STATFLOW STRATEGY

PRESENTED BY: KPO MEMBERS

DOMAIN 2: QUANTITATIVE TRADING

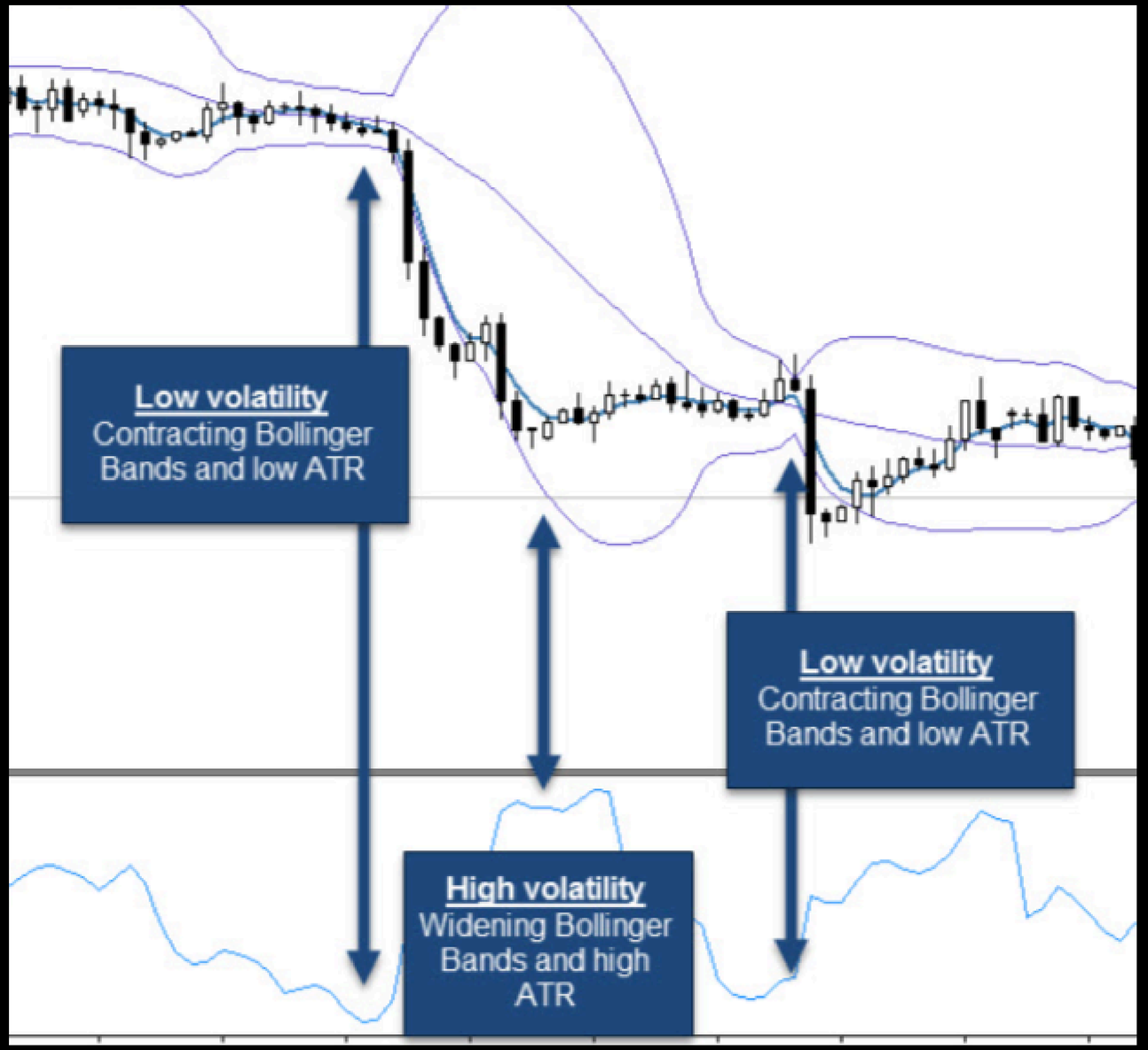
Problem Statement

Problem 1: Traditional Momentum Strategies Are Easily Fooled by Market Noise

Problem 2: High Volatility Periods Cause Unpredictable Losses

Problem 3: Momentum Signals Are Prone to False Breakouts


Euro / U.S. Dollar, 1D, IDC
EMA (200, close)
Negative Momentum
1.21000
1.20000
1.19000
1.18000
1.17000
1.16000
1.15000
1.14000
1.13000
1.12000
1.11000
1.10239
1.09000
Jul
Sep
Nov
2019
Mar
May
Jul
Sep


Low volatility
Contracting Bollinger Bands and low ATR
Low volatility
Contracting Bollinger Bands and low ATR
High volatility
Widening Bollinger Bands and high ATR

PROPOSED SOLUTION

What is the **StatFlow Strategy?**

Volatility-Aware Momentum (VAM) is a trading strategy that combines momentum signals with a volatility filter. It enters trades only when the price is trending strongly and market volatility is within a safe range this help to reduce false signals and manage risk effectively.

Sharpe Ratio (Annualized, Hourly Data): 1.4676

How the **StatFlow Strategy Works**

returns_t = (Close_BTC-USD_t - Close_BTC-USD_{t-1}) / Close_BTC-USD_{t-1}

price_momentum_t = (Close_BTC-USD_t - Close_BTC-USD_{t-3}) / Close_BTC-USD_{t-3}

volatility_t = std(Close_BTC-USD_{t-9}, ..., Close_BTC-USD_t)

flow_z_t = (flow_momentum_t - mean(flow_momentum_{t-19}, ..., flow_momentum_t)) / std(flow_momentum_{t-19}, ..., flow_momentum_t)

volatility_filter_t = 1 if volatility_t < mean(Close_BTC-USD)
volatility_filter_t = 0 if volatility_t >= mean(Close_BTC-USD)

combo_signal_t = 0.3 * flow_momentum_t + 0.3 * inflow_mean_t + 0.2 * price_momentum_t + 0.2 * volatility_filter_t

How the **StatFlow Strategy Works**

Trading Signal Generation

threshold = quantile(combo_signal, 0.8)

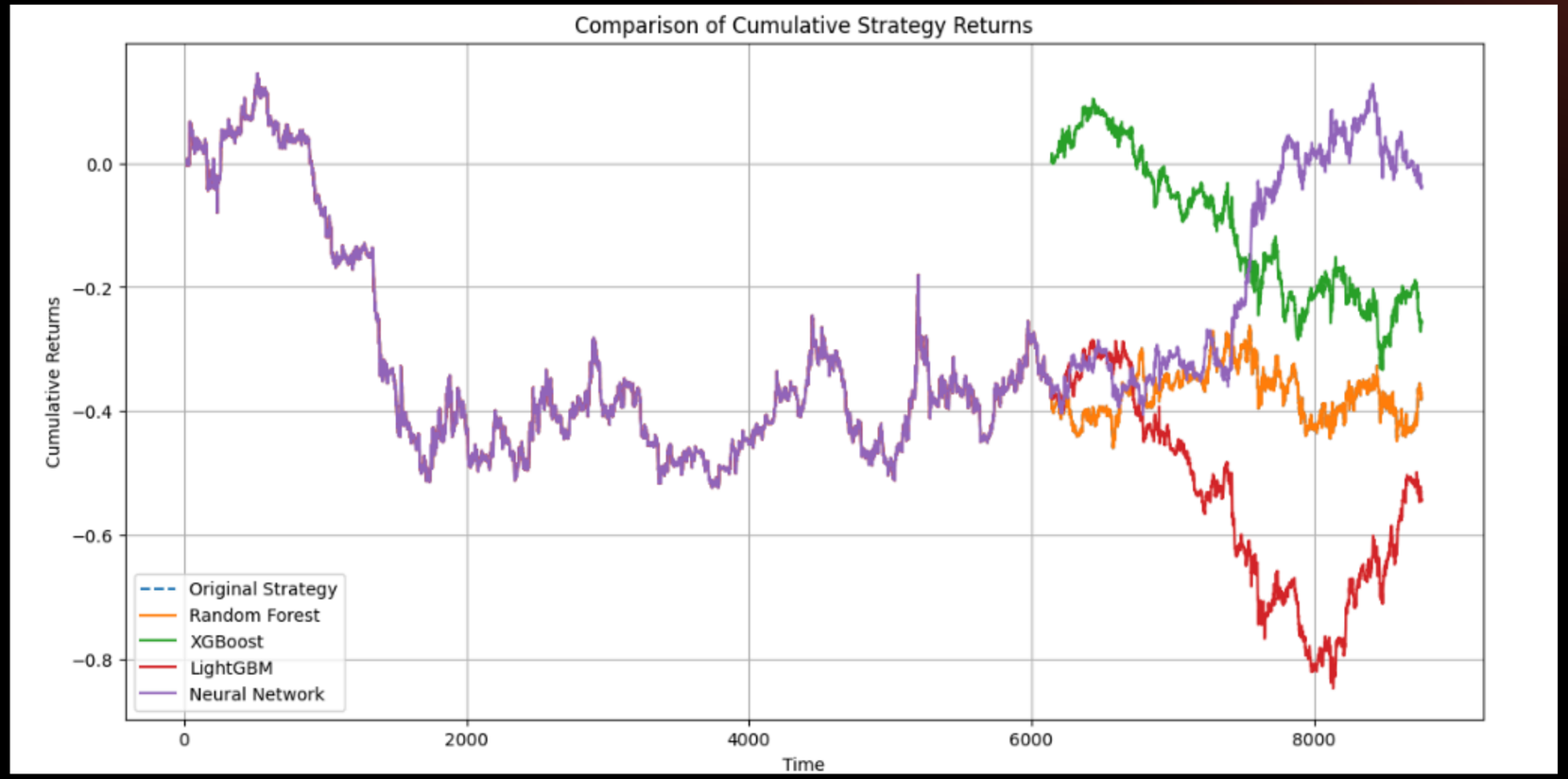
Trading Logic:

- position_t = 1 if combo_signal_t > threshold (Long)
- position_t = -1 if combo_signal_t <= threshold (Short)

Business Value

- Combines momentum and volatility filters to avoid false signals
- Proactively avoids trades during high volatility, reducing drawdowns
- More Predictable Returns: Reduces surprises from sudden market swings.
- Scalable Automation: Easily expanded to multiple assets or strategies.

Machine Learning




```
RandomForestClassifier Accuracy: 0.5186737804878049
XGBoost Accuracy: 0.507621951219512l
LightGBM Accuracy: 0.5
Neural Network Accuracy: 0.5152439024390244
```

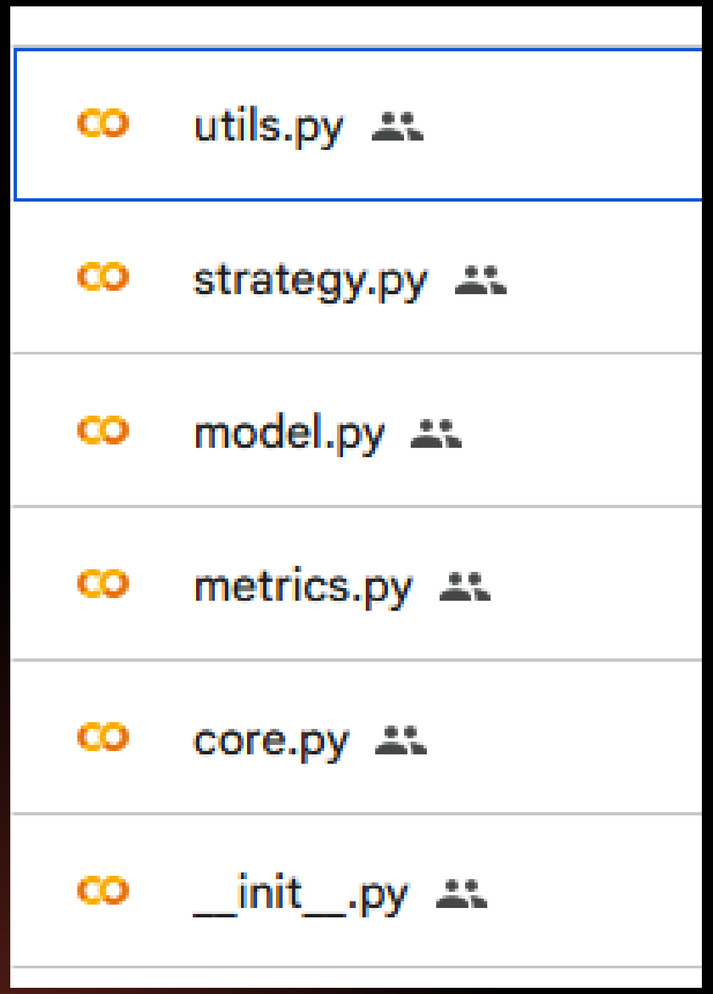
Choosen Model

LSTM Accuracy: 0.5196153846153846

What Our Model do?

- Learns from past 10 time steps of flow + momentum features.
- Predicts whether BTC price will rise next time step.
- Generates binary trading signals.
- Simulates a basic long-short strategy.
- Plots cumulative P&L to evaluate performance.

Architectural Design of Backtest Library



1. Core.py—This is likely where the main engine lives — stuff like data handling, orchestration of strategy logic, or managing the pipeline (e.g., loading prices → calculating metrics → deciding trades).

2. metrics.py - it's the calculator. It computes things like momentum, volatility, returns, maybe drawdowns. Used to filter stocks or generate signals.

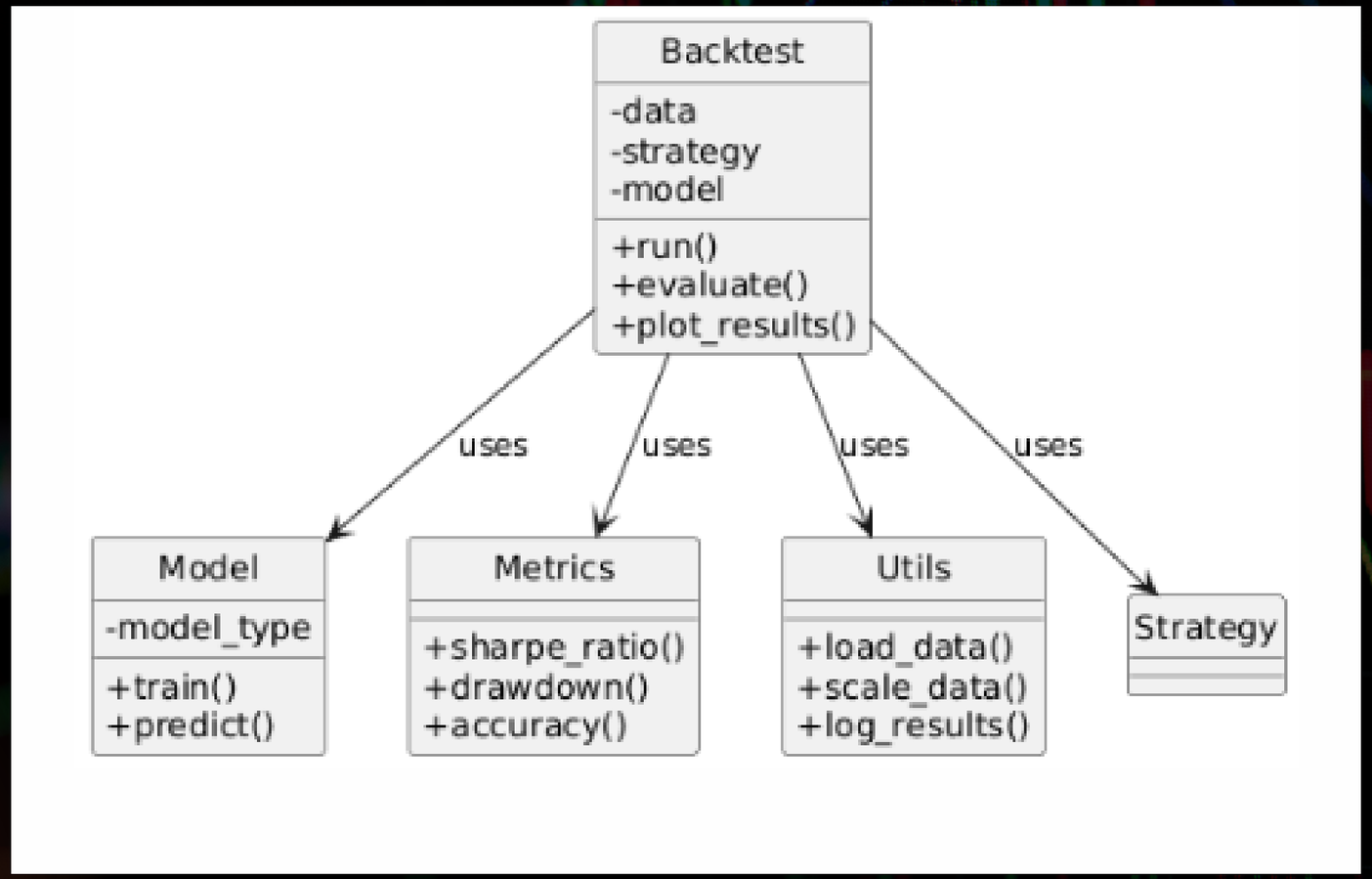
3. model.py - define a machine learning model, or a rule-based decision system. Could be what actually makes the buy/sell call after seeing metrics.

4. strategy.py - "If momentum > X and volatility < Y → buy stock"

5. utils.py - Smoothing data, Converting dates, Normalizing values

6. __init__.py - This makes your folder a Python package. It's like the "hey, I'm ready to be imported" file. Might also expose key functions or classes from the library.

Conceptual Diagram




THANK
YOU