Stock Predictor Framework – ELI5 Guide

Simple words, small steps.

# What is this?

It is a helper robot for stocks. It looks at past prices, learns patterns, and makes a tiny guess about tomorrow. Then it tests a simple rule to see if the guesses could make money.

# The big pieces (folders)

* data: gets price data from the internet and saves it on your computer.
* features: makes smart numbers (indicators) from prices.
* models: teaches a model to guess tomorrow’s return and saves it.
* backtest: plays pretend trading with the guesses to see results.
* cli: buttons you can push in the terminal to run steps.
* streamlit\_app.py: a simple app window with buttons.

# Data – where numbers come from

We use yfinance to download stock prices (Open, High, Low, Close, Volume). We save them as files so we don’t have to download again.

# Features – making smart numbers

* Moving averages (SMA/EMA): smooth the price.
* RSI/MACD/Bollinger: popular indicators to show trend/momentum/volatility.
* Returns: how much price changed over 1, 5, 10 days.
* Lags: we shift features by 1 day to avoid cheating with future info.
* Target: the thing we guess – next day return (percent change).

# Model – the brain

* RandomForest (default): simple and works on most computers.
* XGBoost (optional): sometimes better but needs a macOS library (libomp).
* We split the data by time: train, validate, test. No peeking ahead!
* We save the model to the models folder so we can use it later.

# Backtest – playing pretend

* Rule: if the model says tomorrow is positive by more than a threshold, we buy; otherwise, we hold cash.
* We include simple trade cost.
* We compare to Buy & Hold.
* We look at results: final money (equity), annual return, Sharpe (risk-adjusted), win rate.

# How to run things (easy steps)

* Make a virtual environment and install requirements.
* Train: python -m src.stock\_predictor.cli train AAPL --start 2018-01-01 --end 2024-12-31 --model-type rf
* Backtest: python -m src.stock\_predictor.cli run-backtest AAPL --start 2019-01-01 --end 2024-12-31 --threshold 0.0
* App: streamlit run streamlit\_app.py

# Real-time trading – ELI5

Trading live means: get today’s price, make a fresh guess, decide to buy or not, and place an order at your broker.

* Choose a broker API (Alpaca, Interactive Brokers, Robinhood, etc.).
* Each broker gives you keys (like passwords) and a small library to send orders.
* Write a small script that runs every day after market close:
* 1) Download latest data
* 2) Make features (same steps as training)
* 3) Load saved model and predict tomorrow’s return
* 4) If prediction > threshold, send BUY order; else send SELL/close order.

## Real-time trading – more detail (next steps)

* Scheduler: use cron (Mac/Linux) or launchd to run your script at 3:55pm local time.
* Position sizing: start with small size. You can do a fixed dollar amount or percent of cash.
* Risk: set max position, stop-loss, and a daily limit.
* Logging: save what you did and why (prediction, price, order status).
* Paper trading first: most brokers have a paper (fake money) environment. Try there before real money.

# Where to change things (scope for modification)

* features/engineering.py: add or remove indicators, change windows.
* models/train.py: swap models, tune hyperparameters, add walk-forward CV.
* backtest/backtest.py: add position sizing, shorting, slippage, portfolio of many tickers.
* data/ingest.py: switch to different data source or intraday interval (like 1h).
* cli.py & streamlit\_app.py: add new options and visualizations.

# Safety and expectations

* This is not financial advice.
* Daily returns are noisy: don’t expect magic. Improve step by step.
* Test with paper trading, start small, add risk controls.