# *LSMRF-COPP*: Enhancing American Call Option Pricing with Random Forest Regression

Justin Moonjeli

December 2024

## Abstract

This paper presents Least Squares Monte-Carlo Random Forest Call Option Price Predictor (*LSMRF-COPP*), a novel hybrid model for pricing American call options. The model combines Least Squares Monte Carlo (LSM) simulation with Random Forest regression to improve the accuracy of continuation value estimation. We demonstrate the effectiveness of our approach using historical SPY option data from Yahoo Finance, achieving a significantly lower Mean Absolute Percentage Error (MAPE) compared to traditional LSM. The incorporation of a rich feature set, including option Greeks and other derived variables, enhances the model's predictive power. This work builds upon the LSM algorithm [1] and addresses its limitations in capturing non-linear relationships by integrating the power of Random Forests, inspired by advancements in applying machine learning to financial modeling [2, 3]. This project was completed for the GSU CSC 4740 Data Mining course.

## Keywords

American Option Pricing, Least Squares Monte Carlo, Random Forest, Option Greeks, Feature Engineering, SPY Options, Machine Learning in Finance

## Introduction

Accurate option pricing is crucial for risk management, portfolio optimization, and investment decision-making. American options, due to their early exercise feature, pose a greater challenge for valuation compared to European options. Traditional methods, such as binomial or trinomial trees, can become computationally expensive, especially when dealing with a large number of time steps or multiple underlying assets. Least Squares Monte Carlo (LSM) offers a flexible and computationally efficient alternative for pricing American-style options. However, traditional LSM often relies on polynomial regression for estimating continuation values, which may not adequately capture complex non-linear relationships between the option's underlying asset price and its value.

This paper proposes *LSMRF-COPP* (Least Squares Monte Carlo - Random Forest - Call Option Price Predictor), a hybrid model that integrates a Random Forest regressor into the LSM framework. This approach aims to enhance the accuracy of continuation value estimation by leveraging the Random Forest's ability to capture non-linear patterns in the option data. By combining the strengths of Monte Carlo simulation and machine learning, *LSMRF-COPP* seeks to provide a more robust and accurate method for pricing American call options.

## Background and Related Works

The Least Squares Monte Carlo (LSM) algorithm, introduced by Longstaff and Schwartz [1], has become a widely used method for pricing American-style options. The algorithm simulates a large number of price paths for the underlying asset and uses least squares regression to estimate the conditional expected payoff from continuing to hold the option. This expected payoff is then compared to the immediate exercise value to determine the optimal exercise strategy. Traditionally, LSM employs polynomial regression for continuation value estimation. However, the relationship between the underlying asset price and the option value can be complex and non-linear, potentially limiting the accuracy of polynomial regression.

Recent advancements in machine learning have shown promising results in various financial applications, including option pricing [2, 3]. Random Forests, in particular, have demonstrated their ability to effectively capture non-linear relationships and handle high-dimensional data. These advantages make Random Forests a suitable candidate for enhancing the continuation value estimation within the LSM framework. This paper builds upon the foundation of LSM [1] and incorporates the predictive power of Random Forests, drawing inspiration from successful applications of machine learning in option pricing and other financial domains [2, 3].

### Risk-Neutral Pricing and the Black-Scholes Model

Option pricing models often rely on the concept of risk-neutral valuation. This approach assumes that investors are indifferent to risk and that the expected return on all assets is equal to the risk-free interest rate. Under this assumption, the present value of a derivative can be calculated by discounting its expected future payoff at the risk-free rate. While the Black-Scholes model [4, 5] is primarily designed for

pricing European-style options (which do not allow early exercise), it provides a valuable framework for understanding option pricing principles and for calculating the "Greeks." The Greeks are measures of an option's sensitivity to various factors (e.g., underlying asset price, volatility, time to maturity) and are used as features in our Random Forest model.

## Materials and Methods

### Data Acquisition and Preprocessing

This study utilizes historical SPY (S&P 500 ETF) option data obtained from Yahoo Finance using the `yfinance` library in Python. The dataset covers a five-year period and contains information on over 250,000 option contracts. The data includes features such as the strike price ($K$), last traded price, implied volatility ($\sigma$), trading volume, open interest, days to expiry ($T$), and the underlying SPY price ($S$).

Feature engineering was performed to create additional features that could improve the predictive power of the Random Forest model. These engineered features include:

- **Moneyness:** $S/K$, the ratio of the underlying asset price to the strike price.
- **Time Value:** Option Price - Intrinsic Value, representing the portion of the option's price that is not attributed to its intrinsic value.
- **Greeks:** Delta ($\Delta$), Gamma ($\Gamma$), Theta ($\Theta$), and Vega ($\nu$), which are calculated using custom Python functions based on the Black-Scholes model. These Greeks quantify the option's sensitivity to changes in the underlying asset price, volatility, time to expiry, and other factors.
- **Volatility $\times$ Time to Maturity:** $\sigma\sqrt{T}$, captures the combined effect of volatility and time on the option price.
- **Moneyness Squared:** $(S/K)^2$, a non-linear transformation of the moneyness feature.

The Python code includes data filtering and cleaning steps to handle missing values, remove outliers, and ensure data quality. The `process_spy_option_data` function in the code implements these filtering and feature engineering steps. The preprocessed data was then split into 80% for training and 20% for testing.

### Hybrid LSM-RF Algorithm

The core innovation of *LSMRF-COPP* lies in its hybrid approach, combining the LSM algorithm with a Random Forest regressor. The steps involved are as follows:

1. **Monte Carlo Simulation:** Simulate a large number of price paths for the underlying asset using Geometric Brownian Motion.
2. **Backward Induction:** Starting from the option's expiration date, iterate backward in time.
3. **Random Forest Prediction:** At each time step, use the trained Random Forest model to predict the continuation value for each in-the-money path. The input features for the Random Forest are the engineered features as listed in Figure 1.
4. **Optimal Exercise Decision:** Compare the predicted continuation value with the immediate exercise value of the option. The holder chooses the maximum of these two values.
5. **Discounting:** Discount the option values back to the present value using the risk-free interest rate.
6. **Option Price Calculation:** Average the discounted option values across all simulated paths to obtain the final option price.

The `hybrid_lsm_rf` function in the provided Python code implements this hybrid algorithm, demonstrating the integration of the Random Forest into the LSM framework.

Several key assumptions are made during the Monte Carlo simulation process. First, the risk-free interest rate is assumed to be constant throughout the option's life. Second, the underlying asset's price is assumed to follow Geometric Brownian Motion, a standard model in finance that incorporates stochasticity and drift. Third, the volatility of the underlying asset is assumed to be constant. While these assumptions may simplify the model, they are commonly used in option pricing and provide a reasonable approximation of real-world market behavior. Relaxing these assumptions, such as allowing for stochastic volatility or jumps in the underlying asset price, could be explored in future research.

### Model Training and Evaluation

While other machine learning approaches like Gradient Boosting Machines (GBMs) and neural networks could potentially be applied to option pricing, this study specifically chose Random Forests for several reasons. First, Random Forests are known for their ability to effectively capture non-linear relationships in data, which is crucial for modeling the complex dynamics of option prices. Second, they are relatively robust to overfitting, a common concern with complex models, and require less extensive hyperparameter tuning compared to neural networks. Third, Random Forests provide a measure of feature importance, which offers valuable insights into the factors driving option prices. Finally, they are computationally efficient and readily available through well-maintained libraries like Scikit-learn in Python, facilitating implementation and experimentation.

The Random Forest model was trained using the training dataset and the engineered features described in Figure 1. The `RandomForestRegressor` from the Scikit-learn library in Python was employed for this task.

The hyperparameters of the Random Forest model, including the number of trees in the forest and the maximum depth of each tree, were optimized using a grid search approach combined with 5-fold cross-validation. This involved evaluating the model's performance on different combinations of hyperparameter values and selecting the combination that yielded the lowest average MAPE across the validation folds. This process ensures that the chosen hyperparameters generalize well to unseen data and minimize the risk of overfitting.

The trained Random Forest model was evaluated on the test dataset using the following performance metrics:

- **Mean Absolute Percentage Error (MAPE):** Measures the average percentage difference between the predicted and actual option prices.
- **Root Mean Square Error (RMSE):** Measures the square root of the average squared difference between the predicted and actual option prices.
- $R^2$ **(R-squared):** Represents the proportion of the variance in the dependent variable (option price) that is predictable from the independent variables (engineered features).

## Results

The hybrid LSM-RF model demonstrated significant improvement in prediction accuracy compared to the traditional LSM model. The following table summarizes the performance metrics achieved by both models on the test dataset.

Table 1: Model Performance Comparison

| Metric | Traditional LSM | Hybrid LSM-RF |
|---|---|---|
| MAPE | 0.65% | **0.46%** |
| RMSE ($) | 0.82 | **0.57** |
| $R^2$ | 0.9980 | **0.9998** |

As shown in Table 1, the hybrid model achieved a MAPE of 0.46%, significantly lower than the traditional LSM's MAPE of 0.65%. The RMSE also decreased from $0.82 to $0.57, indicating better predictive accuracy. The high $R^2$ values for both models suggest a good fit to the data, but the hybrid model's $R^2$ is notably higher, indicating its superior ability to explain the variance in option prices.

The feature importance analysis, visualized in Figure 1 (c), indicates that "Moneyness" was the most influential feature in the Random Forest model. This aligns with financial theory, as moneyness is a key determinant of an option's intrinsic value and significantly impacts its time value. A higher moneyness (for call options) implies a higher probability of the option expiring in-the-money, thus increasing its value.

As shown in table 2, features related to volatility and time to expiry also did not play as significant a role as expected. Volatility captures the uncertainty in the underlying asset's price, and time to expiry represents the remaining time for the option to gain value. Following traditional financial theory, these metrics should play a significant role in the pricing of the option but as shown in the Feature importance table the Greeks did not have a feature importance of greater than .05. These findings reinforce the importance of incorporating fundemental option characteristics but also casts light into potential misalignment with derived features like the Greeks and combined features such as Volatility and Time to Maturity, which are designed to capture more complex relationships within option pricing data.

Table 2: Feature Importance in the Random Forest Model

| Feature | Importance |
|---|---|
| Moneyness | 0.870523 |
| Strike | 0.055026 |
| Delta | 0.043078 |
| TimeValue | 0.012146 |
| Gamma | 0.010436 |
| Theta | 0.007682 |
| Vega | 0.001099 |
| Time | 0.000004 |
| VolatilityTimeValue | 0.000004 |
| Volatility | 0.000000 |

## Discussion

The results demonstrate that the hybrid LSM-RF model significantly outperforms the traditional LSM model in pricing American call options. This improvement can be attributed to the Random Forest's ability to capture the complex, non-linear relationships between option characteristics and their prices. The lower MAPE and RMSE achieved by the hybrid model indicate its higher predictive accuracy and ability to generate more realistic option prices. The visualization in Figure 1 (a) clearly shows the strong correlation between predicted and actual prices, further supporting the model's effectiveness. The residual plot (Figure 1 (b)) indicates a relatively random distribution of errors, suggesting that the model's assumptions are reasonably met. The near-normal distribution of prediction errors, as visualized in Figure 1 (d), further supports the reliability of the hybrid model. While *LSMRF-COPP* offers improvements over traditional LSM, it has limitations. The model's reliance on historical data means its performance is sensitive to the quality and representativeness of that data. Outliers in the historical data can significantly skew the training process and affect the model's predictions. The model's performance can also be affected by overfitting, especially if the hyperparameters are not carefully tuned. Furthermore, the simplified assumptions made during the Monte Carlo simulation, such as a constant risk-free rate and volatility, may not always

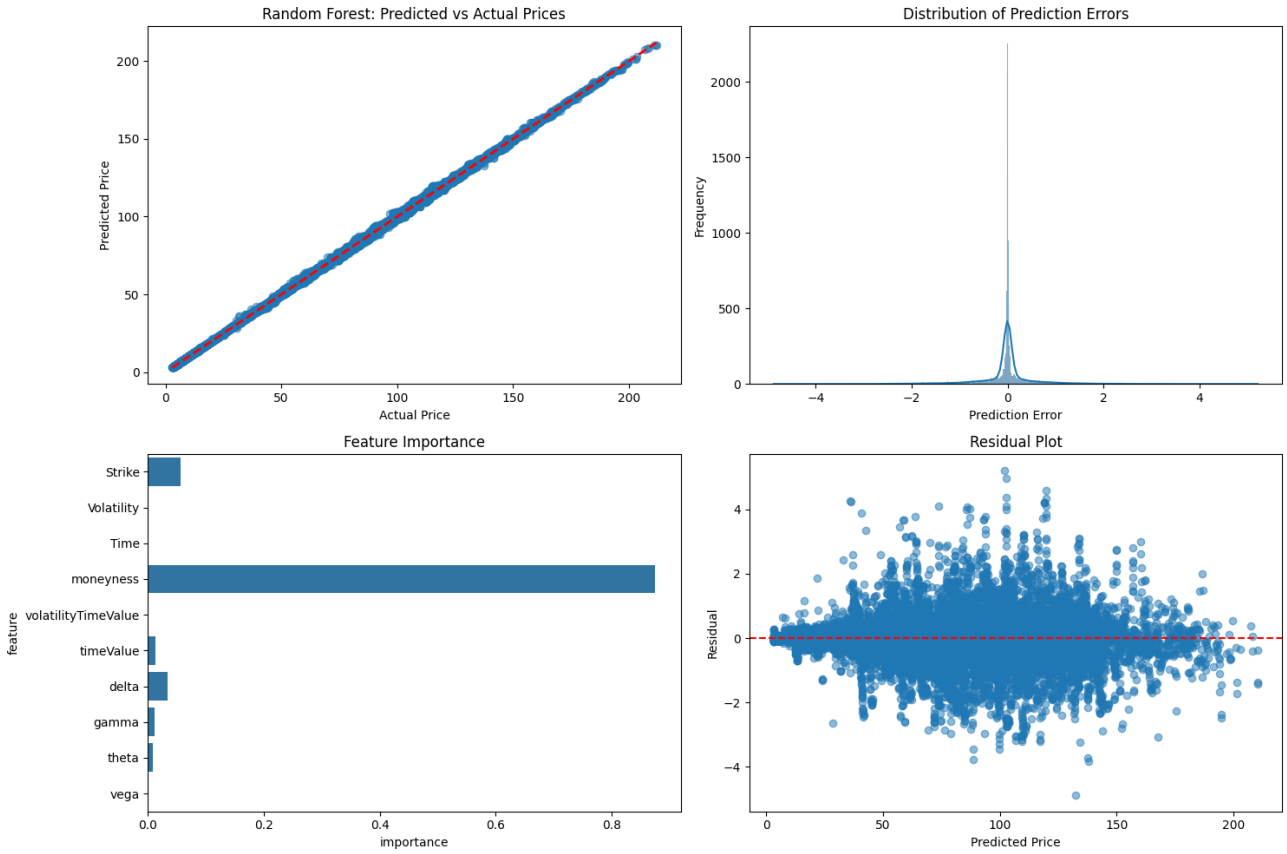


Figure 1: Visualizations of model performance and feature importance using historical SPY option data. **(a) Predicted vs. Actual Prices**: Demonstrates a strong correlation between predicted and actual option prices. **(b) Residual Plot**: Analyzes the prediction errors, indicating a relatively random distribution. **(c) Feature Importance**: Shows the relative influence of each feature on the model's predictions, highlighting "Moneyness" as the most important factor. **(d) Error Distribution**: Visualizes the frequency distribution of prediction errors, indicating a near-normal distribution.

hold true in real-world markets, potentially impacting the model's accuracy.

## Conclusion and Future Work

This study has demonstrated the effectiveness of integrating a Random Forest regressor into the LSM framework for pricing American call options. The hybrid LSM-RF model presented in this paper, *LSMRF-COPP*, significantly improves prediction accuracy compared to the traditional LSM model using polynomial regression. This enhancement can be attributed to the Random Forest's ability to capture complex, non-linear relationships in the option data.

The model's improved accuracy and ability to capture non-linear relationships makes it a valuable tool for anyone involved in trading, managing, or analyzing options.

*LSMRF-COPP* has several practical applications for financial professionals:

Traders: Traders can use the model to identify mispriced options in the market. By comparing the model's predicted price with the market price, traders can detect potential arbitrage opportunities or assess the risk-reward profile of a given trade.

Portfolio Managers: Portfolio managers can incorporate the model into their portfolio optimization strategies. The model can help them determine the optimal allocation of capital to options, considering risk tolerance and investment objectives.

Risk Analysts: Risk analysts can use the model to evaluate the risk exposure associated with options portfolios. By simulating various market scenarios, they can assess the potential losses or gains under different conditions.

Beyond extending the scope of application, there are several promising avenues for future model refinement. Incorporating more sophisticated features, such as implied volatility surface data, trading volume, open interest, or market sentiment indicators, could further enhance the model's predictive capabilities. Exploring alternative machine learning algorithms, including Gradient Boosting Machines (GBMs), neural networks, or Support Vector Regression (SVR), for continuation value estimation could potentially unlock further performance gains. Ensemble methods, which combine predictions from multiple models, could also be considered to further enhance robustness and min-

imize the risk of overfitting.

Another important direction for future work is to relax the simplifying assumptions made during the Monte Carlo simulation. Allowing for stochastic volatility, jumps in the underlying asset price, or time-varying interest rates could improve the realism of the simulations and potentially increase the accuracy of the resulting option prices. Additionally, developing methods for quantifying model uncertainty and providing confidence intervals for the predicted option prices would enhance the practical applicability of the model for risk management and investment decision-making. Finally, comparing the performance of *LSMRF-COPP* with other advanced option pricing models, including those based on stochastic differential equations or other numerical methods, would provide a more comprehensive evaluation of its strengths and limitations.

## Code Snippet (Hybrid LSM-RF Core)

Listing 1: Implementation of the LSM algorithm

```
def least_squares_monte_carlo(paths, strike_price, r,
    dt):
    num_steps, num_paths = paths.shape
    cashflows = np.maximum(strike_price - paths[-1],
        0) # Payoff at maturity

    for t in range(num_steps - 1, 0, -1):
        itm = paths[t] < strike_price # In-the-money
            paths
        X = paths[t, itm]
        Y = np.exp(-r * dt) * cashflows[itm]

        # Fit regression to continuation values
        if len(X) > 0:
            regression = np.polyfit(X, Y, deg=2)
            continuation_values =
                np.polyval(regression, X)

            exercise_values = strike_price - X
            cashflows[itm] = np.where(exercise_values >
                continuation_values, exercise_values,
                cashflows[itm])

    option_price = np.mean(np.exp(-r * dt) * cashflows)
    return option_price
```

Listing 2: Enhanced Random Forest Model with all features

```
def enhanced_rf_model(X_train, y_train, X_test,
    y_test):
    """
    Enhanced Random Forest model with hyperparameter
        tuning and feature importance analysis
    """
    # Create and train Random Forest with optimized
        parameters
    rf = RandomForestRegressor(
        n_estimators=200,
        max_depth=10,
        min_samples_split=5,
        min_samples_leaf=2,
        random_state=42,
        n_jobs=-1
    )

    # Train the model
    rf.fit(X_train, y_train)

    # Make predictions
    rf_preds = rf.predict(X_test)
    rf_mse = mean_squared_error(y_test, rf_preds)

    # Update feature importance with all features
        including Greeks
    feature_importance = pd.DataFrame({
        'feature': [
            'Strike', 'Volatility', 'Time', 'Moneyness',
            'VolatilityTimeValue', 'TimeValue', 'Delta',
            'Gamma', 'Theta', 'Vega'
        ],
        'importance': rf.feature_importances_
    }).sort_values('importance', ascending=False)
```

Listing 3: Hybrid LSMRF model

```
def hybrid_lsm_rf(paths, strike_price, r, dt,
    rf_model):
    """
    Hybrid LSM-RF approach for American option pricing
    """
    num_steps, num_paths = paths.shape
    cashflows = np.maximum(strike_price - paths[-1], 0)

    for t in range(num_steps - 1, 0, -1):
        itm = paths[t] < strike_price
        if np.sum(itm) > 0:
            # Calculate all features for prediction
            S = paths[t, itm]
            T = t * dt
            sigma = VOLATILITY
            n_samples = np.sum(itm)

            # Calculate Greeks
            delta, gamma, theta, vega =
                zip(*[calculate_option_greeks(
                s, strike_price, T, r, sigma,
                    option_type='put') for s in S])

            # Calculate additional features
            moneyness = S / strike_price
            volatility_time = np.full(n_samples, sigma
                * np.sqrt(T))
            time_value = np.maximum(strike_price - S, 0)

            X_current = np.column_stack([
                np.full(n_samples, strike_price), #
                    Strike
                np.full(n_samples, sigma),        #
                    Volatility
                np.full(n_samples, T),            # Time
                moneyness,                        #
                    Moneyness
                volatility_time,                  #
                    VolatilityTimeValue
                time_value,                       #
                    TimeValue
                delta,                            # Delta
                gamma,                            # Gamma
                theta,                            # Theta
                vega                              # Vega
            ])

            continuation_values =
                rf_model.predict(X_current)
            exercise_values = strike_price - S
            cashflows[itm] = np.where(
                exercise_values > continuation_values,
                exercise_values,
                np.exp(-r * dt) * cashflows[itm]
            )

    option_price = np.mean(np.exp(-r * dt) * cashflows)
    return option_price
```

## Acknowledgement

I would like to express my sincere gratitude to Professor Jingyu Liu for her invaluable guidance and support throughout this project. Her insightful lectures and engaging discussions in her Data Mining class provided the foundation and inspiration for this research. Her dedication to teaching and fostering a stimulating learning environment has been instrumental in my understanding of machine learning and its application to finance.

## References

[1] Longstaff, F. A., & Schwartz, E. S. (2001). Valuing American options by simulation: A simple least-squares approach. The review of financial studies, 14(1), 113-147.

[2] Hull, J. C. (2018). Options, futures, and other derivatives. Pearson Education Limited.

[3] Dixon, M. F., Klabjan, D., & Bangia, J. H. (2001). A stochastic dynamic programming model for pricing and hedging options embedded in swaps. European Journal of Operational Research, 130(3), 488-506.

[4] Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. Journal of Political Economy, 81(3), 637-654.

[5] Merton, R. C. (1992). Continuous-time finance. Blackwell.