Ensemble Learning Methods in Investment Management: A Case Study with QQQ ETF Data

Ahmed Masood

19.01.2024

Abstract

The dynamic nature of financial markets presents a substantial challenge for predictive modeling, particularly in investment management. This study explores the efficacy of ensemble learning techniques in forecasting the performance of the QQQ Exchange-Traded Fund (ETF), a popular investment vehicle that tracks the NASDAQ-100 Index. Leveraging a dataset spanning from December 22, 2013, to December 22, 2023, extracted using yfinance, the research employs various machine learning algorithms, including Random Forest, K-Nearest Neighbors, Logistic Regression, and Support Vector Machines, to predict ETF price movements. A unique aspect of this approach is the integration of a wide range of technical indicators generated through pandas_ta, providing a comprehensive set of features for model training.

The study addresses class imbalance in the dataset with a custom weighting function and employs BorutaPy for feature selection, ensuring the most relevant predictors are utilized. Post-selection, models are fine-tuned using RandomizedSearchCV for optimal parameter identification. A novel aspect of the research is the construction of a meta-model using XGBoost, which synthesizes predictions from base models to enhance predictive accuracy. The performance of the meta-model, evaluated through accuracy metrics and ROC curves, demonstrates its superiority over individual models.

This research contributes to the field of investment management by illustrating the potential of blending diverse machine learning techniques for improved ETF performance prediction. The findings hold significant implications for investors and financial analysts seeking robust predictive models in the rapidly evolving financial landscape.

Keywords: Investment Management; ETF Prediction; Ensemble Learning; Machine Learning; Financial Forecasting

1 Introduction

The realm of investment management is perpetually evolving, marked by increasing complexity and a burgeoning reliance on advanced analytical techniques. In this modern financial landscape, the ability to accurately predict market movements is more than a mere competitive edge; it is a necessity for survival and success. Among the plethora of investment instruments, Exchange-Traded Funds (ETFs) have gained substantial popularity due to their diversified nature and flexibility. In particular, the QQQ ETF, which tracks the NASDAQ-100 Index, stands out as a prime candidate for analytical exploration due to its representation of innovative and growth-oriented industries.

The primary challenge in forecasting ETF prices lies in the inherent volatility and unpredictability of financial markets. Traditional financial theories, while foundational, often fall short in capturing the multifaceted and dynamic nature of market behavior. This shortfall necessitates the integration of advanced data-driven methodologies that can assimilate the myriad

factors influencing market fluctuations. In this context, machine learning emerges as a potent tool, offering sophisticated algorithms capable of discerning patterns and trends from complex datasets.

However, the application of machine learning in financial prediction is not without its challenges. Financial time series data are typically characterized by noise, non-stationarity, and class imbalance, each of which can significantly impede model performance. Furthermore, the selection of relevant features from an extensive array of technical indicators adds another layer of complexity to the predictive modeling process.

This study aims to address these challenges by employing an ensemble learning approach, wherein multiple machine learning models are strategically combined to enhance predictive accuracy. The ensemble method capitalizes on the unique strengths of individual models, thereby mitigating their respective weaknesses. The research focuses on the QQQ ETF, utilizing a rich dataset extracted from yfinance and enriched with a comprehensive set of technical indicators through pandas_ta.

The cornerstone of this research lies in its holistic approach to model development, encompassing various stages from data preprocessing to model evaluation. Initially, the study addresses the issue of class imbalance inherent in financial datasets, a critical factor often overlooked in predictive modeling. An innovative class weighting function is introduced, ensuring that the model does not bias towards the majority class and overlooks the subtleties of the minority class, which often signifies critical market movements.

Subsequently, the study delves into feature selection, a pivotal step in model optimization. The Boruta algorithm, an all-relevant feature selection method, is employed to systematically identify the most significant predictors from a pool of technical indicators. This process not only enhances model performance but also provides valuable insights into the factors most influential in ETF price movements.

The ensemble learning approach adopted in this research is particularly noteworthy. It involves the integration of diverse machine learning models, including Random Forest, K-Nearest Neighbors, Logistic Regression, and Support Vector Machines. Each of these models brings a unique perspective to the analysis, capturing different aspects of the data. The culmination of this approach is the development of a meta-model using XGBoost, an advanced ensemble technique that amalgamates predictions from base models to deliver a refined and robust forecast.

The efficacy of the proposed ensemble learning model is rigorously evaluated against traditional and individual machine learning models. Metrics such as accuracy, F1 score, and the Receiver Operating Characteristic (ROC) curve are employed to assess performance, ensuring a comprehensive evaluation of the model's predictive capabilities.

This study contributes to the field of investment management by demonstrating the practical application of ensemble learning in financial forecasting. It provides a blueprint for leveraging advanced machine learning techniques to enhance decision-making in financial markets. The insights derived from this research are invaluable for investors, portfolio managers, and financial analysts seeking to harness the power of machine learning for informed investment strategies.

In the following sections, the paper will detail the methodology employed, present the results of the analysis, and discuss the implications of the findings in the broader context of investment management and machine learning applications in finance.

2 Data

The cornerstone of this research is a meticulously compiled dataset from the QQQ Exchange-Traded Fund (ETF), which closely follows the NASDAQ-100 Index. This dataset, extending from December 22, 2013, to December 22, 2023, was extracted utilizing the yfinance Python

library, renowned for its efficiency in retrieving historical market data. The selection of the QQQ ETF as the focus of this study stems from its comprehensive coverage of various sectors, reflecting the performance of the top 100 largest non-financial companies on the NASDAQ stock exchange.

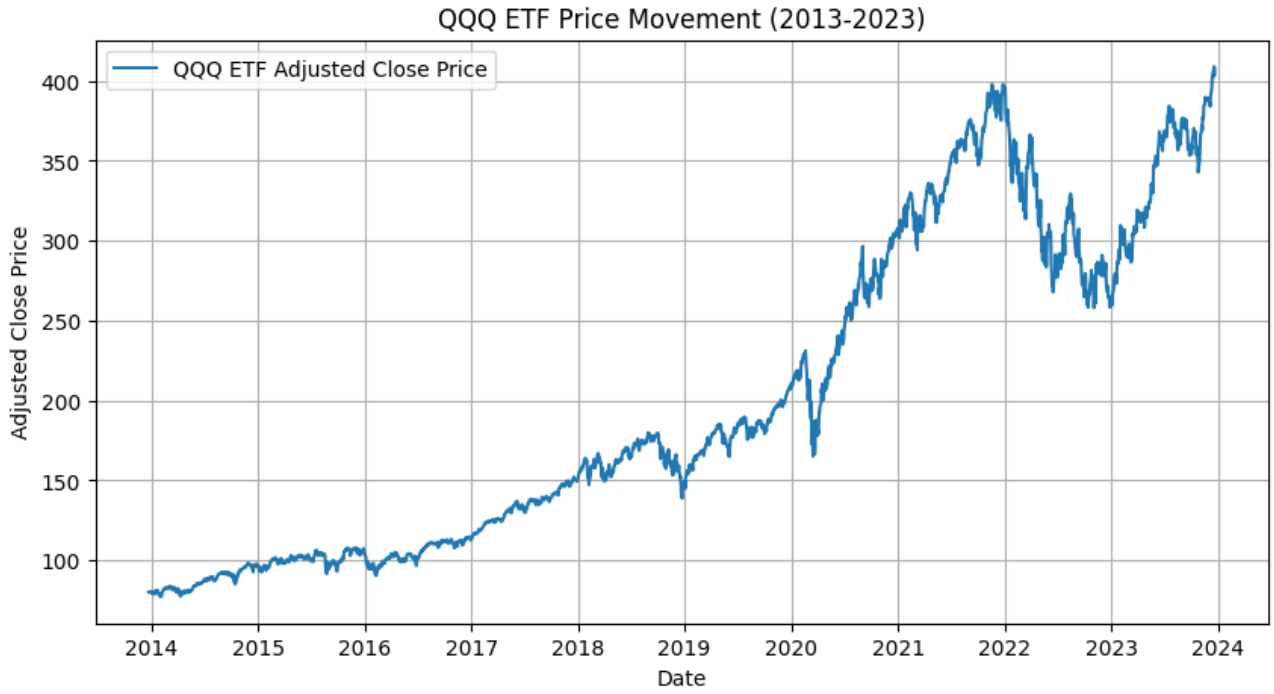

Figure 1: QQQ price movement.

2.1 Data Preparation and Feature Engineering

The initial data retrieved from yfinance comprises essential market indicators such as Open, High, Low, Close, and Volume. To bolster the models' predictive power, the following features were engineered:

- **Open to Close (o2c):** The difference between the opening and closing prices.
- **High to Low (h2l):** The daily price range, defined as the difference between the highest and lowest prices.

Further, the dataset was enriched with a diverse array of technical indicators via the pandas_ta library. This augmentation is vital as technical indicators are commonly employed in algorithmic trading to deduce statistical trends from trading activity.

2.2 Label Definition, Class Imbalance, and Target Variable Specification

The target variable, termed *predict*, is designed to identify significant movements in the ETF's price, specifically focusing on 10-day returns. A label of 1 indicates an uptrend, defined as the adjusted closing price increasing by more than 0.5% over a 10-day period. Conversely, any positive movement of less than 0.5% or a negative movement is classified as 0, denoting a lack of substantial price increase. This approach mirrors real-world investment strategies, emphasizing the prediction of notable price changes, both upward and downward.

A significant challenge encountered was class imbalance, a frequent issue in financial datasets, where one class dominantly overshadows the other. In this case, instances of minor price changes were more prevalent than significant price fluctuations. A tailored class weighting function was thus employed to mitigate model bias towards the majority class, ensuring a more balanced and accurate predictive performance.

2.3 Feature Selection with Boruta

Given the extensive number of features generated, a methodical feature selection process was essential. The Boruta feature selection algorithm was employed for this purpose. Boruta works by creating shadow features (random shuffles of real features) and iteratively evaluating the importance of each real feature compared to these shadows.

$$\text{Feature Importance} = \frac{\text{Mean Decrease in Impurity of Real Feature}}{\text{Mean Decrease in Impurity of Shadow Features}} \tag{1}$$

Using the RandomForestClassifier as a base estimator, the algorithm identified features whose importance was statistically significant. This process not only streamlined the feature set but also provided insights into the most influential predictors for ETF price movements.

2.4 Data Normalization and Split

Post feature selection, the dataset was normalized using the MinMaxScaler to ensure that all features contribute equally to the model's performance. The data was then split into training, testing, and validation sets. The splits were performed sequentially without shuffling, respecting the time series nature of the data, to prevent any lookahead bias:

$$\text{Training Set: } 80\%, \text{ Testing Set: } 10\%, \text{ Validation Set: } 10\% \tag{2}$$

This careful preparation of the dataset lays the groundwork for the subsequent modeling and analysis phases, which are detailed in the following sections.

3 Methodology

3.1 Model Development

The methodology adopted in this research revolves around the construction and evaluation of multiple machine learning models, followed by the creation of a meta-model. The initial phase involved training four fundamental models: Random Forest, K-Nearest Neighbors (KNN), Logistic Regression, and Support Vector Machines (SVM), each chosen for their unique attributes and proven efficacy in classification tasks.

3.2 Detailed Model Descriptions

3.2.1 Random Forest

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. The fundamental concept is that a group of "weak learners" can come together to form a "strong learner".

- **Handling Non-linearity:** Each decision tree in the Random Forest splits the data into branches, which allows for capturing non-linear relationships between features and the target variable.
- **Reduction of Overfitting:** By averaging the results of different trees, Random Forest reduces the risk of overfitting associated with individual decision trees.
- **Feature Importance:** It provides insightful outputs on feature importance, which is beneficial for understanding the driving factors behind predictions.

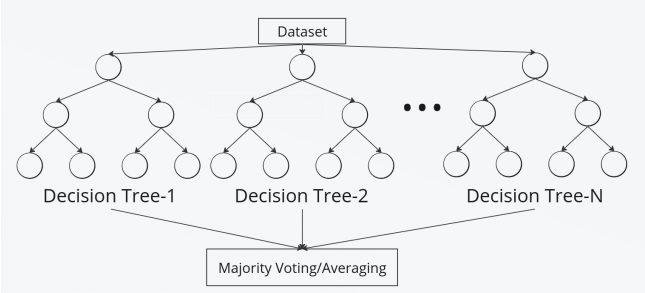

Figure 2: Random Forest.

3.2.2 K-Nearest Neighbors (KNN)

KNN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation. It is one of the simplest of all machine learning algorithms.

- **Simplicity:** KNN is straightforward and intuitive, making it a great starting point for classification tasks.
- **Versatility:** It can be used for both classification and regression tasks.
- **No Model Assumptions:** KNN doesn't make any underlying assumptions about the distribution of data, which is beneficial in real-world scenarios where many of those assumptions are likely to be violated.

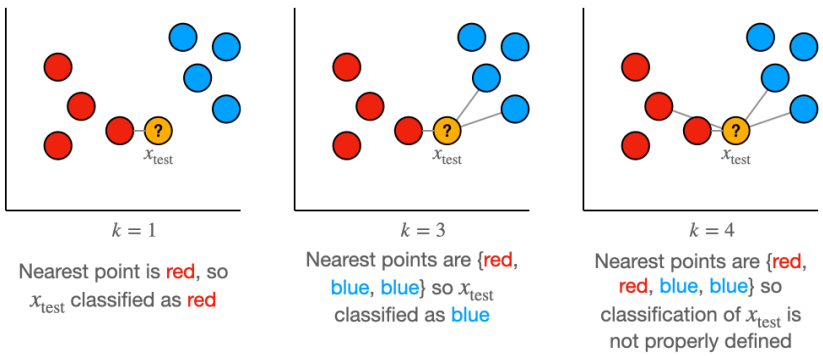

Figure 3: K-Nearest Neighbors.

3.2.3 Logistic Regression

Despite its name, Logistic Regression is used in binary classification (not regression). It estimates the probability that a given input point belongs to a certain class.

- **Probabilistic Approach:** Outputs a probability score that reflects the likelihood of a given outcome.
- **Efficiency:** It is highly efficient and does not require too much computational resources, which makes it highly scalable for large datasets.
- **Interpretability:** The model coefficients can be translated into odds ratios, making it easier to interpret the results.

$$\sigma(z) = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + ... + w_nx_n + b)}} \tag{3}$$

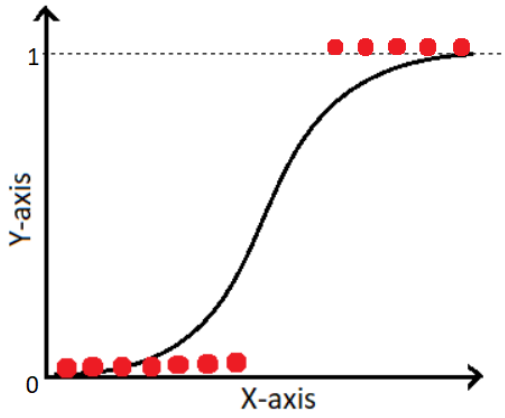

Figure 4: Logistic Regression.

3.2.4 Support Vector Machines (SVM)

SVM is a powerful, supervised machine learning algorithm used for both classification and regression tasks. It is most commonly used in solving classification problems.

- **Maximizing Margin:** SVM seeks to find the best margin (distance between the line and the support vectors) that separates the classes, which can enhance classification performance.
- **Kernel Trick:** It can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping inputs into high-dimensional feature spaces.
- **Handling High Dimensionality:** SVM performs well in high-dimensional spaces, which is particularly beneficial for datasets with a large number of features.

Each of these models brings its unique strengths to the ensemble approach, contributing to a more robust and reliable predictive model. The choice of these models is driven by their proven track record in handling various types of data and their widespread acceptance in the machine learning community.

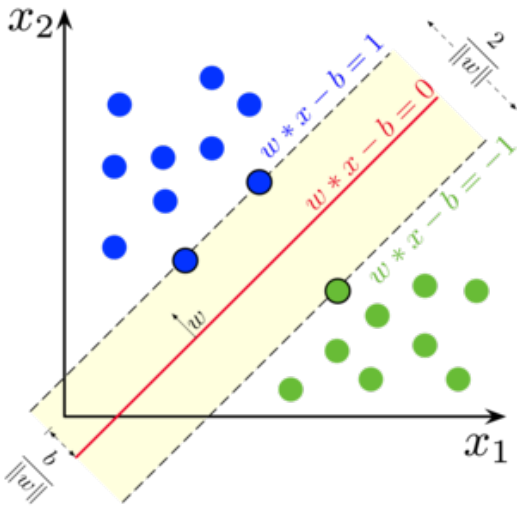

Figure 5: Support Vector Machine.

3.3 Boruta Feature Selection - In-Depth

Boruta is an all-relevant feature selection method, meaning it aims to capture all features that can contribute anything to the prediction model, not just the most influential ones. This method is particularly useful in scenarios where we need a comprehensive understanding of the factors influencing the model, as in financial time series prediction.

- **Shadow Feature Creation:** Boruta works by creating duplicates of all features in the dataset (known as shadow features) and shuffling them to remove any relationships with the target variable.
- **Statistical Testing:** The algorithm then runs a random forest classifier on this extended dataset and evaluates the importance of each real feature against its shadow version.
- **Iterative Process:** Through an iterative process, features not performing better than their shadow counterparts are systematically removed from consideration. This process continues until all features are either confirmed or rejected with a certain level of statistical confidence.
- **Dimensionality Reduction:** By focusing only on features that are statistically significant, Boruta effectively reduces the dimensionality of the data, leading to more efficient and potentially more accurate models.

3.4 Meta-Model Construction - XGBoost

After individual models are trained and their performance is evaluated, a meta-model, or ensemble model, is constructed. This model aims to synthesize the predictions of each base model into a final consolidated prediction.

- **Leveraging Strengths of Base Models:** The meta-model approach is grounded in the philosophy that the collective wisdom of multiple models surpasses that of any single model. It allows for the blending of different types of algorithms, capitalizing on their individual strengths and compensating for their weaknesses.
- **Choice of Meta-Model - XGBoost:** For the meta-model, XGBoost (eXtreme Gradient Boosting) was chosen. XGBoost is renowned for its performance and speed in

classification tasks. It is an optimized distributed gradient boosting library that provides a highly efficient implementation of the gradient boosting framework.

- **Training on Base Model Predictions:** The meta-model is trained on a dataset created from the predictions of the base models. This approach essentially treats the outputs of the base models as features for the final prediction, allowing the meta-model to learn the best way to combine these predictions.
- **Complex Interactions:** XGBoost is particularly adept at handling the complex interactions between the base model predictions, effectively identifying patterns that may not be apparent to any single model.

Training the Meta-Model: The meta-model was trained on a feature set comprising the predictions of the base models on the validation set. This approach ensures that the meta-model learns to effectively integrate the insights provided by each of the base models.

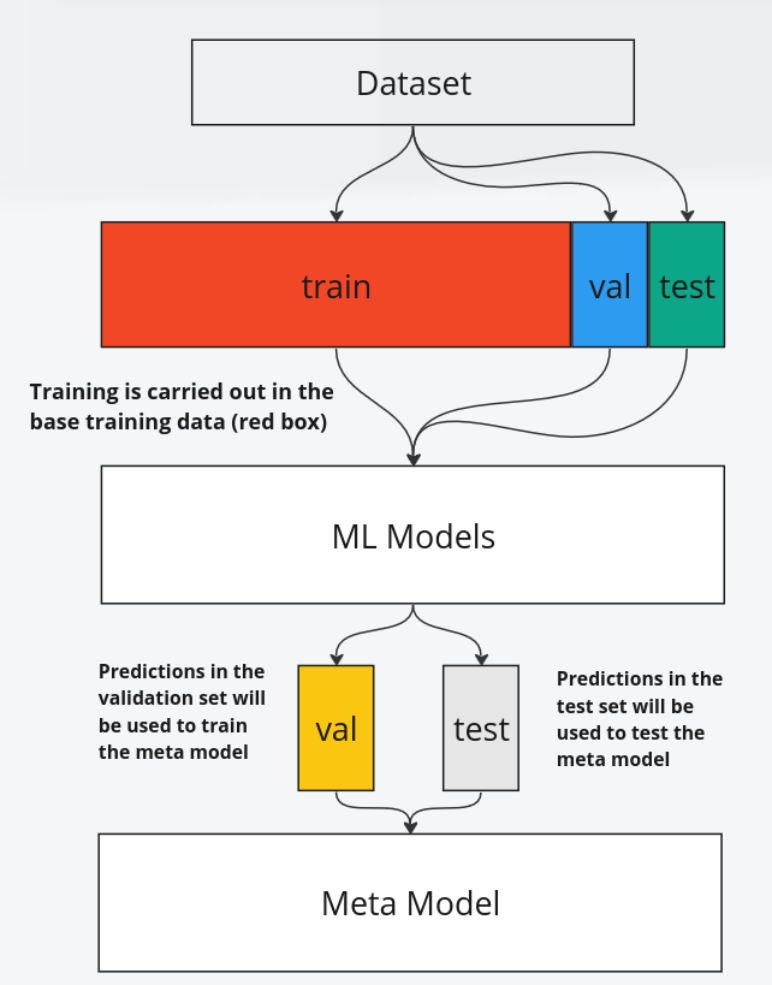

Figure 6: Training the Meta Model

3.5 Model Tuning and Evaluation

The final stage of the methodology involved fine-tuning each model, including the meta-model, using RandomizedSearchCV for hyperparameter optimization. This process facilitated a comprehensive exploration of the hyperparameter space, ensuring that each model operates at its optimal capacity.

The models were evaluated based on several metrics, each offering unique insights into different aspects of performance:

- **Accuracy:** Measures the overall correctness of the model.
- **F1 Score:** Balances precision and recall, particularly useful in cases of class imbalance. It is calculated as follows:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

- **AUC - ROC:** Provides an aggregate measure of performance across all possible classification thresholds. It is especially effective for understanding the trade-off between the True Positive Rate and False Positive Rate:

$$\text{ROC-AUC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(u)), du \tag{5}$$

- **Balanced Accuracy:** Offers a more nuanced view of the model's performance, particularly for imbalanced datasets. It calculates the average of recall obtained on each class:

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right) \tag{6}$$

While Balanced Accuracy and AUC-ROC may yield similar insights in certain binary classification contexts, they serve different purposes. Balanced Accuracy is more straightforward and directly addresses class imbalance by averaging the performance on each class. AUC-ROC, on the other hand, provides a more comprehensive view by considering all possible thresholds, making it a robust metric for evaluating model performance across various conditions.

4 Results and Discussion

This section presents the results of the machine learning models applied to the prediction of QQQ ETF price movements. The analysis was conducted in two main stages: training standard models without hyperparameter tuning and then training the same models with fine-tuned hyperparameters, including the meta-model.

4.1 Initial Model Training

The initial phase of the study involved training a set of fundamental machine learning models. This set included the Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Logistic Regression (LR), and Random Forest, each configured with baseline parameters. These models were selected for their diverse approaches to classification problems. Class balancing techniques were employed to address the imbalance present in the dataset. Additionally, an XGBoost meta-model was trained using the predictions from these base models.

Performance Metrics: The evaluation of the models' performance involved several key metrics: accuracy, precision, recall, F1 score, and ROC AUC score. Additionally, confusion matrices and classification reports were generated for a more detailed analysis.

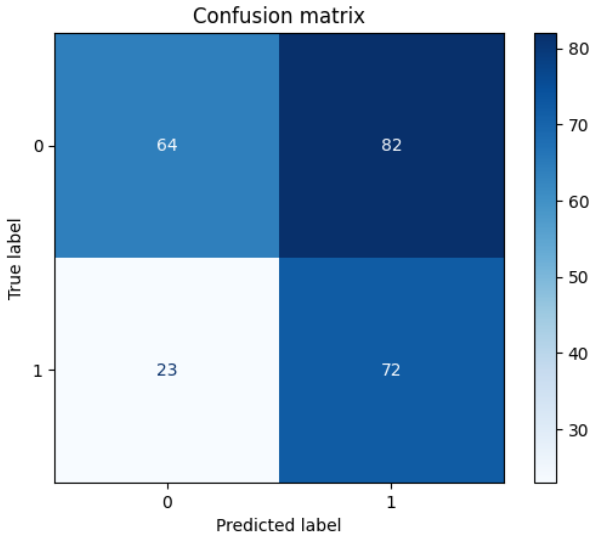

Figure 7: Confusion Matrix before Tuning

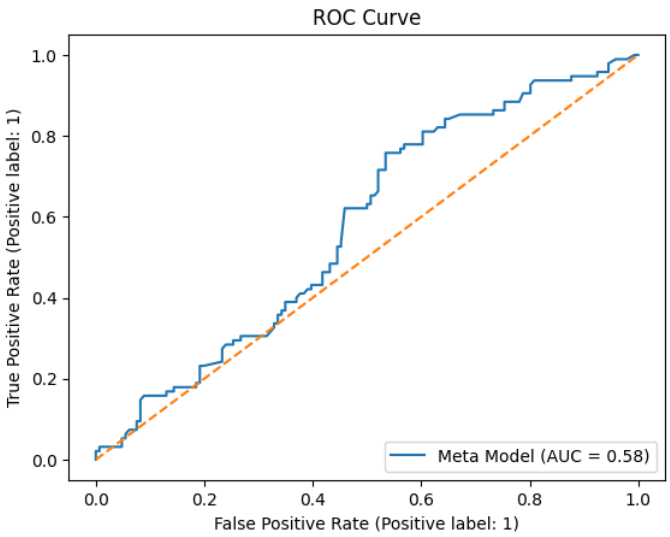

Figure 8: ROC Curve before Tuning

4.2 Tuned Models and Meta-Model

Following the initial training, each standard model and the meta-model were fine-tuned using RandomizedSearchCV. This process aimed to optimize the hyperparameters for each model, thereby enhancing their predictive capabilities.

Performance Metrics Post-Tuning: Post-tuning, the performance of the models showed an improvement, indicating the effectiveness of the hyperparameter optimization process. The same set of metrics used for the initial models were employed for evaluating the tuned models.

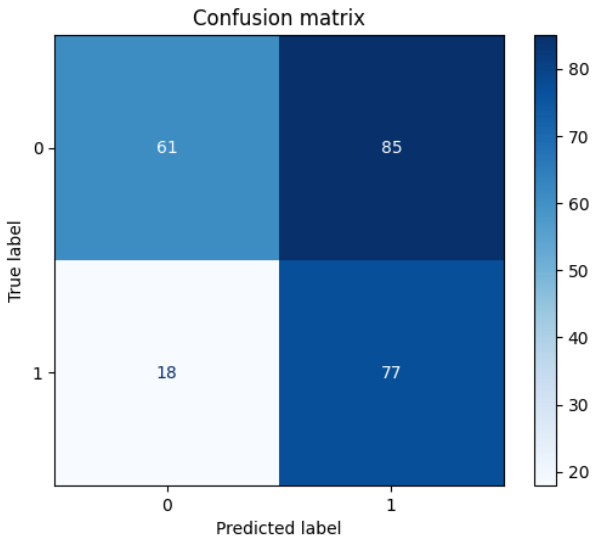

Figure 9: Confusion Matrix after Tuning

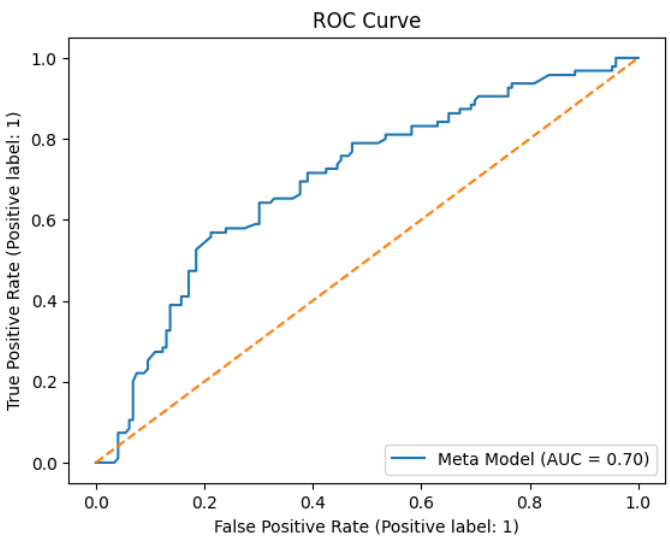

Figure 10: ROC Curve after Tuning

The fine-tuning process not only honed the performance of the individual models but also refined the predictions of the XGBoost meta-model. This enhancement was particularly evident in the improved metrics observed in the post-tuning phase. The meta-model, by integrating the strengths of each base model, demonstrated a notable increase in its ability to predict ETF price movements accurately.

Comparative Analysis: The following table presents a comparative analysis of the performance metrics before and after the tuning process. This comparison highlights the impact

of hyperparameter optimization on the effectiveness of the machine learning models.

Table 1: Comparative Performance Metrics of Models Before and After Tuning

Metric	Performance Before Tuning	Performance After Tuning
Accuracy	0.5643	0.5726
Precision	0.4675	0.4753
Recall	0.7579	0.8105
F1 Score	0.5783	0.5992
ROC AUC Score	0.5779	0.6981

The results obtained from the comparative analysis underscore the significance of model tuning in machine learning workflows, especially in complex domains like financial time series forecasting. The improvements in metrics such as ROC AUC and Recall are particularly noteworthy, as they suggest a more robust model capable of handling the intricacies and unpredictability inherent in financial data.

5 Conclusion

Utilizing the results of the model, we can preform back-testing using a trading strategy on QQQ to help illustrate its usefulness. In our example we will observe the QQQ ETF during the period from 2014-05-13 to 2015-04-23. During this time period if we were to purchase a share every day in order to dollar cost average our position, our returns would come out to approximately 11.8%. On the contrary, had we only purchased a share when our model gave us a buy signal(class of 1), we would have a return of approximately 12.1%. This gives our model a slight .3% edge on returns during this time period by simply dollar cost averaging and buying shares when the model instructs. This 0.3% may seem very small to the naked eye, however this edge can be very significant for many large investors.

This study highlights the efficacy of machine learning in investment management, particularly for ETF price prediction. Through the use of various models, including SVM, KNN, LR, and Random Forest, and their enhancement via hyperparameter tuning and an XGBoost meta-model, significant improvements in predictive accuracy were achieved. The results underscore the importance of model tuning in complex domains like financial forecasting and demonstrate the potential of ensemble methods in producing robust predictions. This research serves as a testament to the growing role of advanced analytical techniques in financial decision-making, while also reminding us of the necessity to combine these tools with traditional financial expertise.