DATA-DRIVEN OPTION PRICING USING SINGLE AND MULTI-ASSET SUPERVISED LEARNING

A Report Submitted
for Research Project-1

by

Ratanpalke Kavya
(419MA5060)

under the guidance of

Dr. Ankur Kanaujiya

Department of Mathematics

National Institute of Technology Rourkela

November, 2023

1 Introduction

The world of finance has witnessed an unprecedented transformation propelled by the advancements in technology and data analytics. Among the myriad applications, option pricing stands as a cornerstone in financial modeling, enabling investors to make informed decisions in volatile markets. Traditional option pricing models, albeit fundamental, often face challenges in capturing the complex dynamics and intricate patterns prevalent in today's financial landscapes.

The emergence of data-driven methodologies, particularly supervised learning techniques, has reshaped the approach to option pricing. Leveraging vast datasets and harnessing the power of machine learning algorithms, these methodologies offer a promising avenue to refine and enhance existing pricing models. Moreover, the integration of single and multi-asset supervised learning approaches introduces a nuanced understanding of interconnected financial instruments, adding depth and accuracy to pricing predictions.

This report embarks on an exploration of the application of supervised learning techniques in the realm of option pricing. It delves into the fundamental concepts, strategies, and algorithms employed in leveraging data-driven approaches to ascertain fair values of financial derivatives. Emphasizing both single and multi-asset scenarios, this study navigates through the intricacies of supervised learning models, shedding light on their adaptability and efficacy in pricing options across diverse market conditions.

The primary aim is to elucidate the practical implications and advantages offered by these data-centric methodologies. By scrutinizing and comparing the outcomes derived from single and multi-asset supervised learning, this report endeavors to offer insights into their applicability, limitations, and potential implications for financial decision-making.

Through this comprehensive analysis, the report aims to contribute to the evolving discourse on option pricing strategies, highlighting the transformative potential of data-driven techniques in navigating the complexities of modern financial markets.

This study is organized as follows: after this introductory section, the subsequent segments will delineate the terminologies, objectives, algorithms, and methodologies employed in data-driven option pricing. The report will culminate in an analysis of the outcomes, concluding reflections, and acknowledgments.

2 Terminologies

Option: An option is a financial derivative contract that provides the holder with the right, but not the obligation, to buy or sell an underlying asset at a specified price within a predetermined time frame. Options are valuable instruments used in various financial strategies and risk management techniques.

Call Option: A call option grants the holder the right to buy the underlying asset at the strike price within a specified time, irrespective of the current market price. The payoff of a call option at maturity is given by:

$$\textbf{Payoff} = C_T = \max(0, S_T - K)$$

Where:

- S_T is the price of the underlying asset at maturity.
- K is the strike price.

Put Option: A put option grants the holder the right to sell the underlying asset at the strike price within a specified time, regardless of the current market price. The payoff of a put option at maturity is given by:

$$\textbf{Payoff} = C_T = \max(0, K - S_T)$$

Where:

- S_T is the price of the underlying asset at maturity.
- K is the strike price.

Underlying Asset: The underlying asset is the financial instrument (e.g., stock, commodity, index) on which the option's value is based. Its price movements affect the option's value.

Strike Price: The strike price, also known as the exercise price, is the predetermined price at which the holder of the option can buy or sell the underlying asset. It is established at the time the option contract is created.

Expiration Date: The expiration date is the date at which the option contract expires. After this date, the option is no longer valid, and the holder loses the right to exercise it.

Premium: The premium is the amount paid by the option buyer to the option seller to acquire the rights of the option. It represents the cost of holding the option and is determined by various factors such as the underlying asset's price, volatility, time to expiration, and the risk-free interest rate.

3 Objective

The primary aim of this study is to harness supervised learning techniques to predict the closing price of the Nifty 50 dataset collected from Yahoo NSE, spanning the years 2014 to 2018.

3.1 Dataset Description

The dataset comprises seven key features:

- **Date**: Represents the date of the recorded trading sessions.
- **Open**: Denotes the opening price of the Nifty 50 index at the beginning of the trading day.
- **High**: Indicates the highest price recorded during the trading session.

- **Low**: Represents the lowest price observed during the trading day.
- **Close**: Represents the closing price of the Nifty 50 index, signifying the final price of the trading day.
- **Adj Close**: Signifies the adjusted closing price, adjusted for corporate actions, dividends, and other factors.
- **Volume**: Represents the volume of Nifty 50 shares traded on a given day.

Additionally, four new columns have been derived from the existing dataset, contributing to a more comprehensive analysis:

- **Log Returns**: The logarithmic change in asset price over a specific period, calculated as:

  $$\text{Log Return} = \log\left(\frac{P_{t-1}}{P_t}\right)$$

  Where:

  - P_t is the price of the asset at time t.
  - P_{t-1} is the price of the asset at time $t-1$.
- **Moneyness**: Moneyness signifies the alignment of an option's strike price with the prevailing market value, crucial for option evaluation without explicit terminology.
- **Time to Maturity**: The duration remaining until the expiration of an option contract, calculated as the difference between the expiration date and the current date.
- **Risk-free Interest Rate**: The theoretical rate of return on an investment with zero risk of financial loss, a crucial factor in option pricing models.

The objective encompasses applying predictive models to the Nifty 50 dataset while considering these features. The study seeks to explore the predictive capacity of various machine learning algorithms and assess their effectiveness in forecasting the closing price of the Nifty 50 index.

4 Methodology

4.1 Supervised Learning Overview

Supervised learning involves training a model on a labeled dataset, where the algorithm learns patterns and relationships between input features and their corresponding target variables. This learning approach enables the model to make predictions or decisions when presented with new, unseen data. In the context of predicting the closing price of the Nifty 50 dataset, supervised learning techniques aim to uncover underlying patterns in the historical data to forecast future price movements.

4.2 XGBoost: Extreme Gradient Boosting

XGBoost is an ensemble learning algorithm that utilizes a decision tree-based approach. It employs a series of weak predictive models, generally decision trees, to form a robust predictive model. The primary components of XGBoost are:

1. **Decision Trees:** XGBoost uses decision trees, which are flowchart-like structures where data is split based on different criteria at each level to make predictions.

2. **Boosting:** XGBoost uses boosting, a technique where models are built sequentially. Each new model focuses on correcting errors made by the previous ones.

3. **Objective Function:** The algorithm aims to minimize a specific objective function while adding new trees. This function includes two parts: the loss function, measuring how well the model fits the data, and a regularization term that helps control the model's complexity to avoid overfitting. The objective function in XGBoost is formulated as:

$$\text{Obj} = \sum_{i=1}^{n} L(Y_i, \hat{Y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

   Where:

   - n represents the number of data points.
   - Y_i is the actual target value.
   - $\hat{Y}_i$ is the predicted value by the model.
   - K is the number of trees (or iterations).
   - f_k represents the k-th tree in the model.

4. **Loss Function:** The loss function measures the difference between predicted ($\hat{Y}_i$) and actual (Y_i) values. It's typically specific to the problem being solved (e.g., regression, classification). Common loss functions include:

   - Squared Error Loss (Regression):

$$L(Y_i, \hat{Y}_i) = (Y_i - \hat{Y}_i)^2$$

   - Logistic Loss (Binary Classification):

$$L(Y_i, \hat{Y}_i) = y_i \log(1 + e^{-\hat{Y}_i}) + (1 - y_i) \log(1 + e^{\hat{Y}_i})$$

5. **Regularization Term:** The regularization term ($\Omega(.)$) controls the complexity of the model to prevent overfitting. It's usually the sum of squared values of the tree's leaf weights. For example, considering a tree f with leaf weights w:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

   Where T is the number of leaves in the tree, γ and λ are regularization parameters.

6. **Gradient Descent Optimization:** Gradient descent is an optimization algorithm used in machine learning to minimize a function by iteratively moving in the direction of steepest descent (downward slope) of the function. In the context of XGBoost, let's focus on how gradient descent is used to optimize the objective function.

   (a) **Gradient Calculation:**

   - **Objective Function:** As mentioned earlier, the objective function in XGBoost consists of a loss function and a regularization term.
   - **Gradient:** To minimize this function, we need to calculate the gradient (first derivative) of the objective function with respect to the predicted values.
   - **Derivative Calculation:** For instance, if we're optimizing a squared error loss function for regression:

     $$L(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

     The gradient of this loss function with respect to the predicted value ($\hat{y}_i$) is:

     $$\frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i} = -2(y_i - \hat{y}_i)$$

   (b) **Update Step:**

   - **Learning Rate (η):** The learning rate controls the step size in each iteration of gradient descent. It's a small value that determines how much the model parameters (weights) should change in the direction of the gradient.
   - **Gradient Descent Update Rule:** The predicted values ($\hat{y}_i$) are updated based on the negative gradient and the learning rate:

     $$\hat{y}_i^{(t+1)} = \hat{y}_i^{(t)} - \eta \cdot \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}$$

     Here, $\hat{y}_i^{(t)}$ represents the predicted value in the current iteration, and $\hat{y}_i^{(t+1)}$ represents the predicted value in the next iteration.

   (c) **Optimization Process:**

   - **Iterative Process:** Gradient descent repeats these update steps for each data point (or batch of data) in the training set, adjusting the predicted values in the direction that minimizes the objective function.
   - **Convergence:** The process continues iteratively until a stopping criterion is met, such as a predefined number of iterations or until the change in the objective function becomes negligible.

   This iterative process of updating predictions based on the negative gradient gradually moves the model parameters (in this case, the predicted values) toward the optimal values that minimize the objective function, leading to better predictions in each iteration.

4.3 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) represent a class of computational models inspired by the human brain's neural structure. These models simulate the learning process and mimic the brain's ability to recognize patterns and relationships within data. ANNs consist of interconnected nodes arranged in layers: an input layer that receives data, one or more hidden layers that process information, and an output layer that generates the network's predictions.

4.3.1 Algorithm Overview and Mathematical Equations

1. **Algorithm Overview**

   - **Initialization:** Initialize the weights and biases of the neural network.
   - **Forward Propagation:** Transmit the input data through the network to compute the output.
   - **Calculate Loss:** Determine the error between the predicted output and the actual output using a chosen loss function.
   - **Backpropagation:** Propagate the error backward through the network to update the weights using gradient descent optimization.
   - **Repeat:** Iterate through steps 2 to 4 until convergence or a stopping criterion is met.

2. **Mathematical Equations**

   - **Forward Propagation** The output of a neuron is calculated using the activation function σ applied to the weighted sum of inputs plus a bias term.

$$
\begin{aligned}
z_j^{[l]} &= \sum_{i=1}^{n^{[l-1]}} w_{ij}^{[l]} a_i^{[l-1]} + b_j^{[l]} \\
a_j^{[l]} &= \sigma(z_j^{[l]})
\end{aligned}
$$

   - **Activation Functions** Common activation functions include:
     - **Sigmoid (Logistic):** $\sigma(z) = \frac{1}{1+e^{-z}}$
     - **ReLU (Rectified Linear Unit):** $\sigma(z) = \max(0, z)$
     - **Tanh (Hyperbolic Tangent):** $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
   - **Loss Function** The loss function measures the difference between predicted and actual output values.
     - **Mean Squared Error (MSE):** $\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(Y_i - \hat{Y}_i)^2$
     - **Cross Entropy (for classification):** $\text{CE} = -\frac{1}{m}\sum_{i=1}^{m} Y_i \log(y_i)$
   - **Backpropagation** The gradients of the loss function with respect to the weights and biases are computed using the chain rule of calculus and propagated backward through the network.

$$\frac{\partial J}{\partial W^{[l]}} = \text{weight update rule using gradient descent}$$

$$W^{[l]} = W^{[l]} - \alpha \cdot \frac{\partial J}{\partial W^{[l]}}$$

where α is the learning rate, and J is the loss function.

5 Concepts used in Algorithms

5.1 Understanding Dataset

The understanding of a dataset involves a comprehensive analysis of its features, structure, and statistical properties.

Characteristics of the Dataset: Consider a dataset D containing n rows and m columns, where each row represents an observation, and each column represents a feature.

- **Exploring Features:** Identify and categorize features into numerical, categorical, or textual types.
- **Statistical Summary:** Compute statistical measures such as mean, median, variance, and standard deviation for numerical features.
- **Data Distribution:** Visualize the distribution of numerical features using histograms, box plots, or density plots.
- **Correlation Analysis:** Calculate correlation coefficients between numerical features to discern relationships.

For a numerical feature X in the dataset D:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \mu)^2}$$

where X_i represents each observation of feature X and n is the total number of observations.

5.2 Handling NULL Values

Handling NULL or missing values is pivotal as they can impact the analysis and modeling process.

Identification of NULL Values:

- **Detection Methods:** Utilize functions like `isnull()` or `info()` to identify missing values.
- **Visualization:** Create visual representations (heatmaps, bar charts) highlighting the presence of missing values across features.

Handling Strategies:

- **Imputation:** Replace NULL values in numerical features with statistical measures like mean, median, or mode.
- **Deletion:** Exclude rows or columns with excessive NULL values if they significantly impact analysis.

Consider NULL values in a numerical feature Y in dataset D. To replace NULL values with the mean μ:

$$Y_{\text{imputed}} = \begin{cases} \mu, & \text{if } Y \text{ is NULL} \\ Y, & \text{otherwise} \end{cases}$$

5.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial step in understanding the dataset's underlying structure and uncovering patterns or anomalies within the data. Characteristics of EDA:

1. Statistical Measures:
   - Compute descriptive statistics like mean, median, mode, variance, and standard deviation for numerical features.
   - Analyze the distribution of data using histograms, box plots, or density plots.
2. Visualization Techniques:
   - Utilize graphical representations to visualize relationships between features using scatter plots, pair plots, or correlation matrices.
   - Identify outliers or anomalies through visual inspections.
3. Data Preprocessing:
   - Handle missing values by imputation or deletion based on the extent of missingness.
   - Scale or normalize numerical features to ensure consistency and facilitate model convergence.

Consider a dataset D with numerical features $X_1, X_2, \ldots, X_n$ and categorical features $C_1, C_2, \ldots, C_m$. To perform EDA:

- Calculate the mean (μ) and standard deviation (σ) for numerical features using:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2}$$

- Pair Plot: A pair plot is a grid of scatterplots where each variable is plotted against every other variable. For n numerical features, it creates an $n \times n$ grid where each cell represents the relationship between two variables.

$$\text{PairPlot}(X_i, X_j) \text{ for } i, j = \{1, 2, \ldots, n\}$$

- Correlation Matrix: A correlation matrix is a table showing correlation coefficients between variables. It is a square matrix where each cell (i, j) contains the correlation coefficient between variables X_i and X_j. The correlation coefficient ρ between two variables x and y is calculated using Pearson correlation:

$$\rho_{xy} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y}$$

  Where:

  - $\text{cov}(x, y)$ is the covariance between x and y.
  - σ_x and σ_y are the standard deviations of x and y respectively.

The correlation matrix provides insights into how strongly each feature is related to every other feature in the dataset. High positive values ($\rho \approx 1$) indicate strong positive correlation, negative values ($\rho \approx -1$) indicate strong negative correlation, and values close to zero ($\rho \approx 0$) indicate weak or no linear correlation.

5.4 Importing Libraries: Scikit-learn and Keras

Scikit-learn and Keras are powerful libraries in Python used for machine learning and deep learning, respectively. They offer various modules and tools to build models, perform optimizations, and train neural networks.

5.4.1 Sequential Models with Keras

Keras provides a high-level API for building neural networks. The Sequential model is a linear stack of layers that enables the creation of a neural network layer-by-layer.

Example of creating a Sequential model in Keras:

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(64, activation='relu', input_dim=100))
model.add(Dense(1, activation='sigmoid'))
```

This code defines a Sequential model with two layers: a Dense layer with 64 units and ReLU activation as the hidden layer, and a Dense layer with 1 unit and sigmoid activation as the output layer.

5.4.2 Adam Optimization Algorithm

Adam (Adaptive Moment Estimation) is an optimization algorithm used for training deep learning models. It combines the benefits of two other extensions of stochastic gradient descent: AdaGrad and RMSProp.

Mathematically, the update rule for Adam can be represented as follows:

$$\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\
v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_{t+1} &= \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}
\end{aligned}$$

Here, m_t and v_t are the first and second moment estimates of the gradients, g_t is the gradient at time step t, θ_{t+1} is the updated parameter at time step $t+1$, β_1 and β_2 are decay rates for moment estimates, α is the learning rate, and ϵ is a small constant to prevent division by zero.

5.4.3 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent is a popular optimization algorithm used in training machine learning models. It updates the parameters based on the gradient of the loss function with respect to a single training example at each iteration.

Mathematically, the update rule for SGD can be represented as:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla J(\theta_t, x_i, y_i)$$

Here, θ_{t+1} is the updated parameter at time step $t+1$, α is the learning rate, $\nabla J(\theta_t, x_i, y_i)$ is the gradient of the loss function J with respect to the model parameters evaluated at a single training example (x_i, y_i).

5.5 Modeling

5.5.1 Test-Train Split

The test-train split is a fundamental step in machine learning model building. It involves dividing the dataset into two subsets: one for training the model and the other for testing its performance.

Mathematically, let D be the dataset with n observations. The test-train split is represented as:

$$D = D_{\text{train}} \cup D_{\text{test}}, \quad D_{\text{train}} \cap D_{\text{test}} = \varnothing$$

Where: D_{train} is the training set containing a fraction γ of the dataset D, D_{test} is the testing set containing the remaining fraction $(1-\gamma)$ of the dataset.

The model learns patterns from the training set (D_{train}) and is evaluated on the unseen testing set (D_{test}).

5.5.2 Hyperparameter Tuning

Hyperparameter tuning involves finding the optimal hyperparameters for a machine learning model. Hyperparameters are set before training the model and cannot be learned directly from the data.

Mathematically, let H be the hyperparameters space, and J be the performance metric. Hyperparameter tuning aims to find:

$$h^* = \arg\min_{h \in H} J(h)$$

Where: h^* is the optimal hyperparameter set that minimizes the chosen performance metric J.

Common techniques for hyperparameter tuning include grid search, random search, and Bayesian optimization.

5.5.3 Predicting Target Values

Once the model is trained and tuned, it can be used to predict target values for new or unseen data. This prediction is based on the learned patterns from the training data.

Mathematically, for a trained model f with parameters θ and a new input x, the prediction $\hat{y}$ is given by:

$$\hat{y} = f(x; \theta)$$

Where: $\hat{y}$ is the predicted target value for the input x, f represents the trained model, θ are the learned parameters.

The accuracy of predictions is evaluated using performance metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or others based on the problem domain.

This structure provides information on important steps in modeling like Test-Train Split, Hyperparameter Tuning, and Predicting Target Values with corresponding mathematical representations. Adjust the content and equations to suit your specific requirements and dataset characteristics.

5.6 Error Metrics

Error metrics are crucial in assessing the performance of machine learning models. These metrics quantify the disparity between predicted values and actual values, providing insights into model accuracy, precision, and robustness.

5.6.1 Mean Squared Error (MSE)

Mean Squared Error is a common metric used to measure the average squared difference between predicted and actual values.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where: n is the number of observations, y_i represents the actual value, $\hat{y}_i$ is the predicted value for observation i.

5.6.2 Root Mean Squared Error (RMSE)

RMSE is the square root of the Mean Squared Error, providing an interpretable measure in the same units as the target variable.

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}}$$

5.6.3 Mean Absolute Error (MAE)

Mean Absolute Error measures the average absolute difference between predicted and actual values.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

5.6.4 R-squared (Coefficient of Determination)

R-squared quantifies the proportion of variance in the dependent variable that is predictable from the independent variables.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where: $\bar{y}$ is the mean of the observed data.

5.6.5 Precision, Recall, and F1-score

In classification problems, precision, recall, and F1-score are used to evaluate model performance.

Precision: Measures the accuracy of positive predictions.

Recall: Measures the proportion of actual positives correctly identified.

F1-score: Harmonic mean of precision and recall.

These error metrics provide insights into the performance of machine learning models, allowing for informed decisions on model selection and refinement.

5.7 Handling Outliers

Outliers are data points that significantly differ from other observations in a dataset. Handling outliers is crucial as they can skew statistical measures and affect model performance.

5.7.1 Identification of Outliers

Outliers can be identified using various statistical methods:

- **Z-score**: Calculate the Z-score for each data point, identifying those beyond a certain threshold (e.g., Z-score > 3 or < -3) as outliers.
- **Interquartile Range (IQR)**: Utilize the IQR to detect outliers lying outside the 1.5 times IQR range above the third quartile or below the first quartile.
- **Visualizations**: Box plots, scatter plots, or histograms can visually highlight data points that deviate significantly from the overall pattern.

5.7.2 Handling Strategies for Outliers

Once outliers are identified, several strategies can be employed:

- **Imputation**: Replace outliers with central measures like the mean, median, or mode of the data.
- **Capping or Flooring**: Set a threshold and cap (or floor) outlier values at this threshold.
- **Transformation**: Apply mathematical transformations (e.g., log transformation) to reduce the impact of outliers.
- **Removal**: Exclude outliers from the dataset if they significantly affect analysis but exercise caution to avoid losing important information.

5.7.3 Mathematical Approaches

For instance, using Z-score to identify outliers:

The Z-score formula for a data point x in a dataset X with mean μ and standard deviation σ is given by:

$$Z = \frac{x - \mu}{\sigma}$$

Outliers are typically defined as data points with Z-scores exceeding a certain threshold, often considered as greater than 3 or less than -3.

Handling outliers mathematically through imputation or capping involves replacing or altering values using statistical methods while ensuring the integrity of the dataset.

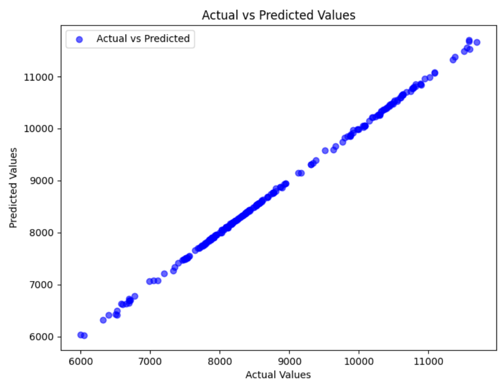
These strategies ensure that outliers do not unduly influence analyses or model performance, leading to more robust and reliable results.

6 Final Outcome

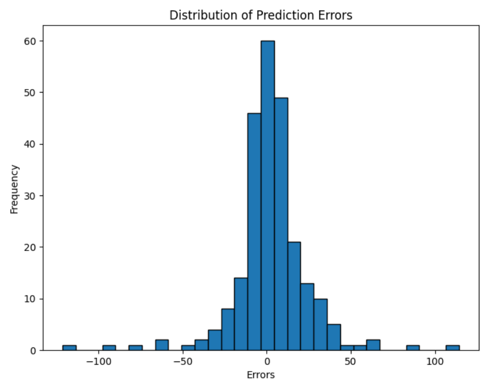
In this section, I have included graphs that display what I found using two methods: XGBoost and ANN. The main goal of my study was to use specific learning methods to predict the closing prices of stocks in the Nifty 50 dataset. This dataset covers data from Yahoo NSE, spanning the years 2014 to 2018.

I spent time writing code for this dataset and looked into various aspects using different graphs like scatter plots to find connections, error distribution plots to understand mistakes, confusion matrices to assess how well the model performed, heat maps to illustrate connections, bar charts to represent categories, and distplots to examine how information is distributed.

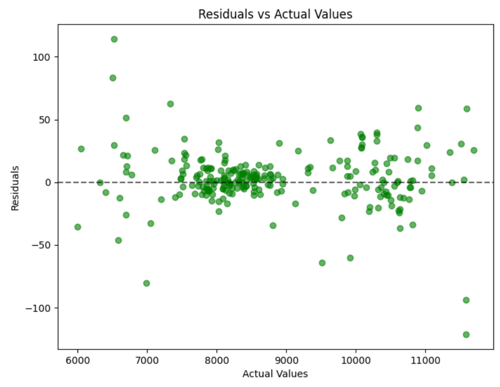
These graphs helped me dig deeper into the dataset, allowing me to understand how different factors relate to one another.



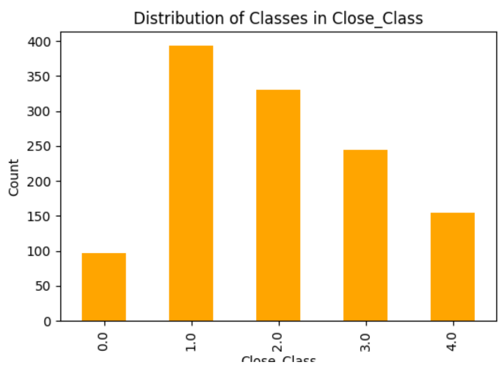
(a) Actual vs Predicted Values



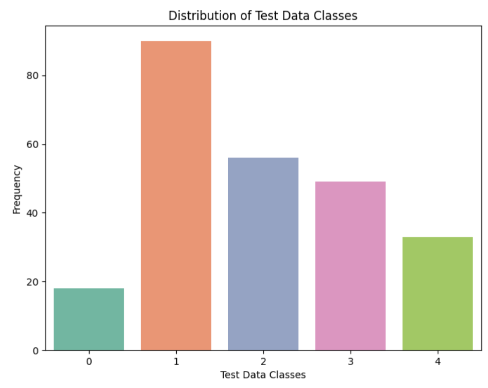
(b) Distribution of prediction errors



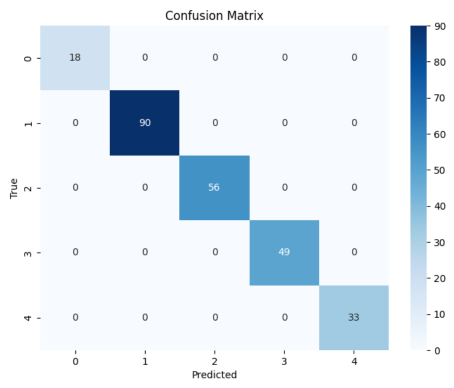
(c) Residual vs Actual Values



(d) Distribution of Classes in Close Class

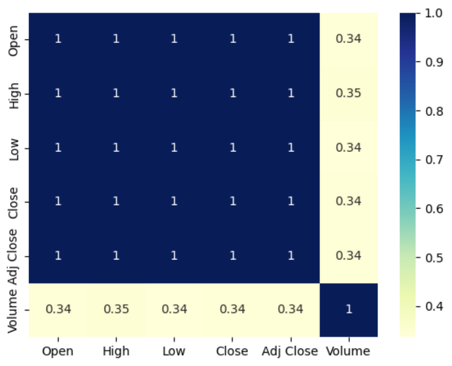


(e) Distribution of Test Data Classes

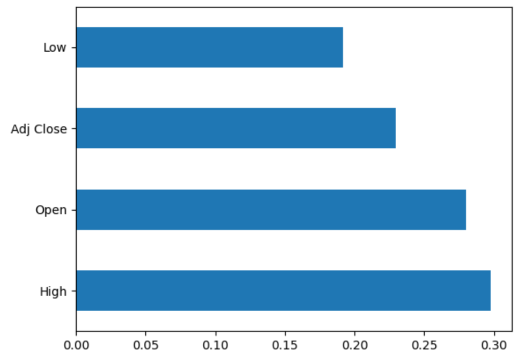


(f) Confusion Matrix

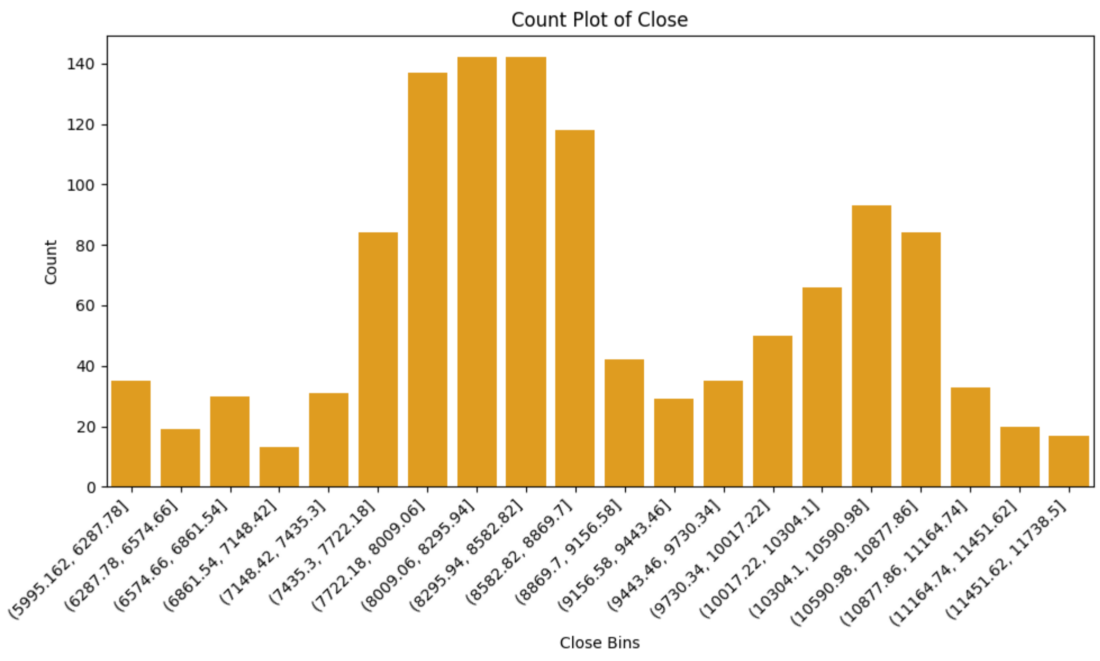
Results obtained using XGBoost Algorithm



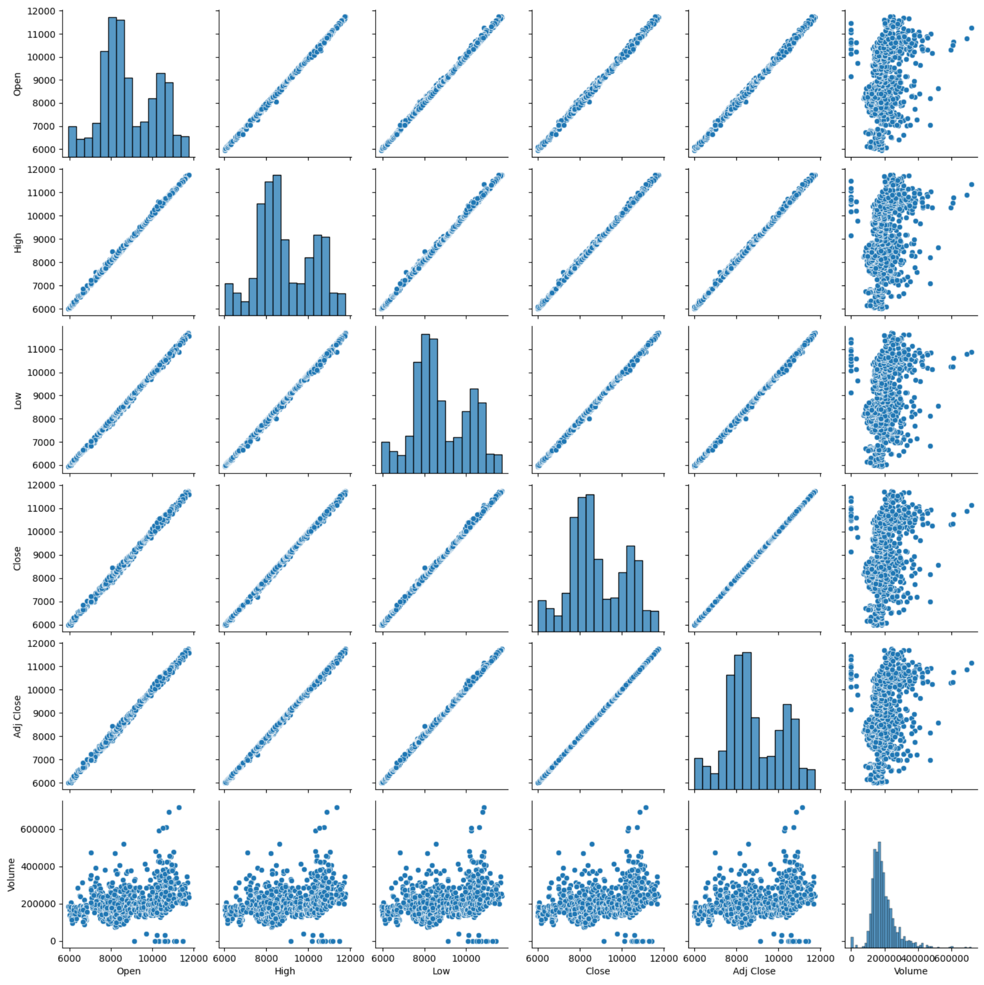
(a) Correlation between features



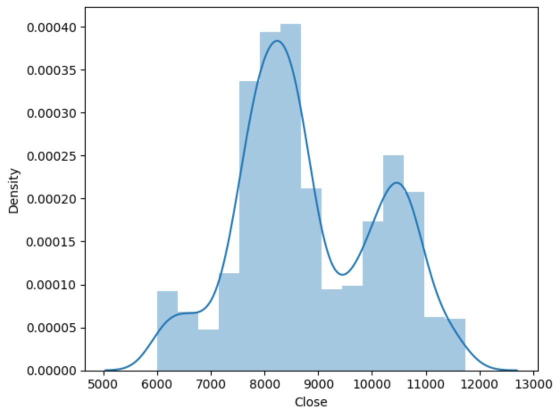
(b) Feature Importance Visualisation

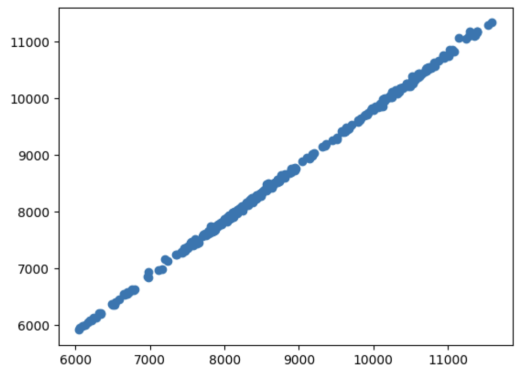


(c) Count Plot of Close



(d) Pairplot





(e) Probability Density Function

(f) Actual vs Predicted values

Results obtained using ANN Algorithm

7 Conclusion

Upon utilizing the XGBoost algorithm to forecast closing prices, the model initially exhibited a mean absolute error (MAE) of 14.2 and a root mean squared error (RMSE) of 22.58.

The starting RMSE of 22.58 served as a benchmark for the model's preliminary performance. Employing an exhaustive exploration involving 216 potential models and employing a 3-fold validation process, I fine-tuned the hyperparameters. The optimized model notably enhanced the RMSE to 16.03.

This substantial reduction underscores the model's heightened predictive capacity, suggesting its potential for superior generalization and accuracy when applied to new, unseen data.

The constructed Sequential ANN model comprises five layers: four dense layers with varying neuron counts (128, 256, 256, 256) and a final output layer yielding a single value. This architecture sums up to a total of 165,633 trainable parameters, occupying 647.00 KB of memory.

Training the model for 300 epochs resulted in promising predictive performance:

Model: *Sequential*

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               768

 dense_1 (Dense)             (None, 256)               33024

 dense_2 (Dense)             (None, 256)               65792

 dense_3 (Dense)             (None, 256)               65792

 dense_4 (Dense)             (None, 1)                 257
```

Total params: 165,633 (647.00 KB)
Trainable params: 165,633 (647.00 KB)
Non-trainable params: 0 (0.00 Byte)

I used 300 epochs and obtained the following performance metrics:

- **Mean Absolute Error (MAE):** 173.25
- **Mean Squared Error (MSE):** 31,493.92
- **Root Mean Squared Error (RMSE):** 177.47
- **R-squared (R2) Score:** 0.98

The R-squared score of 0.98 demonstrates strong explanatory power, indicating the model's ability to capture variance in the data. However, these metrics should be considered in the context of the specific problem domain for practical application.

8 Acknowledgement

I extend my heartfelt gratitude to Dr. Ankur Kanaujiya for guiding me through the captivating realm of data-driven option pricing. Working on this project was an eye-opener, introducing me to complex financial concepts like options and how markets operate. It provided an invaluable insight into how data-driven approaches are employed to foresee closing prices, enabling more informed and prudent investment decisions.

This project deepened my understanding of mathematical models like XGBoost and Artificial Neural Networks (ANN). Learning the ropes of these algorithms not only enhanced my ability to make accurate predictions but also revealed their significant real-world applications. These mathematical tools are instrumental, not just in predicting outcomes but also in shaping trading strategies, risk assessment, and decision-making across various domains.

Dr. Ankur Kanaujiya's mentorship was invaluable, offering profound insights and steering my curiosity toward the multifaceted landscape of financial technology. His guidance has been instrumental in shaping my understanding and nurturing my interest in this dynamic and enriching field.

References

[1] Yahoo Finance - Nifty 50 Historical Data.
`https://finance.yahoo.com/quote/%5ENSEI/history/`

[2] Investopedia - Buying Options.
`https://www.investopedia.com/articles/optioninvestor/09/buying-options.asp`

[3] Corporate Finance Institute - Option Pricing Models.
`https://corporatefinanceinstitute.com/resources/derivatives/option-pricing-models/`

[4] Simplilearn - XGBoost Algorithm in Machine Learning.
`https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article#:~:text=XGBoost%20is%20a%20robust%20machine,optimize%20their%20machine%2Dlearning%20models.`

[5] GeeksforGeeks - XGBoost.
`https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/`

[6] GeeksforGeeks - Artificial Neural Networks and its Applications.
`https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/`