Quantitative Trading Strategy Development

Aguru Venkata Saisantosh Patnaik

1 Hypothesis and Data Overview

I hypothesized that time-based patterns and macroeconomic factors drive price changes in this low-volatility, major-FX-pair series. The data (hourly OHLCV) exhibited muted price movement and moderate volume (hundreds to low thousands). Because the timeline spanned many years, I expected seasonal or cyclical effects. Therefore, I engineered features in three categories: temporal/session features, macro/commodity indicators, and technical indicators. These features would provide context for a machine learning model to predict price movements despite the muted raw volatility.

2 Feature Engineering

2.1 Temporal Features and Session Labeling

I extracted calendar and session information from each timestamp. First, I labeled each hour by trading session: Asian-only, European-only, American-only, or overlapping sessions (e.g. London+NY open). Forex markets trade nearly 24 hours on weekdays but activity patterns differ by session; adding a session label lets the model learn these intraday volatility or trend differences.

I also encoded broader calendar effects: month, quarter, day-of-week, and day-of-month. These capture seasonal patterns (e.g. month-end flows or quarter-end rebalancing) that can affect prices. For instance, I integer-encoded the quarter (1–4) to flag the end-of-quarter, when portfolio rebalancing often occurs. Including day-of-week and day-of-month can capture weekly and monthly seasonality in volume or trends.

2.2 Cyclical Encoding of Date/Time

Simple numeric encodings of cyclical features (like hour or month) can mislead a model because endpoints are artificially far apart. Instead, I applied sine and cosine transforms to map each cyclic feature onto the unit circle. For example, I used $\sin(2\pi \times \text{hour}/24)$, $\sin(2\pi \times \text{day-of-week}/7)$, and similarly for day-of-month and month. This ensures that values at the cycle boundary remain adjacent (e.g. 23:00 and 00:00 hours are close on the circle). As noted in time-series literature, trigonometric encoding preserves the cyclic relationships in the data (e.g. if prices tend to rise every Friday, a sine-of-day feature will capture that pattern smoothly).

2.3 Macro and Commodity Features

Forex prices are sensitive to macroeconomic trends and commodity prices. I augmented the data with external series. First, I included the interest rate differential by taking the spread between U.S. 10-year and 2-year Treasury yields. Empirically, the USD often strengthens when long yields rise relative to short yields. I computed (10yr_yield − 2yr_yield)/24 as a feature (dividing by 24 to roughly scale it into hourly terms).

Second, I merged in commodity prices: daily close of WTI crude oil and gold. I joined these on date so each hour inherits the previous day's close. These commodities often correlate with currency moves (for example, a steep decline in crude oil has historically benefited the USD, since many currencies move inversely to oil prices). I also included oil volume (scaled by 24 for hourly consistency). After merging, new columns like `Diff_10Y_2Y`, `Oil_Close`, `Oil_Volume`, and `Gold_Close` were added to the dataset.

2.4 Technical Indicators

To capture short-term market structure, I computed a suite of bounded technical indicators on the price series. I selected indicators from three groups:

- **Trend indicators:** Average Directional Index (ADX, measures trend strength) and Commodity Channel Index (CCI, indicates cyclical deviation).
- **Momentum indicators:** Percentage Price Oscillator (PPO, a momentum oscillator similar to MACD) and Chande Momentum Oscillator (CMO, bounded between -100 and +100).
- **Volatility indicators:** Chaikin Volatility (based on AD Oscillator range) and Keltner Channel width (based on ATR).

Each indicator was computed for multiple look-back windows based on preliminary autocorrelation analysis. For example, I computed ADX for 2, 5, and 14-day windows; CCI for 2, 8, and 20 days; PPO for (2,6) and (12,26) periods; CMO for 2 and 14 days; Chaikin Volatility for (2,5) and (3,10) periods; and Keltner Channel width for 2, 15, and 24 days. To select useful look-backs, I examined autocorrelation plots (ACF/PACF) of each indicator. For instance, Figure 1 shows the ACF and PACF of the 14-day CMO series. The significant spike at lag 8 suggested adding an 8-day CMO to our features.

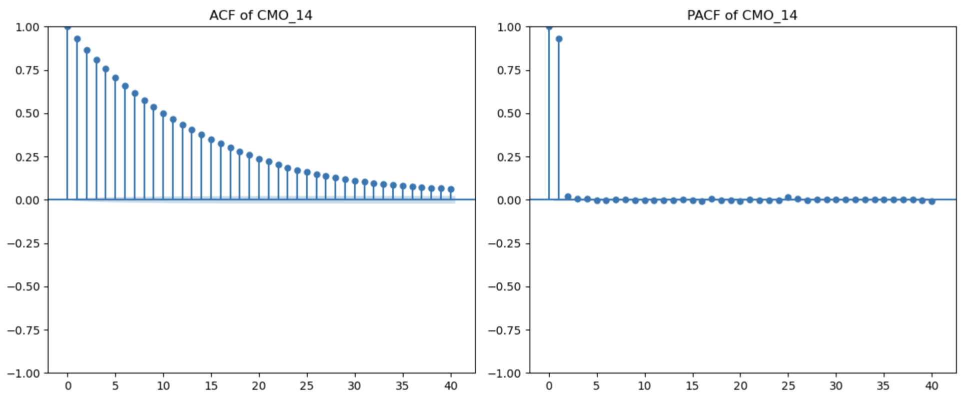


Figure 1: Autocorrelation (ACF) and partial autocorrelation (PACF) of the 14-day CMO indicator.

I also derived composite features to capture higher-order patterns. Specifically, I calculated differences and ratios between different look-back values. For example:

- `CCI_diff_6` $= \text{CCI}_8 - \text{CCI}_2$ (short-term vs. long-term trend difference).
- `KeltnerWidth_diff_13` $= \text{KeltnerWidth}_{15} - \text{KeltnerWidth}_2$.
- `CCI_ratio_2_20` $= \text{CCI}_2/\text{CCI}_{20}$.
- `KeltnerWidth_ratio_15_24` $= \text{KeltnerWidth}_{15}/\text{KeltnerWidth}_{24}$.

These engineered features help the model detect changes in momentum or volatility regimes. After this step, each data row included the original OHLCV, session/time features, macro/commodity features, and a large set of technical features (including the derived differences and ratios).

3 Target Signal Creation

I needed a target trading signal (+1 for long, -1 for short, 0 for flat). I began with an abstracted rule-based signal in `df['signal']` and backtested it to label trades as Profit, Loss, or Neutral. Each time the signal flipped from +1 to -1 or vice versa, a trade was closed and its PnL (percentage return) computed. I then labeled each trade as follows:

1. **Profit:** PnL $> +0.1\%$. Target signal = original signal (i.e. keep the trade).
2. **Loss:** PnL $< -0.1\%$. Target signal = negative of original signal (i.e. flip the trade).
3. **Neutral:** $|\text{PnL}| \leq 0.1\%$. Target signal = 0 (no trade).

Thus, the new `target_signal` column (values -1, 0, +1) represents an "ideal" hindsight signal that avoids previous losses. I then dropped intermediate PnL columns and removed the first few thousand rows (to eliminate partial trades during warm-up). Finally, the data was split chronologically into 70% training and 30% test sets (no shuffling) to avoid any look-ahead bias.

4 Model Building

4.1 Feature Selection via Random Forest

With the target defined, I first pruned the feature set. I trained a Random Forest classifier on the training data to predict the target signal and computed permutation importances across folds. The highest-ranked features included the original baseline signal, momentum indicators (e.g. CMO_14, PPO_2_6), and trend/volatility indicators (e.g. ADX_5, CCI_20, CCI_ratio_2_20, KeltnerWidth_ratio_15_24). In contrast, raw OHLC prices and many basic time flags had very low importance. Based on this, I dropped irrelevant columns and kept those capturing momentum, trend, and volatility dynamics (along with the baseline signal).

4.2 Model Training: XGBoost with Time-Series CV

We then trained a multi-class XGBoost classifier using the selected features. Key steps included:

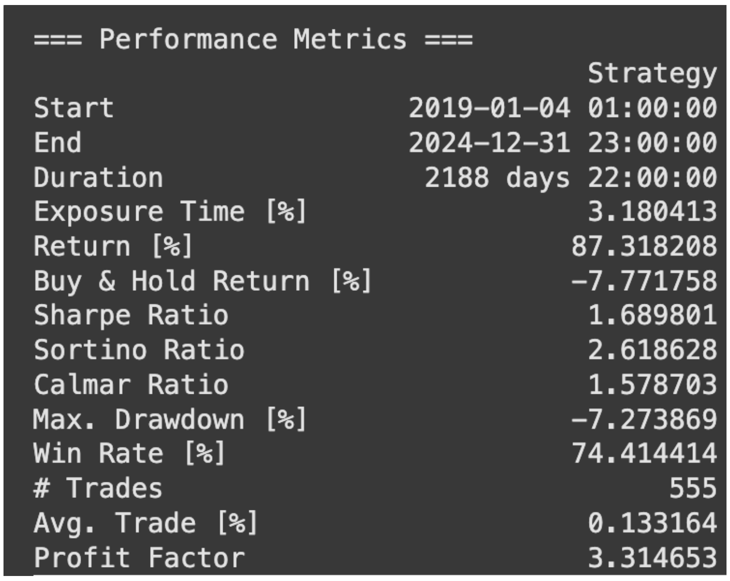
- **Time-series CV:** I used a 5-fold purged time-series cross-validation (each fold had a small gap of 5 bars to prevent leakage). In each fold, earlier bars trained and later bars validated, mimicking live forecasting.
- **Hyperparameter Search:** I performed a randomized search over 30 trials, tuning parameters such as learning rate (η), tree depth (max_depth), subsampling, and regularization. Early stopping (150 rounds) was used to select the optimal number of trees per trial.
- **Class imbalance:** Because the flat (0) class was much more frequent than buy/sell, I applied class weighting in XGBoost to balance the classes.

The best model (e.g. $\eta \approx 0.03$, max_depth $= 4$, subsample=0.8, etc.) was retrained on the full training set. Its in-sample performance was strong: it achieved around 94% precision and recall on the ± 1 classes and near-perfect performance on the flat class. Importantly, no data shuffling was used, ensuring all training respected chronological order.

5 Backtesting Results

I evaluated the model by backtesting its predicted signals, trading at each bar close with round-trip transaction costs.

In this, I examined realistic FX broker fees (typically 0.003–0.007% per side). I ran the backtest with a **0.01% commission**. Figure 3 shows the equity curve, yielding about +87% return. The corresponding metrics are in Figure 2 (Sharpe 1.69, max drawdown 7.3%). Under these plausible costs, the strategy demonstrates a positive risk–return profile.



```
=== Performance Metrics ===
                                    Strategy
Start                    2019-01-04 01:00:00
End                      2024-12-31 23:00:00
Duration                  2188 days 22:00:00
Exposure Time [%]                   3.180413
Return [%]                         87.318208
Buy & Hold Return [%]              -7.771758
Sharpe Ratio                        1.689801
Sortino Ratio                       2.618628
Calmar Ratio                        1.578703
Max. Drawdown [%]                  -7.273869
Win Rate [%]                       74.414414
# Trades                                 555
Avg. Trade [%]                      0.133164
Profit Factor                       3.314653
```

Figure 2: Performance metrics under 0.01% commission.

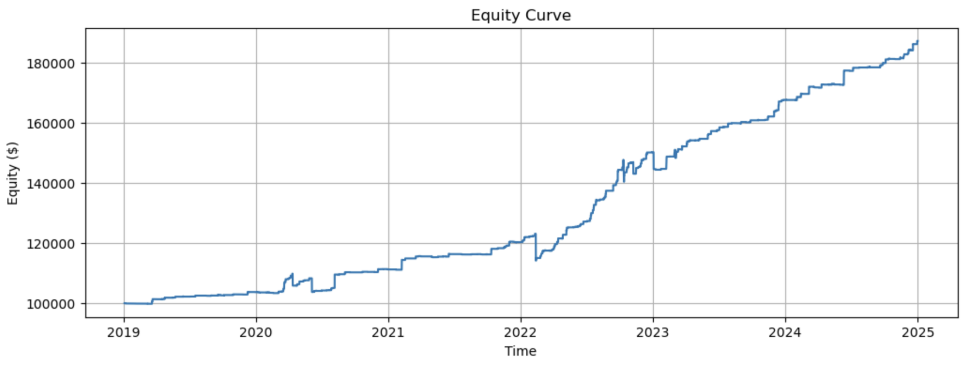


Figure 3: Equity curve under 0.01% commission.

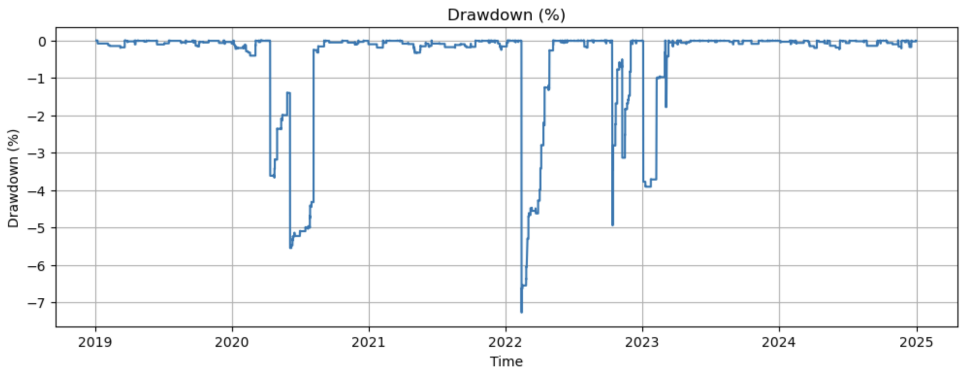


Figure 4: Drawdown plot under 0.01% commission.

6 Conclusion

I have built a structured quantitative pipeline for a low-volatility FX series. By incorporating domain knowledge in feature engineering (session cycles, seasonal effects, interest-rate spreads, commodity prices, and technical patterns), I gave the model a rich context. I ensured all data splits respected chronology to avoid look-ahead bias. The model learned from a hindsight-adjusted target signal that avoided past mistakes. Key lessons include the value of engineered features, the importance of realistic cost assumptions, and strict validation procedures. Future work could refine the signal logic, explore alternative models (e.g. ensembles or deep neural nets), and optimize trade execution to improve returns further.