

Confluence Pipeline: Data, Features & Models

This document describes the end-to-end pipeline: how data flows through the system, how it is cleaned and filtered, what features are computed, and how each model works.

1. Pipeline Overview

The pipeline has two main flows:

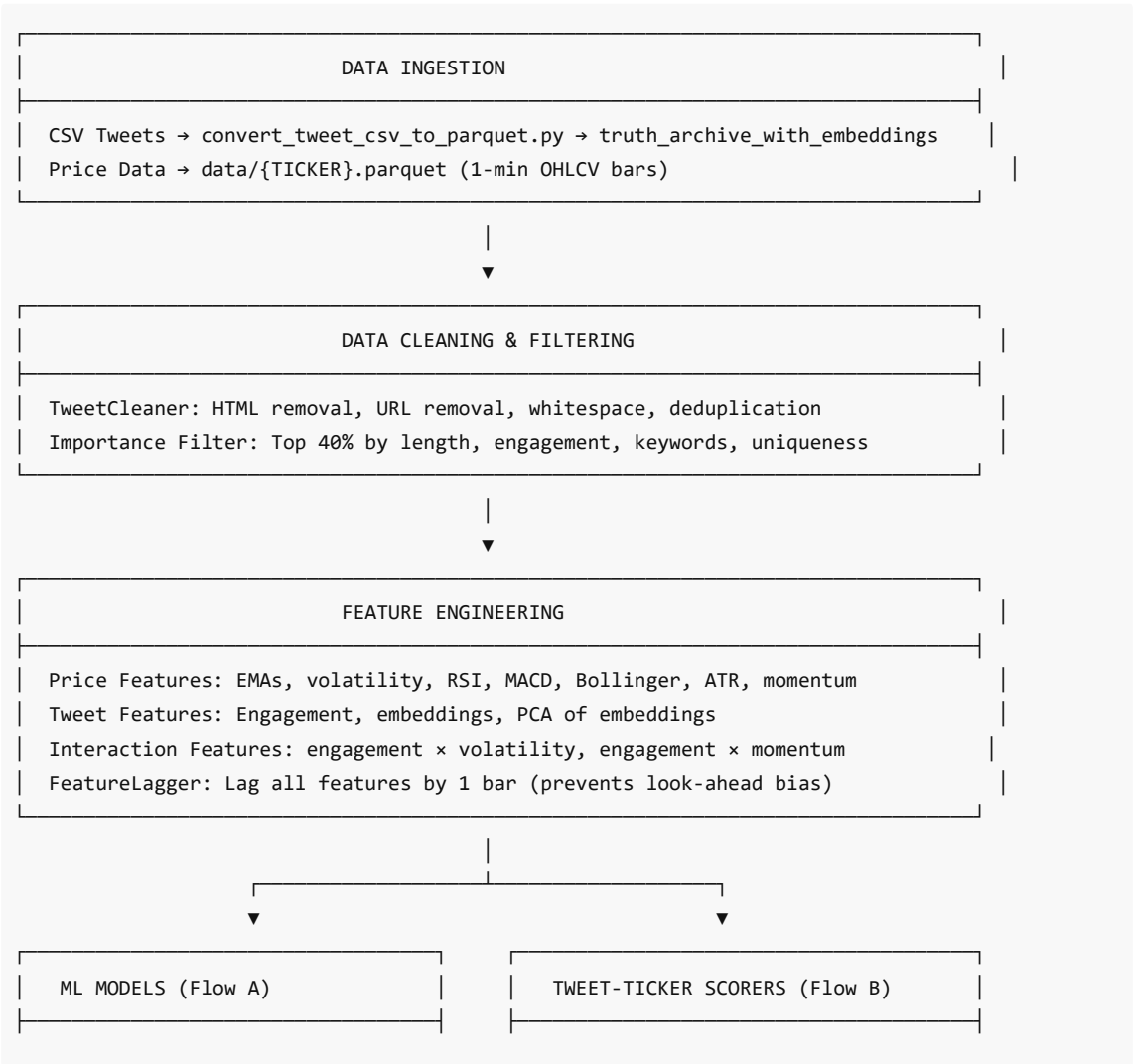
Flow A: ML-Based Trading Strategy (Features → Models → Backtest)

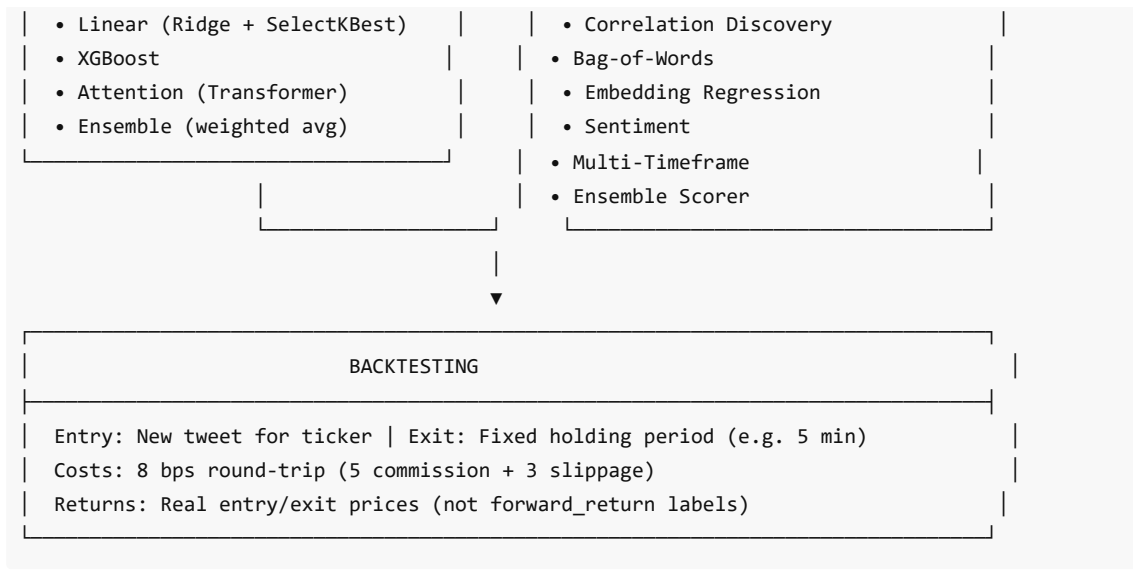
Raw Data → Convert & Embed → Precompute Features → Lag Features → Train ML Models → Backtest

Flow B: Tweet-Ticker Relationship Discovery (Score Methods → Backtest)

Raw Data → Clean & Filter → Compute Forward Returns → Train Score Methods → Backtest

High-Level Architecture





2. Data Cleaning & Filtering

2.1 Tweet Cleaning (TweetCleaner)

Location: `src/tweet_ticker_analysis/tweet_cleaner.py`

Steps:

Step	Description
HTML Tag Removal	Strips all HTML tags (<p>, , etc.) via regex
Entity Decoding	Converts & → &, " → ", ' → ', etc.
URL Removal	Removes http(s):// and www. URLs
Whitespace Normalization	Collapses multiple spaces, trims
Empty/Short Removal	Drops tweets with ≤ 5 characters after cleaning
Deduplication	Removes duplicates by tweet_id (or by text if no id)

Rationale: Raw tweets contain markup and noise that interfere with keyword extraction, embeddings, and sentiment analysis.

2.2 Importance Filtering

Method: `filter_important_tweets()`

Importance Score (0–1 composite):

Factor	Weight	Description
Length	20%	Longer tweets tend to be more substantive
Engagement	40%	replies + reblogs + favourites (normalized)

Keyword Density	30%	Mentions of financial/political terms
Uniqueness	10%	Penalizes very short / repetitive content

Tracked Keywords:

- Financial: tariff , trade , economy , market , stock , inflation , fed , interest , rate , dollar , currency , tax
- Political: policy , regulation , deal , agreement , war , crisis
- Economic: recession , growth , jobs , unemployment , china , mexico

Filtering Logic:

- Default: Keep top 40% by importance score (top_percentile=0.4)
- Optional: min_score to enforce a minimum threshold

Impact: Fewer but higher-quality tweets, better signal-to-noise for trading.

2.3 Correlation Validation (Optional)

Location: src/tweet_ticker_analysis/correlation_validator.py

Before trading, relationships can be validated:

- T-test:** H_0 : mean return = 0; require p-value < 0.1
- Correlation:** Tweet features vs returns; require $|corr| > 0.03$
- Sample Size:** Minimum 20 samples per ticker

3. Features Added

3.1 Price Features (FeatureEngineer.calculate_price_features)

Location: src/features/feature_engineer.py

Category	Features
EMAs	ema_5, ema_10, ema_20, ema_50, ema_100, ema_200
EMA Crossovers	ema_cross_5_20, ema_cross_10_50, ema_cross_50_200
Volatility	vol_5, vol_10, vol_20, vol_50, vol_100
Volatility Ratios	vol_ratio_5_20, vol_ratio_20_50
RSI	rsi_7, rsi_14, rsi_21
MACD	macd, macd_signal, macd_hist
Bollinger Bands	bb_upper, bb_middle, bb_lower, bb_width, bb_position
ATR	atr_7, atr_14
Momentum	momentum_1, momentum_3, momentum_5, momentum_10, momentum_20
Price Ratios	high_low_ratio, close_open_ratio

Distance from Highs/Lows	dist_from_20h, dist_from_20l, dist_from_50h, dist_from_50l
Volume (if present)	volume_ma_20, volume_ratio

3.2 Tweet Features (FeatureEngine.calculate_tweet_features)

Category	Features
Engagement	replies_count, reblogs_count, favourites_count, total_engagement
Engagement Ratios	reply_fav_ratio, reblog_fav_ratio
Log-Scaled	log_total_engagement, log_replies, log_reblogs, log_favourites
Embeddings	embedding_0 ... embedding_N (full vector)
PCA Embeddings	embedding_pca_0 ... embedding_pca_49 (50 components)

3.3 Interaction Features

Feature	Formula
engagement_vol_20	total_engagement × vol_20
engagement_vol_5	total_engagement × vol_5
engagement_mom_5	total_engagement × momentum_5
engagement_mom_20	total_engagement × momentum_20
engagement_rsi	total_engagement × rsi_14

3.4 Feature Lagging (Look-Ahead Bias Prevention)

Location: src/features/feature_lagger.py

- **Lag:** All features shifted by 1 bar
- **Effect:** At time T, only features from T-1 are used
- **Verification:** verify_no_leakage() checks alignment

4. Model Explanations

4.1 ML Models (Flow A: Feature-Based Prediction)

These models predict forward returns from the engineered features.

Linear Model (Baseline)

Attribute	Value
Class	LinearModel
Algorithm	Ridge regression (L2) on top K features
Feature Selection	SelectKBest with F-test (default K=50)

Regularization	alpha=1.0
Pros	Fast, interpretable, low overfitting risk
Cons	Linear only; cannot capture interactions

Use case: Baseline for comparing more complex models.

XGBoost Model

Attribute	Value
Class	XGBoostModel
Algorithm	Gradient boosting decision trees
Hyperparameters	max_depth=5, n_estimators=100, learning_rate=0.1
Early Stopping	10 rounds on validation set
Pros	Non-linear, handles interactions, robust on tabular data
Cons	Slower than linear; needs tuning

Use case: Strong non-linear baseline for tabular features.

Attention Model

Attribute	Value
Class	AttentionModel
Architecture	Multi-head self-attention + feed-forward layers
Config	3 layers, 8 heads, 512 hidden dim
Regularization	Dropout 0.2, weight decay 0.05, batch norm
Training	MSE loss, AdamW, ReduceLROnPlateau, early stopping
Pros	Can capture complex temporal/feature patterns
Cons	Heavier, more prone to overfitting

Use case: When feature interactions and temporal patterns matter.

Ensemble Model

Attribute	Value
Class	EnsembleModel
Composition	Linear + XGBoost + Attention
Method	Weighted average (weights optimized on validation)

Optimization	Scipy SLSQP to maximize validation correlation
Pros	Lower variance, more stable than single models
Cons	More compute; depends on base model quality

Use case: Final production model when combining multiple predictors.

4.2 Tweet-Ticker Scoring Methods (Flow B)

These methods score *how much* a tweet influences a ticker's return, then drive trades.

1. Correlation Discovery

Attribute	Value
Class	CorrelationDiscovery
Method	Statistical correlation between tweet features and ticker returns
Features Used	Embeddings, engagement, price/technical, tweet-derived numerics
Output	Per-ticker correlation scores (and mutual information)

Mechanism: Aligns tweet features with forward returns, computes Pearson correlation and mutual information per ticker.

Use case: Quick discovery of linear relationships between tweet features and returns.

2. Bag-of-Words Scorer

Attribute	Value
Class	BagOfWordsScorer
Method	TF-IDF + regression / t-stat weighting
N-grams	Unigrams through trigrams (1–3)
Filtering	p-value threshold, min occurrences, top N keywords per ticker
Output	Per-tweet, per-ticker influence score from keyword matches

Mechanism: TF-IDF on tweet text → correlates terms with returns → keeps statistically significant keywords → scores new tweets by keyword presence and historical effect.

Use case: Interpretable keyword-based signalling (e.g. "tariff" → EWW).

3. Embedding Scorer

Attribute	Value
Class	EmbeddingScorer
Method	Ridge regression: embedding → forward return

Fallback	k-NN similarity over historical tweets
Output	Influence score from embedding regression or similarity

Mechanism: Trains a Ridge model per ticker (embedding → return). For new tweets, uses model prediction or similarity to high-return historical tweets.

Use case: Semantic similarity without explicit keywords.

4. Sentiment Scorer

Attribute	Value
Class	FinancialSentimentAnalyzer, SentimentScorer
Method	FinBERT (ProsusAI/finbert) for financial sentiment
Output	Positive/negative/neutral + confidence

Mechanism: Uses a financial BERT model to classify sentiment. Falls back to keyword-based sentiment if the model is unavailable.

Use case: Directional signal from sentiment (bullish vs bearish).

5. Multi-Timeframe Scorer

Attribute	Value
Class	MultiTimeframeScorer
Method	Wraps another scorer; scores across multiple horizons
Horizons	5, 15, 30, 60, 240 minutes
Output	Optimal horizon and combined score

Mechanism: Runs base scorer for each horizon, then selects the horizon with the best influence × confidence.

Use case: Adapting holding period to signal type.

6. Ensemble Scorer

Attribute	Value
Class	EnsembleScorer
Composition	Correlation + Bag-of-Words + Embedding (+ optional LLM)
Method	Weighted combination of method scores
Default Weights	Correlation: 0.2, BOW: 0.2, Embedding: 0.6, LLM: 0.2

Mechanism: Trains each enabled method, then combines scores with fixed or learned weights.

Use case: Robust, diversified tweet-ticker scoring.

5. Scripts Reference

Script	Purpose
scripts/convert_tweet_csv_to_parquet.py	CSV → Parquet, embeddings
scripts/precompute_features.py	Build price + tweet + interaction features
scripts/run_all_tweet_ticker_methods.py	Train all scorers, backtest, compare
scripts/run_strategy.py	Run ML-based backtest
scripts/improved_strategy_v3.py	Event-driven keyword strategy

6. Data Flow Summary



For bias fixes, cost assumptions, and model hyperparameters, see [README.md](#) and [config/settings.py](#).