IMPERIAL
BUSINESS SCHOOL

Systematic Trading Strategies with Machine Learning Algorithms Coursework

Mai Luo (06004395)

July 14, 2025

Contents

List of Figures

List of Tables

Part 1: Primary Model Creation for Crypto Data using Trend Scanning Labeling

The Bitcoin price dataset consists of hourly data from 1 January 2018 to 20 March 2024. To ensure the model generalises well to unseen data and avoids forward-looking bias, the dataset will be split into three non-overlapping periods:

- **Test Period (2018-2020):** This period is used to fit the model, including feature engineering, regime modeling, and parameter estimation.
- **Embargo Period (50 hours):** A purging window between training and test data. This is critical for avoiding information leakage, especially when creating labels using looking-ahead windows. Any overlap in look-ahead windows across training and test boundaries would introduce peeking into future data.
- **Test Period (2021):** Used strictly for out-of-sample evaluation. The model is not trained or tuned using this period, making it a fair benchmark for real-world performance.

Feature Engineering

Since the goal of this task is to build a primary model to predict price movement, we want to find features that could indicate whether the price is going to go up or down in the next hour. The following features will be considered:

- **Momentum RSI:** Momentum features give a gauge of the market's speed. RSI indicates overbought/oversold conditions over 14 periods. A high RSI suggests overbought conditions; a low RSI indicates oversold conditions.
- **Volatility (Average True Range):** Captures the average true range of price movement over the past 14 hours. Reflects market volatility and helps the model distinguish between calm and turbulent periods, regardless of direction.
- **Simple Moving Average Crossover:** Measures the difference between the 10-hour and 50-hour simple moving averages. Serves as a momentum and trend indicator; positive values suggest bullish momentum, and negative values indicate bearish momentum.
- **Serial Correlation / Lagged Returns:** Lagged returns (e.g., from k hours ago) capture short-term memory in price movements. Helps detect autocorrelation or short-term trends that may persist or reverse.
- **24-hour Volume Mean:** Captures the average trading volume over the past 24 hours. Acts as a proxy for liquidity; higher values indicate more active trading and potentially stronger conviction behind price moves.

- **On-Balance Volume:** Accumulates volume based on price direction—rises when price increases and falls when price decreases. Reflects volume flow and identifies whether buyers or sellers are driving recent trends.
- **Volume Change:** Measures the hour-to-hour difference in trading volume. Highlights sudden spikes or drops in activity that may precede or confirm price movements.
- **Latent Regime Feature (HMM):** Log-returns and 24-hour volatility will serve as baseline features for the Hidden Markov Model (HMM). The state probabilities refer to the filtered probabilities of being in regime k. These features provides macro context for each observation, allowing the model to condition its predictions on inferred market states.

A Gaussian Hidden Markov Model (HMM) is fitted to log returns and 24-hour volatility, as mentioned above, to infer latent market regimes. Models with 2 to 5 hidden states were evaluated using the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC), both of which measure model fit while penalising complexity. The results are shown in

State	**AIC**	**BIC**	**Convergence**
2	-430894.7698	-430788.4750	No
3	-449752.0345	-449563.9744	No
4	-461802.8133	-461516.6350	No
5	-463350.1113	-462949.4616	No

Table 1: Model Selection Criteria for Different States

Each model is given a maximum of 2,000,000 iterations to converge; however, none reached convergence within this limit, as seen in Table 1, suggesting instability in the Expectation-Maximization (EM) algorithm and some uncertainty in the final regime assignments. It is also notable that the AIC and BIC decreases as the number of states increases, however, 4 states would most likely be enough as the marginal gain on AIC and BIC is limited after this state.

The inferred regime structure is visualised in Figure 1 below, where each colour-coded segment represents the most likely state at each point in time. These latent regimes offer insight into structural shifts in market conditions.

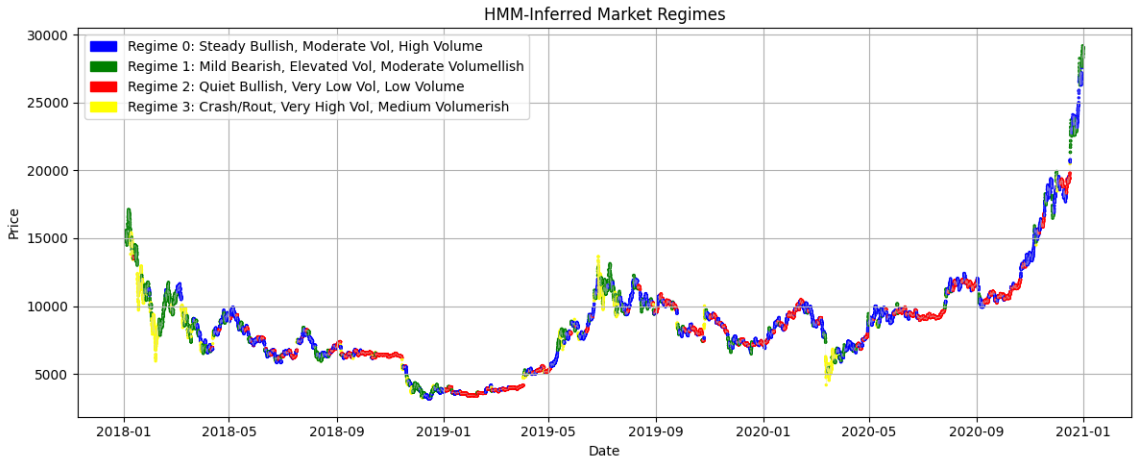


Figure 1: HMM- Inferred Market Regimes

To fully understand the characteristics of each regime, we observe the log returns, volatility, and mean volume per regime, as seen in Figure 2 below.

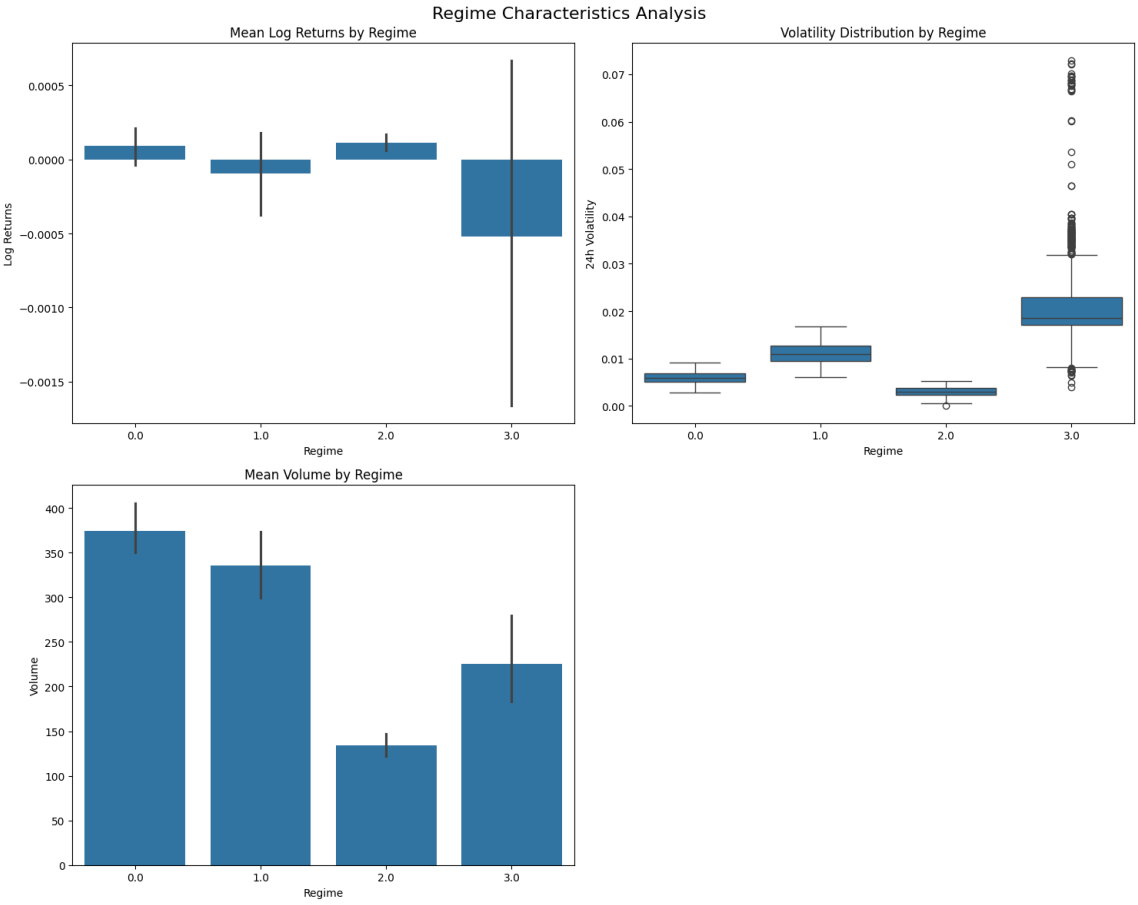


Figure 2: Regime Information

After analysing this, the following can be inferred:

- **Regime 0:** Appears to correspond to a steady bullish phase, characterised by moderately positive returns, mid-tier volatility, and the highest trading volume. This potentially reflects a sustained upward trend driven by steady market participation.
- **Regime 1:** Associated with mildly bearish conditions, where returns are slightly negative and volatility is elevated. This may reflect corrective pull-backs or market indecision.
- **Regime 2:** Captures low-volatility, low-volume periods with positive drift, likely representing passive or quiet bullish regimes.
- **Regime 3:** Displays the highest volatility and the worst returns, suggesting periods of panic selling or broader market drawdowns.

To gain further understanding, we analyse the correlation between the features, as well as the behaviour of all of the features over time. This is shown in Figures 3 and 4 below. The correlation heatmap shows that most features are weakly correlated, reducing concerns around multicollinearity. A few pairs, such as `volume_mean_24h` and `obv`, are highly correlated but were both retained due to their distinct interpretations in capturing volume dynamics. The HMM regime probabilities were reduced to a single `hmm_regime_argmax` feature to avoid redundancy. Based on this analysis, we selected the following features for modeling: `momentum_rsi`, `vol_atr_14h`, `sma_crossover`, `ret_lag_1`, `ret_lag_2`, `ret_lag_6`, `volume_mean_24h`, `obv`, `volume_change`, `volatility_24h`, and `hmm_regime_argmax`.

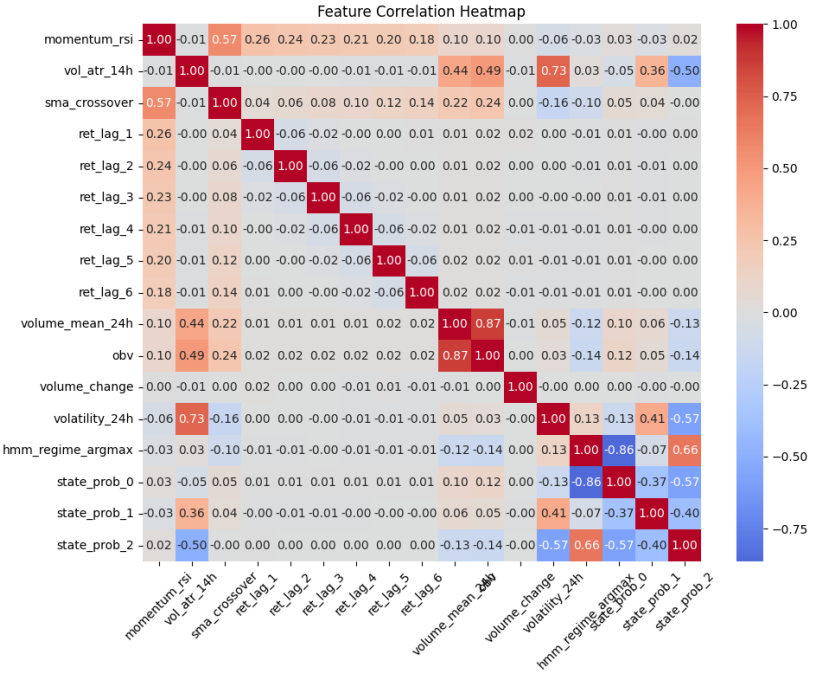


Figure 3: Correlation between features

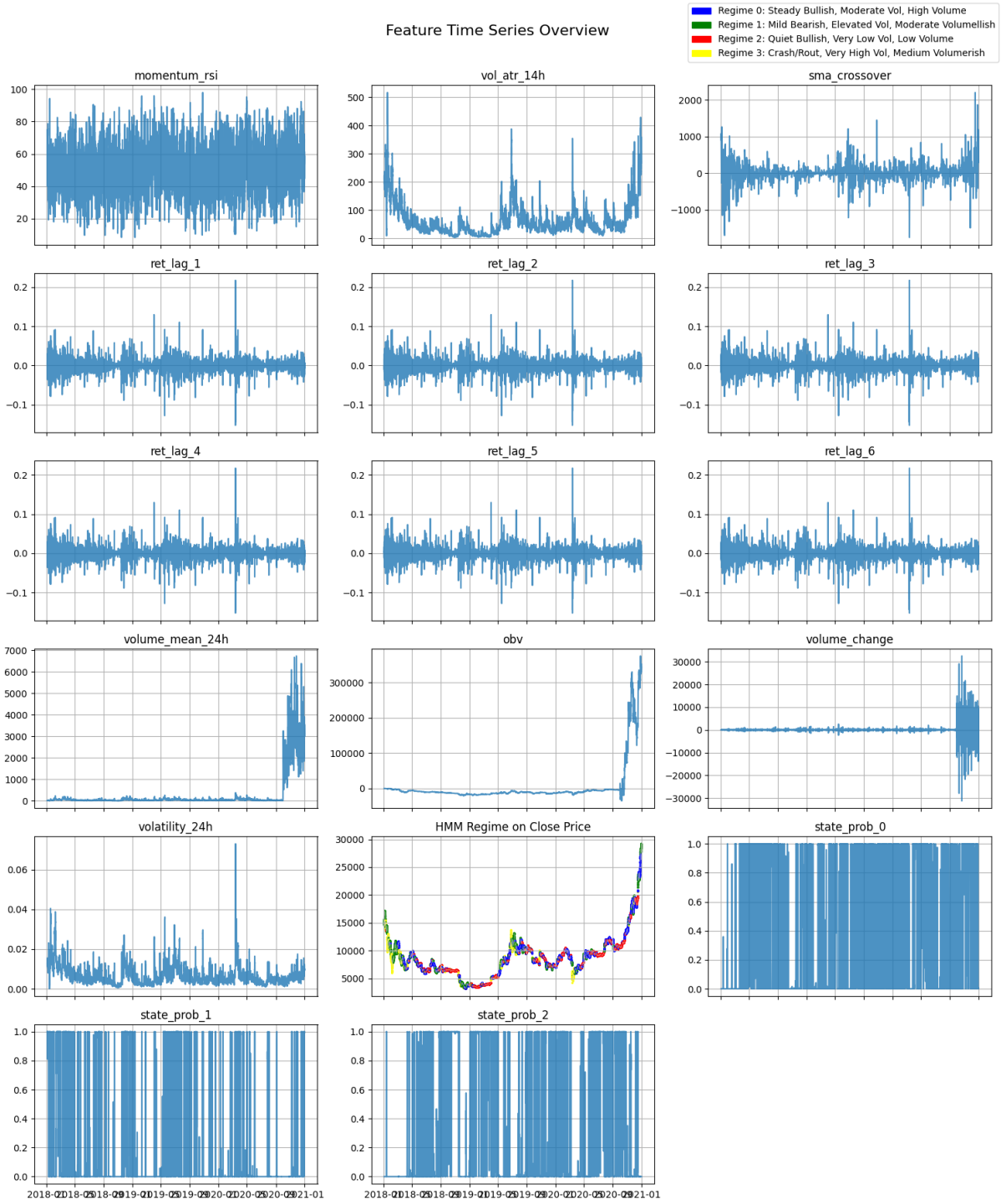


Figure 4: Feature behaviour over time

Creating Labels

To label directional trends in the price data, we applied the trend-scanning method. At each timestamp, this technique fits linear regressions across a range of forward-looking windows and selects the one with the highest absolute t-statistic for the slope. The sign of this t-statistic determines the label: $+1$ for bullish trends and -1 for bearish trends. We scanned horizons between 5 and 25 hours, allowing the model to adapt to varying trend lengths.

Figure 5: Trend Scanning Labels

Figure 5 shows the resulting labels overlaid on price. The majority of t-statistics in the training period exceed common significance thresholds, as seen in Table 2, supporting the statistical relevance of the assigned labels.

Table 2: Sample of Trend-Scanning Output on Training Data

Date	**t1**	**tVal**	**Bin**	**Window Size (hours)**
2018-01-01 01:00	2018-01-01 08:00	3.0679	1	7
2018-01-01 02:00	2018-01-01 07:00	4.1107	1	5
2018-01-01 03:00	2018-01-01 19:00	-3.6500	-1	16
⋮	⋮	⋮	⋮	⋮
2020-12-31 21:00	2021-01-01 17:00	3.3186	1	20
2020-12-31 22:00	2021-01-01 17:00	2.8934	1	19
2020-12-31 23:00	2021-01-01 17:00	2.4107	1	18

Model Development

We explore and compare multiple models to determine which performs best for directional trend prediction. Both expanding and rolling window cross-validation strategies

are considered to reflect realistic trading scenarios where retraining occurs over time.

For the expanding window approach, we use `TimeSeriesSplit` from `scikit-learn` to generate time-respecting folds. Each training set spans from the beginning of the dataset up to a certain point, with the validation set comprising the subsequent period. To prevent look-ahead bias due to our 24-hour future labelling window (used in trend scanning), we implement a 24-hour embargo by setting `gap=24`. This ensures that no label leakage contaminates the training process. In this configuration, the training set grows with each fold.

For the rolling window approach the training set has a fixed size and moves forward in time across the dataset. Specifically, we use a training window that spans 50% of the full training period, ensuring that each fold learns from a consistent amount of historical data. As with the expanding strategy, we enforce a 24-hour embargo between the training and validation periods by setting `gap=24`. This helps prevent any leakage from the forward-looking labelling process. The rolling window method is particularly useful for evaluating how model performance evolves in changing market conditions, as it better mimics a live retraining regime with limited historical memory.

From each strategy, the best-performing models are stored as `model_expanding` and `model_rolling` for final evaluation and comparison.

Random Forest Classifier

We first apply a Random Forest classifier, tuning hyperparameters via `GridSearchCV`. The parameters tuned include: number of trees, maximum tree depth, minimum samples required to split an internal node, maximum number of features considered for a split, and minimum samples required at a leaf node.

The F1-score is used as the scoring metric, as it balances precision and recall—particularly important in our binary classification setup. All cross-validation strategies employ the same 24-hour embargo to mitigate label leakage.

The best results from each validation strategy are summarised in Table 3. The model trained using the rolling window strategy slightly outperformed the expanding window on all metrics, achieving an F1-score of 0.540 compared to 0.527. Precision, recall, and ROC-AUC were also marginally higher under the rolling setup, suggesting that using a fixed-length historical window may better adapt to evolving market dynamics. However, overall performance remained modest, with both setups showing limited predictive power, as reflected in ROC-AUC values close to 0.5.

Table 3: Random Forest Cross-Validation Results

Metric	Expanding Window	Rolling Window

5 Best Hyperparameters	`max_depth=5`
	`max_features='sqrt'`
	`min_samples_leaf=1`
	`min_samples_split=2`
	`n_estimators=200`
		`max_depth=10`
		`max_features='sqrt'`
		`min_samples_leaf=1`
		`min_samples_split=5`
		`n_estimators=200`

F1 Score	0.527	0.540
Accuracy	0.487	0.507
Precision	0.541	0.557
Recall	0.604	0.588
ROC-AUC	0.510	0.523

XGBoost Classifier

We then train an `XGBClassifier` from the `xgboost` library using a similar cross-validation strategy. The model is configured for binary logistic classification with the `logloss` evaluation metric. Labels are encoded as 0 for bearish (-1) and 1 for bullish ($+1$) trends to match the model's binary requirements.

The cross-validation results in Table 4 show that the rolling window strategy again outperforms the expanding window across all metrics. The best F1-score increases from 0.524 (expanding) to 0.536 (rolling), with similar improvements in accuracy (0.486 to 0.504), precision (0.533 to 0.542), and ROC-AUC (0.504 to 0.513). Recall remains constant at 0.578. These gains, while modest, suggest that the rolling window better captures time-varying structure in the data, consistent with the findings from the Random Forest model.

Table 4: XGBoost Cross-Validation Results

Metric	Expanding Window	Rolling Window

5 Best Hyperparameters	`colsample_bytree=0.8`
	`learning_rate=0.1`
	`max_depth=3`
	`n_estimators=100`
	`subsample=0.8`
		`colsample_bytree=0.8`
		`learning_rate=0.1`
		`max_depth=5`
		`n_estimators=100`
		`subsample=0.8`

F1 Score	0.524	0.536
Accuracy	0.486	0.504
Precision	0.533	0.542
Recall	0.578	0.578
ROC-AUC	0.504	0.513

Neural Network Classifier

We implement a feedforward neural network using TensorFlow/Keras with two hidden layers of 128 and 64 units. Hyperparameters such as dropout rate, learning rate, and batch size are tuned using grid search under both expanding and rolling cross-validation strategies. Features are standardised prior to training to improve stability.

The cross-validation results in Table 5 show that the rolling window marginally outperforms the expanding setup, with an F1-score of 0.4950 versus 0.4929. Other metrics remain nearly identical across both strategies, with accuracy at 0.5110 and ROC-AUC at 0.5230. These results indicate that the neural network captures some signal, but falls short of the performance achieved by tree-based models. Further improvements may require deeper architectures or sequence-aware models.

Table 5: Neural Network Cross-Validation Results

Metric	**Expanding Window**	**Rolling Window**
F1 Score	0.4929	0.4950
Accuracy	0.5110	0.5110
Precision	0.5440	0.5440
Recall	0.5570	0.5570
ROC-AUC	0.5230	0.5230

Feature Importance Analysis

To interpret the decision logic of the tree-based models, we evaluate feature importance using two complementary methods: Mean Decrease in Impurity (MDI) and Permutation Feature Importance (PFI). MDI is an in-sample measure and reflects the average reduction in Gini impurity or entropy when a feature is used for splitting, and is directly extracted from the trained model. However, MDI can be biased toward features with more categories or higher cardinality. To mitigate this and validate the robustness of the rankings, we also compute PFI, an out-of-sample measure, which considers the decrease in model performance when a feature's values are randomly shuffled. This allows us to assess each feature's true marginal contribution to predictive performance. Both analyses are visualised across models to identify consistently influential predictors and regime-specific feature behaviours.

In the MDI results in Figure 6, `volume_mean_24h` and `obv` consistently emerge as the top features, underscoring the strong influence of volume-based signals on tree-splitting decisions. Features such as `momentum_rsi`, `volatility_24h`, and `sma_crossover` also appear regularly in the top ranks, reflecting the predictive value of trend and volatility indicators. Notably, `hmm_regime_argmax` gains more prominence in XGBoost models, especially under the rolling window configuration, suggesting that regime-aware inputs are better leveraged by gradient-boosted trees.

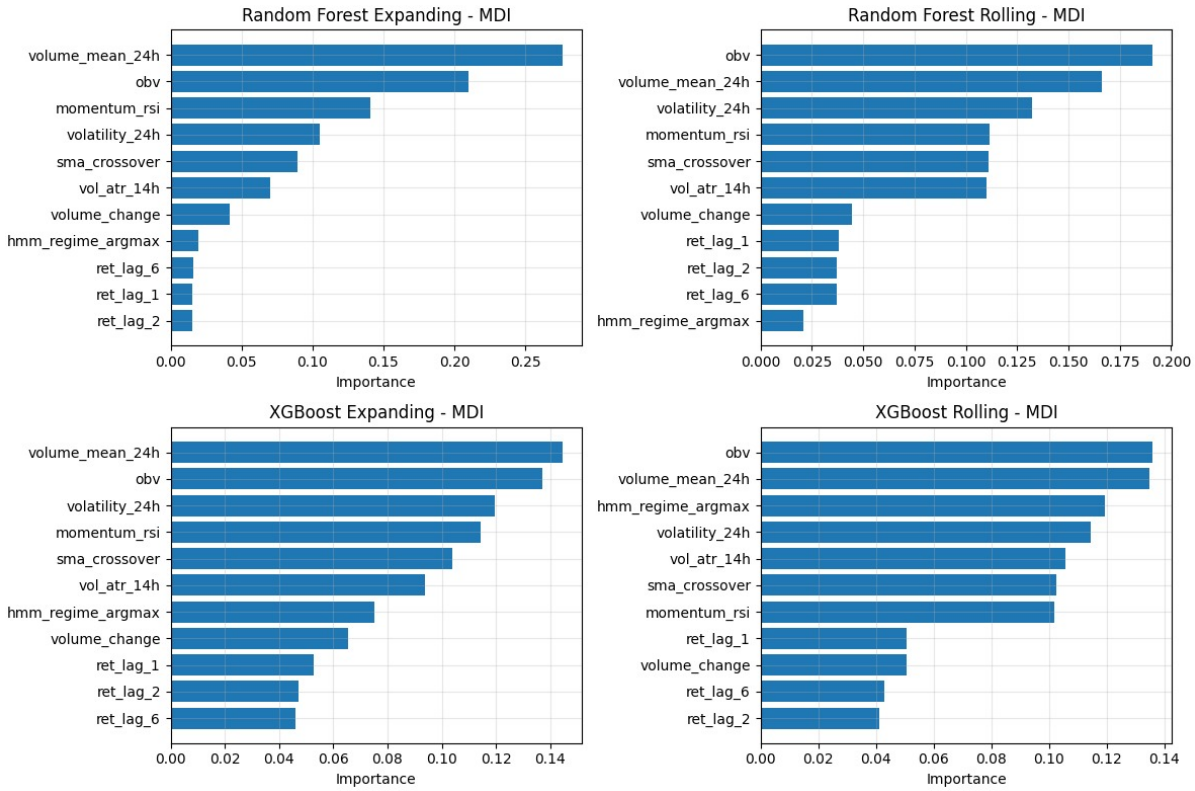


Figure 6: MDI across different models

The PFI plots in Figure 7 provide a complementary view by measuring the marginal impact of each feature on predictive performance. Here, `momentum_rsi` consistently rises in importance, frequently surpassing volume features, particularly in rolling window models. This implies that `momentum_rsi` may be more directly responsible for improving classification outcomes, even if it is used less frequently in splits. `hmm_regime_argmax` also gains relative importance under the PFI framework, most notably in the XGBoost expanding setup, reinforcing its contribution in capturing latent market dynamics.

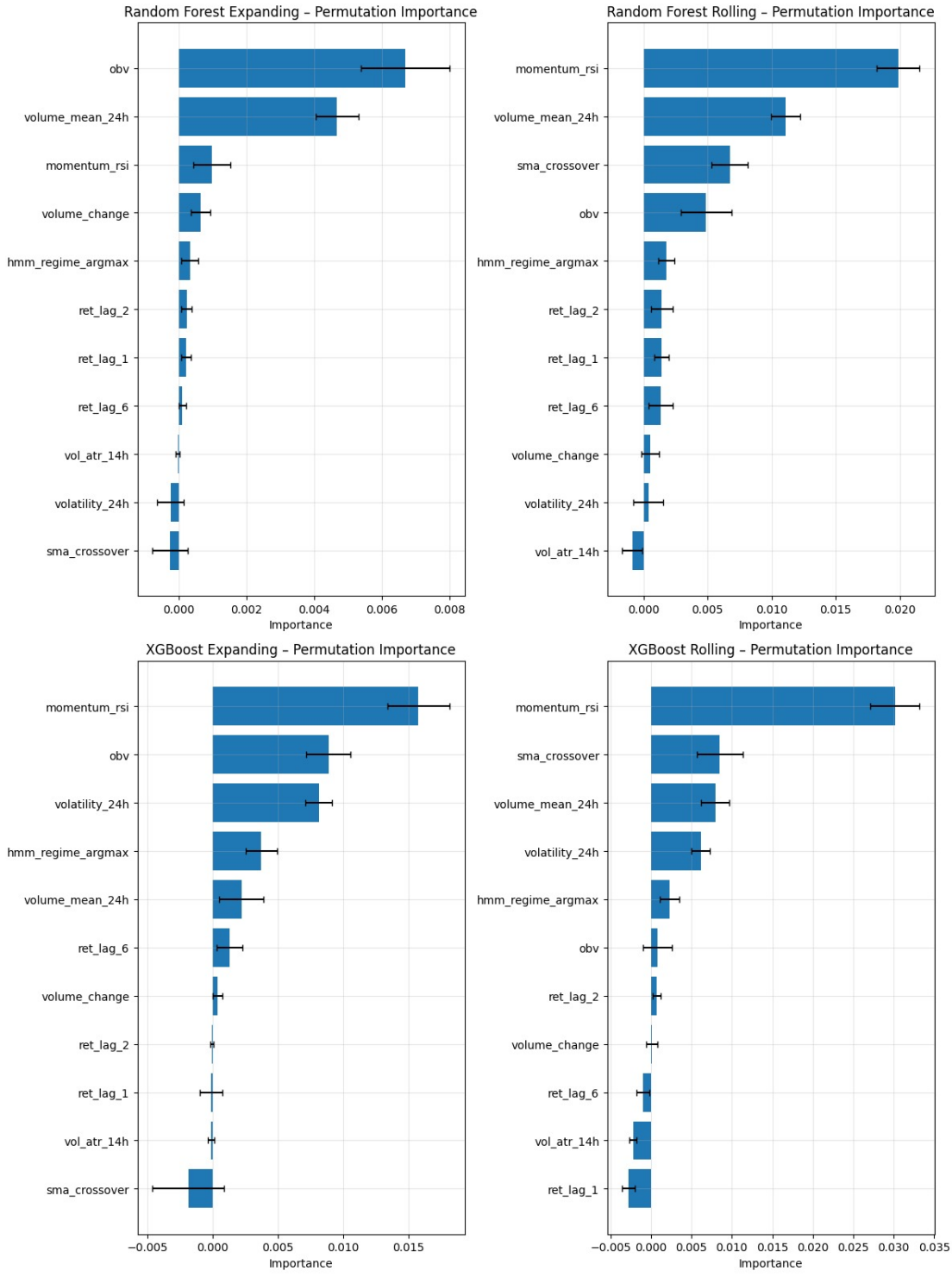


Figure 7: PFI across different models

Taken together, these results highlight the robustness of volume and momentum-based features across models, while also pointing to regime-related information as selectively valuable. On the other hand, some features, especially `ret_lag_1`, `ret_lag_2`, and `ret_lag_6`, consistently rank near the bottom across both MDI and PFI metrics. These

lagged returns may offer limited incremental signal and could be considered for removal or dimensionality reduction in future iterations to reduce complexity and potential noise.

Caution is warranted when interpreting PFI results in isolation, as this method is sensitive to the *substitution effect*: when two features are highly correlated, shuffling one may not impact performance significantly if the model can rely on the other. For instance, `volume_mean_24h` and `obv` are both volume-based and may act as substitutes in the model. As a result, PFI may underestimate their individual contributions.

To further investigate feature redundancy and mitigate the substitution effect observed in standard permutation importance, we grouped features using KMeans clustering and examined the cumulative importance of each cluster. This approach enables us to assess whether certain predictive signals are concentrated within particular groups of features, and whether entire clusters may be over- or under-represented in the model's decision-making.

Across both MDI and PFI cluster-level plots (Figures 8 and 9 on the following pages), we find that volume-related features (`volume_mean_24h`, `obv`) consistently belong to the most important clusters—particularly Cluster 1 in most configurations. This reaffirms earlier findings that volume signals dominate tree-based models. Similarly, Cluster 2, typically associated with trend indicators such as `momentum_rsi`, `sma_crossover`, and `volatility_24h`, also contributes significantly in all models.

In contrast, clusters containing lagged return features (`ret_lag_1`, `ret_lag_2`, `ret_lag_6`) often exhibit lower aggregate importance, especially under PFI. This suggests that these features may not contribute distinct predictive information beyond what is already captured by other clusters. Notably, in the Random Forest models, these return-based clusters rank lowest across both importance methods, strengthening the case for their removal or dimensionality reduction.

Overall, the cluster-based analysis confirms our prior observations: volume and trend features provide the bulk of the predictive signal, while the incremental value of lagged return features remains limited.

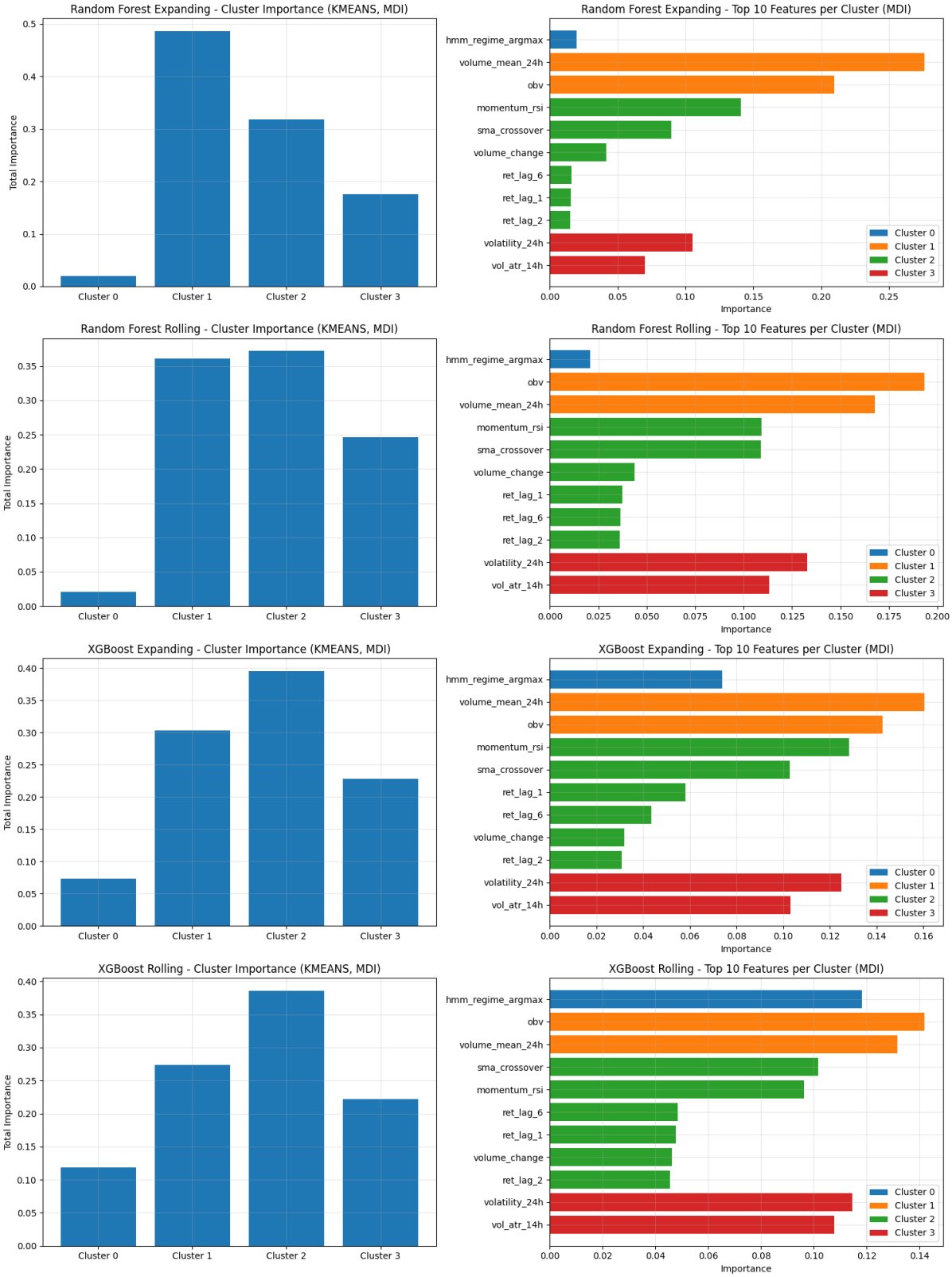


Figure 8: K-means clustering (MDI)

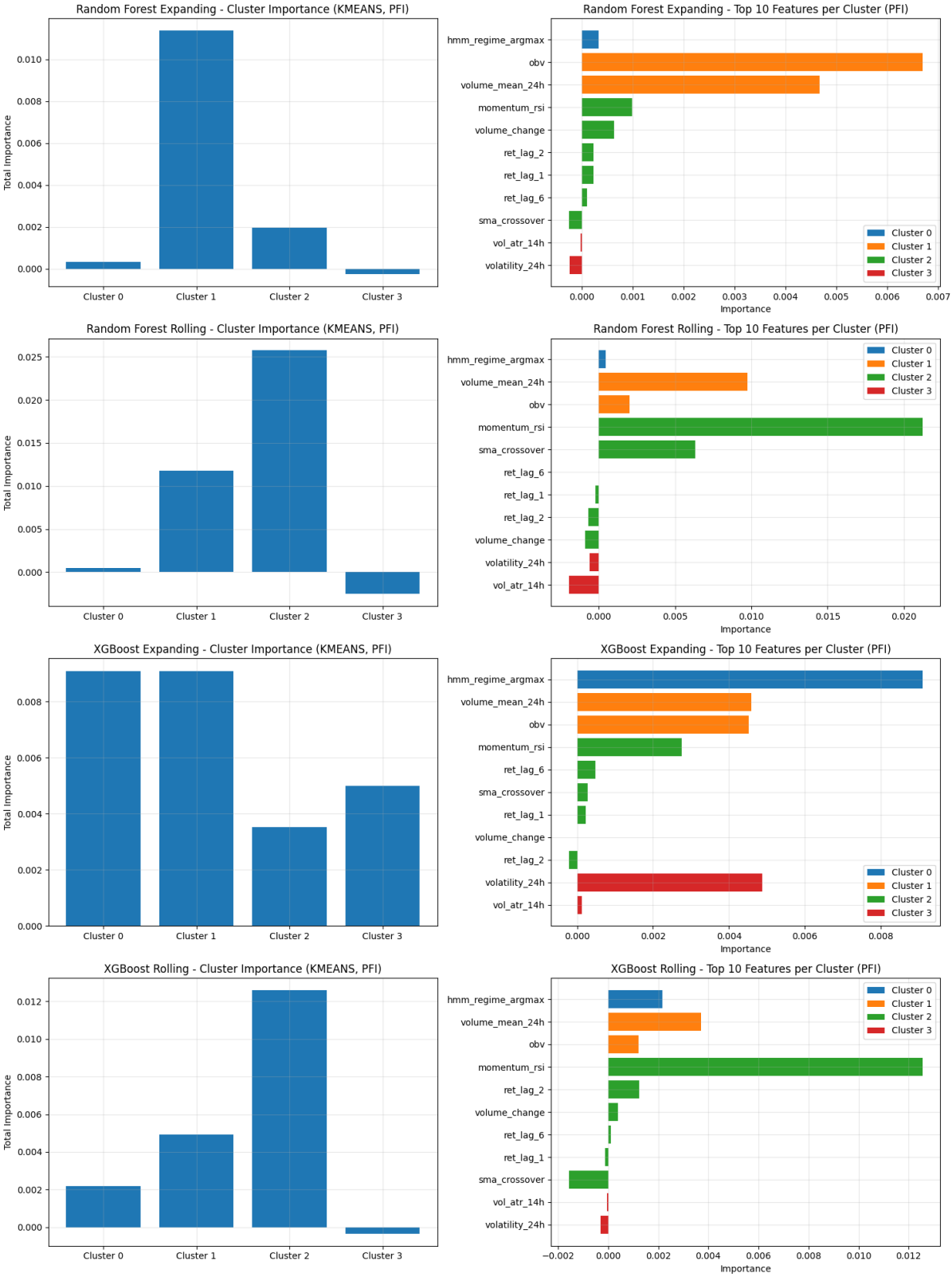


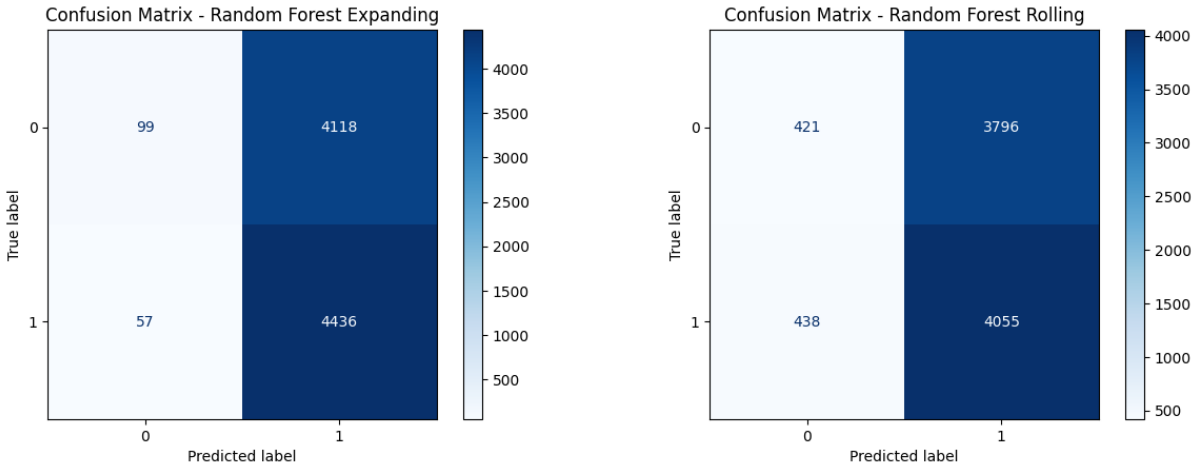
Figure 9: K-means clustering (PFI)

Model Evaluation

Table 6: Model out-of-sample performance

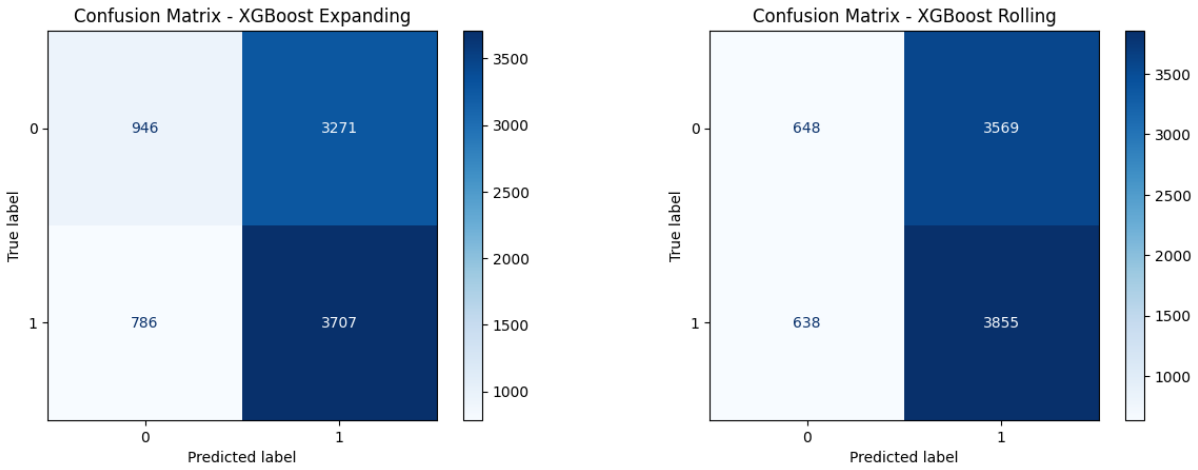
Model	Test Accuracy	Test F1	Test AUC
Random Forest Expanding	0.5207	0.6800	0.5257
Random Forest Rolling	0.5139	0.6570	0.5149
XGBoost Expanding	**0.5342**	**0.6463**	**0.5352**
XGBoost Rolling	0.5170	0.6470	0.5115
Neural Network Expanding	0.5202	0.5784	0.5280
Neural Network Rolling	0.4907	0.1699	0.4905

Table 6 summarises the out-of-sample performance of all models across accuracy, F1-score, and AUC. Overall, the tree-based models (Random Forest and XGBoost) outperform the neural network across all metrics. The highest F1-score is achieved by the Random Forest Expanding model (0.6800), indicating strong precision-recall balance in predicting directional trends. XGBoost Expanding records the best accuracy (0.5342) and AUC (0.5352), although all models exhibit AUCs close to 0.5, suggesting limited discriminative power.



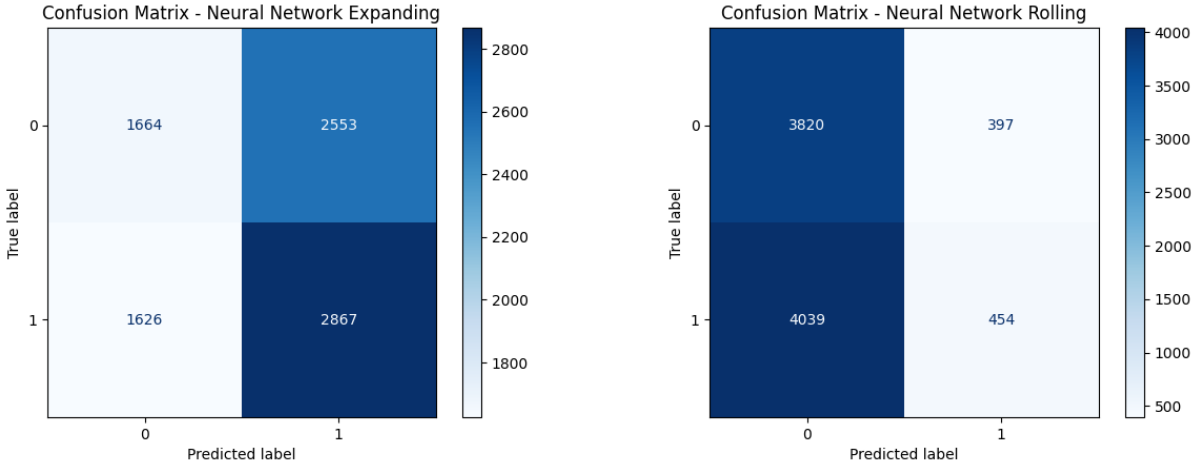
(a) Expanding (b) Rolling

Figure 10: Confusion matrices for Random Forest models



(a) Expanding (b) Rolling

Figure 11: Confusion matrices for XGBoost models



(a) Expanding (b) Rolling

Figure 12: Confusion matrices for Neural Network models

The confusion matrices (Figures 10 to 12) above reveal that most models, particularly Random Forest Expanding, exhibit a strong bias toward predicting the positive class, leading to very high false positive counts. While this boosts recall and F1-score, it comes at the cost of precision and weakens overall classification balance. In contrast, XGBoost Expanding provides a more even distribution of true positives and true negatives, aligning with its stronger AUC and accuracy.

Neural network models underperform overall, with the rolling configuration notably weak, both in confusion matrix structure and ROC performance, indicating poor generalisation and difficulty adapting to changing regimes.

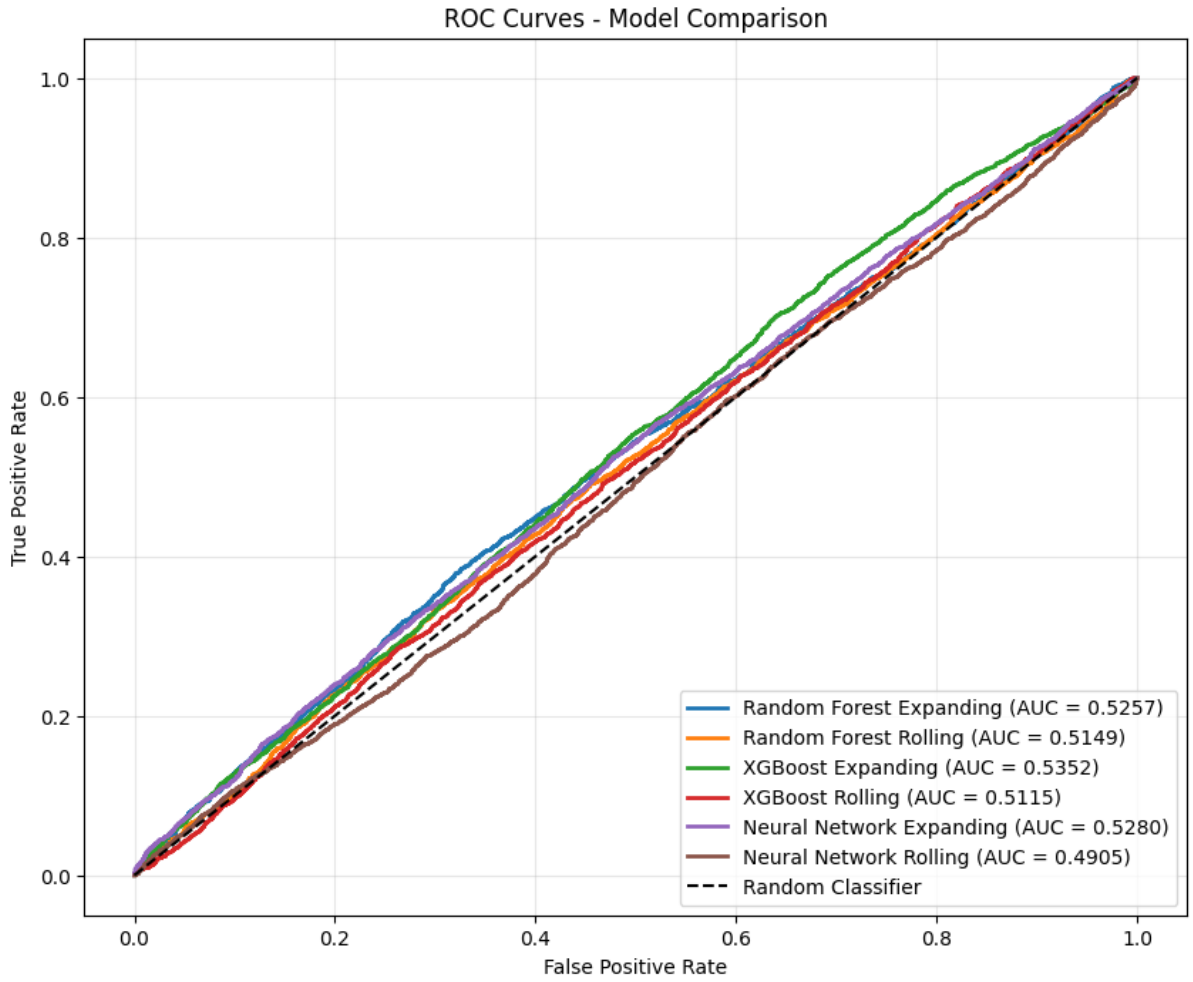


Figure 13: ROC Curves

The ROC curves in Figure 13 above reinforce these findings. All models operate close to the random classifier line, with only marginal improvements in ranking ability. Despite this, XGBoost Expanding shows consistently better balance across metrics, making it the most reliable model under current conditions.

In summary, while none of the models demonstrate strong separation capability, XGBoost Expanding offers the most consistent performance across all metrics. Ensemble tree models remain the most robust choice for this directional classification task, and expanding windows appear marginally more effective than rolling configurations under current conditions.

Backtesting

We designed a long-only trading strategy using hourly Bitcoin data and evaluated its performance over the full year of 2021, based on machine learning-generated signals. The objective is to assess how the strategy compares to a buy-and-hold benchmark in terms of risk-adjusted performance metrics, including Sharpe ratio, Sortino ratio, and maximum

drawdown.

Note: No leverage or shorting is used; capital is fully invested only when a bullish signal is generated.

Over the 2021 test period, the XGBoost-based trend-following strategy achieved a Compound Annual Growth Rate (CAGR) of 62.45%, compared with 42.15% for a simple buy-and-hold approach. A CAGR of 62.45% implies that, over the one-year testing window, an initial investment of $1 would have grown to roughly $1.624 under the strategy's signals, versus $1.421 under buy-and-hold. In absolute terms, the strategy captured about 20 percentage points more annual growth. Since Bitcoin's price swung widely in 2021—with several strong rallies and steep declines—the model's ability to move to cash around predicted downturns appears to have amplified overall returns. Crucially, outperforming buy-and-hold by ∼20% in a single year reflects both successful bullish signal captures and the avoidance of substantial negative moves.

This is further supported by the volatility measures, which reflect the standard deviation of hourly returns scaled to an annualised figure. As shown in Table 7, the strategy's 85.56% annualised volatility is appreciably lower than buy-and-hold's 94.15%. In other words, the strategy produced a smoother return path, even as it pursued higher absolute returns. For an asset as notoriously volatile as Bitcoin, trimming nearly 8 percentage points of volatility is nontrivial—it suggests that exiting positions before large drawdowns, albeit imperfectly, helped reduce fluctuations. Both the Sharpe and Sortino ratios also show modest improvements over the benchmark.

Table 7: Backtest Results (Test Set)

Metric	**Strategy**	**Buy & Hold**
CAGR	62.45%	42.15%
Annualized Volatility	85.56%	94.15%
Sharpe Ratio (rf=0%)	1.00	0.85
Sortino Ratio (rf=0%)	1.11	1.06
Max Drawdown	-56.14%	-54.76%
Avg Holding Period (hrs)	18.4	8709.0

Despite better volatility and risk-adjusted metrics, the strategy's maximum drawdown was slightly deeper than buy-and-hold (−56.14% vs. −54.76%). This seemingly paradoxical result can arise for two reasons: (1) *Signal lag or whipsaws* — the model may have entered long positions just before sharp reversals, resulting in concentrated losses that deepened the equity trough; and (2) *Missed rallies vs. drawdown timing* — buy-and-hold may have stayed invested through a long drawdown and fully recovered, while the model exited partway through and re-entered late, increasing realised losses. A 1.38 percentage-point difference is relatively minor; the more important takeaway is that the model performed comparably on drawdown despite much more frequent trading.

Buy-and-hold maintained one continuous position for the entire year (8,709 hours). In contrast, the model's average holding period was just 18.4 hours—indicating frequent switches between long and cash. This reflects a highly active approach focused on short-term momentum signals. While this can improve responsiveness, it also increases exposure

to transaction costs and slippage. If round-trip trading costs were 0.1%, even a few dozen trades could erode a portion of the strategy's edge. However, staying in cash during predicted downturns likely contributed to the observed reduction in volatility and improved Sortino ratio.

To visualise the difference, Figure 14 plots the equity curves of both strategies. At $t = 0$ (January 1st), both curves begin at 1.00. Every upward tick reflects a positive hourly return $((P_{t+1}/P_t) - 1)$; downward ticks reflect losses. The blue strategy curve moves only when a "go long" signal is active—remaining flat when in cash—while the orange buy-and-hold curve reflects continuous exposure.

Throughout 2021, the machine learning strategy (blue) closely tracks or outperforms buy-and-hold (orange) during strong rallies, and flattens out during major drawdowns. This is evident during the April–May crash, the October correction, and the December decline—periods where the model preserved capital by reducing exposure. As a result, the strategy ends the year significantly ahead of buy-and-hold, with superior CAGR, lower volatility, and fewer prolonged drawdowns.

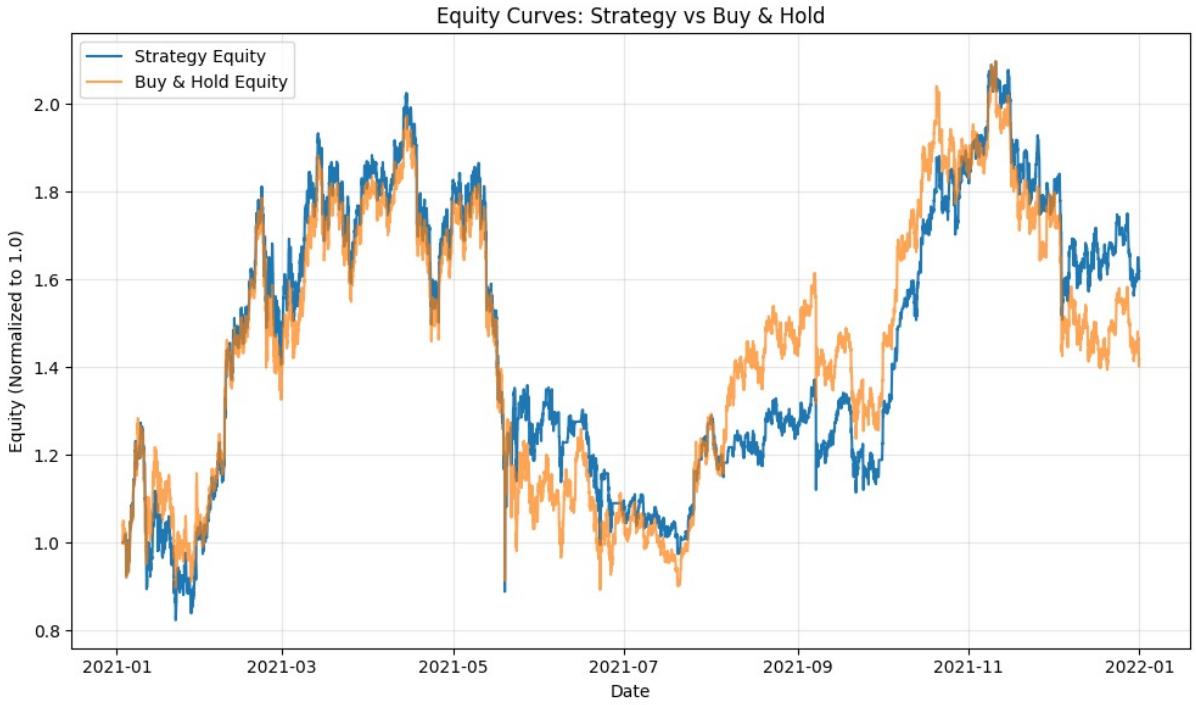


Figure 14: Backtesting performance: equity curves for ML strategy vs. buy-and-hold

All code can be viewed in `Part 2.ipynb`