2024/11/1

Research Report

Machine Learning for Short-Term Bitcoin Price Prediction

CHEUNG, Chit Wang

Introduction

In this section, we develop a predictive model for positive price movements in Bitcoin (BTC) using a Support Vector Machine (SVM) classifier. Bitcoin, the world's first decentralized cryptocurrency, has shown significant price volatility over the years. Predicting its price movements can be challenging but crucial for traders and investors. The historical data used in this model spans from January 1st, 2019, to the present and was downloaded using the **CCXT** library, which fetches data directly from the Binance exchange. The dataset includes Open, High, Low, Close prices, and trading volume for BTC/USDT with a 6-hour time interval.

Data Overview

The historical price data for Bitcoin was collected from January 1st, 2019, using the Binance exchange's OHLCV (Open, High, Low, Close, Volume) data. The adjusted close price of Bitcoin from January 2019 to the present is shown in Figure 1. The data was used to calculate returns and create various technical indicators. The continuous 6-hour returns are shown in Figure 2.

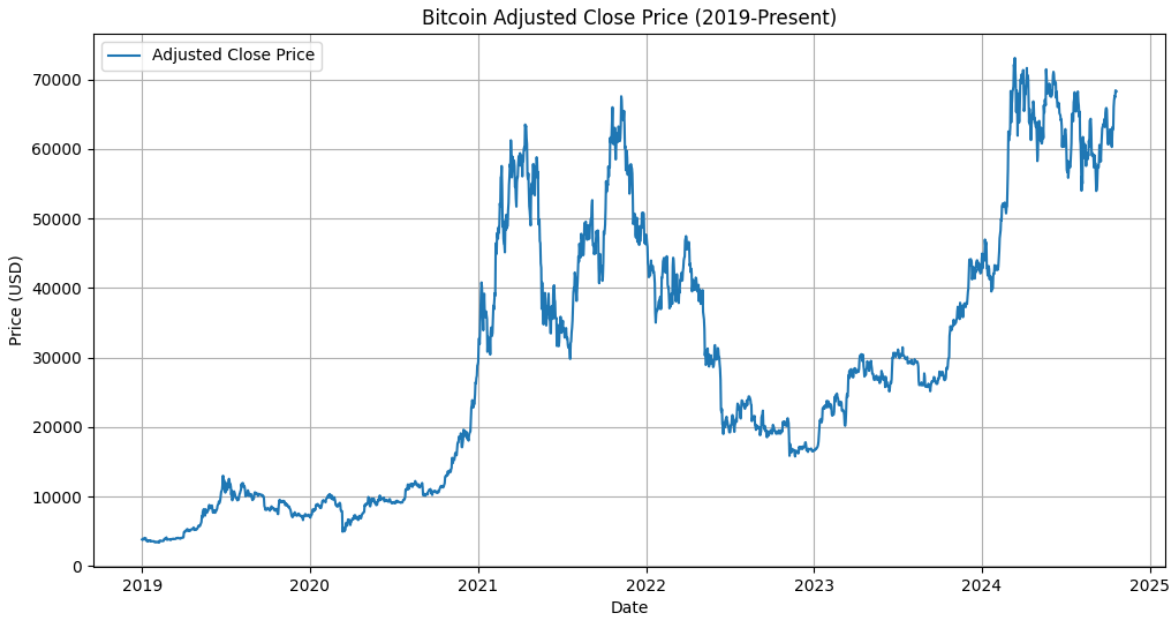

Figure 1

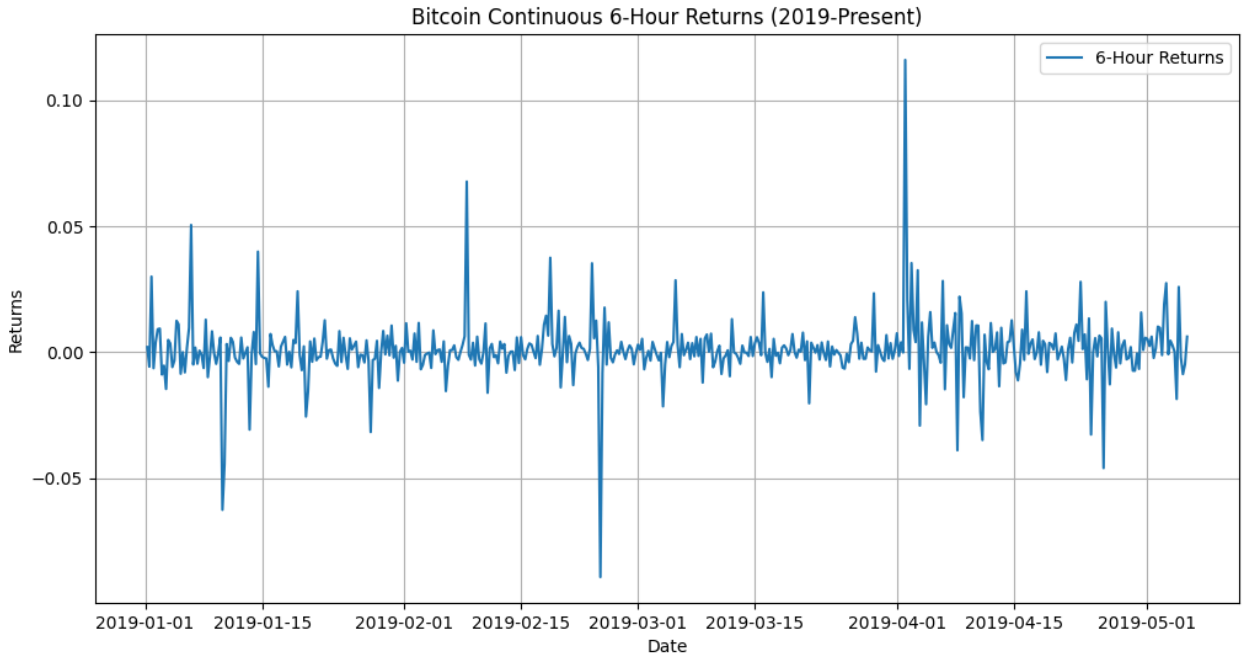

Figure 2

For labeling, we generated a binary target where:

- 1 represents a positive return (price increase),

- 0 represents a negative return (price decrease).

The threshold for labeling returns was set as 0.25%, meaning that if the return is greater than the threshold, it is labeled as 1 (positive), and vice versa. This binary classification allows the SVM model to predict whether the next price movement will be positive or negative.

Data Preprocessing and Feature Engineering

To enhance the predictive power of our model, we computed several technical indicators based on the historical price data:

1. Simple Moving Averages (SMA): 5-period and 15-period moving averages.

2. Exponential Moving Averages (EMA): 10-period and 20-period exponential moving averages.

3. Relative Strength Index (RSI): A momentum oscillator that measures the speed and change of price movements, computed over a 14-period window.

4. Moving Average Convergence Divergence (MACD): The difference between the 12-period and 26-period EMAs, along with a 9-period signal line.

5. Bollinger Bands: Upper and lower bands based on a 20-period moving average and 2 standard deviations.

6. High-Low Spread and Open-Close Spread: The difference between the high and low prices, and the open and close prices, respectively.

These features are designed to capture various aspects of price movements, such as trend direction, momentum, and volatility.

Model Setup

We used Support Vector Machine (SVM) to classify whether the next 6-hour return would be positive or negative. SVM is a powerful classification algorithm that works well for time-series data, especially when combined with the right feature engineering.

The dataset was split into training (80%) and testing (20%) sets, with data scaling applied using StandardScaler to standardize the features. Given the time-series nature of the data, we avoided shuffling to maintain the temporal order.

Hyperparameter Tuning

To optimize the performance of the SVM model, we used GridSearchCV to find the best hyperparameters. The SVM classifier's key parameters, including the regularization parameter `C`, the kernel type, and the kernel coefficient `gamma`, were tuned using a grid search over the following ranges:

- C: [0.1, 1, 10, 100, 300]

- Kernel: ['linear', 'rbf']

- Gamma: ['scale', 'auto']

We used TimeSeriesSplit with 5 splits to ensure that the model was validated on unseen data during cross-validation.

Results and Model Performance

After training the model and selecting the best parameters, the SVM model was evaluated on the test set. The best performing parameters were:

- C: 10

- Kernel: linear

- Gamma: scale

The model achieved the following results:

- Training Accuracy: 62.967%

- Test Accuracy: 64.519%

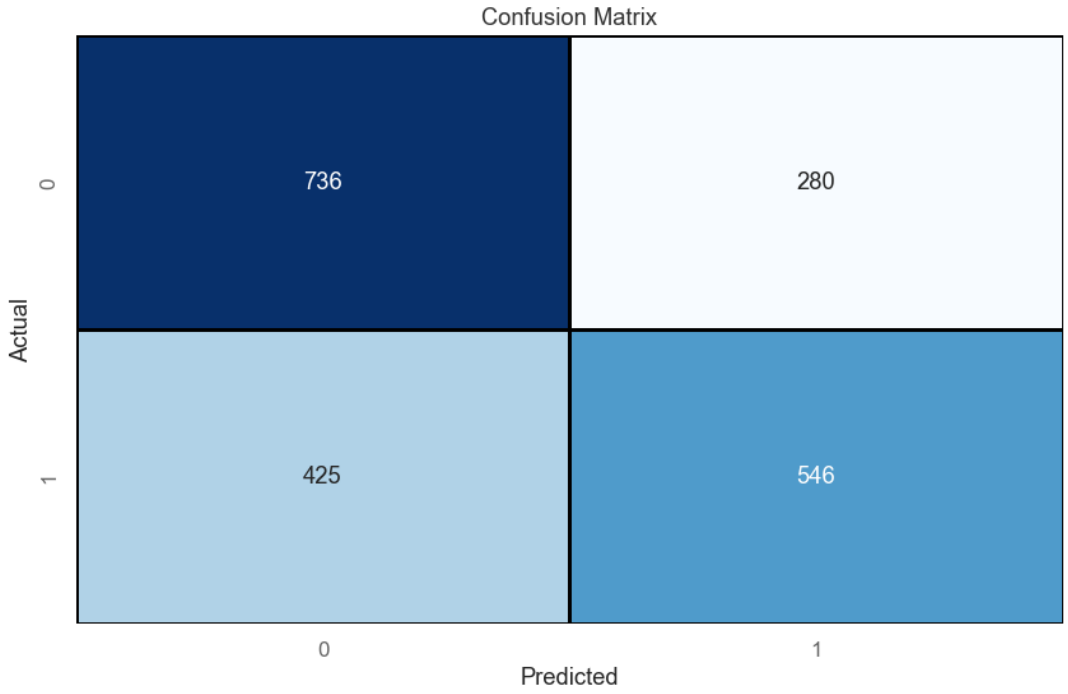

Figure 3

The confusion matrix in Figure 3 shows the distribution of true positives, true negatives, false positives, and false negatives. In total, the model performed well in distinguishing positive and negative price movements, with a slight bias toward predicting positive returns.

Classification Report provides precision, recall, and F1-score for both classes. The model achieved a precision of 66% for predicting positive returns (label 1), meaning it was fairly reliable at predicting when the price would increase. However, the recall for positive returns was slightly lower, indicating that the model missed some positive movements.

Classification Report:

	precision	recall	f1-score	support
0	0.63	0.72	0.68	1016
1	0.66	0.56	0.61	971
accuracy			0.65	1987
macro avg	0.65	0.64	0.64	1987
weighted avg	0.65	0.65	0.64	1987

ROC-AUC Analysis

To further evaluate the model, we computed the Area Under the Receiver Operating Characteristic (ROC) curve (AUC-ROC). The ROC curve, shown in Figure 4, plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The AUC score was 0.7037, indicating that the model has a good ability to discriminate between positive and negative price movements.

Figure 4

Discussion

The SVM model demonstrated reasonable predictive power for forecasting short-term Bitcoin price movements. With an accuracy of approximately 67% on the test set and an AUC-ROC score of 0.728, the model can serve as a useful tool for traders looking to anticipate short-term price changes. However, there are areas for improvement, such as increasing the recall for positive returns to reduce missed opportunities.

Further Exploration: XGBoost Classifier

Using XGBoost for comparison with SVM in your trading strategy analysis serves several purposes. It allows for assessing performance differences, particularly in capturing complex patterns and handling non-linear interactions, which are crucial in dynamic market conditions. XGBoost's feature importance insights help identify influential factors, guiding data analysis and feature engineering. Additionally, it evaluates error handling, potentially reducing costly prediction mistakes. The model's generalization capabilities to unseen data are crucial for adapting to market shifts. Lastly, XGBoost's extensive hyperparameter tuning provides opportunities for optimization, enhancing overall strategy effectiveness.

Hyperparameter Tuning

Parameter Grid:

- learning_rate: [0.05, 0.1, 0.15, 0.2]
- max_depth: [3, 4, 5, 6]
- min_child_weight: [1, 3, 5]
- gamma: [0.0, 0.1, 0.2]
- colsample_bytree: [0.3, 0.5, 0.7]

Cross-Validation: Utilized TimeSeriesSplit with 5 splits to maintain data sequence integrity.

Search Method: RandomizedSearchCV with 50 iterations to efficiently explore the hyperparameter space.

Evaluation Metrics

Best Parameters: Identified and printed after fitting the model.

Accuracy: Computed on the test set.

Classification Report: Displays precision, recall, and F1-score.

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
Best Parameters: {'min_child_weight': 3, 'max_depth': 3, 'learning_rate': 0.05, 'gamma': 0.0, 'colsample_bytree': 0.7}
Test Accuracy: 0.5963764469048818

Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.64      0.62      1016
           1       0.59      0.55      0.57       971

    accuracy                           0.60      1987
   macro avg       0.60      0.60      0.60      1987
weighted avg       0.60      0.60      0.60      1987
```

Confusion Matrix: Visualized using a heatmap to show prediction distribution.

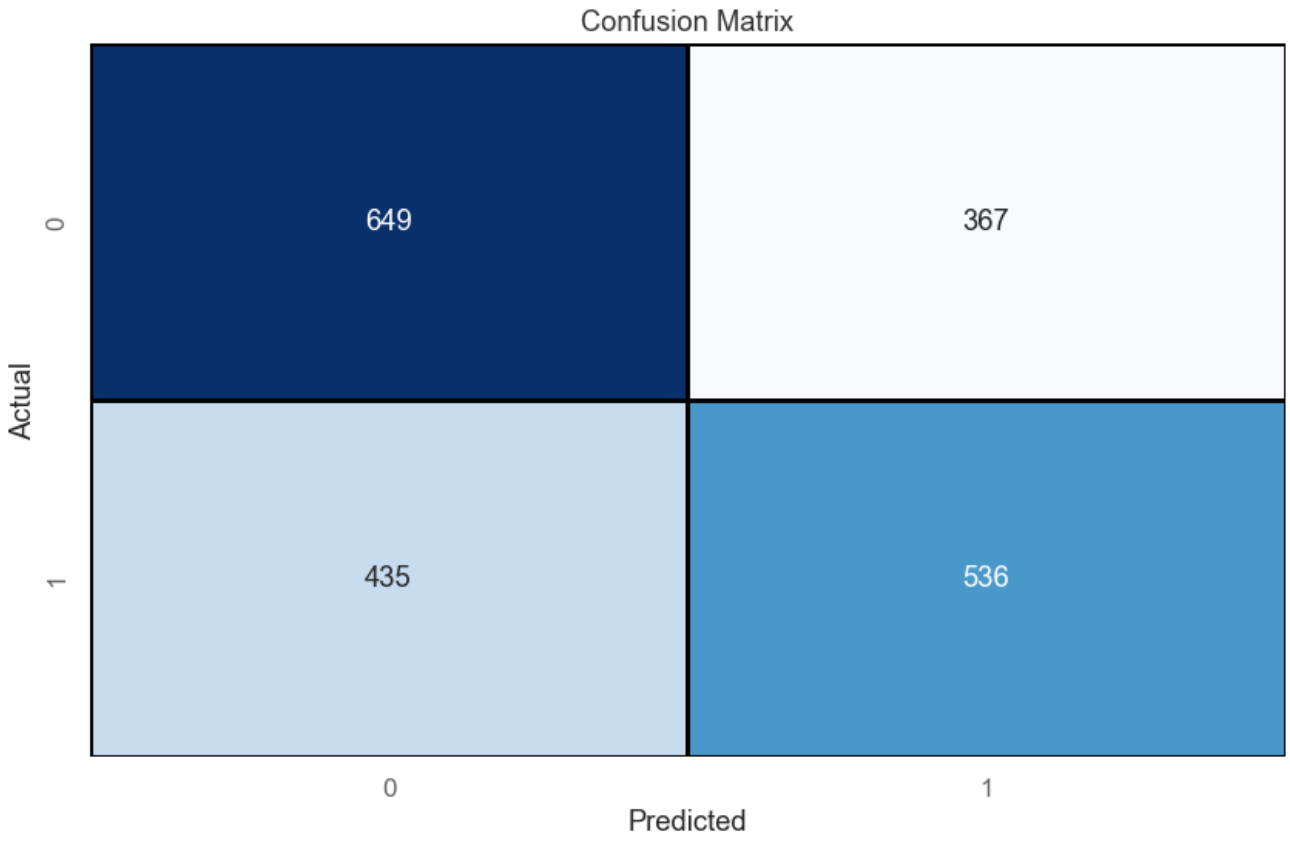

Figure 5

ROC Curve and AUC: Plotted to assess the model's discrimination ability, with AUC reported.

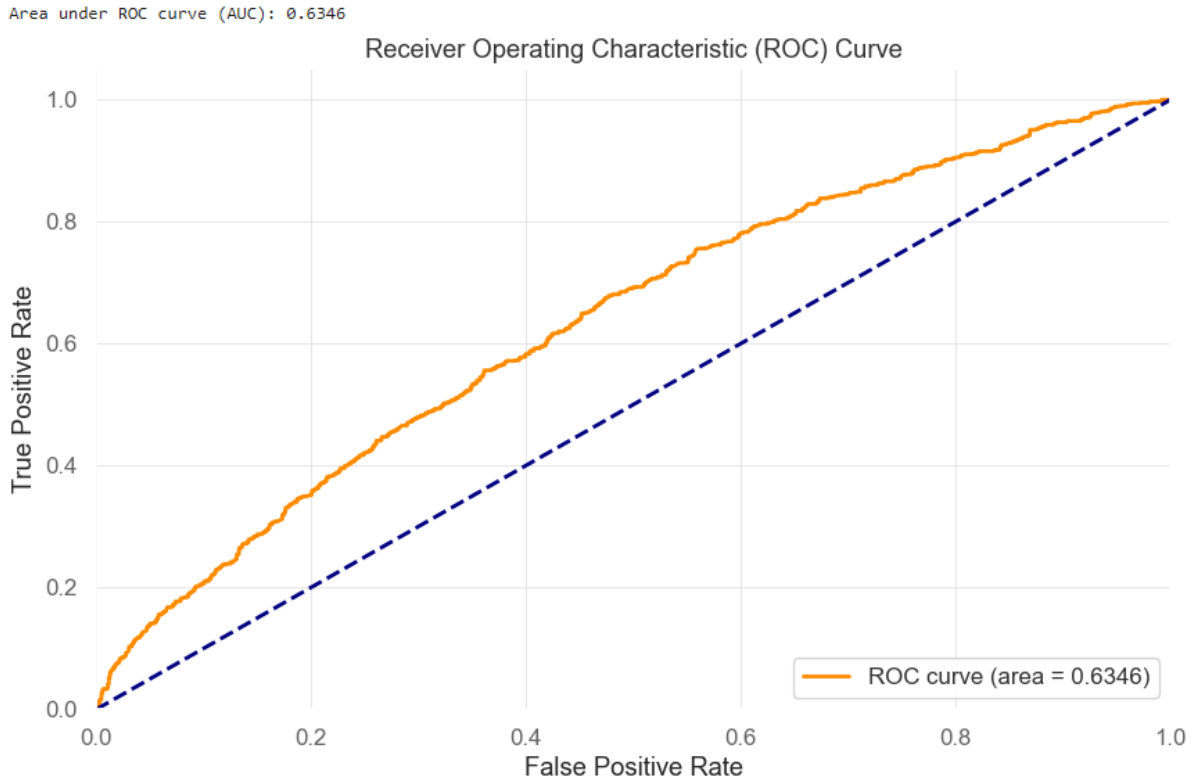

Figure 6

Feature Importance: Visualized using XGBoost's built-in importance plot.

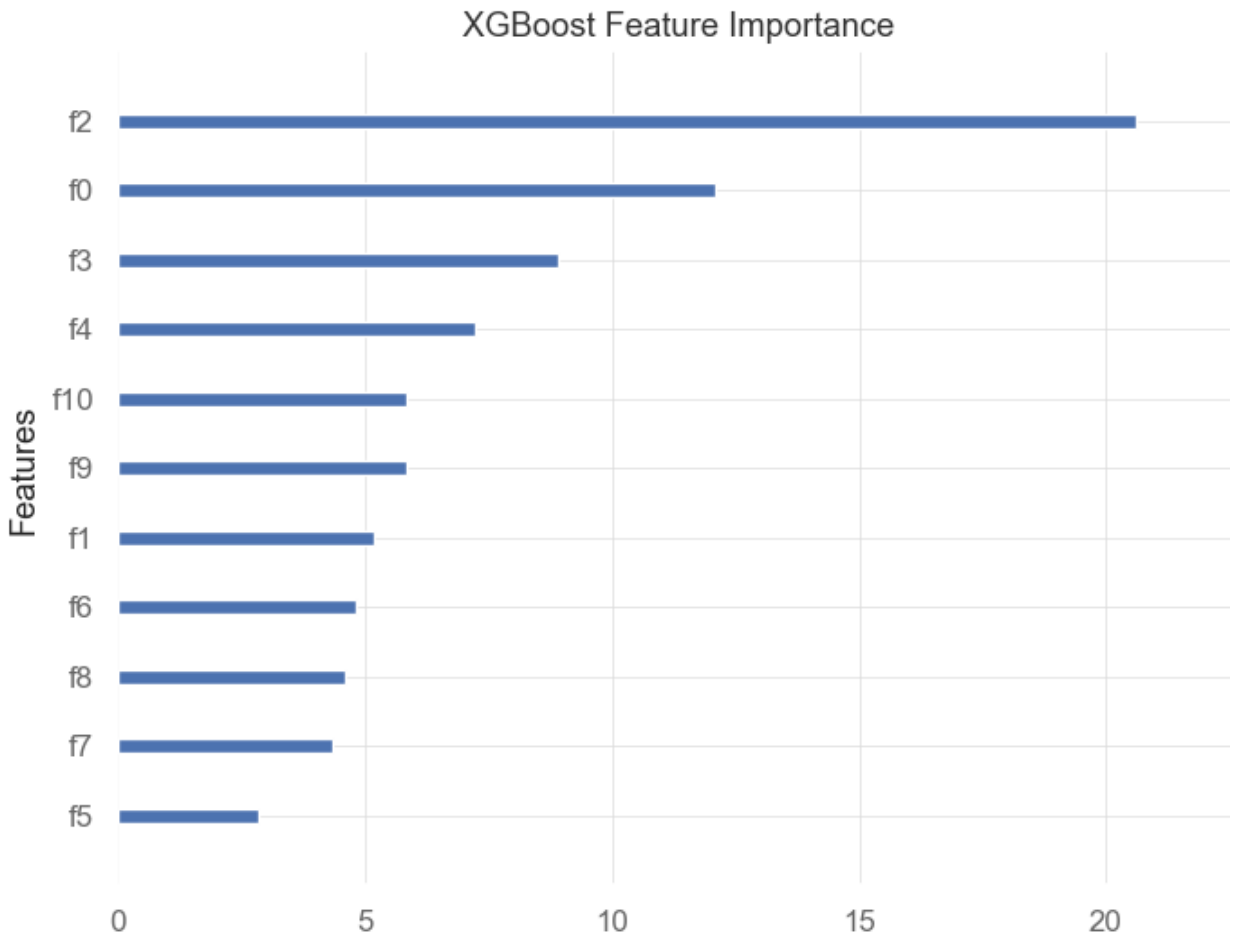

Figure 7

SHAP Analysis: Provides insights into feature impact using SHAP values and summary plots.

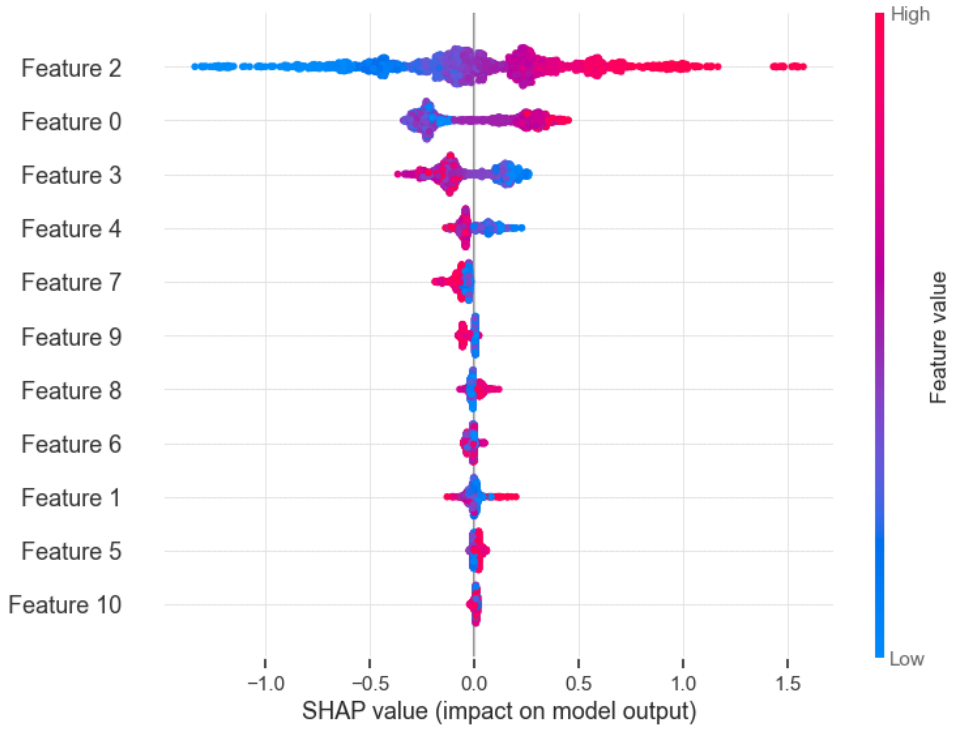

Figure 8

Summary of Results

1. **Hyperparameter Tuning**:
    - Best Parameters: min_child_weight = 3, max_depth = 3, learning_rate = 0.05, gamma = 0.0, colsample_bytree = 0.7.
    - Achieved a test accuracy of approximately 59.64%.
2. **Classification Metrics**:
    - Precision, recall, and F1-score are around 0.60, indicating balanced but modest predictive performance.
3. **Confusion Matrix**:
    - Correctly predicted class 0: 649 instances.
    - Correctly predicted class 1: 536 instances.
    - Misclassifications are notable, with 367 false positives and 435 false negatives.
4. **ROC Curve**:
    - Area under the ROC curve (AUC) is 0.6346, suggesting the model is better than random guessing but has room for improvement.

Feature Importance and Impact

1. **Feature Importance**:
    - Feature 2 is the most influential, followed by Feature 0 and Feature 3.
    - Suggests focus on these features for further model improvement or analysis.
2. **SHAP Values**:
    - SHAP values indicate how each feature impacts the model's predictions.
    - Feature 2 has the most significant impact, with both positive and negative contributions to predictions.

Trading Strategy Backtesting

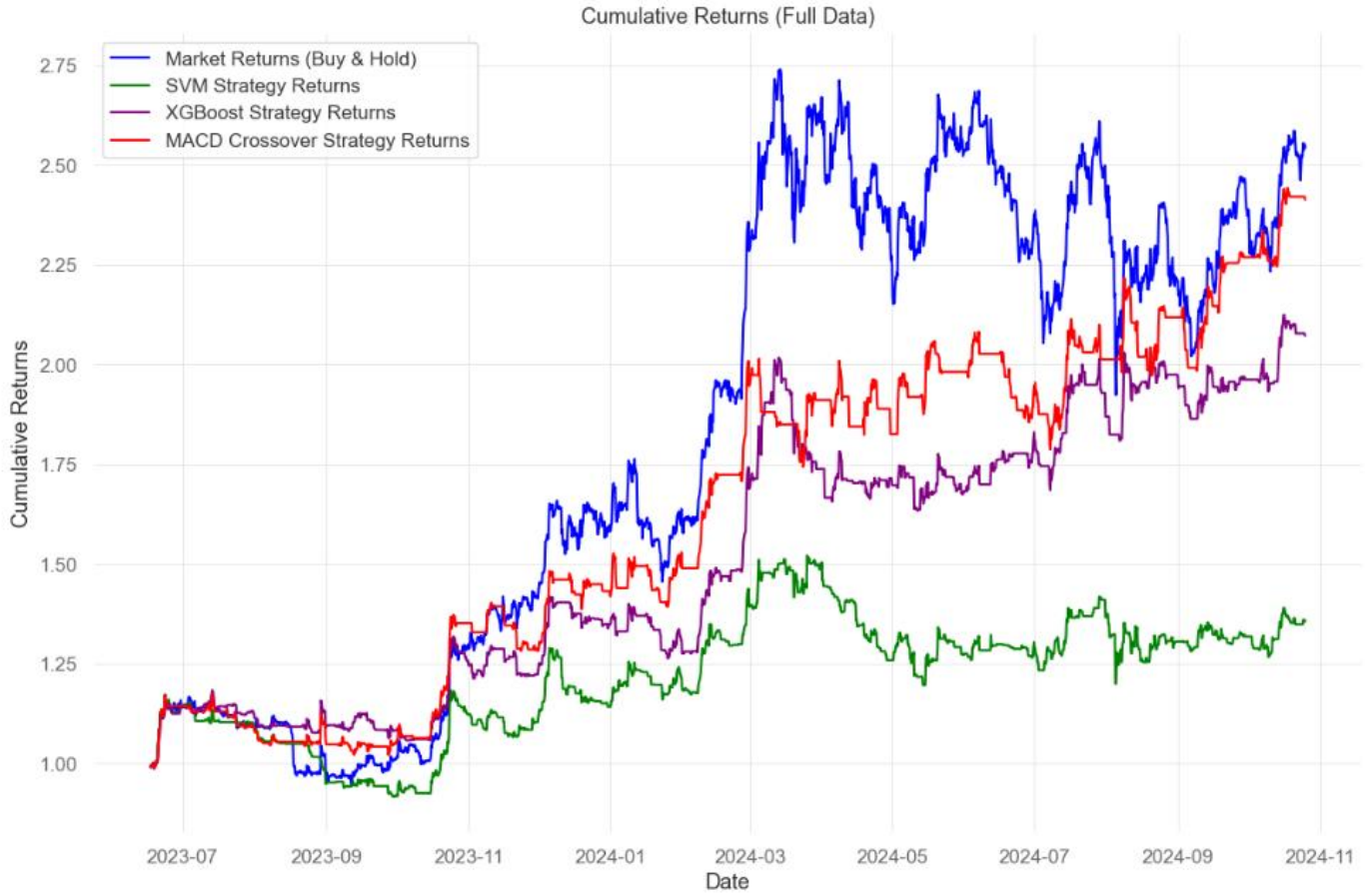

Figure 9

Analysis:

This graph presents the cumulative returns of various trading strategies.

- **Market Returns (Buy & Hold)**: Represented by the blue line, this strategy shows significant volatility and the highest returns in the longer time frames, indicating strong market performance during the period.
- **SVM Strategy Returns**: Shown in green, this strategy exhibits stable but moderate growth, underperforming compared to other strategies, especially over longer periods.
- **XGBoost Strategy Returns**: Depicted in purple, this strategy consistently outperforms the SVM strategy, indicating better adaptability and pattern recognition over time.
- **MACD Crossover Strategy Returns**: Illustrated in red, this strategy shows steady performance, often outperforming the SVM strategy but slightly trailing the XGBoost strategy in longer periods.

Overall, the XGBoost strategy demonstrates superior performance, particularly for longer durations, highlighting its effectiveness in capturing market trends and generating higher returns.

Discussion

The underperformance of the SVM strategy compared to XGBoost stems from several critical factors that affect their ability to adapt to the complexities of financial data.

Firstly, the SVM classifier's use of a linear kernel imposes significant limitations in scenarios where the underlying relationships in the data are non-linear, as is often the case in financial markets. The linear kernel is effective at creating sharp, well-defined decision boundaries, which can lead to high training accuracy, especially in clean, well-structured datasets. However, financial data, particularly price movements, is frequently characterized by non-linearities, volatility, and noise—none of which are easily captured by a linear model. As a result, SVM with a linear kernel tends to overfit on the training data, as it tries to fit every pattern, including noise, with a linear boundary. This overfitting reduces its generalization capability when applied to new, unseen data. Furthermore, while you applied hyperparameter tuning, the scope was somewhat limited, particularly with respect to kernel selection. A linear kernel is inherently restrictive in its ability to model complex financial patterns, and expanding the hyperparameter search to include non-linear kernels (such as Radial Basis Function (RBF) or polynomial kernels) could allow the model to capture more intricate relationships between features.

Additionally, the feature engineering process, while comprehensive, may also contribute to SVM's limitations. Financial indicators like Moving Averages (MA), Relative Strength Index (RSI), and MACD inherently exhibit non-linear correlations with future price movements. The use of a linear kernel in SVM cannot fully exploit these non-linear dependencies, which constrains the model's ability to predict accurately. Moreover, the class imbalance—where one class (e.g., negative returns) may dominate the dataset—can further degrade SVM's performance. Without adjusting for this imbalance (e.g., using class weights or resampling techniques), the model may favor the majority class, leading to biased predictions and lower recall for the minority class (e.g., positive returns).

In stark contrast, XGBoost is naturally better suited for handling the complexities of financial data due to its ensemble of decision trees. By building a collection of weak learners (trees), XGBoost is able to capture non-linear relationships and interactions between features that a linear SVM would miss. Each tree in the ensemble focuses on different parts of the data, enabling the model to adapt to the complex, non-linear patterns in financial time series, such as lagged effects, trend reversals, and volatility spikes. This is particularly important when working with technical indicators like

Bollinger Bands or MACD, which are designed to capture dynamic shifts in market momentum that a linear model often overlooks.

Moreover, XGBoost benefits from extensive hyperparameter tuning through RandomizedSearchCV, which optimizes critical parameters such as learning rate, tree depth, min_child_weight, and colsample_bytree. This flexibility allows XGBoost to strike a balance between bias and variance, ensuring that the model generalizes well to unseen data without overfitting to the noise in the training set. Additionally, XGBoost incorporates built-in regularization techniques, such as L1 (lasso) and L2 (ridge) penalties, which further help to control overfitting by discouraging overly complex models that may fit spurious patterns in the data. This regularization is particularly useful in financial markets, where market noise and random price fluctuations can easily mislead models that lack proper safeguards against overfitting.

Another key strength of XGBoost is its robustness to class imbalance and its ability to handle skewed datasets more effectively. Unlike SVM, which may require explicit adjustments (such as resampling or cost-sensitive learning), XGBoost can implicitly manage imbalanced data through its objective function and boosting framework. By focusing on hard-to-predict examples in each boosting round, XGBoost naturally adjusts to cases where one class is underrepresented, ensuring that both classes (positive and negative returns) are adequately captured in the final model predictions.

Lastly, XGBoost's ability to provide feature importance metrics (through techniques like SHAP values) allows for deeper insights into the driving forces behind the predictions. This can be particularly valuable in financial modeling, where understanding which factors (e.g., which technical indicators or market conditions) are most influential in predicting future price movements can help refine the model further and guide feature selection in future iterations. SVM, on the other hand, does not provide this level of interpretability, making it harder to diagnose where the model might be going wrong or how to improve it.

In summary, SVM's reliance on a linear kernel and limited adaptability to non-linear patterns, combined with its susceptibility to overfitting and challenges in handling class imbalance, contribute to its underperformance in this financial context. In contrast, XGBoost's ensemble nature, flexibility in capturing non-linear relationships, extensive hyperparameter tuning, and built-in regularization make it a more suitable model for predicting financial price movements.

While SVM can be improved through the use of non-linear kernels and better handling of class imbalance, XGBoost's inherent strengths make it a more robust and adaptable choice for this type of problem, especially in real-world financial markets characterized by noise and volatility.

Figures

Reference

1. Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20(3), 273-297.

2. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.