Domain 2

QUANTITTIVE TRADING

Presented by: Ctrl+Defeat

ABOUT US

01

Lee Pei Yu

02

Ngoh Zhiyu

03

Cheang Ka Sin

04

Tee Qing Fui

05

Ong Jing Qi

KEY ASSUMPTION

VOLATILITY

THE VOLATILITY BETWEEN PRICE SHOW HOW THE MARKET FEELS, AND MIGHT HELP US GUESS WHAT HAPPENS NEXT AND DO A DECISION.

HOW THE VOLATILITY AFFECTS THE MARKET AND PEOPLE DECISION?

WE ASSUME THESE:

1. If a candle has a much bigger range than usual, it might mean a breakout or trend is starting.

2. If a market is moving a lot today, it will probably keep moving a lot tomorrow too.

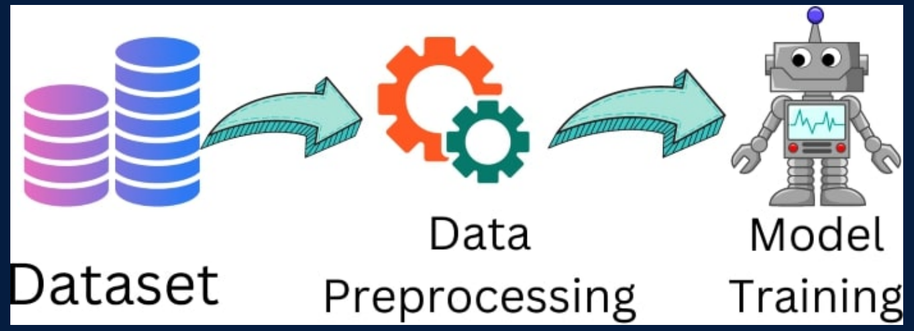
3. If the shadows are large compared to the full candle, it means the market is unsure – price was pushed away from the highs/lows. That could mean a potential reversal.

WE CAME OUT A CONCEPT

BACKTESTING FRAMEWORK

Our backtesting framework plays a critical role in validating the effectiveness of our HMM-based quant trading strategy. By leveraging historical multi-source candle data and volatility-driven features, we simulate realistic trading environments to assess performance before live deployment.

DATA PREPROCESSING



Aggregates and cleans high-frequency candle data (≤1-day intervals) from platforms like CryptoQuant, Glassnode, and Coinglass.

Feature Extraction & Signal Generation

Computes engineered volatility features (RangeVol, VolCluster, WickRatio).

Calculates a SignalScore, weighted by these features, and integrates HMM-predicted market regimes to generate Buy/Sell/Hold signals.

STRATEGY EXECUTION

Executes trades based on model outputs, accounting for a 0.06% transaction fee.

Ensures minimum 3% trade signal activity per data row to maintain trading frequency.

PERFORMANCE EVALUATION

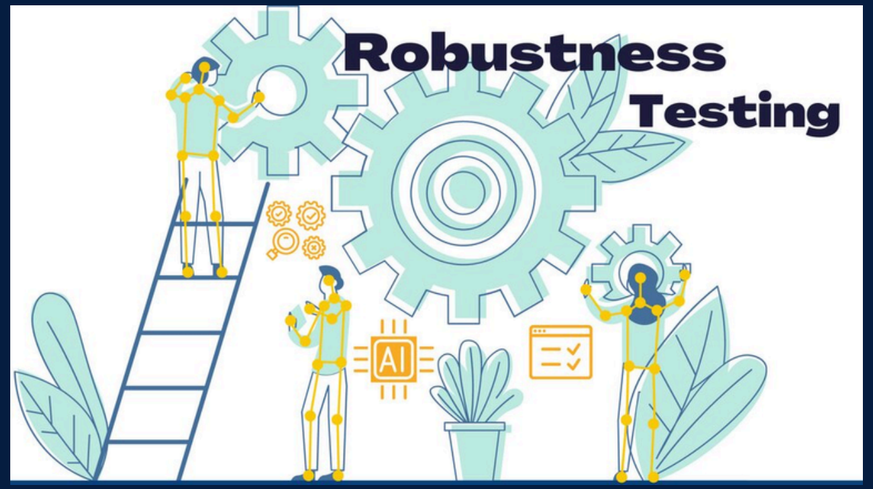
Assesses strategy via key metrics:

Sharpe Ratio ≥ 1.8 (risk-adjusted return)

Maximum Drawdown ≥ -40% (drawdown control)

Trade Frequency ≥ 3% (signal sufficiency)

ROBUST TESTING TIMELINE

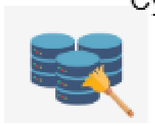


Covers multiple years of historical backtesting, followed by at least one year of forward testing on unseen data.

This end-to-end backtesting system ensures our quant strategy is not only theoretically sound but also practically robust, paving the way for confident real-world trading execution.

CONCEPTUAL DIAGRAM

Data Collection and Preprocessing



CyboTrade API (Live)

Collect

Glassnode, CryptoQuant, Coinglass

<= 1 days intervals

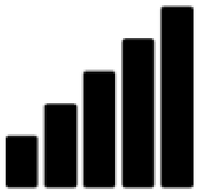
Data (Candle Model) Preprocessing:

Data Cleaning,
Data Formatting,
Extract Features

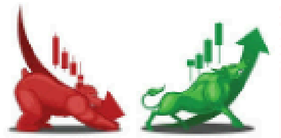
Volatility Feature Extraction

```
SignalScore =
        w1 * RangeVol
        +
        w2 * VolCluster
        -
        w3 * WickRatio

where;

    - RangeVol =
        High - Low
    - VolCluster =
        std(Close - N:t)
    - WickRatio =
        (High - max(open,close)
        + min(open,close) - low)
        / (high - low + e)
```

Calculate Signal Score



HMM modelling



- Regime Detection

(HMM predicted market)

- Match Pattern

Signal Generation

- Buy, Sell, Hold Signals
- Processing fees (0.06%)

Strategy Execution

Executes trades based on model outputs, accounting for a 0.06% transaction fee.

Ensure minimum 3% trade signal activity per data row to maintain trading frequency

Performance Evaluation - Key Metrics

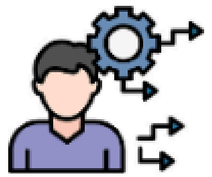
- ✅ **Sharpe Ratio ≥ 1.8** (risk-adjusted return)
- ✅ **Maximum Drawdown ≥ -40%** (drawdown control)
- ✅ **Trade Frequency ≥ 3%** (signal sufficiency)

Backtesting & Forward Testing

Backtesting

Apply model to historical dataset

Observe how the strategy performs during both bull and bear market states

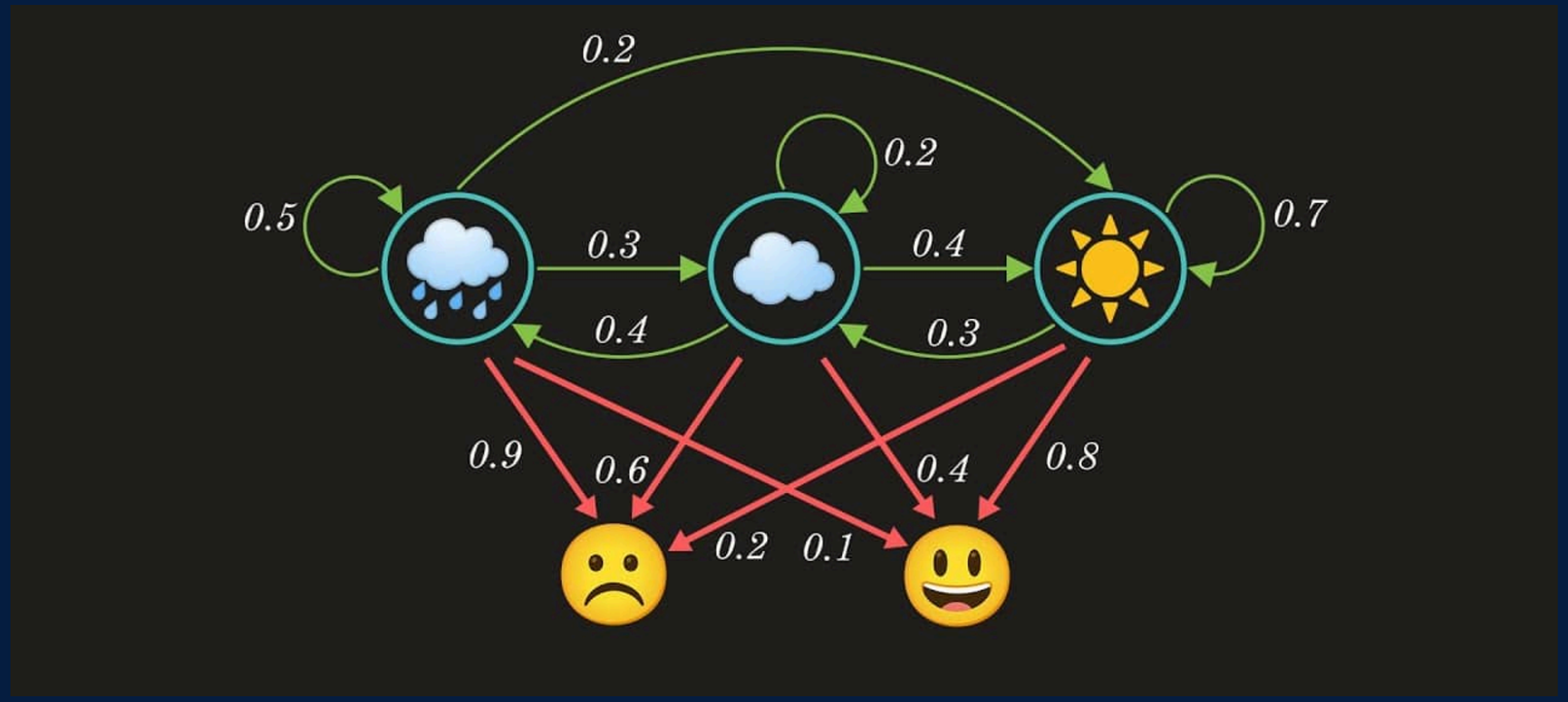


Forward Testing

Forwad-test on unseen data

Evaluate how it performs in real-time conditions

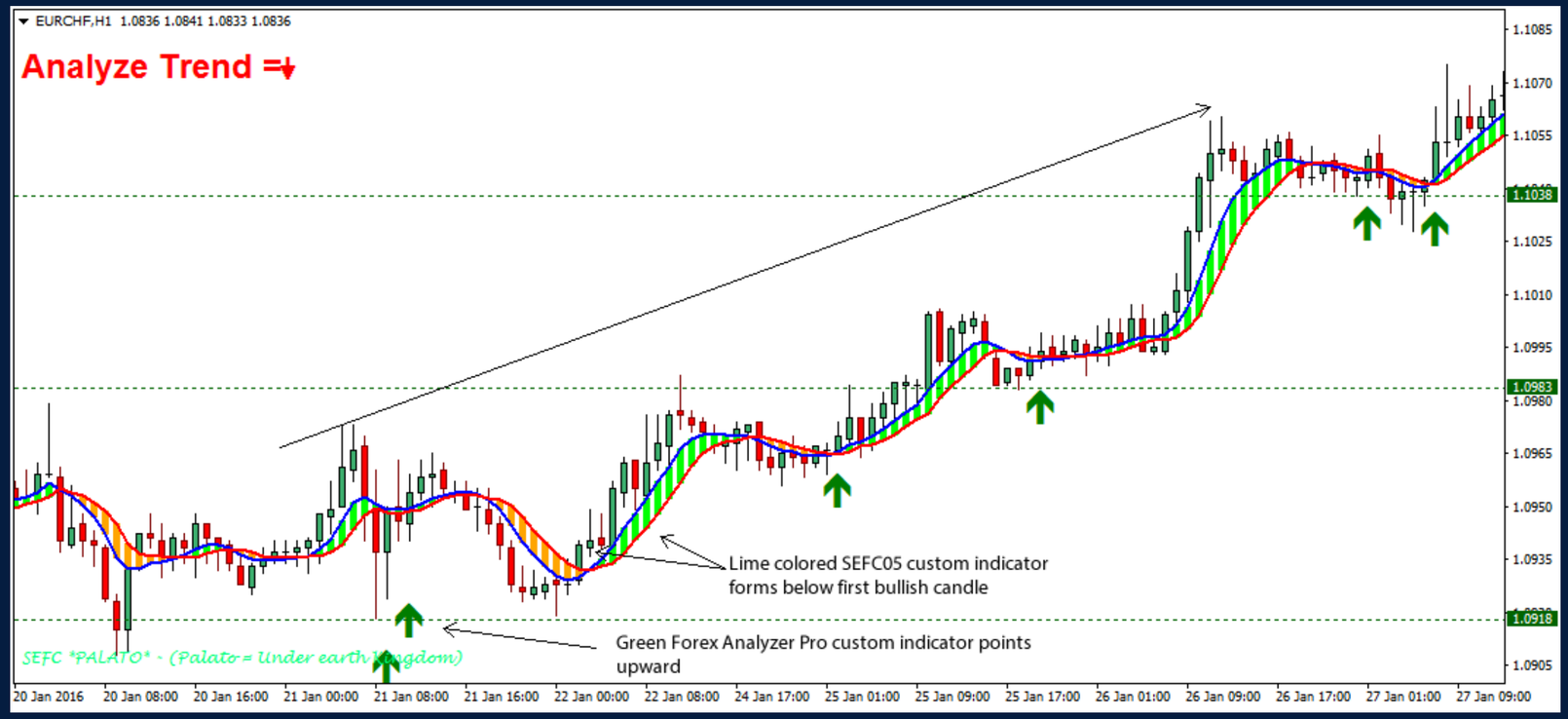
WHY WE USED HMM MODEL FOR OUR DATA MODELLING?



- HELPS DETECT HIDDEN MARKET REGIMES (LIKE BULLISH OR BEARISH) BASED ON OBSERVED PRICE AND VOLATILITY PATTERNS

- ALLOWING US TO MAKE BETTER TRADING DECISIONS FROM NOISY MARKET DATA

WHAT ARE THE BACKTEST STRATEGIES THAT WE USED FOR?



Trend Following Strategy

Trends often form when volatility expands after a regime change.

High RangeVol or VolCluster indicates a breakout or sustained trend.

HMM detects trend regime → strategy rides the trend.

📈 Higher volatility reflects stronger sentiment = stronger trend confirmation.

🚨 Breakout Strategy

This strategy depends on volatility spikes to detect breakouts.

Sudden increase in RangeVol or VolCluster can signal a pre-breakout phase.

Your SignalScore, which combines volatility factors, becomes critical to predict breakout opportunities.

Momentum Strategy

Momentum thrives in volatile, trending markets — which is exactly where your volatility indicators (e.g., large RangeVol, increasing VolCluster) can detect strong moves.

🔧 SignalScore helps confirm whether the trend strength is high enough to enter a position.

📊 HMM captures the market regime (bullish/bearish), while volatility confirms momentum strength.

FIRST STAGE OF MODELLING

Define feature:

```
[8]
    global_df['Volatility'] = global_df['high'] - global_df['low']
    X = global_df[['Volatility']].values
    print(global_df['Volatility'])
```

```
0        80.6
1       106.6
2        49.7
3       151.1
4       102.6
        ...
9995     24.4
9996     45.4
9997     48.1
9998     32.1
9999     45.1
Name: Volatility, Length: 10000, dtype: float64
```

FIRST STAGE OF MODELLING

Train /test split

```
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
```

Define and Train HMM

```
model = GaussianHMM(n_components=3, covariance_type='full', n_iter=100, random_state=42)
model.fit(X_train)
```

GaussianHMM

```
GaussianHMM(covariance_type='full', n_components=3, n_iter=100, random_state=42)
```

Predict on training and test set

```
train_states = model.predict(X_train)
test_states = model.predict(X_test)

global_df['State'] = np.concatenate([train_states, test_states])
```

FIRST STAGE OF MODELLING

Define features

```
# Calculate the range volatility (difference between high and low prices)
global_df['RangeVol'] = global_df['high'] - global_df['low']

# Calculate rolling standard deviation (volatility) over the last 10 periods
global_df['VolCluster'] = global_df['close'].rolling(window=10).std()

# Calculate the upper wick (distance from the close or open to the high)
epsilon = 1e-6  # Small constant to avoid division by zero
global_df['upper_wick'] = global_df['high'] - global_df[['open', 'close']].max(axis=1)

# Calculate the lower wick (distance from the close or open to the low)
global_df['lower_wick'] = global_df[['open', 'close']].min(axis=1) - global_df['low']

# Calculate the wick ratio (relative size of the wicks)
global_df['wick_ratio'] = (global_df['upper_wick'] + global_df['lower_wick']) / (global_df['high'] - global_df['low'] + epsilon)
```

✓ 0.0s

Handle missing data

```
global_df = global_df.dropna(subset=['VolCluster']).reset_index(drop=True)
```

✓ 0.0s

Standardize features

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
global_df[['RangeVol', 'VolCluster', 'wick_ratio']] = scaler.fit_transform(
    global_df[['RangeVol', 'VolCluster', 'wick_ratio']]
)
```

✓ 0.1s

FIRST STAGE OF MODELLING

Evaluate Performance metric

```
sharpe = global_df['strategy_return_net'].mean() / global_df['strategy_return_net'].std() * np.sqrt(252)

cumulative_return = cumulative.iloc[-1] - 1

drawdown = cumulative / cumulative.cummax() - 1
max_dd = drawdown.min()

trade_freq = (global_df['signal'].diff() != 0).sum() / len(global_df)

print(f"Sharpe Ratio: {sharpe:.2f}")
print(f"Total Return: {cumulative_return:.2%}")
print(f"Max Drawdown: {max_dd:.2%}")
print(f"Trade Frequency: {trade_freq:.2%}")
```

✓ 0.0s

```
Sharpe Ratio: -0.11
Total Return: -5.74%
Max Drawdown: -7.84%
Trade Frequency: 2.23%
```

FIRST STAGE OF MODELLING


```
Sharpe Ratio: -0.17
Total Return: -9.03%
Max Drawdown: -10.98%
Trade Frequency: 2.95%
```

Ctrl+Defeat

That’s all from us.
Thanks