Stock Direction Prediction — Code Explanation

This document explains the project's structure, what has been used so far, and a line-by-line explanation of core files.

What you've used so far

- Python 3.8+ environment (`venv`) with packages listed in `requirements.txt`.

- Data: per-ticker CSV files in `data/raw/` (NIFTY 50 constituents and `NIFTY50_all.csv`).

- Feature engineering: `src/features.py` computes returns, lag features and technical indicators (SMA, EMA, RSI, MACD, Bollinger Bands, rolling volatility).

- Baseline training: `src/train_baseline.py` runs time-series CV with `LogisticRegression` and saves a final model.

- Batch training: `src/train_all.py` trains the baseline across all CSVs and writes `models/metrics.csv`.

- EDA: `notebooks/eda.py` produces `notebooks/eda_summary.csv` and sample visualizations.

- Advanced training: `src/train_advanced.py` (this file) trains `RandomForest` and `XGBoost` per ticker and writes feature importances and `advanced_metrics.json`.

File-by-file explanation

`src/data.py`

- `ensure_dir(path)`: utility to create directories.

- `download_ticker(ticker, start, end, out_dir)`: uses `yfinance` to download ticker history and save CSV into `out_dir`.

- `download_tickers_from_list(ticker_list, start, end, out_dir)`: loops over tickers and downloads each.

- `load_csv(path)`: loads a CSV into a Pandas DataFrame with date index.

- `if __name__ == '__main__'`: provides CLI to download tickers from a comma list or file.

`src/features.py` (line-level highlights)

- `_resolve_price_col(df, preferred='Close')`:

- Tries to find a suitable price column among common names: `Close`, `Adj Close`, `Last`, etc.

- Returns the matching column name or raises KeyError.

- `add_returns(df, price_col='Close')`:

- Uses `_resolve_price_col` to select the price column, computes `pct_change()` and stores in `df['return']`.

- `add_direction_label(df, price_col='Close')`:

- Creates `direction` label: 1 if next day's price > today's price else 0. Implemented via `shift(-1)` on the price column.

- `add_lags(df, cols=['Close'], n_lags=5)`:

- Adds lagged columns for each column requested (resolves price column names if necessary).

- `sma`, `ema`, `rsi`, `macd`, `bollinger_bands`:

- Technical indicator helpers. `rsi` uses rolling averages of gains and losses.

- `add_technical_indicators(df, price_col='Close')`:

- Adds SMA(10,20), EMA(12,26), MACD line & signal, RSI(14), Bollinger bands and BB width, and rolling vol (10,20).

- `prepare_for_classification(df, price_col='Close', n_lags=5, add_tech=True, dropna=True)`:

- Full pipeline: returns, optional technicals, direction label, lags, dropna, and returns numeric features X and label y.

`src/train_baseline.py`

- Reads a single ticker CSV, calls `prepare_for_classification`, performs `TimeSeriesSplit` (n_splits=5), trains `LogisticRegression` on each fold, stores fold accuracies, trains final model on all data and saves it to `models/baseline_lr.pkl`.

- CLI: `--csv` path and `--out` model path.

`src/train_all.py`

- Finds all `*.csv` files in `data/raw/`, calls the baseline training per CSV (using `prepare_for_classification`), saves per-ticker models into `models/` and writes `models/metrics.csv` with fold accuracies and mean accuracy.

`src/train_advanced.py` (this file)

- Trains `RandomForestClassifier` (n_estimators=100 in CV, 200 final) and (optionally) `XGBoost` if `xgboost` is installed.

- Uses `TimeSeriesSplit` (n_splits=5) and computes per-fold metrics (accuracy, precision, recall, f1).

- Saves final RF model and feature importances per ticker under `models_advanced/` and aggregates metrics in `models_advanced/advanced_metrics.json`.

How the pieces fit together (workflow)

1. Place CSVs in `data/raw/`.

2. Run `python -m src.train_all` to train the logistic baseline across tickers.

3. Run `python notebooks/eda.py` to inspect dataset balance and visualize indicators.

4. Run `python -m src.train_advanced` to train RandomForest/XGBoost and save advanced metrics.

5. Use the Streamlit app (`streamlit run app/app.py`) to interact with models (upload CSV, show predictions).

Progress summary (what was run)

- Baseline training across tickers: `python -m src.train_all` produced `models/metrics.csv`.

- EDA summary: `python notebooks/eda.py` produced `notebooks/eda_summary.csv` and sample figures.

- Advanced training scaffolding added in `src/train_advanced.py` (callable via `python -m src.train_advanced`).

- Documentation PDF generated at `docs/code_explanation.pdf` from `docs/code_explanation.md`.

Inventory of data files (data/raw)

The following CSV files are present in `data/raw/`:

ADANIPORTS.csv

ASIANPAINT.csv

AXISBANK.csv

BAJAJ-AUTO.csv

BAJAJFINSV.csv

BAJFINANCE.csv

BHARTIARTL.csv

BPCL.csv

BRITANNIA.csv

CIPLA.csv

COALINDIA.csv

DRREDDY.csv

EICHERMOT.csv

GAIL.csv

GRASIM.csv

HCLTECH.csv

HDFC.csv

HDFCBANK.csv

HEROMOTOCO.csv

HINDALCO.csv

HINDUNILVR.csv

ICICIBANK.csv

INDUSINDBK.csv
INFRATEL.csv
INFY.csv
IOC.csv
ITC.csv
JSWSTEEL.csv
KOTAKBANK.csv
LT.csv
MARUTI.csv
MM.csv
NESTLEIND.csv
NIFTY50_all.csv
NTPC.csv
ONGC.csv
POWERGRID.csv
RELIANCE.csv
SBIN.csv
SHREECEM.csv
stock_metadata.csv
SUNPHARMA.csv
TATAMOTORS.csv
TATASTEEL.csv
TCS.csv
TECHM.csv
TITAN.csv
ULTRACEMCO.csv
UPL.csv
VEDL.csv
WIPRO.csv
ZEEL.csv

Installed packages (venv - pip freeze)

Key packages and versions installed in the virtual environment (output of `pip freeze`):

- pandas==2.3.3
- numpy==2.3.3
- scikit-learn==1.7.2
- xgboost==3.0.5

- ta==0.11.0
- yfinance==0.2.66
- matplotlib==3.10.7
- seaborn==0.13.2
- tensorflow==2.20.0
- keras==3.11.3
- streamlit==1.50.0
- reportlab==4.4.4
- joblib==1.5.2

For a complete list refer to the `pip freeze` output included in the project environment.