📝Articlel

📌Overview

Articlel is a Python-based supervised learning framework designed for stock trading prediction.
The project provides tools to generate adaptive BUY/HOLD/SELL labels, extract technical features, train machine learning models, and evaluate predictions using comprehensive metrics and visualizations.

The main goal is to create actionable trading signals based on historical stock data and technical indicators while accounting for market volatility and risk-adjusted returns.

🕐Objectives

- Generate adaptive trading targets using multi-horizon returns, technical indicators, and risk-adjusted scoring.
- Build a supervised learning pipeline for predicting trading actions (BUY, HOLD, SELL).
- Develop and train machine learning models to predict adaptive trading signals.
- Evaluate model performance and visualize results effectively.

📂Repository Structure

File	Purpose
main.py	Main orchestration script: coordinates labeling, feature extraction, model training, evaluation, and visualization.
labeling.py	Adaptive labeling module: calculates multi-horizon returns, technical indicator scores, and generates BUY/HOLD/SELL targets.
baselines.py	Implements simple baseline trading strategies (MACD crossover, RSI thresholds, random) for benchmarking.
features.py	Feature extraction and preprocessing pipeline from stock data.
models.py	Core ML models, including training, hyperparameter tuning, and prediction logic.
graph_intro.py	Visualization scripts for metrics, results, and action distributions.
evaluate.py	Evaluation of model predictions against generated targets using supervised learning metrics and visualizations. (rename from evaluate (1).py for clarity)

⚙️Installation & Setup

Clone the repository:

```
git clone https://github.com/Hichri-Hassan/Articlel.git
cd Articlel
```

(Optional) Create a virtual environment:

```
python3 -m venv venv
source venv/bin/activate   # Windows: venv\Scripts\activate
```

Install dependencies:

```
pip install -r requirements.txt
```

(If no requirements.txt yet, generate one with `pip freeze > requirements.txt`.)

🗜️ Usage Examples

Run baseline models:

```
python baselines.py --input data/your_stock_data.csv --output results/
```

Extract features:

```
python features.py --source data/your_stock_data.csv --out features.pkl
```

Train models:

```
python models.py --config configs/model_config.json
```

Evaluate predictions:

```
python evaluate.py --pred predictions.csv --true ground_truth.csv
```

Generate graphs/visuals:

```
python graph_intro.py --metrics results/metrics.json --out figures/
```

Orchestrate full pipeline:

```
python main.py --input data/your_stock_data.csv --config configs/
model_config.json --results results/ --figures figures/
```

📊Methodology

Data Preparation & Feature Engineering
Features include historical returns, volatility, RSI, MACD, Bollinger Bands, moving averages, volume, momentum, and other statistical indicators.

Adaptive Labeling (labeling.py)

- Generates BUY/HOLD/SELL labels based on multi-horizon returns and risk-adjusted scores.
- Dynamically adjusts thresholds for different market conditions (high vs low volatility).
- Creates time-based rolling thresholds to adapt to market trends.

Baseline Strategies (baselines.py) Provides simple benchmarks to compare against advanced ML models:

- MACD Crossover Strategy: BUY when MACD > signal, SELL when MACD < signal.
- RSI Threshold Strategy: BUY when RSI < 30 (oversold), SELL when RSI > 70 (overbought).
- Random Strategy: Random BUY/SELL/HOLD assignments for comparison.

Evaluation Module (evaluate.py)

- Computes supervised learning metrics: Accuracy, Precision, Recall, F1-score, and ROC AUC.
- Generates visual reports such as confusion matrices, ROC/PR curves, and detailed classification reports.
- Includes a trading performance simulation to compare cumulative returns of model signals against a baseline.
- Confirms the framework as a supervised classification system since predictions are compared directly to ground truth labels.

Machine Learning Models (models.py)

- Implements supervised ML models for classification of trading signals.
- Includes model training, hyperparameter tuning, and prediction generation.
- Supports multiple algorithms for robust experimentation and benchmarking.

Evaluation & Visualization

- Predictions are compared against generated targets.
- Metrics and visualizations allow analysis of model accuracy and trading signal distributions.

📖Future Improvements

- Include Jupyter notebooks for demonstrations.
- Integrate deep learning models for enhanced prediction.

- Provide pre-trained weights for reproducibility.
- Extend evaluation metrics to include profit/loss simulations.

Contributing

Contributions are welcome!

- Fork the repo
- Create a new branch (feature/your-feature)
- Commit changes
- Open a pull request

📬 Contact

Developed by **Hicheri Hassen**.
For questions, open an issue on GitHub or contact me directly.
