes-Merton (BSM) model using randomized inputs. The BSM model takes in several parameters to determine the price of a call or put option: stock price (S), strike price (K), dividend yield (q), risk-free rate (r), implied volatility (v), time to expiry (T) in years.

$$C = Se^{-qT}N(d_1) - Ke^{-rT}N(d_2) \qquad P = Ke^{-rT}N(-d_2) - Se^{-qT}N(-d_1)$$

N is a normal (gaussian) distribution with input parameters d_1 or d_2 that depend on volatility. The parameters S, K, q, r, v, T all served as features to be used in the training data set for the random tree model. These would be used to predict C or P depending on the call or put data set. The basic data structure that both historical and synthetic data sets are mapped into depend on these parameters.





Historical Data

The historical data was not normally distributed, so perhaps using a data structure that dependent on BSM input parameters was a misstep. Looking at the collected data, we see a clear asymmetry in option expiry times that was not present with the random data sets. Definite improvements would include more relevant parameters, option greeks, correlation, and more. Training on more of the data, using more random trees, and collecting more historical data could improve predictive performance. Additionally, perhaps the choice of random forest implementation was naïve. Using regression on such complex data sets can run into prediction quality issues and ultimately, random forests struggle with extrapolation, which means the ultimate test of using a model likes this on live data may struggle. Neural Networks, and gradient boosting can be future alternatives for this project.