Forecasting Implied Volatility to Identify Undervalued Options: Model Creation

Nicholas Stranges (251150534)

Goal and Overview

The main goal of this project is to combine finance and machine learning to gain a better understanding of how machine learning can be utilized in financial markets. Modern traders are frequently using machine learning to predict market outcomes and improve returns. As new artificial intelligence (AI) algorithms improve, hedge funds and banks need traders that can understand and implement state of the art tools.

We wanted to use the Black-Scholes model in this project as it was a method we focused on in class and is a common options pricing model in the greater market. It also offers a closed form solution where variables can be directly substituted, making it easier to build a model around. This means that we were looking to trade European options on an underlying of our choosing. Apple stock was chosen as the underlying because both Tristan and I are interested in software companies. Apple also has low and stable volatility and a high market cap making it easier to build a prediction model on its data.

Prediction Method

In the Black-Scholes model, all inputs except for implied volatility are given by the market or the option data. Implied volatility cannot be directly calculated as it is a prediction about how much the stock will move in the future. Equation 1 shows the formula for calculating d_1 and d_2 which are the essential components of the Black-Scholes model and are used for both call and put options. As all other variables are fixed, changing the implied volatility prediction changes the price of the option directly.

$$d_1 = \frac{\ln\left(\frac{S}{X}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

Equation 1: d_1 and d_2 for the Black-Scholes Model

The market price of the option can therefore be underpriced (beneficial to buy) or overpriced (beneficial to sell) depending on its implied volatility relative to the realized volatility until the time to expiry. To create a scenario where a machine learning model can be used predict the "true" implied volatility we need a way to calculate what the implied volatility should have been based on the future movements of stock performance. The log returns of Apple from the current date to the expiry date were used as a metric for the "true" stock performance over the option's life, seen in Equation 2. Annualized volatility was then calculated to give the "true" implied volatility over the option's life, seen in Equation 3. The volatility was annualized so that options with different lives would be represented equally, an important consideration when training machine learning models. Once the implied volatility of an option was predicted, Tristan would use that information to make profitable trades based on purchasing undervalued options.

$$R_i = ln(\frac{S_{i+1}}{S_i}), i = 1, 2 \cdots, N-1$$

Equation 2: Log Returns of Stock Movements

$$\sigma = \sqrt{252} * STDEV\{R_1, R_2, \cdots, R_{N-1}\}$$

Equation 3: Annualized Volatility of Log Returns

Data Collection

The main motivation behind the data collection process was to collect data from a wide range of sources and let the model decide what features are important in prediction. Data was collected from three categories: economic indicators, fundamental financial data, and option chain data. The economic indicators included long-term macroeconomic datapoints such as Gross Domestic Product (GDP) and the Consumer Price Index (CPI) which are announced on a quarterly basis. The economic indicators data was collected from Federal Reserve Economic Data (FRED) using the Pandas data reader Python library [1]. The fundamental finance data was data usually used in value investing such as Price-to-Earnings Ratio and the Beta value. This financial data was gathered from the Yahoo Finance API [2]. The last category of data was the option chain including information such as bid and ask prices, option "Greeks", and strike price. Historical option chain data was hard to find as much of this data was offered as paid packages. We were able to find a historical S&P 500 options database that has been collected since 2019 on Dolthub [3]. We confirmed that this data was accurate and selected only options that were applicable to Apple stock. This data was formatted and organized using the Pandas Python package [4]. Data was split into two sets, 2021-2023 which would be used to train and test the model for prediction and 2024 which was used by Tristan to backtest our trading algorithm.

Model Choice

A XGBoost model was chosen for prediction as it is simple to implement and is relatively fast to train. XGBoost uses many decision tree models as base learners and combines them to create a final regression prediction result [5]. The eXtreme Gradient Boosting Python library was chosen as this is the most common library for implementing XGBoost models [6]. To format the collected data into a row that the XGBoost model could train on, 50 options with the same

expiration date and current date were combined into one row with the corresponding economic and fundamental finance data. Very little feature engineering was done on this data, only removing missing values, changing expiry date into days to expiry, and changing put/call into a binary metric.

Model Results

The first important realization about the model was through its original feature importance breakdown. This showed that both the economic indicators and the fundamental finance data did not help in making predictions for the implied volatility. This is understandable because many of these factors are released on a quarterly basis and therefore might not change over the option's life. Also, these factors are most likely already priced into the option's price as this data is the first source of information around the company and the market. Both the economic indications and the fundamental finance data was removed, only relying on option data for 50 options at the same date and time to expiry.

The model showed that it could learn from this data finishing with a R^2 value of 64.4%. A sufficient R^2 value depends on the situation that you are trying to predict, and I believe that this result is sufficient given it is predicting a future value in the stock market. The final percentage error of the model was 16% corresponding to a 4.1% prediction error on the implied volatility.

The graph in Figure 1 shows a histogram of residual values from the model predictions. Residuals are the model errors between the predicted and true values. It is important to see that the residuals are normally distributed meaning that the errors are random and unbiased. This demonstrates that the model did learn the underlying distribution of the data and with model improvements, prediction accuracy can be improved.

The graph in Figure 2 shows a scatter plot of the predicted vs actual volatility. The red linear line is perfect prediction where predicted and actual correspond directly with each other. The model's performance in low volatility regimes is much better as data points are clustered tightly in this area. As volatility increases, model performance decreases, consistently underestimating the true volatility level. This points to the fact that the dataset being used is unbalanced and has much more examples of low volatility outcomes, understandable for Apple as it exhibits relatively stable trading patterns.

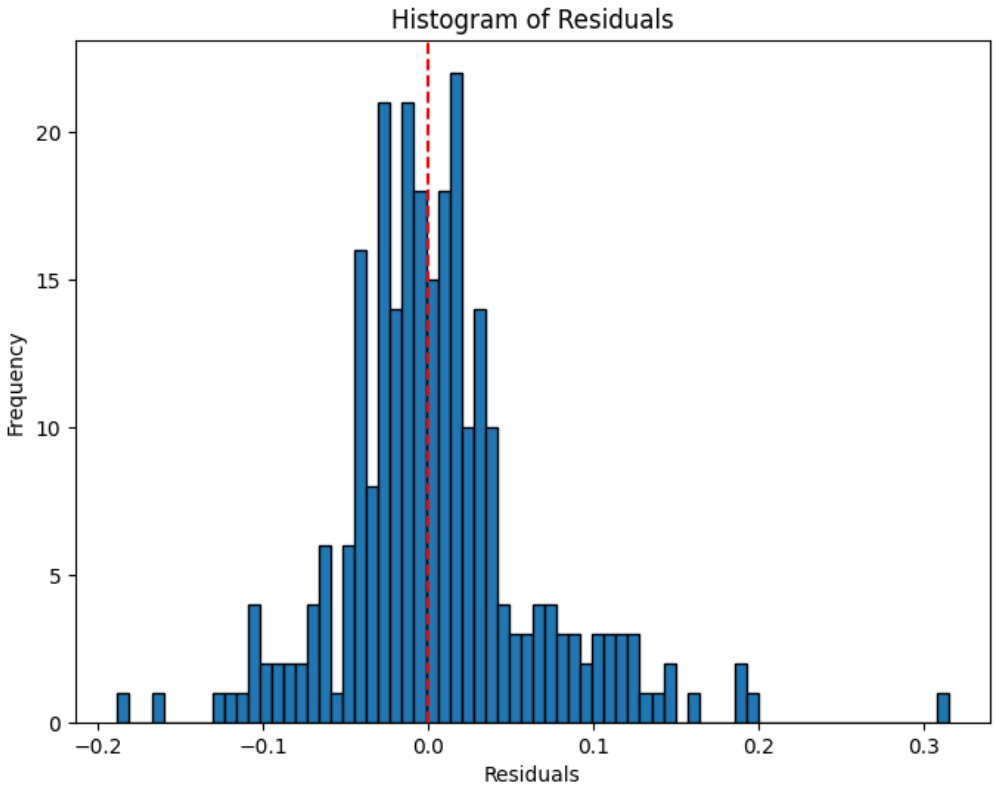

Figure 1: Residual Histogram Plot

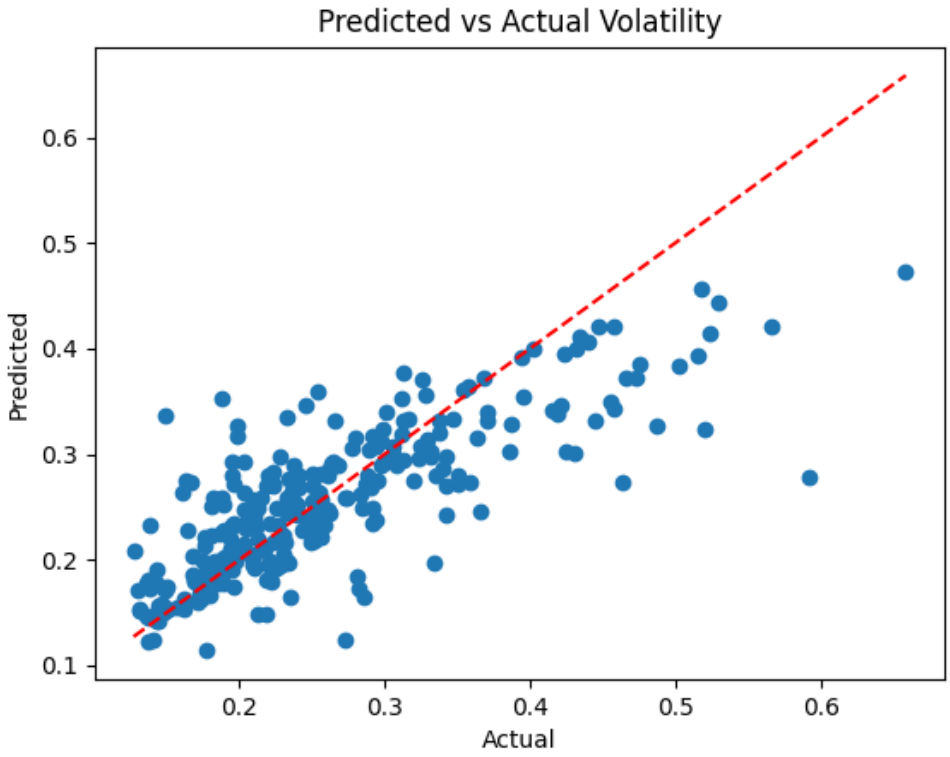

Figure 2: Predicted vs Actual Volatility Scatter Plot

Future Improvements

As shown Figure 2, model performance could be directly improved by balancing the training dataset to better represent high volatility data. This can be done many ways, but I would consider oversampling high volatility samples. As discussed before, I did minimal feature engineering on the raw data. To improve performance, I could transform the data into a normal distribution depending on what the original distribution of the feature was, such as a power distribution. I could also derive more features other than the "Greeks" such as bid/ask ratio to help the model extract a better prediction distribution.

I would also be interested in testing more non-linear models that could learn more complex representations. For example, Long Short Term Memory (LSTM) networks have been

shown to do well with time series modelling as it is based on recurrent neural networks. The training dataset could be changed to represent the option chain as a time series instead of only using a single day of the option chain as an input to the model. More experimentation is needed to confirm the viability of a deep learning approach on this dataset.

Conclusion

This project showed that machine learning can be used to predict implied volatility and help identify undervalued options for trading. By using the Black-Scholes model and training an XGBoost model on grouped option chain data, we were able to predict implied volatility with reasonable accuracy. The model performed well in low volatility regimes, which makes sense given Apple's typical trading behavior, but struggled with higher volatility predictions due to an unbalanced dataset. With better representation of high volatility data, more feature engineering, and testing of deep learning approaches like LSTM, there's a clear path forward for improving the model. Overall, this project was a strong starting point for understanding how machine learning can be applied to options trading.

References

[1] The PyData Development Team, "Federal Reserve Economic Data (FRED)," pandas-datareader Documentation, Jul. 13, 2021. [Online]. Available: https://pandas-datareader.readthedocs.io/en/latest/readers/fred.html. [Accessed: Mar. 31, 2025].

[2] R. Aroussi, "yfinance," PyPI, Mar. 20, 2025. [Online]. Available: https://pypi.org/project/yfinance/. [Accessed: Mar. 31, 2025].

[3] post-no-preference, "Options," DoltHub, [Online]. Available: https://www.dolthub.com/repositories/post-no-preference/options/data/master. [Accessed: Mar. 31, 2025].

[4] The pandas Development Team, "pandas - Python Data Analysis Library," pandas.pydata.org, Sep. 20, 2024. [Online]. Available: https://pandas.pydata.org/. [Accessed: Mar. 31, 2025].

[5] GeeksforGeeks, "XGBoost," GeeksforGeeks, Feb. 2, 2025. [Online]. Available: https://www.geeksforgeeks.org/xgboost/. [Accessed: Mar. 31, 2025].