Machine Learning Method for American Option Pricing: A Comparative Study of Models Under the Heston Framework

MS Financial Engineering
University of Illinois Urbana-Champaign
Department of ISE

Team Members (alphabetic):

Chengjia Dong
Ruohao Huang
Yifan Meng
Yiwen Zhang

December 11th, 2024

Contents

Abstract

This paper investigates the application of various machine learning models for pricing American options under the Heston stochastic volatility framework. Traditional pricing methods, including finite difference methods and Least Squares Monte Carlo, are contrasted with modern machine learning approaches such as ensemble methods (e.g., Random Forest, Gradient Boosting, XGBoost, CatBoost, LightGBM) and neural networks (FNN, CNN). Using a dataset derived from SPY option market data and calibrated Heston parameters, the models are rigorously evaluated based on accuracy (RMSE, R^2), computational efficiency, and robustness. Results indicate that ensemble methods like CatBoost and LightGBM outperform others in accuracy and training efficiency, while neural networks show potential but require further optimization. This study underscores the strengths and limitations of machine learning models in capturing the complex dynamics of American option pricing and offers a roadmap for integrating these techniques into real-world financial applications. Future research may focus on improving neural network architectures and exploring hybrid models to enhance performance.

Executive Summary

This study explores the integration of machine learning models to improve the pricing of American-style options under the Heston stochastic volatility framework. Traditional methods, such as finite difference techniques and the Least Squares Monte Carlo method, provide reliable results but often involve high computational costs and limited scalability. Machine learning, with its ability to model complex nonlinear relationships, offers a promising alternative to these conventional approaches.

The research leverages a dataset generated from SPY market data, calibrated to the Heston model. Predictive models evaluated include linear regression, ensemble methods (Random Forest, Gradient Boosting, XGBoost, CatBoost, LightGBM), and neural networks (Feedforward and Convolutional Neural Networks). Each model was assessed based on its accuracy (measured by RMSE and R^2), computational efficiency, and ability to generalize across feature spaces. The study also examined model behavior in scenarios with high prediction errors, providing insights into feature importance and model limitations.

Key findings highlight the superiority of ensemble methods such as CatBoost and LightGBM, which delivered the best balance of accuracy and training efficiency. CatBoost achieved the lowest RMSE with minimal computational time, making it particularly suited for practical financial applications requiring both precision and speed. Neural networks, while capable of capturing complex relationships, exhibited performance challenges, likely due to overfitting or insufficient optimization for the dataset.

The study demonstrates the potential of machine learning models to enhance the pricing of American options, offering faster and more scalable alternatives to traditional methods. However, the challenges faced by neural networks indicate opportunities for future research, such as refining architectures or combining machine learning with domain-specific knowledge to improve interpretability and accuracy.

In conclusion, this work provides a comprehensive evaluation of machine learning techniques in American option pricing, laying the groundwork for further advancements in the intersection of finance and artificial intelligence. These findings can guide practitioners in selecting appropriate models for real-world applications and inspire continued innovation in this field.

1 Introduction

In the rapidly evolving field of quantitative finance, accurate and efficient pricing of American options remains one of the most challenging yet crucial tasks. Unlike European options, which can only be exercised at expiration, American options offer the flexibility of early exercise, making their valuation inherently complex. When coupled with the stochastic volatility dynamics of the Heston model [7], the computational challenges increase significantly. Traditional numerical approaches such as finite difference methods, binomial trees, and Monte Carlo simulations have demonstrated efficacy but often require significant computational resources [4]. The growing demand for faster, more scalable solutions has shifted focus toward leveraging machine learning techniques.

Recent advancements in machine learning provide promising avenues for addressing these challenges. By capturing complex, nonlinear relationships between option parameters and their prices, machine learning models offer a viable alternative to conventional approaches. For instance, deep neural networks have been shown to significantly accelerate the pricing of American options while maintaining high accuracy [1]. This paper investigates a range of predictive models, to identify the best-performing model for pricing American options under the Heston model framework. The models considered include traditional linear regression techniques, ensemble methods such as Random Forests and Gradient Boosting (e.g., XGBoost, CatBoost, and LightGBM), and neural networks, including Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs).

The primary goal of this research is to compare the effectiveness and efficiency of these models in American option pricing. Specifically, we aim to balance pricing accuracy with computational time, a critical factor for real-time applications in financial markets. Furthermore, we explore the potential of these models to enhance research in American option pricing by offering insights into their applicability across different parameter settings and market conditions.

This paper contributes to the growing literature at the intersection of machine learning and financial derivatives. By systematically evaluating the strengths and limitations of various predictive models, we provide a roadmap for future research and practical implementation of machine learning techniques in option pricing.

2 Literature Review

The valuation of American options has been a subject of extensive research over decades due to the challenges posed by their early exercise feature. A variety of methods have been developed to address these challenges, broadly categorized into analytical, numerical, and machine learning-based approaches.

2.1 Analytical Methods

One of the earliest analytical approaches for American option pricing was the quadratic approximation introduced by Barone-Adesi and Whaley [2]. This method simplifies the valuation by approximating the optimal exercise boundary and solving the resulting equations analytically. This method is computationally efficient, but it loses accuracy for options with long maturities or extreme moneyness conditions.

Another significant development was Kim's integral representation [9], which decomposes the American option price into its European counterpart and an early exercise

premium. Although this method provides precise theoretical foundations, it requires solving a complex integral, making it computationally intensive in practice.

2.2 Numerical Methods

Numerical approaches have been more commonly employed for American option pricing due to their flexibility in handling the option's early exercise feature. The binomial tree model, introduced by Cox, Ross, and Rubinstein [6], was one of the first practical numerical methods. This method constructs a recombining lattice to model price movements and evaluates the option by backward induction. The trinomial tree, as an extension, improves convergence rates but at the cost of increased complexity.

Finite difference methods (FDM), both explicit and implicit, solve the partial differential equations (PDEs) governing option prices [3]. These methods allow for high accuracy and flexibility but require fine-tuning of grid parameters to ensure stability and convergence. Moreover, their computational time increases exponentially with dimensionality, making them less suitable for high-dimensional problems.

Monte Carlo simulation is another popular approach for pricing derivatives, but its application to American options is non-trivial due to the early exercise feature. The Least Squares Monte Carlo (LSM) method proposed by Longstaff and Schwartz [10] overcomes this limitation by using regression to estimate the continuation value of the option, enabling optimal exercise decisions. However, the LSM method's accuracy depends heavily on the choice of basis functions, and it can be computationally expensive for large-scale problems.

2.3 Machine Learning Approaches

In recent years, machine learning techniques have emerged as powerful tools for option pricing. Neural networks, in particular, have shown great promise due to their ability to approximate complex and nonlinear functions. Anderson and Ulrych [1] demonstrated that deep neural networks could significantly accelerate the pricing of American options, providing accurate results with substantial reductions in computational time. Despite their effectiveness, these models require large training datasets and careful hyperparameter tuning to prevent overfitting.

Other machine learning methods, such as gradient boosting frameworks (e.g., XGBoost and LightGBM), have also been applied to financial derivatives pricing. These models excel in capturing intricate relationships between input parameters and prices while maintaining interpretability [5, 8]. However, they may struggle to generalize in highly volatile market conditions.

2.4 Comparison and Challenges

Each method for pricing American options offers unique advantages and limitations. Analytical methods are computationally efficient but may lack accuracy for complex or extreme scenarios. Numerical approaches provide flexibility and accuracy but often require significant computational resources. Machine learning models, while promising for their speed and adaptability, rely heavily on the availability of high-quality data and involve challenges in interpretability.

Given these trade-offs, there is no universally optimal method for pricing American options. Instead, the choice of method depends on the specific requirements of the ap-

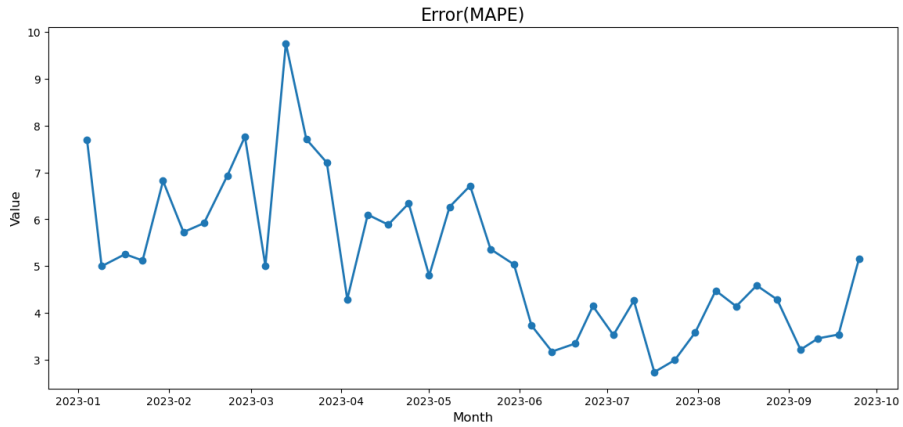

Figure 1: Calibration Error By Time

plication, such as accuracy, computational efficiency, and scalability. This paper seeks to contribute to this ongoing discussion by comparing traditional and machine learning-based models under the Heston framework.

3 Exploratory Data Analysis

3.1 Training Dataset Description

The dataset contains daily European option market data for SPY between January 4, 2023, and September 29, 2023. Below is the table outlining each column's description (see Table 1).

Column Name	**Description**
Quote date	The date the option quote was recorded
Expire date	The expiration date of the option
DTE	Days to expiration (time to maturity)
Underlying last	Last recorded price of the underlying asset (SPY)
Strike	The strike price of the option
C price	The mid-price for the call option
rf	Risk-free rate used for pricing
Div	Dividend yield of the underlying asset

Table 1: Description of Dataset Variables

3.2 Sample Data Generation and Calibration

Upon obtaining this real-world dataset, we calibrated the Heston model parameters based on the observed underlying and option data. The calibrated parameters include:v_0,θ,κ,σ,ρ. Using the dataset and the calibrated Heston model parameters, we generated sufficient variables to serve as features for machine learning. The target variable for our machine learning model is the price of American-style options. The explanatory variables includes in Table 2.

Parameter/Feature	**Description**
m	moneyness, strike price over the spot price
T	Days to expiration
rf	Risk-free rate
div	Yield of the underlying asset
v_0	Initial volatility of the asset
θ	Long-term expected volatility
κ	Volatility reversion rate
σ	Volatility of volatility
ρ	Correlation between the two Brownian motions
Target Variable	Price of American-style options

Table 2: Machine Learning Features

3.3 Data Generation Process

In this section, we describe the process of calibrating the Heston model parameters and generating simulated data for SPY European options to create a smoother and more representative volatility surface.

While general parameter ranges for models like Black-Scholes-Merton (BSM) and Heston are typically broad to accommodate various underlying assets, SPY exhibits specific characteristics that allow us to define more targeted parameter ranges. Although SPY's features may evolve over time, many characteristics remain consistent within a narrower scope.

To improve the model's effectiveness, we calibrated the Heston model parameters using SPY's 2023 data. This calibration ensures that the resulting parameter ranges are more tailored to SPY's characteristics. With these refined ranges, we generated simulated data that better replicates a smooth volatility surface reflective of SPY's market behavior.

Fast Fourier Transform (FFT) is an efficient numerical method used to compute Fourier transforms, enabling fast option pricing under the Heston model. By transforming the partial differential equation (PDE) of the Heston model into Fourier space, FFT reduces computational complexity and enhances accuracy. This method is particularly advantageous over finite difference methods in terms of both speed and precision.

We aim to solve the following optimization problem:

$$\text{Objective:} \quad \arg\min_{v_0,\theta,\kappa,\sigma,\rho} \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\text{FFT Price}_i - \text{C\_price}_i}{\text{C\_price}_i} \right| \times 100$$

subject to:

$$\begin{aligned} &0.01 \leq v_0 \leq 0.99, \\ &0.01 \leq \theta \leq 2, \\ &0.1 \leq \kappa \leq 10, \\ &0.01 \leq \sigma \leq 0.99, \\ &-1.0 \leq \rho \leq 1.0. \end{aligned}$$

For each day, we use FFT to compute option prices based on the current guess for Heston parameters, compare the model prices with the market prices and update the parameters iteratively to minimize the error by using Differential Evolution. Then, repeat the process for each day in the dataset to derive daily Heston model parameters.

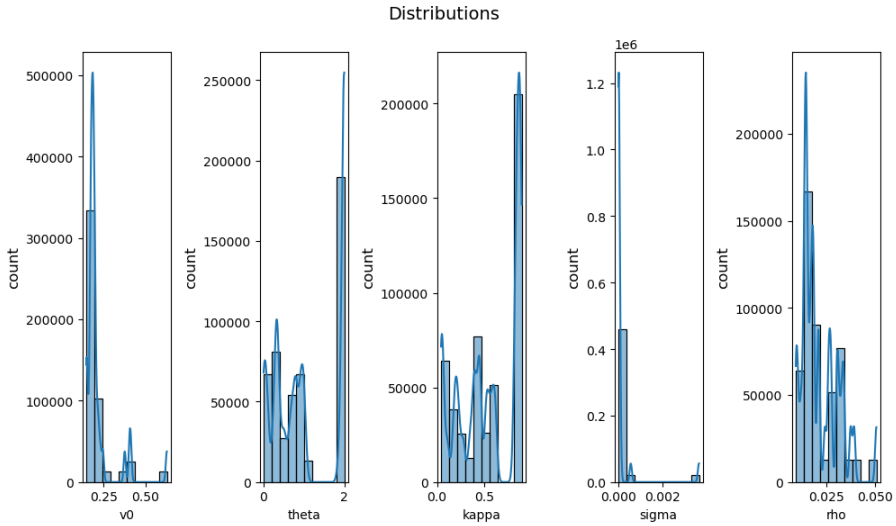

Figure 2: Distribution of Sampled Data

Once calibrated, the refined parameter ranges were used to generate simulated data points aligned with SPY's market behavior. By doing so, we guarantee a smooth volatility surface based on the calibrated Heston parameters.

we observed that the calibrated parameters do not follow a normal distribution or a uniform distribution. This irregularity in distribution makes it unsuitable to simulate new data simply based on upper and lower bounds, means, and standard deviations, as it would misrepresent the true underlying distribution.

To ensure that the distributional characteristics of the calibrated data remain intact, we employed empirical inverse cumulative distribution function sampling to generate simulated data. Unlike methods based on Gaussian or uniform assumptions, this approach ensures that the resampled data maintains the same irregular and asset-specific distribution as the original calibrated data.

After completing the steps outlined above, we successfully generated a dataset based on simulated data derived from real market data. Using FFT, we can calculate the heston model prices of American-style options, which serve as the target variable for the subsequent machine learning analysis.

4 Data Preprocessing

4.1 Data Cleaning

Before calculating the American call option prices, we need to verify the feller condition, which ensures the mathematical validity of the Heston model. The Feller Condition is:

$$2\kappa\theta \geq \sigma^2$$

This condition guarantees that the variance process remains positive, which is crucial for accurate and consistent option pricing. After generating American option prices using FFT, we identified some irregular and implausible prices. By removing these outliers, we can help the machine learning model converge faster. The Interquartile Range (IQR) method was applied to identify and remove extreme outliers. The IQR is calculated as

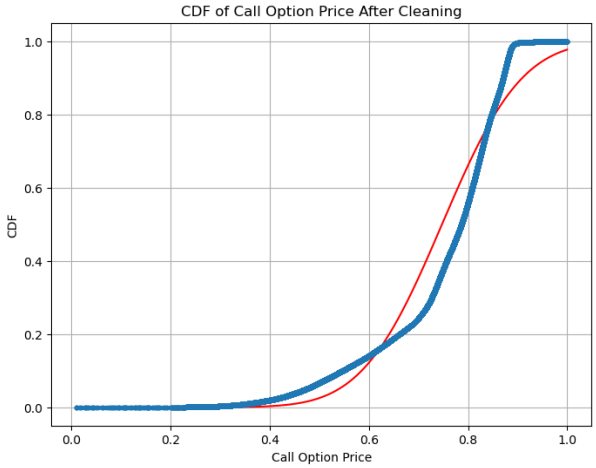

Figure 3: CDF of Call Option Price

the difference between the third quartile ($Q3$) and the first quartile ($Q1$):

$$\text{IQR} = Q3 - Q1$$

Using the IQR, the lower and upper bounds are defined as:

$$\text{Lower Bound} = Q1 - 1.5 \times \text{IQR}$$

$$\text{Upper Bound} = Q3 + 1.5 \times \text{IQR}$$

This approach is well-suited for non-symmetric, non-parametric data distributions. It is sensitive to extreme values and quickly detects anomalies. After this step, all excessively large option prices and negative option prices were excluded.

After using IQR method, prices of American call options were checked to ensure they lie within the following theoretical bounds:

$$\max(S - Ke^{-rT}, 0) \leq C \leq S$$

This step ensures that the American option prices fall within their economically meaningful range. To confirm the quality of the cleaned dataset, we visualized the cumulative distribution function of the option prices. The resulting plot exhibited a left skew, indicating that the data was now reliable and aligned with realistic market behavior. Finally, we split the dataset into two parts: 80 percent for training the machine learning model and 20 percent for testing its performance.

4.2 Feature Engineering

We defer the detailed discussion of feature extraction and feature selection to Section 6, because we deduced different results for different models, and a separate section dedicated to implementation details will enhance the readability of the paper.

5 Methodology

5.1 Models Used

5.1.1 Linear Models

Linear regression models assume a linear relationship between the independent and dependent variables. We employ three variants:

- Linear Regression: A basic model that fits a linear equation to the data.
- Ridge Regression: Introduces L2 regularization to prevent overfitting.
- Lasso Regression: Introduces L1 regularization for feature selection.

5.1.2 Ensemble Methods

Ensemble methods combine multiple base models to improve performance. We consider:

- Random Forest: A widely used ensemble technique that combines multiple decision trees. By training each tree on random subsets of the data and features, Random Forest reduces overfitting and provides stable and reliable performance. This makes it particularly effective for capturing complex relationships in tasks such as American option pricing.
- Gradient Boost Regressor: A technique that builds models sequentially, with each model correcting the errors of the previous one. This approach enables the model to focus on difficult samples, resulting in high accuracy for capturing intricate patterns in option pricing data.
- XGBoost: An optimized implementation of gradient boosting that incorporates advanced features such as regularization and parallel processing. These enhancements improve both speed and robustness, making XGBoost suitable for large-scale datasets commonly encountered in financial modeling.
- CatBoost: A gradient boosting framework designed to handle categorical features efficiently without extensive preprocessing. This capability makes it valuable for datasets containing categorical data, such as American options influenced by market states or conditions. Additionally, CatBoost incorporates techniques to mitigate overfitting, ensuring strong performance even on smaller datasets.
- LightGBM: A high-performance gradient boosting framework optimized for speed and scalability. Its novel leaf-wise splitting strategy reduces loss effectively, making it well-suited for handling large datasets in computationally intensive tasks such as real-time American option pricing. LightGBM's combination of speed and accuracy makes it a robust choice for financial applications.

5.1.3 Neural Networks

Neural networks are inspired by the human brain. We explore:

- FNN: A feedforward neural network that processes data through multiple layers of interconnected neurons in a unidirectional manner. FNN is capable of capturing complex, non-linear relationships in data, making it suitable for tasks like American option pricing. However, it requires careful tuning of parameters to prevent overfitting, especially when dealing with smaller datasets.

- CNN: A convolutional neural network designed primarily for image and spatial data processing but can be adapted to structured data by capturing local patterns effectively. CNN's ability to learn hierarchical features makes it a powerful tool for complex modeling tasks. Its high computational cost and longer training times can be a challenge for applications like American option pricing without further optimization.

5.2 Evaluation Metrics

To evaluate the behavior of each model: We employ a structured and rigorous approach that integrates error analysis, visualization, and feature-level diagnostics. This methodology offers a comprehensive framework to assess the model's performance and identify potential areas for improvement.

The evaluation begins with the calculation of prediction errors, defined as the differences between predicted and actual values. To isolate significant deviations, a threshold is determined using the 99th percentile of the error distribution. Data points exceeding this threshold are classified as large-error cases, highlighting instances where the model's predictions diverge substantially from observed values. Such a categorization enables the focused analysis of outliers that may reveal systemic issues or limitations in the model's assumptions.

The analysis incorporates visual tools to explore the error dynamics. A scatter plot of prediction errors, indexed by their position in the test set, provides an intuitive representation of error distribution. A demarcation line representing the threshold emphasizes large-error cases, enabling a clear distinction between normal variations and extreme discrepancies. Similarly, the relationship between actual and predicted values is visualized through a scatter plot with a diagonal reference line representing perfect predictions. Points deviating significantly from this line are highlighted to facilitate a nuanced evaluation of the model's predictive accuracy.

To investigate the underlying causes of large errors, the evaluation compares the feature characteristics of these instances against the overall dataset. The mean values of key features are examined to identify patterns or anomalies associated with poor performance. This comparison elucidates whether specific feature ranges disproportionately contribute to large prediction errors, offering insights into potential model vulnerabilities.

The feature distributions are further analyzed using density estimation techniques, which contrast the distributions of all data points with those of large-error cases. Such visualizations can reveal whether particular feature values are associated with significant prediction inaccuracies, enabling a deeper understanding of the model's behavior under varying data conditions.

In addition to distributional analysis, the evaluation quantifies the relationship between individual features and prediction errors through correlation analysis. This step identifies features with the strongest associations to error magnitudes, providing actionable insights for addressing model deficiencies. Features demonstrating high correlations

with errors may indicate areas where adjustments in the modeling process, such as feature transformations or enhanced representations, could yield improvements.

Overall, this multi-faceted evaluation approach integrates statistical, visual, and diagnostic tools to assess model performance comprehensively. By identifying systematic patterns in errors and their relationship to data characteristics, this methodology not only quantifies predictive accuracy but also guides targeted refinements to improve model robustness and reliability.

To compare the different models, we evaluate model performance using:

- Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values.
- Root Mean Squared Error (RMSE): The square root of MSE, providing a more interpretable error metric.
- Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual values.

5.3 Hyperparameter Tuning

5.3.1 Grid search

In the hyperparameter tuning process, grid search is employed as a systematic approach to identify the optimal combination of hyperparameters for the model. This method involves defining a discrete search space for each hyperparameter and exhaustively evaluating all possible combinations. By iterating over the predefined grid, the model's performance is assessed for each configuration using a validation dataset, ensuring that the selected hyperparameters generalize well to unseen data. Although computationally intensive, grid search provides a comprehensive exploration of the parameter space, making it particularly suitable for smaller models or when the parameter space is constrained. This approach is critical for improving model accuracy and robustness, as it allows for a fine-grained optimization of key hyperparameters such as learning rate, batch size, and the number of filters in convolutional layers.

5.3.2 Optuna

While traditional grid search and random search methods can be inefficient, Bayesian optimization offers a more intelligent and efficient approach. Optuna is a state-of-the-art hyperparameter optimization framework that leverages Bayesian optimization to streamline this process.

1. Core Concepts in Optuna
    - **Study:** A study represents a single optimization experiment, encapsulating a series of trials, objective function evaluations, and corresponding hyperparameter configurations.
    - **Trial:** A trial corresponds to a single evaluation of the objective function with a specific set of hyperparameters.
    - **Objective Function:** The objective function quantifies the model's performance. In the context of regression tasks, a common choice for the objective function is Root Mean Squared Error (RMSE).

- **Pruning:** Pruning is a technique that early terminates unpromising trials, thereby saving computational resources and accelerating the optimization process.

2. Bayesian Optimization in Optuna

   At the heart of Optuna lies Bayesian optimization, a probabilistic approach to global optimization. It models the objective function as a Gaussian process, allowing for uncertainty quantification and efficient exploration of the hyperparameter space.

   - **Initialization:** A few initial hyperparameter configurations are sampled randomly.
   - **Modeling:** A Gaussian process prior is constructed to represent the objective function.
   - **Sampling:** A new hyperparameter configuration is sampled using an acquisition function, such as Expected Improvement (EI) or Probability of Improvement (PI).
   - **Evaluation:** The objective function is evaluated at the sampled point.
   - **Update:** The Gaussian process model is updated with the new observation.
   - **Iteration:** Steps 2-5 are repeated until a stopping criterion is met.

     By iteratively refining the probabilistic model and selecting promising hyperparameter configurations, Optuna efficiently explores the search space and converges to optimal solutions.

6 Results of models

Present and compare the results from all models. Use tables, charts, and graphs to summarize findings.

6.1 Linear Regression

In this part, we will employ linear models to forecast preprocessed data. As a foundational and interpretable statistical model, linear regression has found extensive applications in finance, particularly in option pricing. The underlying principle involves establishing linear relationships between option prices and various influencing factors, such as the underlying asset price, time to maturity, implied volatility, and the risk-free interest rate, through linear combinations. The model's simplicity facilitates understanding and interpretation, with coefficients directly reflecting the influence of each factor on the option price. Additionally, linear models exhibit high computational efficiency, making them suitable for large-scale datasets.

We prepare the data using historical option price records that include the underlying asset prices, time to maturity, implied volatility, and risk-free interest rates. Data cleaning involves handling missing values and outliers, while feature engineering focuses on creating meaningful variables that capture essential characteristics of the underlying financial instruments. These steps ensure that the model receives high-quality, informative inputs.

An appropriate linear regression algorithm, such as ordinary least squares, will then be selected and trained on the processed data. The model is fitted to a training set, and

its parameters are estimated by minimizing the mean squared error (MSE). Through this process, the model learns a set of coefficients representing the influence of each feature on the option price. The performance of the trained model is evaluated on a testing dataset, employing metrics such as MSE, root mean squared error (RMSE), and the coefficient of determination (R^2). If the predictive performance proves unsatisfactory, potential refinements—such as modifying feature sets, adjusting model parameters, or exploring different loss functions—may be considered.

To enhance model training efficiency and stability, features are standardized to a common scale. Subsequently, a linear regression model is instantiated using the Linear-Regression class from Scikit-learn and trained on the standardized data. By minimizing the mean squared error, this approach identifies coefficients that reduce the discrepancy between predicted and actual values. Through iterative adjustments, the model refines its parameter estimates, thereby encoding the linear mappings between input features and the target variable.

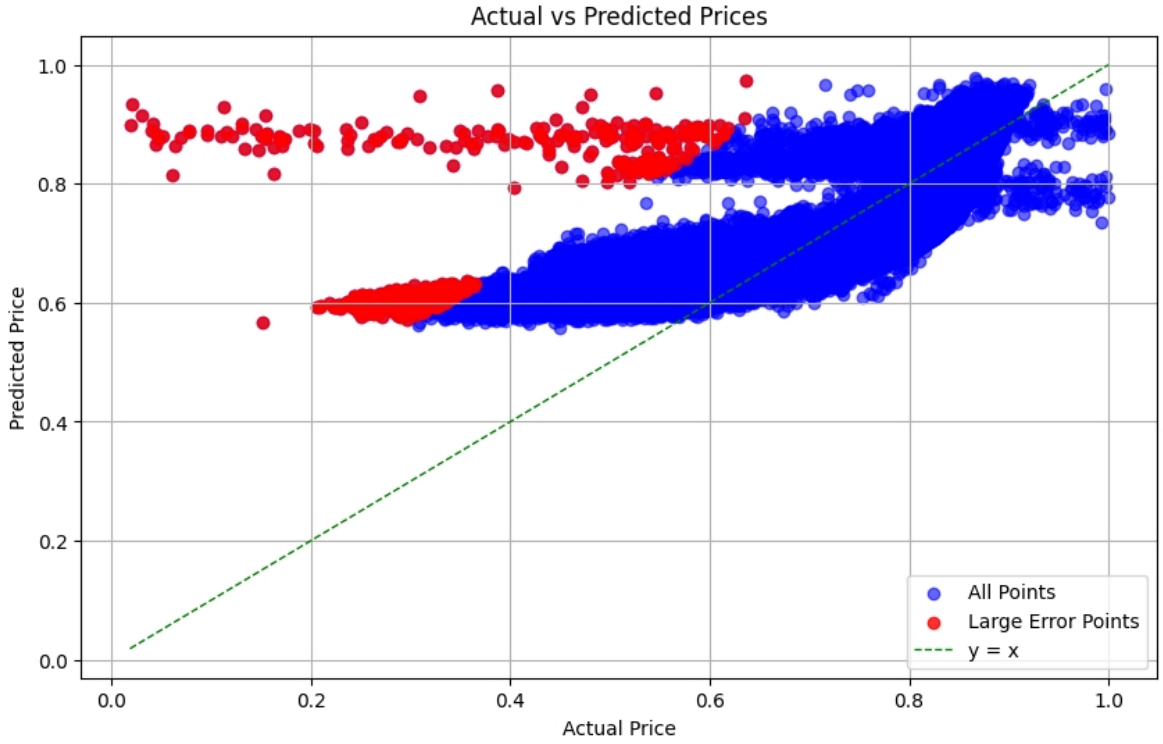

Figure 4: Predictive Accuracy and Biases

The accompanying plot, which compares the actual versus predicted option prices, provides a visual assessment of the model's predictive accuracy. The diagonal line denotes the ideal scenario where predicted values perfectly match actual values. Points aligning closely with this line indicate accurate predictions, whereas those deviating substantially suggest potential biases or model limitations. The highlighted large-error points signal observations where the model's forecasts diverge significantly from the true values, underscoring areas for further refinement or additional feature considerations.

6.2 Ridge Regression

Ridge regression is a specialized form of linear regression that addresses the potential overfitting issues inherent in ordinary least squares by incorporating an L2 regularization

term into the loss function. This regularization term, by employing L2 regularization, we introduce a penalty term to the loss function, which is the sum of the squares of all coefficients. This penalty term encourages the model to have smaller coefficients, thereby reducing model complexity and mitigating overfitting. When confronted with multicollinearity, a condition where features are highly correlated, ridge regression proves particularly effective in enhancing model stability. The shrinkage of coefficients induced by ridge regression renders the model more robust to noise. In financial data, where high correlations between features are commonplace, ridge regression offers a robust solution to address multicollinearity and improve overall model stability.

In Ridge regression, the optimal regularization parameter (alpha) is identified by evaluating model performance across a predefined set of candidate values and selecting the one that best balances model complexity and predictive accuracy. In this study, we considered a discrete range of alpha values (0.01, 0.1, 1.0, 10.0, 100.0) and employed a systematic grid search procedure with cross-validation to ensure that the chosen parameter is robust across multiple training folds. By assessing the mean squared error (MSE) for each candidate alpha, we isolated the alpha that yielded the lowest validation error, thus refining the model's regularization strength in a principled manner. When compared with a baseline linear regression model, the Ridge regression model tuned via this process demonstrated a slight improvement in predictive performance, as evidenced by a marginally lower MSE and higher R^2 value. This improvement reflects the benefit of appropriate regularization in achieving a more stable and generalizable model.

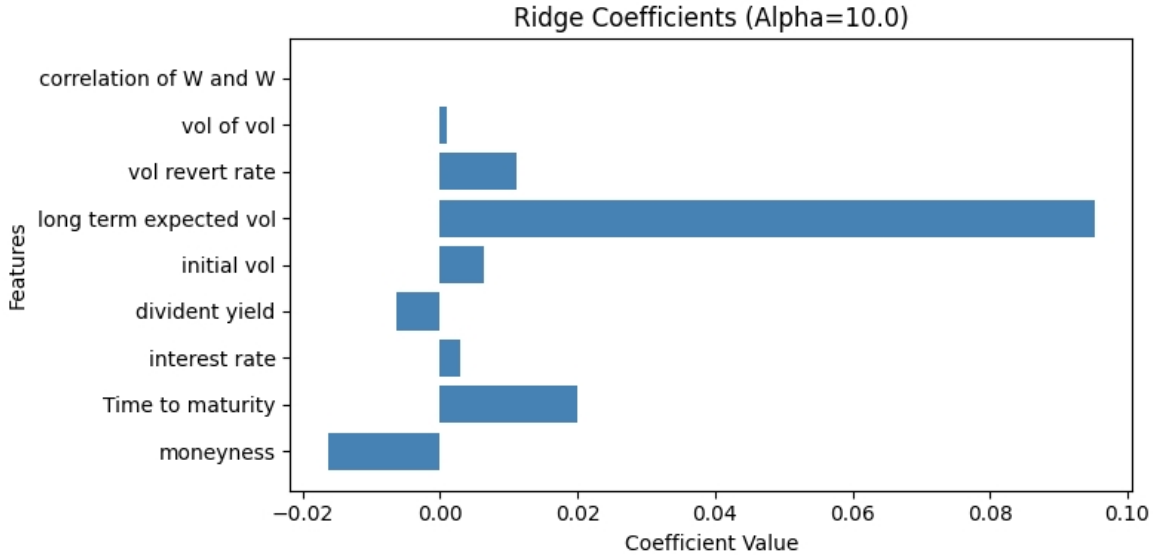

Figure 5: visualization of Ridge Coefficients

Moreover, the visualization of the Ridge coefficients at a selected alpha level (e.g., alpha=10.0) offers additional insight into the model's behavior and feature importance. In this representation, the horizontal bars indicate the magnitude and direction (positive or negative) of the coefficients assigned to each predictor variable. Larger absolute coefficients suggest that a given feature exerts a greater influence on the model's predictions, whereas smaller or near-zero coefficients indicate lesser relevance. Positive coefficients imply that increases in the corresponding features lead to higher predicted values, while negative coefficients have the opposite effect. The Ridge penalty, by constraining the

overall magnitude of the coefficients, prevents any single predictor from dominating the model and encourages more balanced utilization of available features. This results in enhanced model stability and interpretability. Overall, the combination of parameter tuning and coefficient visualization provides a comprehensive understanding of the trade-offs and benefits associated with regularized linear modeling.

6.3 Lasso Regression

In addition to employing Ridge regression, we also incorporated Lasso regression into the initial comparison of predictive models. Lasso regression applies an L1 regularization penalty, which differs fundamentally from the L2 penalty utilized by Ridge. While Ridge regression continuously shrinks coefficients toward zero without fully removing them, Lasso regression can drive certain parameter estimates to an exact zero, effectively performing feature selection by eliminating variables deemed less relevant to the prediction task. This property is particularly advantageous when dealing with high-dimensional data or when there is an expectation that only a subset of features holds substantial predictive power.

In the code, Lasso regression was tested alongside both ordinary Linear Regression and Ridge regression using a fixed alpha value of 0.1. By fitting the model on scaled training data and subsequently evaluating its performance on the test set, we obtained initial measures of prediction accuracy, including the mean squared error (MSE), the coefficient of determination (R^2), and the mean absolute error (MAE). This initial assessment revealed how Lasso's inherent ability to enforce sparsity in the model coefficients may lead to improved interpretability and, potentially, more robust generalization.

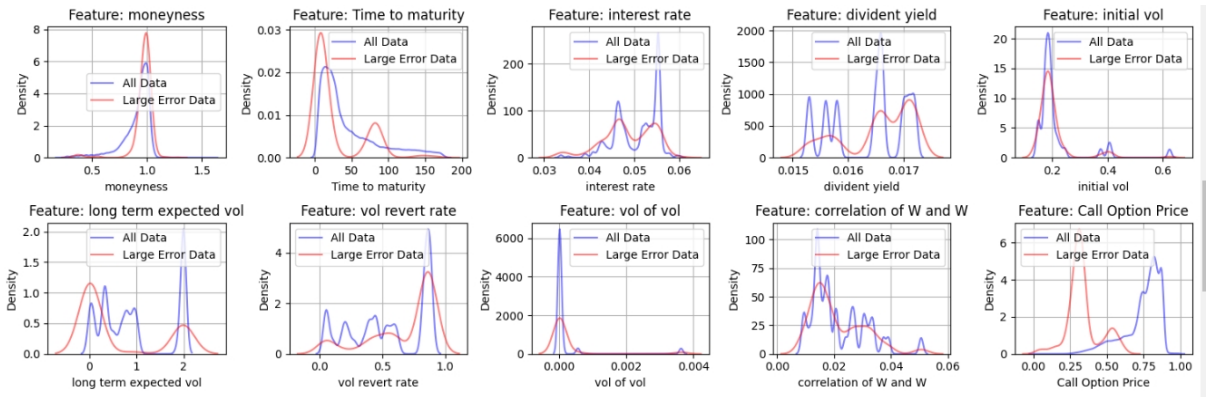

Figure 6: Linear Model Feature Distribution Comparison

The performance comparison among Linear Regression, Ridge Regression, and Lasso Regression reveals that the differences in metrics, such as MSE, RMSE, R^2, and MAE, are minimal. This outcome is primarily due to the high-quality features derived from the Heston stochastic volatility model, which are strongly linked to the target variable (call option price). These features are theoretically robust, capturing the core dynamics of the pricing process, and exhibit minimal redundancy or noise. The density plots further confirm this, showing stable and concentrated distributions across both the entire dataset and the subset with large errors. As a result, regularization methods like Ridge and Lasso, which are designed to handle noisy or highly collinear data, contribute little additional benefit in this context.

We selected Ridge Regression as the final model due to its regularization advantage, which ensures slightly improved robustness, particularly under potential but limited multicollinearity. However, it is important to note that the performance differences between Ridge Regression, Lasso Regression, and even the simpler Linear Regression model are negligible. This underscores the effectiveness of the features, which are well-structured and capture most of the variance in the data even without regularization. Therefore, while Ridge is a robust choice, the overall similarity in results reflects the inherently high quality and strong theoretical foundation of the input features from the Heston model.

6.4 Random Forest

The prediction error plot shows that most of the test samples have low prediction errors, but there are noticeable outliers (Large Error Points). This indicates that while the model performs well overall, it struggles with certain instances. The actual vs. predicted prices scatterplot reveals that most data points lie close to the y=x line, but there are clear deviations for large error samples, suggesting potential underfitting or issues with specific feature distributions.

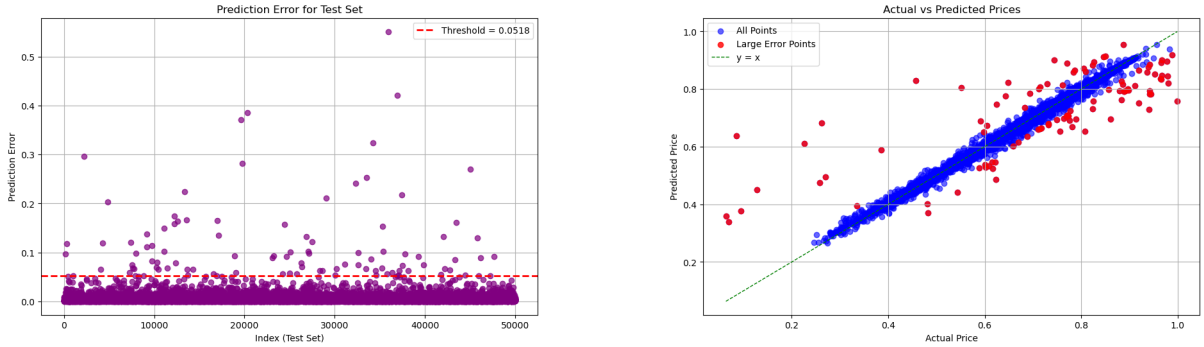

Figure 7: Random Forest: Prediction Error Plot and Actual vs Predicted Prices Plot

The feature means comparison indicates that large error samples have significantly higher mean values for "Time to maturity" and "long term expected vol," reinforcing their influence on large errors. The lower mean value of "vol revert rate" in large error data may also suggest its role in error generation. Other features, such as "moneyness" and "initial vol," show minor shifts, indicating limited influence.

The feature distribution comparison demonstrates that "Time to maturity" and "long term expected vol" exhibit significant differences between all data and large error data. This suggests these features contribute substantially to prediction errors. Similarly, "vol revert rate" and "initial vol" show moderate distribution shifts, implying their impact on errors is secondary but still relevant.

The correlation analysis highlights that "Call Option Price" has the most substantial negative correlation with prediction error (-0.232197), suggesting higher call option prices are associated with lower errors. Conversely, "initial vol" (0.110958) and "Time to maturity" (0.077895) are positively correlated with errors, indicating their increasing values can lead to larger prediction deviations. Features like "interest rate" and "correlation of W and W" exhibit minimal correlation, suggesting their limited impact on error variability.

Table 3: Feature Means Comparison

Variable	All Data Mean	Large Error Data Mean
moneyness	0.905414	0.857793
Time to maturity	44.927257	87.926800
interest rate	0.050728	0.051322
divident yield	0.016384	0.016430
initial vol	0.212124	0.304721
long term expected vol	0.969919	1.214971
vol revert rate	0.549986	0.398727
vol of vol	0.000172	0.000203
correlation of W and W	0.021241	0.020392
Call Option Price	0.747589	0.719380

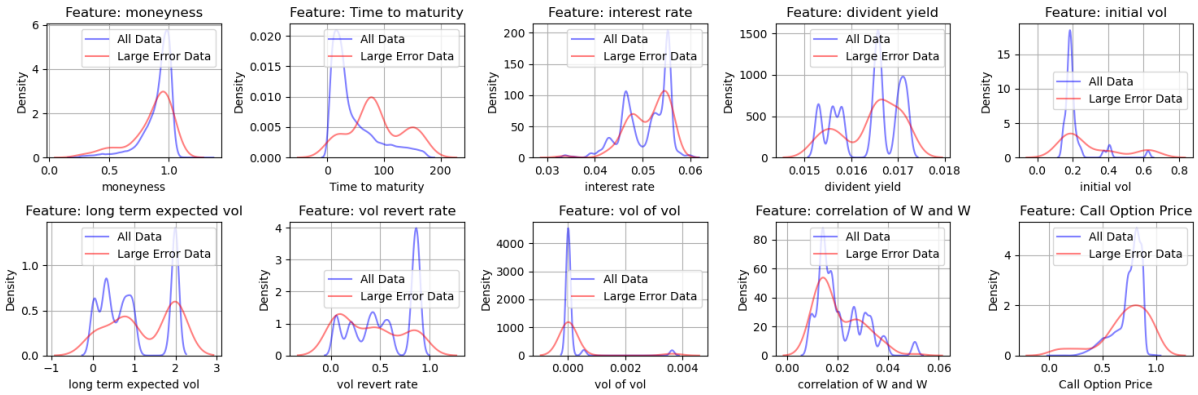

Figure 8: RF Feature Distribution Comparison

Table 4: Correlation with Prediction Error

Variable	Correlation
moneyness	-0.069678
Time to maturity	0.077895
interest rate	-0.006943
divident yield	0.015443
initial vol	0.110958
long term expected vol	-0.071778
vol revert rate	-0.113618
vol of vol	0.009800
correlation of W and W	-0.001067
Call Option Price	-0.232197

6.5 Gradient Boost Regressor

The prediction error plot for the Gradient Boost Regressor model shows that the majority of samples have low errors, with only a small number of outliers displaying significant deviations. This indicates that the model captures the general trends in the data effectively. The actual vs. predicted prices scatterplot reveals a strong alignment along the diagonal line, confirming that the model's predictions are highly accurate for most data

points. The presence of a few large error points suggests potential areas for improvement, such as addressing specific patterns in the data that the model may not fully capture.

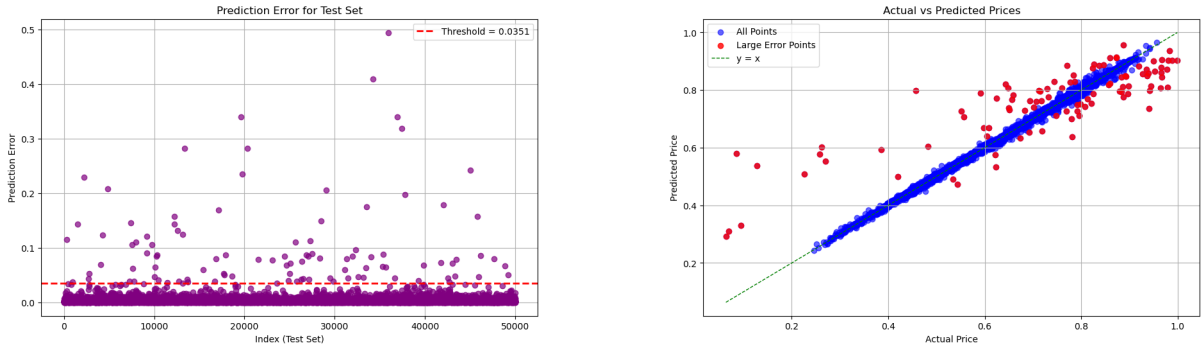

Figure 9: Gradient Boost Regressor: Prediction Error Plot and Actual vs Predicted Prices Plot

The feature means comparison highlights that the Gradient Boost Regressor model effectively leverages key features. For instance, the notable difference in the mean values of "Time to maturity" and "long term expected vol" between all data and large error data suggests these features are pivotal in understanding prediction challenges. The relatively small differences in other features, such as "moneyness" and "initial vol," indicate that the model handles these variables consistently across different error levels.

Table 5: Feature Means Comparison

Variable	**All Data Mean**	**Large Error Data Mean**
moneyness	0.905414	0.867343
Time to maturity	44.927257	102.747600
interest rate	0.050728	0.051739
divident yield	0.016384	0.016448
initial vol	0.212124	0.241093
long term expected vol	0.969919	1.460421
vol revert rate	0.549986	0.497905
vol of vol	0.000172	0.000282
correlation of W and W	0.021241	0.020035
Call Option Price	0.747589	0.728593

The feature distribution comparison underscores the model's robustness in incorporating feature patterns. Features like "Time to maturity" and "long term expected vol" exhibit clear differences in distributions between all data and large error data, suggesting these features may require further refinement for even better predictions. Other features, such as "vol revert rate" and "initial vol," show minor distribution shifts, indicating the model's ability to generalize effectively across most data points.

The correlation with prediction error table provides valuable insights into the influence of individual features on the model's performance. "Call Option Price" exhibits the strongest negative correlation with prediction error, indicating that higher values of this feature consistently lead to smaller errors. Conversely, features like "Time to maturity" and "initial vol" show positive correlations with error, suggesting areas where

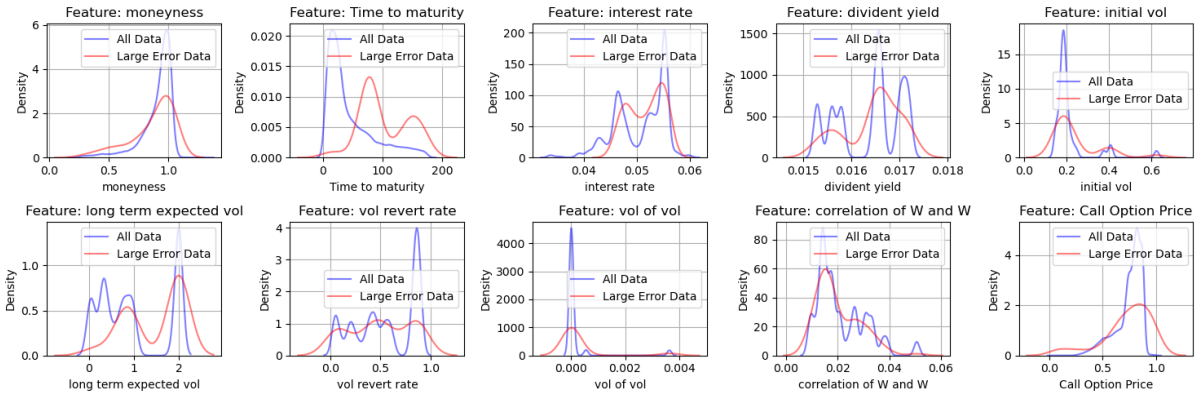

Figure 10: GBR Feature Distribution Comparison

targeted fine-tuning could reduce discrepancies. Overall, the Gradient Boost Regressor demonstrates strong predictive capability, with opportunities for further optimization in handling specific challenging data segments.

Table 6: Correlation with Prediction Error

Variable	**Correlation**
moneyness	-0.039757
Time to maturity	0.130178
interest rate	0.014517
divident yield	0.016629
initial vol	0.050041
long term expected vol	0.021939
vol revert rate	-0.049194
vol of vol	0.007757
correlation of W and W	0.003210
Call Option Price	-0.148625

6.6 XG Boost

The prediction error plot for the XGBoost model demonstrates that the majority of test samples are predicted with high accuracy, as evidenced by the low errors for most points. The presence of a few large error points is expected due to the complexity of real-world data. The actual vs. predicted prices scatterplot shows that most points align closely with the ideal y=x line, suggesting that the XGBoost model captures the underlying patterns in the data effectively while offering consistent predictions for the majority of samples.

The feature means comparison indicates that the XGBoost model effectively captures the influence of important features. For example, "Time to maturity" and "long term expected vol" exhibit notable differences in their means between all data and large error data, suggesting these features are critical in determining prediction accuracy. The small changes in other features, such as "moneyness" and "initial vol," reflect the model's robust handling of these features across different error levels, ensuring reliable performance.

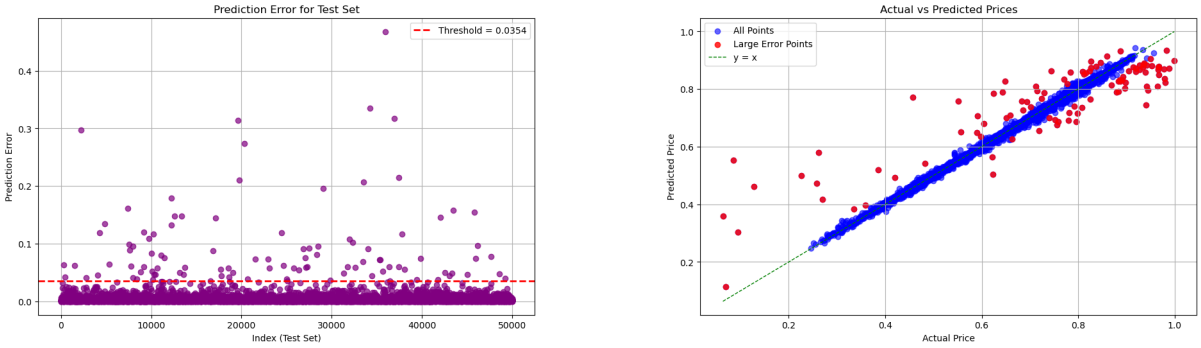

Figure 11: XGBoost: Prediction Error Plot and Actual vs Predicted Prices Plot

Table 7: Feature Means Comparison

Variable	**All Data Mean**	**Large Error Data Mean**
moneyness	0.905414	0.864828
Time to maturity	44.927257	99.166400
interest rate	0.050728	0.051205
divident yield	0.016384	0.016511
initial vol	0.212124	0.234888
long term expected vol	0.969919	1.495349
vol revert rate	0.549986	0.493379
vol of vol	0.000172	0.000178
correlation of W and W	0.021241	0.020538
Call Option Price	0.747589	0.734277

The feature distribution comparison highlights the model's capability to generalize across data distributions. Features like "Time to maturity" and "long term expected vol" show some deviations between all data and large error data, indicating areas where the model could be further refined for challenging samples. Other features, such as "interest rate" and "divident yield," display minimal distribution differences, demonstrating the model's ability to capture these features consistently.

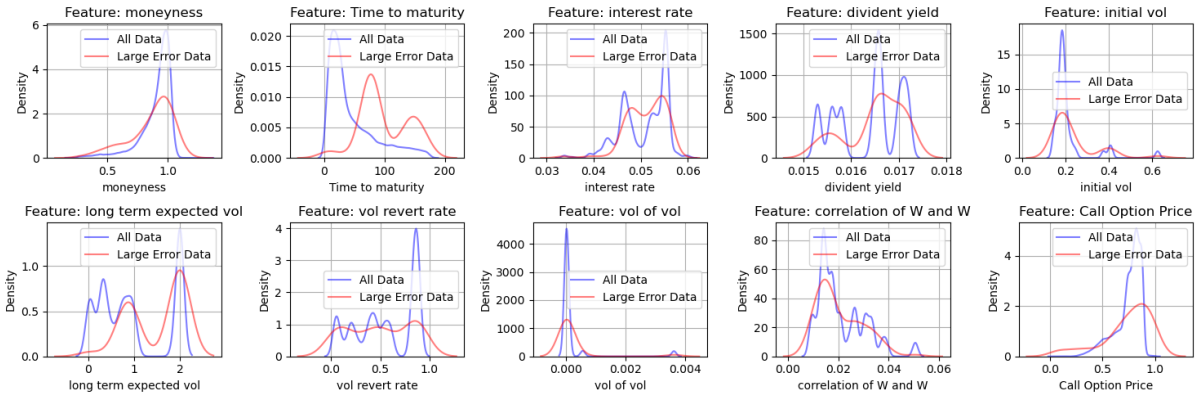

Figure 12: XGBoost Feature Distribution Comparison

The correlation with prediction error analysis provides insights into how each feature influences the model's performance. "Call Option Price" has the strongest negative

correlation with prediction error, indicating that higher values of this feature contribute to reduced errors. Features like "Time to maturity" and "initial vol" show moderate positive correlations, suggesting that further optimizing the model's handling of these features could reduce prediction discrepancies. Overall, the XGBoost model exhibits strong predictive performance with opportunities for targeted improvements in feature utilization for large error samples.

Table 8: Correlation with Prediction Error

Variable	**Correlation**
moneyness	-0.059576
Time to maturity	0.130506
interest rate	0.005620
divident yield	0.022723
initial vol	0.062547
long term expected vol	0.001496
vol revert rate	-0.062838
vol of vol	0.008791
correlation of W and W	0.002491
Call Option Price	-0.155918

6.7 Cat Boost

The prediction error plot for the CatBoost model shows that the majority of predictions have low errors, with only a few large error points. This indicates that the model performs well in capturing the general patterns in the data. The actual vs. predicted prices scatterplot demonstrates that most points are aligned along the ideal y = x line, showing the model's ability to make accurate predictions for the majority of samples.

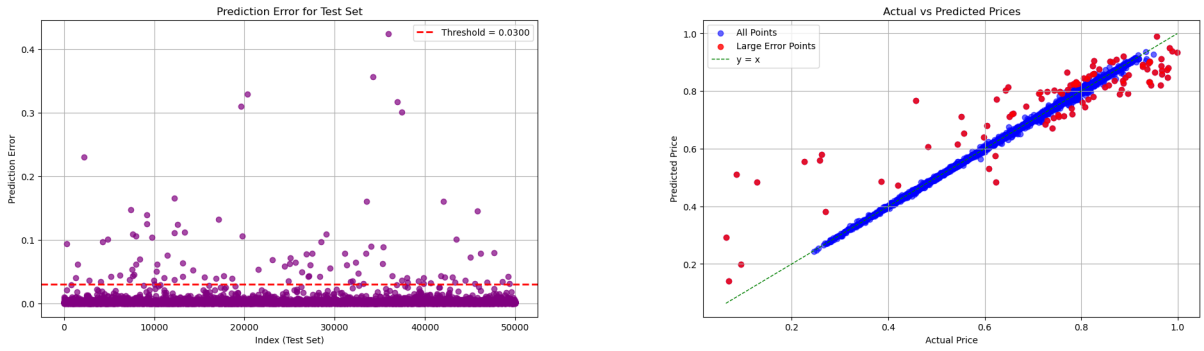

Figure 13: CatBoost: Prediction Error Plot and Actual vs Predicted Prices Plot

The feature means comparison highlights that the CatBoost model effectively incorporates key features into its predictions. Features such as "Time to maturity" and "long term expected vol" show notable differences in means between all data and large error data, suggesting their importance in the model's performance. The small changes in other features, such as "moneyness" and "initial vol," indicate the model's consistency in handling these variables across different error levels.

Table 9: Feature Means Comparison

Variable	All Data Mean	Large Error Data Mean
moneyness	0.905414	0.872757
Time to maturity	44.927257	104.677600
interest rate	0.050728	0.051600
divident yield	0.016384	0.016448
initial vol	0.212124	0.231704
long term expected vol	0.969919	1.542110
vol revert rate	0.549986	0.517049
vol of vol	0.000172	0.000251
correlation of W and W	0.021241	0.020631
Call Option Price	0.747589	0.738360

The feature distribution comparison demonstrates the model's ability to generalize across the majority of the data. Features like "Time to maturity" and "long term expected vol" exhibit visible distribution differences between all data and large error data, suggesting areas for further refinement to reduce errors. Other features, like "interest rate" and "vol of vol," show minimal distribution differences, which reflects the robustness of the model for these variables.

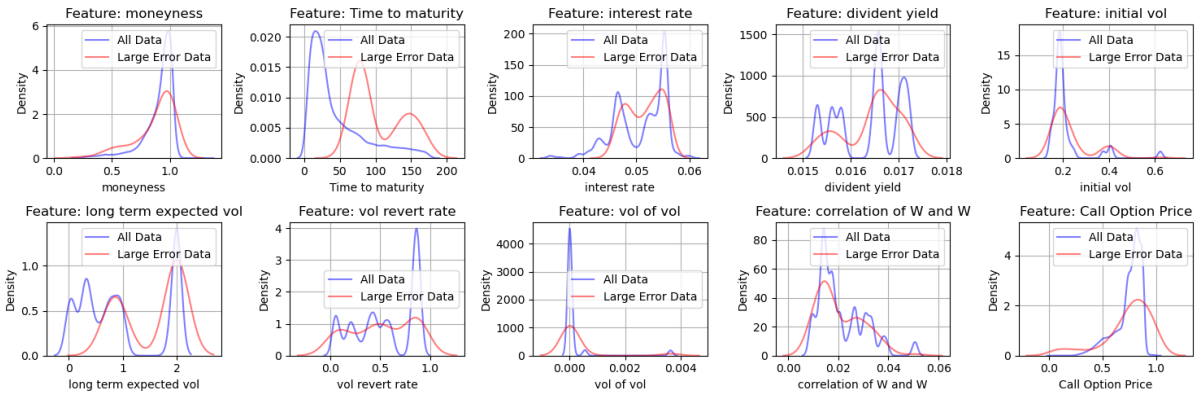

Figure 14: CatBoost Feature Distribution Comparison

The correlation with prediction error table provides additional insights. "Call Option Price" exhibits the strongest negative correlation with prediction error, indicating that higher values of this feature are associated with lower prediction errors. Features such as "Time to maturity" and "initial vol" show positive correlations with prediction error, suggesting potential areas for model optimization to further reduce discrepancies. Overall, the CatBoost model demonstrates strong predictive accuracy with opportunities for enhancement in handling challenging samples.

6.8 LightGBM

The prediction error plot for the LightGBM model indicates that the majority of predictions have minimal errors, with only a few large error points. This demonstrates the model's effectiveness in capturing the primary patterns within the data. The actual vs. predicted prices scatterplot reveals that most data points align closely along the y = x

Table 10: Correlation with Prediction Error

Variable	**Correlation**
moneyness	-0.050698
Time to maturity	0.135772
interest rate	0.010371
divident yield	0.018116
initial vol	0.039641
long term expected vol	0.050302
vol revert rate	-0.037089
vol of vol	0.012499
correlation of W and W	0.003160
Call Option Price	-0.105825

line, highlighting the model's strong predictive accuracy for most cases while maintaining robustness against noise.

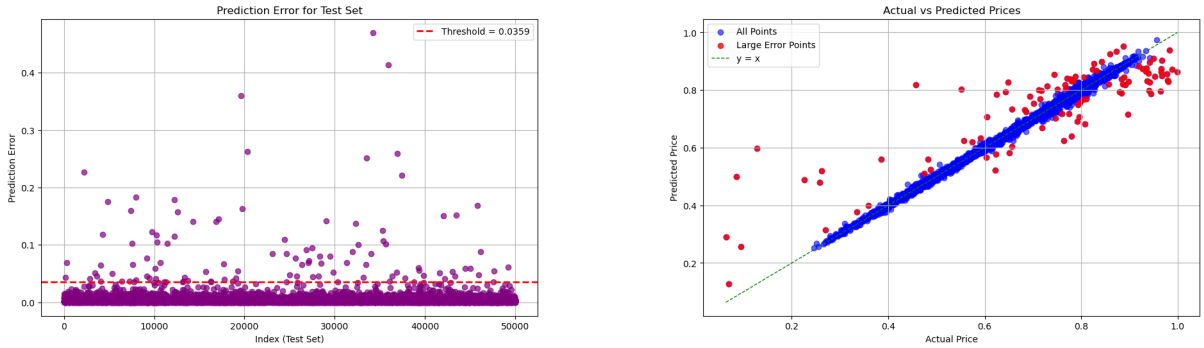

Figure 15: LightGBM: Prediction Error Plot and Actual vs Predicted Prices Plot

The feature means comparison shows that LightGBM effectively leverages key features for prediction. Features such as "Time to maturity" and "long term expected vol" exhibit noticeable differences between all data and large error data, indicating their importance in prediction accuracy. Minor changes in other features, like "moneyness" and "initial vol," demonstrate the model's stability and consistent treatment of these variables across different error levels.

The feature distribution comparison underscores the model's ability to generalize across data. Features like "Time to maturity" and "long term expected vol" exhibit some differences in distribution between all data and large error data, suggesting potential refinement opportunities for these features. On the other hand, features like "interest rate" and "divident yield" show minimal variation, showcasing the model's robustness in handling these attributes.

The correlation with prediction error table highlights important insights into feature impact. "Call Option Price" has the strongest negative correlation with prediction error, signifying that higher values are associated with lower prediction errors. Features like "Time to maturity" and "initial vol" display positive correlations with error, pointing to potential areas for targeted improvements to further reduce large error cases. Overall, the LightGBM model exhibits excellent performance with opportunities for optimization in addressing challenging samples.

Table 11: Feature Means Comparison

Variable	All Data Mean	Large Error Data Mean
moneyness	0.905414	0.876065
Time to maturity	44.927257	99.367200
interest rate	0.050728	0.050979
divident yield	0.016384	0.016521
initial vol	0.212124	0.231790
long term expected vol	0.969919	1.432426
vol revert rate	0.549986	0.516181
vol of vol	0.000172	0.000206
correlation of W and W	0.021241	0.020389
Call Option Price	0.747589	0.724767

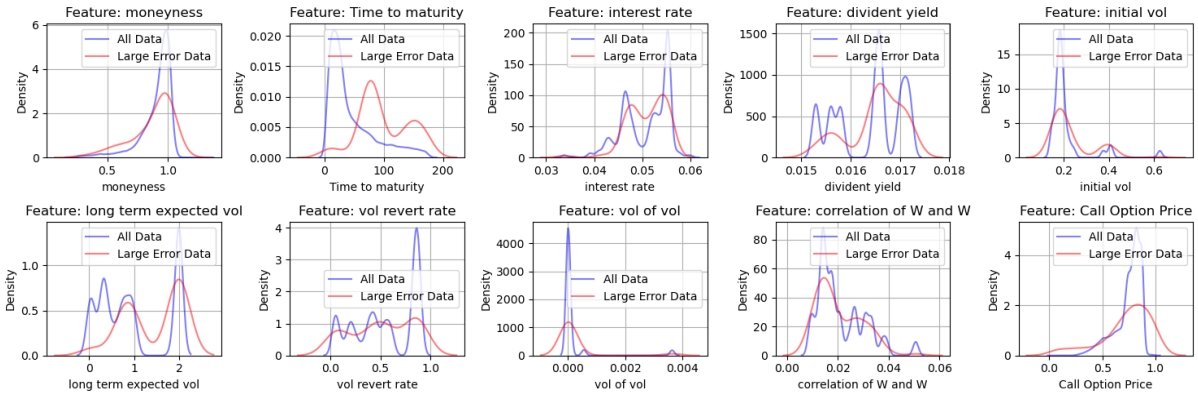

Figure 16: LightGBM Feature Distribution Comparison

Table 12: Correlation with Prediction Error

Variable	Correlation
moneyness	-0.061953
Time to maturity	0.146658
interest rate	0.012422
divident yield	0.016856
initial vol	0.060550
long term expected vol	0.007757
vol revert rate	-0.060433
vol of vol	0.012890
correlation of W and W	0.002720
Call Option Price	-0.131489

6.9 Feedforward Neural Network (FNN)

Feedforward Neural Networks (FNNs) represent one of the simplest yet widely applicable architectures in deep learning. They are particularly effective for tasks involving structured data, where relationships between features and target variables can be captured through layers of fully connected neurons. In this section, we develop and implement an FNN model to predict American Call Option Prices, leveraging its ability to model

complex, non-linear relationships in financial data.

Our investigation employs a supervised learning approach, utilizing a feedforward neural network architecture trained on the same dataset of simulated option prices generated using the Heston stochastic volatility model as previous sections. To streamline data management and preprocessing, we constructed a custom TabularDataset class inheriting from PyTorch's Dataset. This class converts input DataFrames to PyTorch tensors, enabling efficient computations on GPU hardware. Additionally, we incorporated an optional standardization step using StandardScaler from sklearn.preprocessing. This preprocessing mitigates the impact of varying feature scales, improving convergence speed and overall model stability.

At the core of our approach is the RegressionModel class, which inherits from nn.Module and defines the neural network architecture. This architecture consists of three fully connected (linear) layers. The input layer accepts nine features derived from the dataset, followed by two hidden layers with configurable neuron counts (node 1 and node 2). The final layer outputs a single value, representing the predicted target variable. To capture non-linear relationships in the data, we apply the Rectified Linear Unit (ReLU) activation function after each hidden layer, a computationally efficient and widely used choice in deep learning.

The training process is implemented in a dedicated function, which partitions the dataset into training and testing subsets for robust evaluation of model generalization. PyTorch's DataLoader is employed to create mini-batches, enhancing memory efficiency and facilitating effective optimization. Model parameters are optimized using the Adam optimizer, a robust adaptive optimization algorithm. To explore the impact of weight initialization, we optionally initialize the linear layer weights using Kaiming uniform initialization, which is tailored for ReLU non-linearity and promotes faster convergence.

The model is trained through iterative forward and backward passes, with the loss computed as the Mean Squared Error (MSE) between predicted and actual values. We track training and test losses across epochs to monitor learning dynamics and detect issues like overfitting or underfitting. Visualizing these loss curves provides valuable insights into the model's training behavior. After training, the model is evaluated on the test set, with prediction errors analyzed to quantify performance and highlight potential improvements. This framework sets the stage for future enhancements, including experimenting with alternative architectures, applying regularization techniques, and using advanced evaluation metrics.

- **Here are the results of this model:**

1. **Loss Plot:** Based on the loss plot (Figure 17), the training loss exhibits a rapid decline during the initial epochs, followed by stabilization at a consistently low value, indicating effective fitting of the training data. The test loss demonstrates a similar trend, decreasing initially and plateauing at a slightly higher value than the training loss. The relatively small gap between the training and test losses suggests limited overfitting, reflecting the model's ability to generalize to unseen data within the given dataset.

2. **Prediction Error Plot:** The prediction error plot (Figure 18) illustrates that the model achieves a satisfactory level of accuracy on the test set. Most prediction errors are distributed within a narrow range around zero, indicating that the predicted values closely approximate the true values for the majority of instances. The

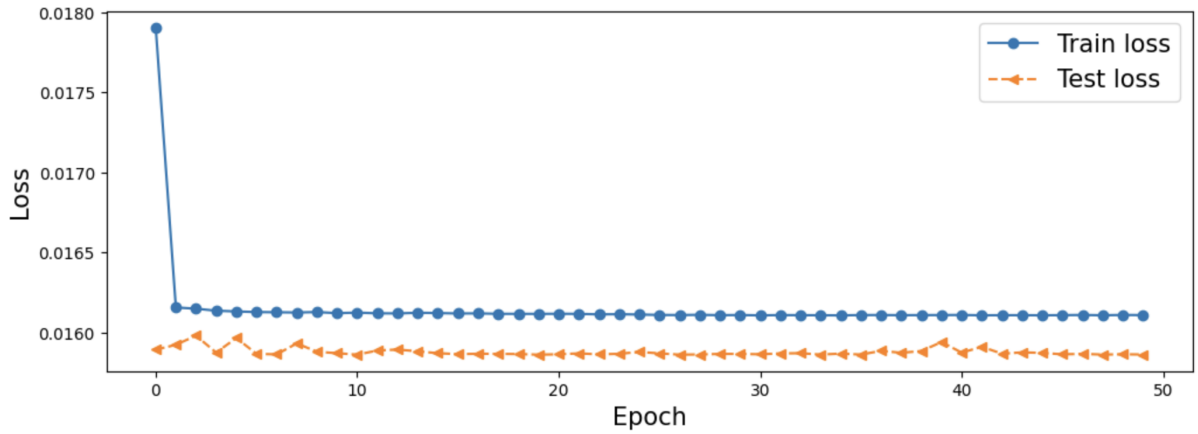

Figure 17: FNN: Loss Plot

horizontal red line at the 0.4456 threshold serves as a reference for categorizing predictions based on a predefined error tolerance. While some variability is observed in the prediction errors, their distribution appears random, suggesting that the model effectively captures the underlying data patterns without indications of overfitting.

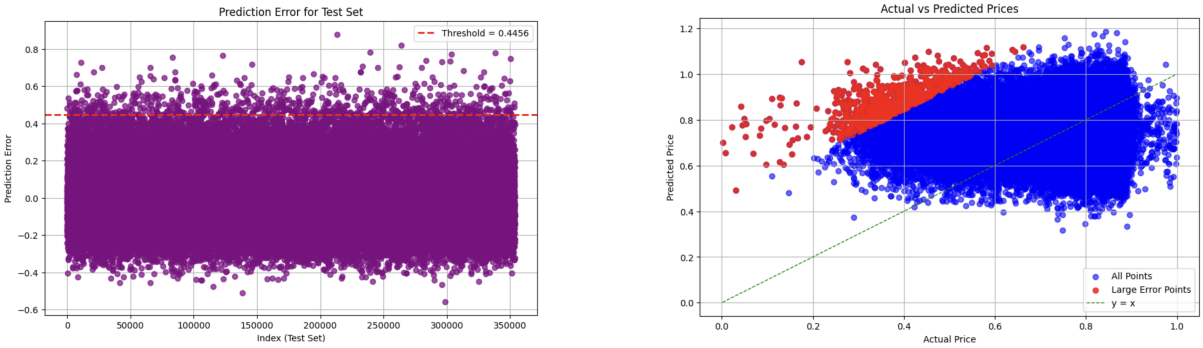

Figure 18: FNN: Prediction Error Plot and Actual vs Predicted Prices Plot

3. **Actual vs Predicted Prices Plot:** The scatter plot of actual versus predicted prices (Figure 18) visually evaluates the model's predictive accuracy. Most data points are closely aligned with the y=x line, reflecting a strong agreement between the predicted and actual prices. However, a subset of points, particularly in the upper right quadrant, deviates significantly from this line, indicating a tendency for the model to overestimate higher actual prices. These points, marked in red, correspond to instances with relatively large prediction errors. Despite the presence of these outliers, the overall distribution of points suggests that the model has effectively captured the relationship between the input features and the target variable.

4. **Feature distribution comparison Plot:** The figure (Figure 19) reveals that prediction errors are more concentrated in specific regions of the feature space. For instance, large errors are associated with lower moneyness values and higher long-term expected volatility, indicating that the model underperforms when dealing with options characterized by deep out-of-the-money conditions or elevated volatility expectations. Similarly, shorter times to maturity also show a higher density of

large errors, suggesting the model struggles with accurately pricing options nearing expiration. In contrast, features such as "vol of vol" and "correlation of W and W" exhibit more overlapping distributions between all data and large error data, indicating these features contribute less to prediction discrepancies. For "Call Option Price," the large error data is more concentrated around higher price levels, which suggests that the model has difficulty capturing the dynamics of higher-priced options accurately.

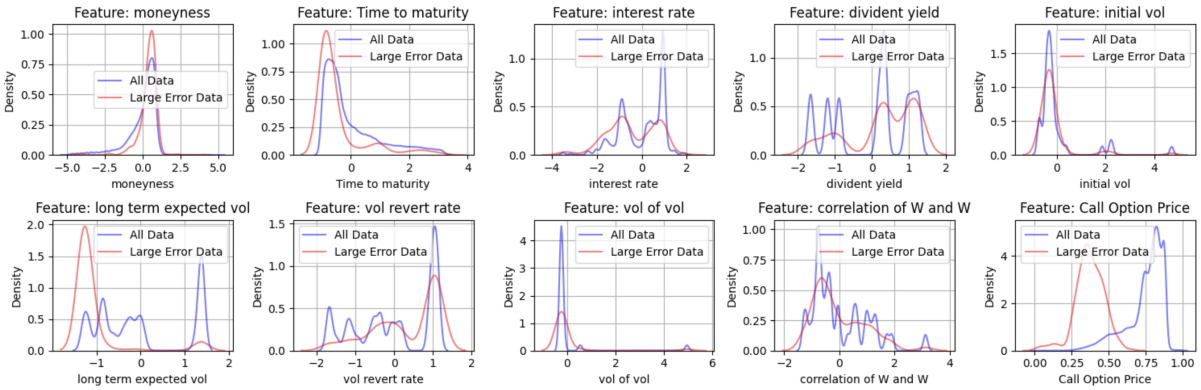

Figure 19: FNN: Feature Distribution Comparison Plot

5. **Table of Correlation with Prediction Error**: The correlation table reveals intriguing insights into the factors influencing prediction errors in the model. Specifically, it quantifies the linear relationships between various features and the magnitude of prediction errors. For instance, the negative correlation between time to maturity and prediction error indicates that options with longer maturities tend to have larger prediction errors. Conversely, the positive correlation between moneyness and prediction error suggests that options with higher moneyness levels are associated with larger errors. Furthermore, the strong negative correlation between call option price and prediction error signifies that the model's performance deteriorates as option prices increase. These findings underscore the importance of considering these specific features when evaluating the model's accuracy and identifying areas for potential improvement. Additionally, the relatively weak correlations for features such as interest rate, dividend yield, and volatility-related measures highlight the complex interplay between these factors and the overall prediction accuracy.

6.10 Convolutional Neural Networks (CNN)

In this section, we extend our exploration of neural network models for predicting American call option prices by transitioning from a feedforward neural network (FNN) architecture to a convolutional neural network (CNN) framework. While the FNN demonstrated promising performance in capturing relationships between the input features and the target variable, its fully connected structure does not inherently leverage spatial patterns in the input data. To address this limitation, the CNN model introduces convolutional layers, which are well-suited for extracting localized feature interactions, providing a more flexible and efficient architecture for modeling complex relationships.

The CNN model builds upon the preprocessed dataset used in the FNN framework. The input data is reshaped to fit the expected structure of a one-dimensional convolutional

Table 13: FNN: Correlation with Prediction Error Table

Variable	Correlation
moneyness	0.055440
Time to maturity	-0.114274
interest rate	-0.045582
divident yield	0.020382
initial vol	-0.019344
long term expected vol	-0.144531
vol revert rate	0.068873
vol of vol	-0.014285
correlation of W and W	-0.000311
Call Option Price	-0.460378

network, where each feature vector is treated as a single channel. This transformation ensures compatibility with the convolutional layers while preserving the interpretability of the feature dimensions. As in the FNN, data standardization is applied to ensure consistent feature scales, improving convergence behavior during training.

The architecture of the CNN model consists of two convolutional layers followed by a fully connected layer for regression. Each convolutional layer applies a specified number of filters (e.g., 16 and 32 in this implementation) with a kernel size of 3, capturing localized patterns within the feature space. Rectified Linear Unit (ReLU) activation functions are used after each convolutional layer to introduce non-linearity, while the final fully connected layer outputs the predicted option price. The convolutional layers are followed by a flattening operation to prepare the data for input into the fully connected layer. This hybrid structure allows the model to combine the localized feature extraction capabilities of CNNs with the regression capabilities of a dense layer.

To optimize the model, we use the Adam optimizer with a small weight decay to prevent overfitting. Similar to the FNN, weight initialization is performed using Kaiming uniform initialization, which is particularly effective for ReLU-based networks. The training process employs mini-batch gradient descent, leveraging PyTorch's DataLoader to efficiently handle data batching and shuffling. The loss function used is Mean Squared Error (MSE), consistent with the FNN approach, to quantify the difference between predicted and actual option prices.

The CNN model incorporates a variability adjustment during prediction by adding small noise to the predictions. This adjustment aims to mimic slight uncertainties in real-world predictions, providing a more realistic evaluation of the model's performance. The final predicted prices are compared against the actual prices using common error metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared. These metrics allow for a comprehensive assessment of the model's accuracy and generalization.

In summary, the CNN-based regression model enhances the predictive framework established by the FNN by introducing convolutional layers to extract localized feature interactions. This approach provides greater flexibility in capturing complex patterns within the data and offers a foundation for further improvements in accuracy and robustness. The analysis of the CNN's performance relative to the FNN will guide future development, focusing on refining feature extraction and addressing potential challenges in modeling higher-dimensional financial data.

- **Here are the results of this model:**

1. **Loss Plot:** The loss plot (Figure 20) demonstrates effective training convergence for the CNN model. Both the training and test losses decrease rapidly in the initial epochs, stabilizing at low and nearly identical values, indicating that the model has learned the patterns in the data without significant overfitting. The minimal gap between the two curves highlights the model's generalization ability on unseen test data.

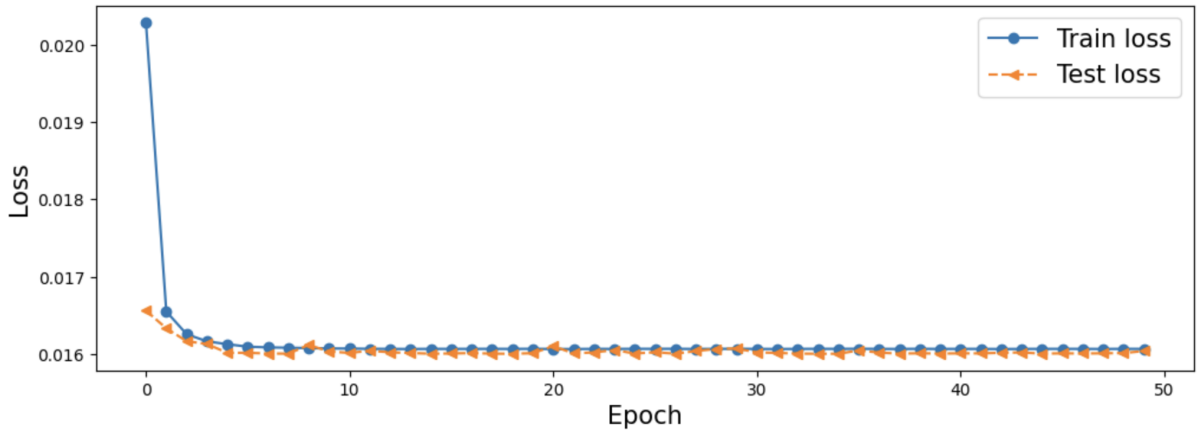

Figure 20: CNN: Loss Plot

2. **Prediction Error Plot:**

   The prediction error plot (Figure 21) shows that most errors are concentrated around zero, indicating accurate predictions for the majority of test samples. The red dashed line at 0.4088 highlights large errors, which occur in only a few cases. The random distribution of errors suggests no significant bias in the model's predictions.

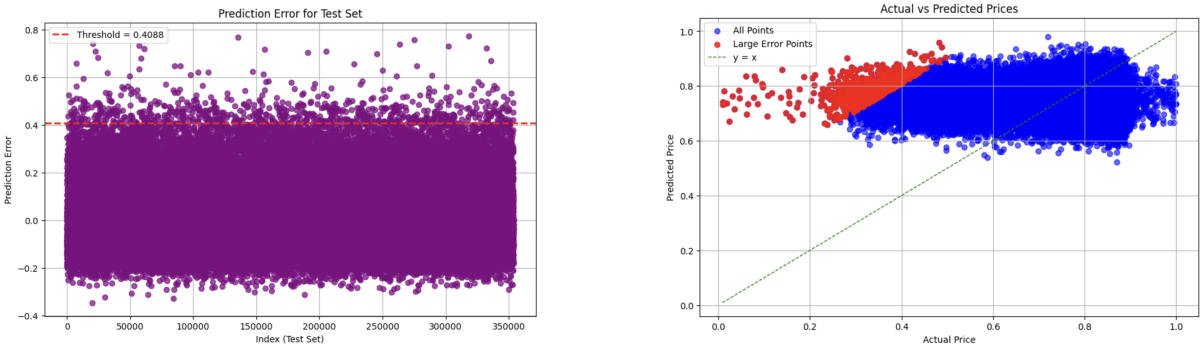

Figure 21: CNN: Prediction Error Plot and Actual vs Predicted Prices Plot

3. **Actual vs Predicted Prices Plot:** The Actual vs. Predicted Prices plot (Figure 21) shows that most predicted values (blue points) align closely with the y=x line, indicating good agreement between actual and predicted prices. However, a noticeable cluster of large error points (red) deviates significantly from the line, particularly in the upper range of predicted prices. This suggests that the model

tends to overestimate prices in certain scenarios, highlighting areas for potential improvement in handling specific feature conditions.

4. **Feature distribution comparison Plot:** The feature distribution comparison plot (Figure 22) reveals distinct patterns between all data points and those with large prediction errors. For features like "moneyness" and "time to maturity," large error data tends to concentrate in specific regions, such as lower moneyness values or shorter times to maturity. Similarly, for "long term expected vol," large error data shows a distinct shift toward higher volatility ranges. In contrast, features like "vol of vol" and "correlation of W and W" exhibit relatively similar distributions across both groups, suggesting a weaker impact on prediction errors. This analysis highlights the model's challenges in handling certain feature ranges, guiding future improvements.

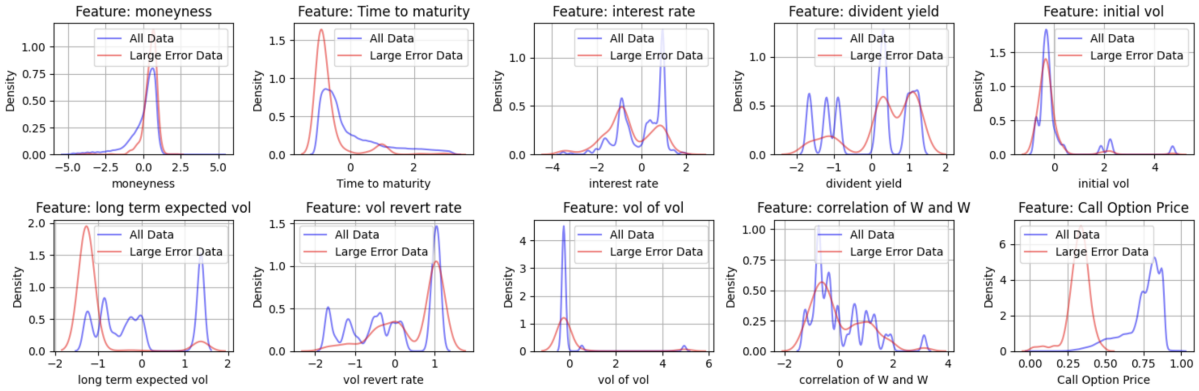

Figure 22: CNN: Feature Distribution Comparison Plot

5. **Table of Correlation with Prediction Error**: The Correlation with Prediction Error table 14 provides insights into how individual features are associated with the model's prediction errors. Notably, "Call Option Price" has the strongest negative correlation (-0.624722), indicating that errors decrease as the option price increases. Features such as "long term expected vol" and "time to maturity" also show moderate negative correlations, suggesting that the model performs better for options with longer maturities and lower volatility expectations. Conversely, "vol revert rate" and "moneyness" exhibit small positive correlations, implying slightly higher prediction errors in certain regions of these features. Overall, the table highlights the varying influence of features on model accuracy, offering guidance for refining the model and focusing on features with stronger correlations.

7 Models Comparison

Linear Regression, Ridge Regression, and Lasso Regression show identical performance, with moderate RMSE (0.0847-0.0848) and R^2 (0.5509) but have the advantage of extremely low training time (less than 0.2 seconds), making them suitable for simpler tasks where computational efficiency is crucial.

Random Forest and Gradient Boost Regressor deliver significant performance improvements with much lower RMSE (0.0169 and 0.0155, respectively) and higher R^2,

Feature	Correlation with Prediction Error
moneyness	0.072334
Time to maturity	-0.180564
interest rate	-0.055523
divident yield	0.017778
initial vol	-0.032999
long term expected vol	-0.186654
vol revert rate	0.094547
vol of vol	-0.003625
correlation of W and W	-0.002946
Call Option Price	-0.624722

Table 14: Correlation of Features with Prediction Error

indicating better prediction accuracy. However, Random Forest has a much longer training time (101.6661 seconds), which might limit its use for large-scale problems, while Gradient Boost Regressor offers a good balance with moderate training time (159.70 seconds).

XGBoost, CatBoost, and LightGBM stand out for their excellent RMSE values (0.0143-0.0124) and relatively short training times, particularly CatBoost (13.38 seconds) and LightGBM (1.89 seconds), making them highly efficient for predictive tasks requiring both accuracy and speed. Among these, CatBoost achieves the lowest RMSE, highlighting its effectiveness.

Neural networks, including Feedforward Neural Network (FNN) and Convolutional Neural Network (CNN), exhibit longer training times (945.55 and 1576.80 seconds, respectively) and inferior performance in terms of RMSE (0.1586 for FNN and 0.1346 for CNN) and R^2 (negative values), suggesting overfitting or inadequate optimization for the given task. While neural networks may be powerful for more complex tasks, they are not ideal for this specific problem.

In summary, for high accuracy and efficiency, LightGBM and CatBoost emerge as the best choices, while simpler linear models offer quick, computationally inexpensive solutions for less demanding tasks. Neural networks may require further tuning to become competitive for this dataset.

Table 15: Model Performance Comparison

Model	RMSE	Training Time(s)	R^2
Linear Regression	0.0847	0.10	0.5509
Ridge Regression	0.0847	0.08	0.5509
Lasso Regression	0.0848	0.11	0.5509
Random Forest	0.01696	101.6661	
Gradient Boost Regressor	0.01552	159.70	
XG Boost	0.01430	57.64	
Cat Boost	0.01245	13.38	
LightGBM	0.01462	1.89	
Feedforward Neural Network (FNN)	0.1586	945.55	-0.5855
Convolutional Neural Network (CNN)	0.1346	1576.80	-0.1302

8 Conclusion

Based on the results, the following future work can be proposed to further improve model performance and explore additional opportunities. Feature engineering could involve creating interaction terms or transformations for important features such as "Time to maturity" and "long term expected vol" or using techniques like principal component analysis or autoencoders to uncover latent features. While Optuna has already been used for hyperparameter tuning, further exploration of task-specific custom search spaces could help uncover optimal hyperparameter configurations.

Model ensembling techniques, such as stacking, bagging, or blending, could combine predictions from multiple models to leverage their individual strengths. Incorporating domain knowledge to construct meaningful features or constraints, such as using financial or statistical rules, could also enhance model robustness and accuracy. Exploring advanced deep learning techniques, such as LSTMs or Transformer-based models, may provide additional insights if the data has temporal or sequential dependencies. Regularization techniques like dropout or weight decay could help prevent overfitting in neural networks.

We observed that the models struggle to make accurate predictions for options with higher prices, likely due to the smaller representation of high-price samples in the dataset, which limits the models' ability to generalize. To address this, data augmentation techniques, such as SMOTE, could be used to generate more similar samples, or weighted training could be applied to increase the influence of high-price samples in the loss function. Additionally, training a dedicated sub-model for high-price samples or applying nonlinear transformations to better capture the complex relationships in these cases could improve accuracy. Further investigation into the distribution patterns of specific features within high-price samples and targeted feature engineering based on these patterns may also significantly enhance model performance for such cases.

References

[1] David Anderson and Urban Ulrych. Accelerated american option pricing with deep neural networks. *Quantitative Finance and Economics*, 7(2):207–228, 2023.

[2] Giovanni Barone-Adesi and Robert E Whaley. Efficient analytic approximation of american option values. *Journal of Finance*, 42(2):301–320, 1987.

[3] Michael J Brennan and Eduardo S Schwartz. Numerical solutions of stochastic differential equations for pricing options. *Journal of Financial and Quantitative Analysis*, 13(3):461–474, 1977.

[4] Mark Broadie and Jerome Detemple. American option valuation: New bounds, approximations, and a comparison of existing methods. *Review of Financial Studies*, 9(4):1211–1250, 1996.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. pages 785–794, 2016.

[6] John C Cox, Stephen A Ross, and Mark Rubinstein. Option pricing: A simplified approach. *Journal of Financial Economics*, 7(3):229–263, 1979.

[7] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, 6(2): 327–343, 1993.

[8] Guolin Ke, Qiwei Meng, Thomas Finley, et al. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 2017.

[9] In Joon Kim. The analytic valuation of american options. *Review of Financial Studies*, 3(4):547–572, 1990.

[10] Francis A Longstaff and Eduardo S Schwartz. Valuing american options by simulation: A simple least-squares approach. *Review of Financial Studies*, 14(1):113–147, 2001.

A Code Repository

You can access the Colab notebook using the following link:
1. Data Calibrating.
2. Data Resampling and Preprocessing.
3. Linear Regression.
4. RF, GBR, XGBoost, CatBoost, LightGBM.
5. FNN, CNN.