A Machine-Learning Based Exploration into Quantitative Finance and Portfolio Theory

Andrew Tran - Erdos Quant Finance Summer 2025

Statement of Completion

This report serves as a personal statement of completion for all four mini-projects in the Summer 2025 Quantitative Methods in Finance course. Each project built upon key concepts in quantitative finance, covering portfolio analysis, statistical testing of asset returns, and advanced hedging techniques using the Heston model. All projects were implemented in Jupyter notebooks, which analyses supported by simulation, visualization, and interpretation of results.

Self-Implemented Techniques

In Mini-Project 4, I extended the assignment beyond the core curriculum by integrating machine learning models to evaluate hedging performance under the Heston stochastic volatility framework. This involved framing a binary classification task, to predict whether a delta hedge on a short call option would result in high or low hedging error, based on features observable at the beginning of the trading process.

I utilized feature engineering on the final asset price at maturity, the realized volatility, and the moneyness of the option, and implemented ML models such as logistic regression, random forest, and XGBoost. XGBoost achieved the strongest performance, with ~93.5% accuracy, and ~98% precision in detecting high-error scenarios. The process revealed that final price and realized volatility were the most predictive variables.

While the ML models performed well on the synthetic dataset generated by the Heston model, several limitations remain. Although the use of synthetic data allowed for controlled experiment, real market dynamics include factors not captured by the model, such as jumps, liquidity shocks, or macroeconomic/political events. Future work could train and evaluate models on real market data to validate robustness, and go beyond the currently somewhat simple feature set of price, volatilities, and moneyness. Additionally, while metrics like accuracy and precision were high, these were based on a static testing/training split. A more rigorous evaluation could use technqiues like cross validation to ensure more robust generalziation. Finally, models like XGBoost, while powerful, could suffer from overfitting.

This addition required self-learning in areas such as dataset creation from simulated paths, classification thresholding, model evaluation (confusion matrix and F1 scores), and XGBoost's handling of imbalanced classes.