Market Trend Fill Modelling with EMA and MACD Features

Contents

Problem Statement

This project addresses the challenge of making weekly trading decisions using only historical price data available at the close of the previous week. In many realistic trading scenarios, orders must be placed before markets open on Monday, without the ability to adjust positions during the week or react to intra-week news.

The goal is to develop a modular, reusable system that can forecast short-term price direction and evaluate the likelihood of order execution, based purely on lagged technical indicators. The system is designed to work with any sufficiently liquid instrument with weekly OHLC data, including ETFs and individual stocks.

Although developed and tested using the iShares Physical Gold ETC (SGLN), the pipeline is fully general and suitable for assets such as VUSA, ISF, or equity tickers like AAPL or MSFT.

Summary of Approach

The modelling pipeline consists of two stages. The first classifies the upcoming weekly trend as Buy, Sell, or Sideways, using lagged technical indicators such as Exponential Moving Averages (EMA) and the MACD histogram. This provides a directional signal for the week ahead.

The second stage, applied only to Buy and Sell weeks, predicts whether the price movement will be filled — that is, whether a limit order placed below (for Buy) or above (for Sell) the previous week's close would be executed before the trend reverses. This allows the model to assess the practical execution risk associated with directional signals.

Trend labels are based on week-to-week closing prices. For fill detection, the model evaluates intra-week price movements to determine whether a price threshold was crossed. Both models

are trained using a small set of engineered features and rule-based logic designed to reflect realistic trading constraints.

Project Structure and Deployment

The project is organised into a modular Python script structure designed for clarity, reproducibility, and ease of extension. Each major step in the modelling pipeline is handled by a dedicated script, while a single controller script (**run_project.py**) coordinates the entire process from start to finish.

Table 1 summarises the core scripts used in the project.

Script	Purpose
prepare_data.py	Loads and cleans the historical price data (e.g., SGLN), preparing it for feature engineering
features.py	Computes lagged technical indicators, including EMAs and MACD histogram
model_trend.py	Builds and evaluates the weekly trend classifier
model_fill.py	Trains a secondary classifier for predicting intra-week fill likelihood
wrap_up.py	Handles post-processing, saving outputs and visualisations
run_project.py	Top-level script that executes the full modelling pipeline

Table 1 Core scripts in the modelling pipeline, including their function and sequencing.

In addition to these scripts, the repository contains supporting resources summarised in Table 2.

Component	Contents
data/	Raw price data in CSV format (e.g., SGLN Historical Data.csv)
outputs/	Structured output folders for figures, reports, and intermediate data
README.md	Project overview, setup instructions, and usage guidance
requirements.txt	List of Python dependencies required to run the project

Table 2 Supplementary files and folders included in the project directory.

To run the complete pipeline, the user simply executes the following command from the root directory:

```
python run_project.py
```

This modular design ensures that each step can be tested or replaced in isolation. It also supports easy extension — for example, using different technical indicators, adding alternative classifiers, or adapting the pipeline to other instruments.

Feature Engineering

The model relies entirely on technical indicators derived from historical price data. The goal was to extract directional and momentum-related features that might help anticipate short-term trends and detect execution potential for limit orders.

Three types of indicators were used:

Exponential Moving Averages (EMA):

EMAs with periods 5, 12, and 26 were calculated based on weekly closing prices. These were chosen to reflect short-term (5), medium-term (12), and longer-term (26) price movements, capturing momentum shifts across different time horizons. All EMAs were lagged by one week to prevent data leakage into the prediction window.

MACD Components:

The full MACD suite was included. This comprised the MACD line (the difference between EMA_12 and EMA_26), the signal line (an EMA of the MACD line), and the MACD histogram (the distance between these two). These features collectively capture both the direction and acceleration of momentum. All were lagged to align with how data would be available in a real trading scenario.

Weekly Price Change Features:

Three additional features were engineered to capture basic price movement characteristics across each week: percentage change from open to close, and the relative changes from close to both the weekly low and the weekly high. These provided useful context for interpreting short-term price dynamics.

Table 3 summarises all nine features used in the trend classification model.

Feature	Description
ema_5	5-period Exponential Moving Average of weekly closing price
ema_12	12-period Exponential Moving Average of weekly closing price
ema_26	26-period Exponential Moving Average of weekly closing price
fast_line	MACD line: difference between ema_12 and ema_26, capturing trend direction
slow_line	Signal line: 9-period EMA of MACD line, used for smoothing
macd_h	MACD histogram: distance between MACD and signal line, showing momentum shifts
change_pct	Percentage change from weekly open to close
change_to_low	Percentage change from close to weekly low (intra-week downside)
change_to_high	Percentage change from close to weekly high (intra-week upside)

Table 3 Summary of engineered features used in the trend classification model.

Each feature was calculated using only historical prices and lagged appropriately to reflect real-world availability. This design ensured the model remained fully causal and could be realistically used to place limit orders before the market opens each week.

Modelling and Results

Trend Classification Model

The first stage of the modelling pipeline classifies each upcoming week as Buy, Sell, or Sideways, based on the direction of the following week's closing price. A Buy label indicates an expected increase, a Sell label a decrease, and Sideways represents minimal change. These classes were defined using a simple rule-based threshold: if the percentage change between the current and next closing price exceeded a predefined boundary, it was labelled as Buy or Sell; otherwise, Sideways.

The classifier used was a Decision Tree, chosen for its interpretability, simplicity, and ability to capture non-linear relationships. It was trained using the nine lagged features described earlier, including technical indicators such as EMAs, MACD components, and price change metrics.

The model was trained and tested on a rolling window of weekly data using a hold-out strategy. A portion of the dataset was reserved as a test set to simulate forward-looking performance. The outputs include both the predicted class labels and the model's decision paths, allowing for visual inspection of how different features contribute to the classification.

Evaluation results are included alongside each model, with confusion matrices, performance metrics, and visualisations of the decision tree and feature importance provided where relevant.

Evaluation and Results

The trend classifier was evaluated on a held-out test set consisting of 23 weeks. Table 4 shows the confusion matrix for the three-class prediction task: Buy, Sell, and Sideways.

	Predicted Buy	Predicted Sell	Predicted Sideways
Actual Buy	0	2	6
Actual Sell	0	2	2
Actual Sideways	3	1	7

Table 4 Confusion matrix for the trend classification model.

The classifier performed best on the Sideways class, with 7 correct predictions out of 11, resulting in a recall of 0.64. Performance on Buy signals was notably poor, with no correct predictions and all Buy weeks misclassified as Sell or Sideways. Sell predictions were slightly better, with 50% recall.

Overall accuracy was 39%, and the macro-averaged F1-score was 0.33, suggesting limited generalisation on this test set. While these results are modest, they provide a meaningful first step in generating directional signals.

Figure 1 shows the truncated structure of the decision tree (depth ≤ 2), illustrating how features like **macd_h**, **ema_12**, and **change_to_low** contribute to the model's decisions. Figure 2 displays the relative importance of all nine features used in the classifier. Momentum indicators and intra-week change percentages ranked highest, while MACD components and short EMAs contributed less.

Figure 3 shows the distribution of trend labels in the dataset. Sideways labels were the most common, followed by Buy and then Sell, highlighting a mild class imbalance. This likely contributed to the model's stronger performance on the Sideways class.

Figure 4 presents a PCA projection of the feature space coloured by trend label. While some clustering is visible, especially for Sideways points, the overlap between classes explains the model's difficulty in consistently separating Buy and Sell weeks.

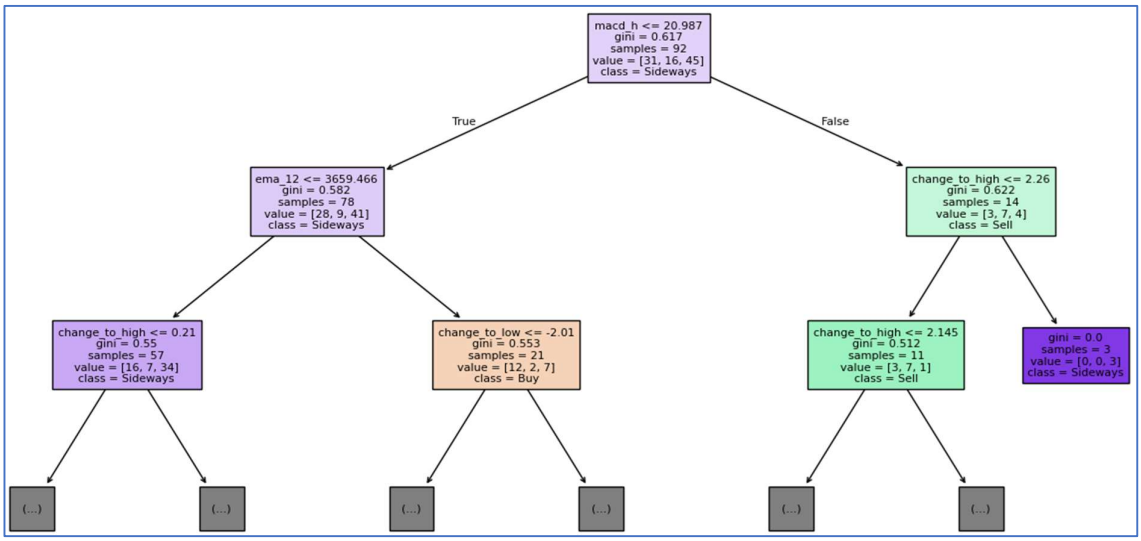

Figure 1 Truncated decision tree structure (maximum depth = 2) used for weekly trend classification

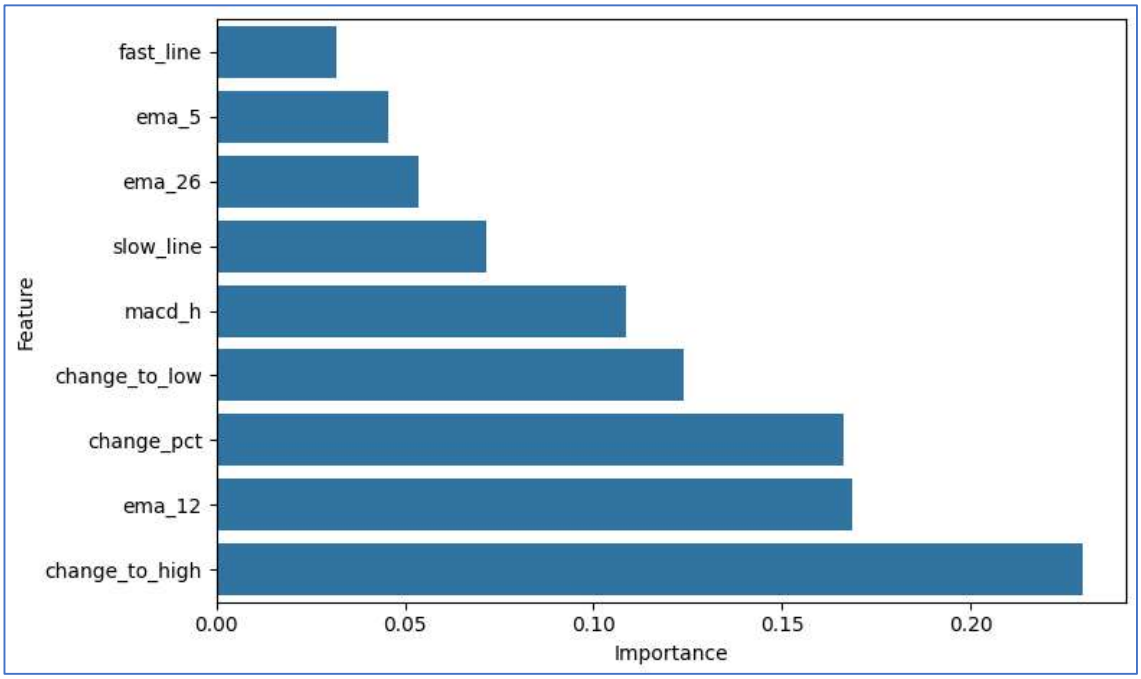

Figure 2 Feature importances from the trend classification model

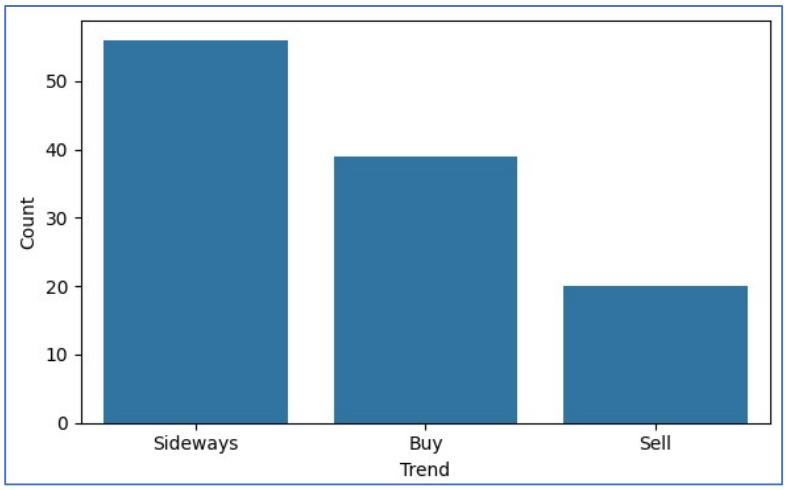

Figure 3 Distribution of true trend labels in the dataset

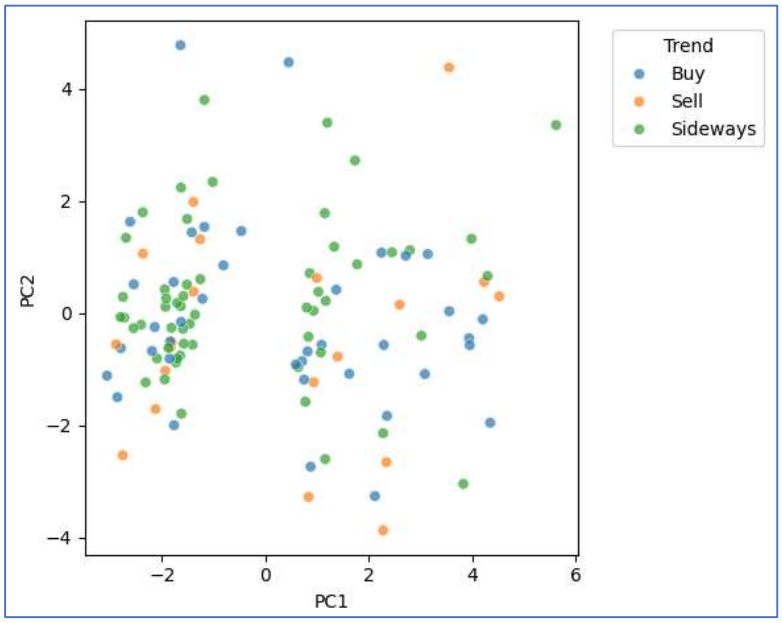

Figure 4 PCA projection of model features coloured by true trend label

Fill Detection Model

The second stage of the pipeline models whether a directional movement — once predicted — is likely to be filled. This step focuses on practical execution risk: given a Buy or Sell label for the upcoming week, will a limit order placed at the previous week's close be executed before the trend reverses?

Separate binary classifiers were trained for Buy and Sell weeks, as the conditions influencing fill likelihood differ. A fill was considered successful if the intra-week low (for Buy) or high (for Sell)

crossed the prior week's close. The target variables **buy_filled** and **sell_filled** were constructed based on these conditions.

Both models used the same engineered feature set as the trend classifier, and were implemented using Decision Tree classifiers. Training was restricted to directional weeks, with Sideways cases excluded. The dataset was stratified during the train-test split to reflect the underlying class balance.

This second model allows the pipeline to output not just directional signals, but realistic guidance on whether those signals are likely to result in executed trades — a crucial step for applying the system in practice.

Evaluation and Results

The fill detection models were evaluated separately for Buy and Sell signals using a balanced hold-out set of 30 samples in each case. **Tables 5 and 6** show the confusion matrices for each classifier.

	Predicted False	Predicted True
Actual False	1	5
Actual True	1	23

Table 5 Confusion matrix for the Buy fill model.

	Predicted False	Predicted True
Actual False	2	3
Actual True	1	24

Table 6 Confusion matrix for the Sell fill model.

Both models performed well on the positive (filled) class, with recall of 0.96 for both Buy and Sell predictions. The Buy fill model achieved 80% overall accuracy, with a weighted F1-score of 0.76. The Sell fill model was slightly stronger overall, reaching 87% accuracy and a weighted F1-score of 0.85.

However, precision and recall for the False (not filled) class were notably weaker in both cases, particularly for Buy signals, where most unfilled examples were misclassified as filled. This reflects a class imbalance challenge and highlights the importance of interpreting True predictions with caution — especially in low-fill environments.

Figure 5 provides further insight into the distinguishing characteristics of filled versus unfilled orders. For Buy signals, filled orders tend to have significantly lower **change_to_low** values, suggesting that stronger downward price movements after entry increase the likelihood of execution. For Sell signals, the clearest separation occurs in **change_to_high** and **change_pct**, with filled orders showing greater upward momentum. These visual patterns reinforce the model's ability to detect directional conviction and provide a useful lens for assessing execution risk.

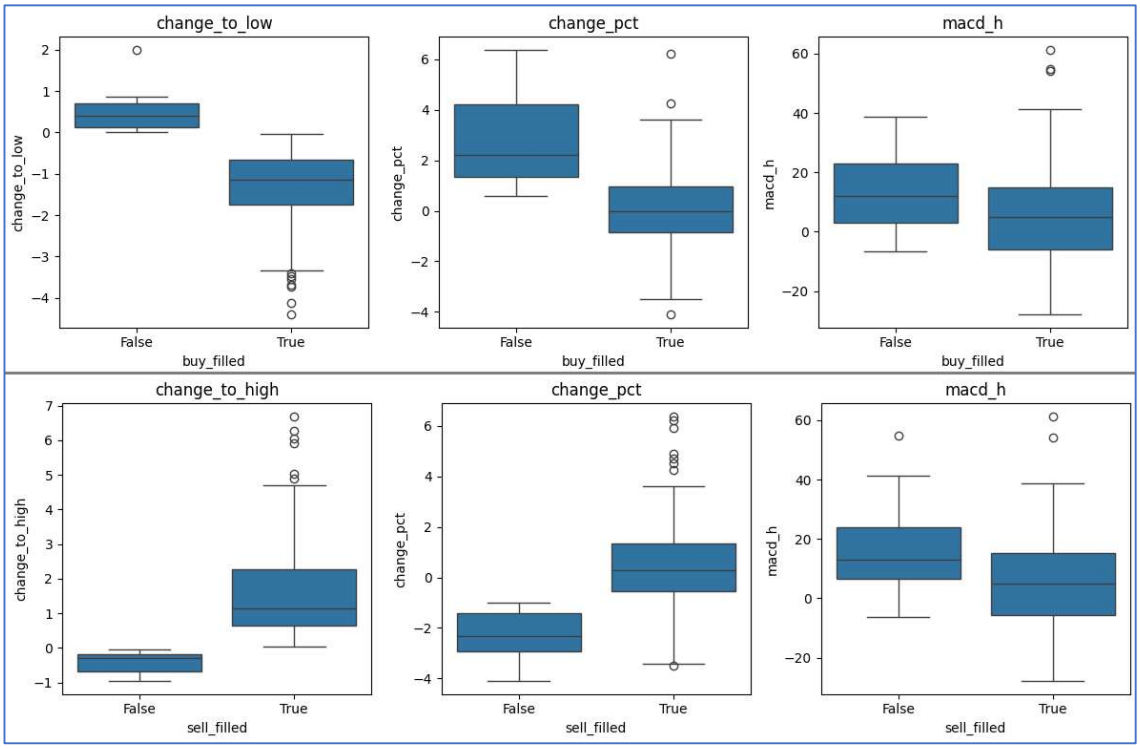

Figure 5: Boxplots comparing key features for filled vs unfilled limit orders for Buy (top) and Sell (bottom) signals

Despite some limitations, the models offer practical guidance on the viability of limit order strategies and help refine directional predictions into actionable trading decisions.

Conclusion and Next Steps

This project developed a modular, script-based system for classifying weekly price trends and assessing the likelihood of limit order execution, using only historical OHLC data. The approach is fully generalisable to any sufficiently liquid asset, including ETFs and stocks, and is designed to simulate a realistic trading constraint where decisions must be made before the weekly market open.

The modelling pipeline consisted of two stages. The first model classified upcoming trends as Buy, Sell, or Sideways using lagged technical indicators such as EMAs and the MACD histogram. The second model, applied only to directional signals, predicted whether a limit order placed below (for Buy) or above (for Sell) the previous week's close would have been filled during the subsequent week's trading range. Both models were implemented using decision tree classifiers and evaluated using held-out data and visual diagnostics.

While the trend classification model showed only moderate performance, the fill detection models for both Buy and Sell signals achieved strong accuracy and recall. The system successfully filtered directional predictions based on execution viability, a key step toward making such strategies actionable in practice.

Future improvements could include more sophisticated classifiers (e.g., XGBoost or LightGBM), the introduction of cross-validation for more robust evaluation, and the incorporation of simple

risk-based rules to adjust or skip trades with low confidence or poor reward-to-risk ratios. The current modular design allows these upgrades to be integrated with minimal disruption.