StockAI Pipeline Overview

StockAI (defined in ai_predictor.py) orchestrates end-to-end price forecasting. It pulls one year of OHLCV data from Yahoo Finance, enriches it with dozens of technical indicators via StockCalculations, trains a RandomForestRegressor on those engineered features, and complements the statistical forecast with a scenario-based simulation that projects the next 150 trading days under controlled trend, volatility, and cyclical assumptions.

Data Acquisition

1. Symbol normalization appends .NS for Indian equities before requesting data. 2. Yahoo Finance provides a minimum 200 trading days of Open/High/Low/Close/Volume. 3. Rows containing NaNs or missing mandatory columns are discarded to keep the learning set consistent.

Technical Indicator Engine

StockCalculations derives multiple indicator families before any modeling occurs: - Trend/price action: SMA20/50/200, ADX, and custom trend heuristics. - Momentum: RSI plus MACD, its signal line, and histogram. - Volatility: Bollinger Bands (upper/middle/lower) and Average True Range. - Volume: On Balance Volume and rolling volume summaries. These enriched columns populate both the visualization layer and the ML feature set.

Feature Matrix For Random Forest

prepare_features() constructs the supervised learning table. Each row includes: Returns (daily % change), RSI, MACD, Bollinger upper/lower bands, raw Volume, and the plain OHLC columns. NaNs are dropped and all values cast to float, ensuring the MinMaxScaler can normalize the matrix before fitting the forest.

Model Training And Prediction

predict_price() shifts the Close column by -1 to create a next-day target. After scaling the features, a RandomForestRegressor (100 estimators, seed 42) fits the full historical window each time a prediction is requested. The most recent feature vector becomes the inference input, producing the upcoming closing price.

Deterministic Future Path Simulation

predict_future_prices() provides a complementary projection for ~5 months of trading days. It blends several deterministic components: - Trend strength computed from 20/50/200-day returns. - Volatility estimated from historical return std-dev and modulated with a sine wave to mimic regime changes. - Market cycles injected via low-frequency sine/cosine terms. - A hard 3% daily cap prevents unrealistic jumps. This routine seeds NumPy's RNG for reproducible trajectories.

Visualization Layer

Two plotting utilities summarize both the backtest window and projections. generate_prediction_graph() builds a 2x2 dashboard (price+MAs, RSI, MACD, Bollinger) with confidence zones, while generate_future_prediction_graph() overlays historical closes and the simulated forward path with confidence bands and a regression trend line.

Market Context And Confidence

StockCalculations augments raw predictions with qualitative metrics: - analyze_market_regime() inspects SMA stacking and ADX to classify bullish, bearish, or sideways regimes. - calculate_confidence() blends RSI distance from neutral, MACD magnitude, SMA trend alignment, and annualized volatility to produce a bounded 0-100 confidence score. - analyze_sentiment() scrapes Google News headlines, applies TextBlob polarity, and backs off to technical sentiment if news is sparse. These signals accompany the Random Forest output in predict_stock().