Design patterns

Pipeline Pattern

The pipeline pattern processes data through a sequence of steps, with each step transforming the input for the next.
Usage:
This pattern is evident in:
- Technical Analysis:

```
def get_signal_dataframe(data):
    return aggregate_signals(
        generate_signals(
            calculate_indicators(
                preprocess_data(data)
            )
        )
    )
```

- Steps: Data preprocessing -> Indicator calculation -> Signal generation -> Signal aggregation.
- Ensures modular and reusable components for financial data analysis.

Model-View-Controller (MVC)

MVC separates concerns into three components:
- Model: Data and business logic.
- View: User interface or output.
- Controller: Coordinates the model and view.
Usage:
- Main Pipeline (Main.py):
  -Model: SQL database and LSTM model.
  - View: JSON responses for endpoints.
  - Controller: Flask app routes coordinate between the database and analysis functions.

Functional Programming Style

Independent functions with no side effects process and transform data in a clean and reusable manner.
Usage:
- Example Functions:
  - preprocess_data: Prepares raw data for analysis.
  - calculate_indicators: Computes financial indicators.
  -generate_signals: Creates buy/sell/hold signals.
- Benefits: Functions remain modular and can be independently tested.

Conclusion

The provided codebase demonstrates the effective application of multiple design patterns, ensuring modularity, scalability, and maintainability. These patterns facilitate the integration of diverse functionalities like data scraping, analysis, machine learning, and web application development.