Stock Trend Predictor - Complete User Manual

📋 Table of Contents

📊 Project Overview

What This Does

- ✅ Predicts stock price direction (UP/DOWN) for next trading day
- ✅ Analyzes 40+ technical indicators (RSI, MACD, Bollinger Bands, etc.)
- ✅ Integrates news sentiment analysis from recent articles
- ✅ Supports multiple ML models (Random Forest, LightGBM, LSTM, XGBoost, Ensemble)
- ✅ Provides web interface and command-line interface
- ✅ Trains on single or multiple stocks for better generalization

What This Doesn't Do

- ❌ Predict exact prices (only direction)
- ❌ Guarantee profits (markets are unpredictable)

- ❌ Provide financial advice (educational only)

Expected Accuracy

- Single stock: 52-58%
- Multi-stock balanced: 58-65%
- Advanced training: 62-72%

🛠️ Installation

Step 1: Install Python Dependencies

```bash
# Install all requirements
pip install -r requirements.txt
```

Core Requirements:

- Python 3.8+
- yfinance (stock data)
- pandas, numpy (data processing)
- ta (technical analysis)
- scikit-learn, lightgbm, xgboost (ML models)
- tensorflow (LSTM models)
- newsapi-python, vaderSentiment (sentiment analysis)
- flask (web application)

Step 2: Get NewsAPI Key (Free)

1. Go to: https://newsapi.org/register
2. Sign up (free, no credit card)
3. Copy your API key
4. Open `src/sentiment_config.py`
5. Replace line 8: `'api_key': 'YOUR_KEY_HERE'`

Step 3: Verify Installation

```bash
python -c "import yfinance, lightgbm, sklearn, tensorflow, flask; print('✅ All packages installed')"
```

🚀 Quick Start

Option 1: Web Interface (Recommended for Demos)

```bash
# 1. Start web server
python app.py

# 2. Open browser
# http://localhost:5000

# 3. Enter stock ticker (e.g., AAPL)
# 4. Click "Predict"
# 5. View results!
```

Option 2: Command Line

```bash
# 1. Train a model (15 minutes)
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN NVDA --model ensemble --max-features 35

# 2. Make prediction
python src/predict.py AAPL

# 3. View results in terminal
```

🎓 Training Models

You have **three training scripts** to choose from:

Script	When to Use	Time	Accuracy
`main.py`	Single stock, basic	2-3 min	52-58%
`train_multi.py`	Multiple stocks, balanced	5-10 min	58-65%
`train_advanced.py`	**Best accuracy**	10-20 min	62-72%

Option 1: Single Stock Training (`main.py`)

Train on one stock's historical data.

```bash
# Basic usage
python src/main.py AAPL --model ensemble

# With options
python src/main.py AAPL \
    --model lightgbm \
    --features basic momentum volatility \
    --start 2020-01-01 \
    --end 2024-12-31
```

Available options:

- `--model`: random_forest, lightgbm, lstm, stacker, ensemble
- `--features`: basic, momentum, volatility, volume, patterns, advanced
- `--start`: Start date (YYYY-MM-DD)
- `--end`: End date (YYYY-MM-DD)
- `--output`: Output model path

Examples:

```bash
# Train Random Forest on Apple
python src/main.py AAPL --model random_forest

# Train LightGBM with specific features
python src/main.py TSLA --model lightgbm --features basic momentum volatility
```

Option 2: Multi-Stock Training (`train_multi.py`)

Train on multiple stocks for better generalization.

```bash
# Using presets
python src/train_multi.py --preset tech --model ensemble --balance

# Custom stock list
python src/train_multi.py \
    --tickers AAPL MSFT GOOGL AMZN NVDA \
    --model lightgbm \
    --balance
```

Available presets:

- `tech`: AAPL, MSFT, GOOGL, AMZN, META, NVDA, TSLA, AMD
- `sp500_sample`: 20 diverse stocks
- `diverse`: 10 stocks across sectors
- `mega_cap`: 8 largest companies

Available options:

- `--preset` or `--tickers`: Choose stocks
- `--model`: Model type
- `--balance`: **Balance UP/DOWN samples (recommended!)**
- `--features`: Feature groups to use
- `--start/--end`: Date range
- `--no-fetch`: Skip downloading (use cached data)

Examples:

```bash
```

```
# Tech stocks with balancing (recommended)
python src/train_multi.py --preset tech --model ensemble --balance

# Custom list with specific features
python src/train_multi.py \
    --tickers AAPL JPM XOM JNJ WMT \
    --model lightgbm \
    --features basic momentum volatility \
    --balance

# S&P 500 sample for diverse model
python src/train_multi.py --preset sp500_sample --model stacker --balance
```

Option 3: Advanced Training (`train_advanced.py`) ⭐ RECOMMENDED

Best accuracy with feature selection, augmentation, and optimization.

bash

```
# Basic usage
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN NVDA \
    --model lightgbm

# Full options
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD JPM BAC \
    --model ensemble \
    --start 2018-01-01 \
    --max-features 35
```

What makes it "advanced":

- ✅ Automatic feature selection (picks top N most important)
- ✅ Data augmentation (creates synthetic samples)
- ✅ Lagged features (uses past 1-3 days)
- ✅ Rolling features (moving averages, std dev)
- ✅ Feature scaling (normalizes all features)
- ✅ Cross-validation (5-fold time series CV)

Available options:

- `--tickers`: Stock symbols (required)
- `--model`: lightgbm, random_forest, stacker, ensemble
- `--start/--end`: Date range
- `--max-features`: Number of features to select (default: 30)
- `--no-augmentation`: Disable data augmentation
- `--no-feature-selection`: Use all features
- `--no-scaling`: Don't scale features

Examples:

```bash
# Quick test (5 minutes)
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL \
    --model lightgbm \
    --max-features 25

# Best accuracy (15 minutes) ⭐ RECOMMENDED
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD JPM BAC \
    --model ensemble \
    --start 2018-01-01 \
    --max-features 35

# Fast with good accuracy (8 minutes)
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN NVDA TSLA META AMD \
    --model lightgbm \
    --max-features 30 \
    --start 2019-01-01
```

🔮 Making Predictions (CLI)

Basic Prediction

```bash
```

```
# Predict for Apple (with sentiment)
python src/predict.py AAPL
```

Prediction Options

```bash
# Without sentiment analysis
python src/predict.py AAPL --no-sentiment

# Using specific model
python src/predict.py AAPL model/advanced_ensemble_model.pkl

# Using custom model without sentiment
python src/predict.py TSLA model/custom_model.pkl --no-sentiment
```

Predict Multiple Stocks

```bash
# One by one
python src/predict.py AAPL
python src/predict.py MSFT
python src/predict.py GOOGL
python src/predict.py TSLA
python src/predict.py NVDA
```

Batch Predictions

Windows (batch file):

```batch
@echo off
echo Predicting Tech Stocks...
python src/predict.py AAPL
python src/predict.py MSFT
python src/predict.py GOOGL
python src/predict.py AMZN
python src/predict.py NVDA
echo Done!
```

Save as `predict_batch.bat` and run it.

Linux/Mac (bash script):

```bash
#!/bin/bash
echo "Predicting Tech Stocks..."
for ticker in AAPL MSFT GOOGL AMZN NVDA TSLA META; do
    echo "==== $ticker ===="
    python src/predict.py $ticker
    echo ""
done
echo "Done!"
```

Save as `predict_batch.sh`, make executable with `chmod +x predict_batch.sh`, then run `./predict_batch.sh`.

Understanding CLI Output

```
============================================================
STOCK PREDICTION RESULTS
============================================================

Last trading day: 2025-01-10
Last close price: $175.43

============================================================
MODEL PREDICTION:
============================================================
📈 PREDICTION: UP tomorrow
Confidence: 68.7%

Individual Model Predictions:
  - random_forest: UP
  - lightgbm: UP
  - stacker: DOWN

============================================================
NEWS SENTIMENT ANALYSIS:
============================================================

✅✅ Overall Sentiment: POSITIVE
   Sentiment Score: +0.45 (range: -1 to +1)
📊 Sentiment Trend: Improving
📑 Articles Analyzed: 12 (last 3 days)
```

```
============================================================
COMBINED ANALYSIS:
============================================================

✅✅ Strong signal - Both technical indicators and news agree
Overall Confidence: HIGH

Breakdown:
  • Technical Indicators: UP (68.7% confidence)
  • News Sentiment: POSITIVE (0.45 strength)

💡 Suggestion: This is a high-confidence signal. Both technical and fundamental indicators align.

============================================================
RECENT TECHNICAL INDICATORS:
============================================================
  RSI: 58.32 (Neutral)
  MACD: 2.14 (Bullish)
  Volatility: 0.0234 (Normal)

============================================================
⚠️  DISCLAIMER: Not financial advice. For educational purposes only.
============================================================
```

🌐 Web Application

Starting the Web App

bash

```bash
# Make sure you're in project root
cd stock-trend-predictor

# Start the server
python app.py
```

You should see:

```
🚀 Starting Stock Trend Predictor Web App...
📊 Access at: http://localhost:5000
 * Running on http://127.0.0.1:5000
```

Accessing the Web Interface

Open your browser and go to:

- **Local**: http://localhost:5000
- **Network**: http://YOUR_COMPUTER_IP:5000 (accessible from other devices on same network)

Using the Web Interface

1. **Enter Stock Ticker**
   - Type any stock symbol (AAPL, TSLA, GOOGL, etc.)
   - Must be 1-5 characters
   - Press Enter or click "Predict"
2. **Toggle Sentiment Analysis**
   - Check/uncheck "Include News Sentiment"
   - Useful if you want technical-only analysis
3. **View Results**
   - **Prediction Box**: Shows UP/DOWN with confidence
   - **News Sentiment**: Overall sentiment from articles
   - **Combined Analysis**: Agreement between model and news
   - **Technical Indicators**: RSI, MACD, Volatility displayed
   - **Individual Models**: Votes from each model (for ensemble)

Web Interface Features

- 🎨 Beautiful gradient design
- 📱 Mobile responsive (works on phones/tablets)
- ⚡ Real-time predictions
- 🎯 Clean, professional look
- 📊 Visual confidence bars

- 🔄 Smooth animations
- ❌ Clear error messages
- ⏳ Loading indicators

Customizing the Web App

Change Port: Edit `app.py`, line 212:

```
python

app.run(debug=True, host='0.0.0.0', port=8080)  # Change to 8080
```

Change Default Model: Edit `app.py`, line 18:

```
python

MODEL_PATH = "model/your_custom_model.pkl"
```

Disable Debug Mode (Production):

```
python

app.run(debug=False, host='0.0.0.0', port=5000)
```

📰 News Sentiment Analysis

How It Works

1. **Fetches News**: Gets last 3 days of articles for the stock
2. **Analyzes Sentiment**: Each article rated -1 (negative) to +1 (positive)
3. **Aggregates Score**: Calculates overall sentiment
4. **Checks Agreement**: Compares with model prediction
5. **Provides Context**: Shows trend and article count

Sentiment Score Scale

Score Range	Label	Icon	Meaning
+0.6 to +1.0	Very Positive	✅✅✅	Excellent news
+0.3 to +0.6	Positive	✅✅	Good news

Score Range	Label	Icon	Meaning
+0.1 to +0.3	Slightly Positive	✅	Somewhat positive
-0.1 to +0.1	Neutral	➖	Mixed signals
-0.3 to -0.1	Slightly Negative	⚠️	Somewhat concerning
-0.6 to -0.3	Negative	❌	Bad news
-1.0 to -0.6	Very Negative	❌❌	Very bad news

Agreement Scenarios

High Confidence (Best):

```
Model: UP (68%) + Sentiment: Positive (+0.45)
→ ✅ ✅ Strong UP signal - Both indicators agree
```

Low Confidence (Caution):

```
Model: UP (62%) + Sentiment: Negative (-0.38)
→ ⚠️ Mixed signals - Trade with caution
```

Neutral Sentiment:

```
Model: UP (65%) + Sentiment: Neutral (0.05)
→ 📊 Rely on technical indicators
```

Configuring Sentiment Analysis

Edit `src/sentiment_config.py`:

python

```python
SENTIMENT_CONFIG = {
    'enabled': True,           # Turn on/off
    'news_days': 3,            # Look back N days (change to 7 for more)
    'min_articles': 2,         # Minimum articles needed
    'max_articles': 20,        # Maximum to analyze
    'show_headlines': 3,       # Headlines to display (CLI only)
    'method': 'vader',         # 'vader' or 'finbert'
}
```

Using FinBERT (Better Accuracy)# Stock Trend Predictor - Complete Usage Guide

📋 Table of Contents

📊 Project Overview

This project predicts whether a stock will go UP or DOWN tomorrow using machine learning. It supports multiple models and advanced feature engineering for better accuracy.

What This Does

- ✅ Fetches historical stock data from Yahoo Finance
- ✅ Engineers 40+ technical indicators (RSI, MACD, Bollinger Bands, etc.)
- ✅ Trains multiple ML models (Random Forest, LightGBM, LSTM, Ensemble)
- ✅ Predicts tomorrow's direction (UP/DOWN) with confidence scores
- ✅ Supports training on single or multiple stocks

What This Doesn't Do

- ❌ Predict exact prices (only direction)
- ❌ Guarantee profits (markets are unpredictable)
- ❌ Provide financial advice (educational only)

🛠️ Installation

Step 1: Install Python Dependencies

```bash
# Install all requirements
pip install -r requirements.txt
```

Requirements include:

- yfinance (stock data)
- pandas, numpy (data processing)
- ta (technical indicators)
- scikit-learn (traditional ML)
- lightgbm, xgboost (gradient boosting)
- tensorflow (deep learning)

Step 2: Verify Installation

```bash
python -c "import yfinance, lightgbm, sklearn, tensorflow; print('✅ All packages installed')"
```

Step 3: Project Structure

```
stock-trend-predictor/
├── src/
│   ├── main.py                # Single-stock training
│   ├── train_multi.py         # Multi-stock training
│   ├── train_advanced.py      # Advanced training (RECOMMENDED)
│   ├── predict.py             # Make predictions
│   ├── compare_models.py      # Compare model performance
│   ├── compare_configs.py     # Configuration helper
│   ├── feature_engineering.py # Feature creation
│   ├── model_training.py      # Model definitions
│   ├── data_preprocessing.py  # Data cleaning
│   ├── data_fetch.py          # Data download
│   └── config.py              # Configuration settings
├── data/                      # Stock data (auto-created)
├── model/                     # Trained models (auto-created)
├── results/                   # Results & logs (auto-created)
```

```
├── requirements.txt
└── README.md
```

🚀 Quick Start

1. Train Your First Model (5 minutes)

```bash
# Quick test with 3 stocks
python src/train_advanced.py --tickers AAPL MSFT GOOGL --model lightgbm --max-features 25
```

Expected output:

```
[1/7] Fetching stock data...
[2/7] Engineering features...
...
✅ Final Test Accuracy: 64.32%
Model saved to: model/advanced_lightgbm_model.pkl
```

2. Make a Prediction

```bash
# First, update predict.py to use the new model
# Edit src/predict.py, line 8:
# MODEL_PATH = "model/advanced_lightgbm_model.pkl"

python src/predict.py AAPL
```

Expected output:

```
============================================================
STOCK PREDICTION RESULTS
============================================================

Last trading day: 2025-01-05
Last close price: $175.43

📈 PREDICTION: UP tomorrow
Confidence: 67.3%

Recent Technical Indicators:
  RSI: 58.32
  MACD: 2.14
  Volatility: 0.0234
============================================================
```

3. That's it! 🎉

You've trained a model and made your first prediction!

🎓 Training Models

You have **three training scripts** to choose from:

Script	When to Use	Training Time	Accuracy
`main.py`	Single stock, basic training	2-3 min	52-58%
`train_multi.py`	Multiple stocks, balanced data	5-10 min	58-65%
`train_advanced.py`	**Best accuracy** (recommended)	10-20 min	62-72%

Option 1: Single Stock Training (`main.py`)

Train on one stock's historical data.

```bash
```

```
# Basic usage
python src/main.py AAPL --model ensemble

# With options
python src/main.py AAPL \
    --model lightgbm \
    --features basic momentum volatility \
    --start 2020-01-01 \
    --end 2024-12-31
```

Available options:

- `--model`: random_forest, lightgbm, lstm, stacker, ensemble
- `--features`: basic, momentum, volatility, volume, patterns, advanced
- `--start`: Start date (YYYY-MM-DD)
- `--end`: End date (YYYY-MM-DD)
- `--output`: Output model path

Example:

```
bash

python src/main.py TSLA --model lightgbm --start 2019-01-01
```

Option 2: Multi-Stock Training (`train_multi.py`)

Train on multiple stocks for better generalization.

```
bash

# Using presets
python src/train_multi.py --preset tech --model ensemble --balance

# Custom stock list
python src/train_multi.py \
    --tickers AAPL MSFT GOOGL AMZN NVDA \
    --model lightgbm \
    --balance
```

Available presets:

Available options:

- `--preset` or `--tickers`: Choose stocks
- `--model`: Model type
- `--balance`: Balance UP/DOWN samples (recommended!)
- `--features`: Feature groups to use
- `--start/--end`: Date range
- `--no-fetch`: Skip downloading (use cached data)

Examples:

```bash
# Tech stocks with balancing
python src/train_multi.py --preset tech --model ensemble --balance

# Custom list with specific features
python src/train_multi.py \
    --tickers AAPL JPM XOM JNJ WMT \
    --model lightgbm \
    --features basic momentum volatility \
    --balance
```

Option 3: Advanced Training (`train_advanced.py`) ⭐ RECOMMENDED

Best accuracy with feature selection, augmentation, and optimization.

```bash
```

```
# Basic usage
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN NVDA \
    --model lightgbm

# Full options
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD JPM BAC \
    --model ensemble \
    --start 2018-01-01 \
    --max-features 35
```

What makes it "advanced":

- ✅ Automatic feature selection (picks top N most important)
- ✅ Data augmentation (creates synthetic samples)
- ✅ Lagged features (uses past 1-3 days)
- ✅ Rolling features (moving averages, std dev)
- ✅ Feature scaling (normalizes all features)
- ✅ Cross-validation (5-fold time series CV)

Available options:

- `--tickers`: Stock symbols (required)
- `--model`: lightgbm, random_forest, stacker, ensemble
- `--start/--end`: Date range
- `--max-features`: Number of features to select (default: 30)
- `--no-augmentation`: Disable data augmentation
- `--no-feature-selection`: Use all features
- `--no-scaling`: Don't scale features

Examples:

```
bash
```

```
# Quick test (5 minutes)
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL \
    --model lightgbm \
    --max-features 25

# Best accuracy (15 minutes)
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD JPM BAC \
    --model ensemble \
    --start 2018-01-01 \
    --max-features 35

# Fast with good accuracy (8 minutes)
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN NVDA TSLA META AMD \
    --model lightgbm \
    --max-features 30 \
    --start 2019-01-01
```

🔮 Making Predictions

After training a model, use `predict.py` to make predictions.

Step 1: Update predict.py

Edit `src/predict.py` (line 8) to use your trained model:

```python
# For single-stock model
MODEL_PATH = "model/stock_predictor.pkl"

# For multi-stock model
MODEL_PATH = "model/multi_stock_predictor.pkl"

# For advanced model
MODEL_PATH = "model/advanced_lightgbm_model.pkl"
# or
MODEL_PATH = "model/advanced_ensemble_model.pkl"
```

Step 2: Make Predictions

```bash
python src/predict.py AAPL
python src/predict.py TSLA
python src/predict.py GOOGL
```

Understanding the Output

```
============================================================
STOCK PREDICTION RESULTS
============================================================

Last trading day: 2025-01-05
Last close price: $175.43

📈 PREDICTION: UP tomorrow
Confidence: 67.3%

Individual Model Predictions:  # (only for ensemble)
  - random_forest: UP
  - lightgbm: UP
  - stacker: DOWN

Recent Technical Indicators:
  RSI: 58.32
  MACD: 2.14
  Volatility: 0.0234
============================================================
```

What each means:

- **Prediction**: UP (price will increase) or DOWN (price will decrease)
- **Confidence**: How certain the model is (higher = more confident)
- **Individual Predictions**: Each model's vote (for ensemble only)
- **Technical Indicators**: Recent market signals

📊 Comparing Models

Use `compare_models.py` to compare different models' performance.

Basic Usage

```bash
python src/compare_models.py AAPL
```

This compares:

- Single-stock model (`model/stock_predictor.pkl`)
- Multi-stock model (`model/multi_stock_predictor.pkl`)

Custom Comparison

```bash
python src/compare_models.py AAPL model/custom1.pkl model/custom2.pkl
```

Example Output

```
================================================================================
MODEL COMPARISON FOR AAPL
================================================================================

Test set size: 50 samples

================================================================================
COMPARISON RESULTS
================================================================================

Model           Accuracy    Precision   Recall      F1-Score
--------------------------------------------------------------------
Single-Stock       58.00%      56.52%      62.50%      59.38%
Multi-Stock        64.00%      65.22%      65.22%      65.22%

🏆 Multi-stock model performs better by 6.00%
================================================================================
```

📈 Understanding Results

Accuracy Metrics

Metric	What It Means	Good Value
Accuracy	% of correct predictions	>60%
Precision	Of predicted UPs, how many were correct	>60%
Recall	Of actual UPs, how many did we catch	>60%
F1-Score	Balance of precision & recall	>60%

Training Output Explained

```
[1/7] Fetching stock data...
  ✓ AAPL: 1250 rows
  ✓ MSFT: 1250 rows

[2/7] Engineering features...
Adding basic features...
Adding momentum features...
...

[3/7] Creating lagged and rolling features...
Total features after engineering: 87

[4/7] Augmenting data...
Original class distribution:
  Majority: 5124
  Minority: 4876
After augmentation:
  Total samples: 14876

[5/7] Preparing train/test split...
Train samples: 11900
Test samples: 2976

[6/7] Selecting best 30 features...
Top 10 most important features:
  RSI: 0.0842
  MACD: 0.0731
  BB_Width: 0.0689
  ...
```

```
[7/7] Scaling features...

Performing 5-fold time series cross-validation...
  Fold 1: 0.6234
  Fold 2: 0.6512
  Fold 3: 0.6387
  Fold 4: 0.6445
  Fold 5: 0.6521
CV Mean Accuracy: 0.6420 (+/- 0.0112)

Training final model...
✅ Final Test Accuracy: 65.32%

Classification Report:
              precision    recall  f1-score
DOWN             0.64      0.68      0.66
UP               0.67      0.63      0.65

Model saved to: model/advanced_lightgbm_model.pkl
```

Key points:

- **CV Mean Accuracy**: Average across 5 folds (should be stable)
- **Std deviation**: Low = consistent, high = unstable
- **Final Test Accuracy**: Performance on unseen data
- **Classification Report**: Detailed breakdown per class

🎯 Advanced Usage

Testing Different Configurations

Use `compare_configs.py` to see recommended configurations:

```bash
python src/compare_configs.py
```

This shows 6 pre-configured setups you can try manually.

Batch Predictions

Create a script to predict multiple stocks:

Windows (batch file):

```batch
@echo off
for %%t in (AAPL MSFT GOOGL AMZN TSLA) do (
    python src/predict.py %%t
)
```

Linux/Mac (bash script):

```bash
#!/bin/bash
for ticker in AAPL MSFT GOOGL AMZN TSLA; do
    python src/predict.py $ticker
done
```

Custom Feature Engineering

Edit `src/feature_engineering.py` to add your own features:

```python
def _add_custom_features(self, df):
    """Add your custom features"""
    # Example: Price distance from 200-day MA
    df["Custom_MA_Dist"] = (df["Close"] - df["SMA_200"]) / df["SMA_200"]

    # Example: Volume surge detection
    df["Volume_Surge"] = df["Volume"] > (df["Volume_SMA"] * 1.5)

    return df

# Register in __init__
self.feature_groups['custom'] = self._add_custom_features
```

Then train with your custom features:

```bash
```

```
python src/train_advanced.py \
  --tickers AAPL MSFT \
  --model lightgbm \
  --features basic momentum custom
```

💡 Tips & Best Practices

For Best Accuracy

1. **Use multi-stock training** (8-15 diverse stocks)

```
bash

  python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN JPM XOM JNJ WMT --model ensemble
```

2. **Include more historical data** (5+ years)

```
bash

  --start 2018-01-01
```

3. **Use ensemble model** for final production

```
bash

  --model ensemble
```

4. **Start with LightGBM** for experiments (faster)

```
bash

  --model lightgbm
```

5. **Balance your dataset** (for multi-stock training)

```
bash

  --balance
```

For Faster Training

1. **Use fewer stocks** (3-5)

2. **Use LightGBM** instead of ensemble
3. **Reduce features** (`--max-features 20`)
4. **Shorter date range** (`--start 2021-01-01`)

For Different Stock Types

Tech Stocks:

```
bash

python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD \
    --model lightgbm
```

Blue Chips:

```
bash

python src/train_advanced.py \
    --tickers AAPL JPM JNJ PG KO WMT DIS \
    --model lightgbm
```

High Volatility:

```
bash

python src/train_advanced.py \
    --tickers TSLA GME AMC NVDA \
    --model ensemble \
    --max-features 40
```

Retraining Schedule

- **Monthly**: Retrain with latest data
- **After major events**: Retrain after market crashes, Fed announcements
- **When accuracy drops**: Monitor predictions vs actual results

---

🐛 Troubleshooting

Common Errors

1. "No module named 'xgboost'"

```bash
pip install xgboost
```

2. "No module named 'feature_engineering'"

Make sure you're running from the project root:

```bash
cd /path/to/stock-trend-predictor
python src/train_advanced.py --tickers AAPL --model lightgbm
```

3. "Model not found"

Train a model first:

```bash
python src/train_advanced.py --tickers AAPL MSFT GOOGL --model lightgbm
```

4. "Not enough data"

Use longer date range:

```bash
--start 2018-01-01
```

5. Low accuracy (<55%)

Try these solutions:

```bash
```

```
# More stocks
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD JPM BAC --model ensemb

# More data
--start 2017-01-01

# More features
--max-features 40

# Better model
--model ensemble
```

6. Training takes forever

Reduce complexity:

```
bash

# Fewer stocks
--tickers AAPL MSFT GOOGL

# Fewer features
--max-features 20

# Faster model
--model lightgbm
```

Performance Issues

Issue	Solution
Training too slow	Use LightGBM, fewer stocks, shorter date range
Low accuracy	More stocks, more data, ensemble model
Out of memory	Fewer features, shorter date range
Predictions too slow	Use single model (not ensemble)

📚 Recommended Workflows

For Beginners

```bash
# Day 1: Quick test
python src/train_advanced.py --tickers AAPL MSFT GOOGL --model lightgbm --max-features 25

# Day 2: Better model
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN NVDA --model lightgbm --max-features 30

# Day 3: Best model
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD --model ensemble --max-fe
```

For Production Use

```bash
# Step 1: Train best model
python src/train_advanced.py \
    --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD JPM BAC XOM JNJ WMT PG V MA \
    --model ensemble \
    --start 2017-01-01 \
    --max-features 40

# Step 2: Update predict.py
# MODEL_PATH = "model/advanced_ensemble_model.pkl"

# Step 3: Test predictions
python src/predict.py AAPL
python src/predict.py TSLA

# Step 4: Monitor accuracy weekly
# Compare predictions vs actual outcomes

# Step 5: Retrain monthly
# Run Step 1 again with latest data
```

For Experimentation

```bash
```

```
# Test different models
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN NVDA --model lightgbm
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN NVDA --model random_forest
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN NVDA --model stacker
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN NVDA --model ensemble

# Compare results
python src/compare_models.py AAPL model/advanced_lightgbm_model.pkl model/advanced_ensemble_model.pkl
```

📖 Summary

Quick Reference Commands

bash

```
# Best command for accuracy (15 min)
python src/train_advanced.py --tickers AAPL MSFT GOOGL AMZN META NVDA TSLA AMD JPM BAC --model ensemb

# Fast command for testing (5 min)
python src/train_advanced.py --tickers AAPL MSFT GOOGL --model lightgbm --max-features 25

# Make prediction
python src/predict.py AAPL

# Compare models
python src/compare_models.py AAPL

# See configuration options
python src/compare_configs.py
```

Expected Performance

Training Method	Accuracy	Time
Single stock (main.py)	52-58%	2-3 min
Multi-stock (train_multi.py)	58-65%	5-10 min
Advanced (train_advanced.py)	62-72%	10-20 min

Important Notes

⚠️ **This is for educational purposes only**

⚠️ **Not financial advice**

⚠️ **Past performance ≠ future results**

⚠️ **Markets are unpredictable**

⚠️ **Always do your own research**

🎓 Next Steps

1. ✅ Train your first model
2. ✅ Make predictions
3. ✅ Compare different configurations
4. ✅ Monitor accuracy over time
5. ✅ Retrain regularly with new data
6. ✅ Experiment with custom features

Good luck with your stock predictions! 📈