Detecting Market Structure Using Long-Short Term Memory Neural Networks

Boston University Faculty of Computing & Data Sciences
Yulian Gogov & Sviatoslav Shevchenko
CDS DS 340 | Dr. Cleary
December 8, 2025

1. Introduction

1.1 Background & Motivation

Daily market movements appear noisy and unpredictable, but research in econometrics and machine learning suggests that beneath this noise there are slower, repeatable patterns. These patterns—or market structure—do not guarantee precise forecasts, but they imply that certain combinations of returns, volatility, volume, and sentiment tend to recur in similar environments. Because these relationships are nonlinear and context-dependent, they are difficult to detect with traditional statistical models.

LSTM neural networks are well suited to this problem because they analyze sequences rather than isolated observations. By "remembering" relevant conditions and discarding irrelevant ones, an LSTM can learn how patterns evolve across weeks of market behavior. This makes it a strong candidate for detecting gradual directional pressure, regime transitions, and other subtle forms of temporal structure that simple models typically overlook.

1.2 Research Objective & Methodology Overview

The goal of this project is to test whether NASDAQ price behavior contains detectable statistical structure, not to build a trading strategy or predict exact returns. If an LSTM can consistently identify medium-term directional tendencies or anticipate turning points better than chance, then the underlying market dynamics must contain learnable patterns.

To evaluate this, we built a full data pipeline combining historical price data, engineered technical and volume features, volatility measures, and sentiment inputs derived from financial news. We then constructed labeled target periods using trade segmentation and trained LSTM models on sequences of these features. By comparing predicted and realized returns, as well as examining directional accuracy and turning-point timing, we assess whether the model is capturing genuine structure in the data. The following sections describe this workflow in detail.

2. Data Collection & Preprocessing

To evaluate structural relationships in market behavior, we first constructed a unified, high-integrity dataset combining daily, hourly, and minute-level price data for QQQ (NASDAQ-100 ETF). This multi-resolution design enables the model to incorporate long-term context while preserving recent, fine-grained market information.

2.1 Multi-Frequency Data Collection

We gathered historical data from Yahoo Finance under the following schedule:

- 1-minute data: last 5 trading days (limited by Yahoo's 7-day maximum window)
- 1-hour data: the two years preceding the minute window

- 1-day data: all available history prior to the start of the hourly dataset (typically back to 1999)

Each segment includes open, high, low, close, and volume fields. All timestamps were converted to Eastern Time and trimmed to regular U.S. market hours (9:30-16:00 ET) to eliminate overnight and extended-session distortions.

2.2 Dynamic Splicing & Integrity Validation

Because the data arrives from three independent queries, a key part of the pipeline is ensuring perfect chronological stitching. For each segment we:

- removed duplicates
- enforced strictly increasing timestamps
- verified session boundaries
- ensured no overlaps between daily/hourly/minute segments
- checked expected row counts per session
- detected and flagged missing intervals or gaps

After cleaning, the segments were concatenated in daily → hourly → minute order. A frequency column was added to indicate the resolution of each row. The output of this stage is a continuous, gap-free dataset suitable for consistent feature engineering and sequential modeling.

2.3 External Sentiment and Risk Indicators (News, Fed Tone, VIX, and Fear & Greed)

In parallel with the price-based dataset, we constructed a comprehensive set of daily sentiment and risk indicators to provide a richer context for regime detection. This pipeline consists of (i) building a large financial-news corpus from multiple sources, (ii) running FinBERT to extract headline-level sentiment, (iii) integrating macro sentiment proxies (Fed news tone, CNN Fear & Greed index, and VIX), and (iv) aligning all series on a common daily calendar.

2.3.1 Building the News Corpus from CSV Sources

We first assembled a pooled news dataset by merging several CSV-based sources:

- S&P 500 headlines (sp500_headlines_2008_2024.csv)
- Guardian headlines (FNHD_guardian_headlines.csv)
- Reuters headlines (FNHD_reuters_headlines.csv)
- CNBC headlines (FNHD_cnbc_headlines.csv)

For each source, we standardized columns to a common schema and cleaned date fields:

- S&P 500 CSV:
    - Loaded the file into pandas and renamed Title → headline and Date → date.
    - Kept only date and headline, added an empty description column, and parsed date as datetime using the format %Y-%m-%d.
    - Dropped rows with missing headline or date.

- Guardian CSV:

    - Loaded FNHD_guardian_headlines.csv and renamed Headlines → headline and Time → date.
    - Kept date and headline, dropped rows with missing headline, and added description = None.
    - Cleaned date strings by stripping whitespace and handling partial month-year patterns (e.g., "Jul-18") via regular expressions.
    - Replaced incomplete or malformed date entries with NA, then forward-filled and back-filled to propagate valid dates.
    - Parsed the resulting date strings using a consistent format and dropped any remaining invalid dates.

- Reuters CSV:
    - Loaded FNHD_reuters_headlines.csv and renamed Headlines → headline, Time → date, and Description → description.
    - Retained only date, headline, and description, dropping rows missing headline.
    - First attempted to parse dates with a standard format (e.g., "%b %d %Y").
    - For rows that failed parsing, used a flexible regex-based helper that extracted (MonthName, Day, Year) and tried both long and short month-name formats before marking a date as invalid.

- CNBC CSV:
    - Loaded FNHD_cnbc_headlines.csv and printed diagnostics for rows missing any of [Headlines, Time, Description].
    - Renamed columns to headline, date, and description and dropped rows where both headline and date were missing.
    - Implemented a custom extract_cnbc_date() function that:
        - Normalized whitespace and standardized month names (e.g., "Sept" → "Sep").
        - Extracted patterns of the form "day month year" via regex.
        - Tried multiple date formats (full month name and abbreviated month) and returned NaT if parsing failed.

    - Applied this parser and tracked how many rows retained a valid date.

After per-source cleaning, each dataset contained standardized columns [date, headline, description] and a consistent datetime index.

2.3.2 Combining and Deduplicating CSV Sources

Once all four CSV feeds were standardized, we:

- Added a source column with values "sp500", "guardian", "reuters", or "cnbc" to retain provenance.
- Concatenated all sources into a single merged_raw DataFrame and dropped rows with missing date or headline.
- Performed deduplication at the (date, headline) level:
    - Counted rows before and after dropping duplicates on ["date", "headline"].
    - Kept the first occurrence within each duplicate group to avoid double-counting identical stories syndicated across feeds.
- Audited duplicates by:
    - Grouping the full pre-deduplicated data by ["date", "headline"] to obtain duplicate counts.
    - Filtering to pairs with count ≥ 2 to see all clusters of repeated headlines.

    - Joining back to the source-tagged dataset to identify which feeds contributed to each duplicate.

2.3.3 Harvard / JSON News Data and Master Headline File

In addition to CSV feeds, we processed a collection of JSON news datasets (e.g., HuffPost/Harvard-style archives) named data_*.json:

- Discovered all JSON files via glob("data_*.json").
- For each file:
    - Loaded JSON using json.load and checked that the root object was a non-empty list.
    - Converted the list into a DataFrame and normalized column names to lowercase.
    - Identified candidate columns for date, headline, and description based on substring matches (e.g., columns containing "date", "headline", "short" or "desc").
    - Renamed these to standard names (date, headline, description), dropped rows missing any of date or headline, and filled missing descriptions with empty strings.
    - Parsed date with pd.to_datetime(errors="coerce") and removed any rows where date remained invalid.
- Concatenated all per-file DataFrames into json_all, sorted by date, and converted date to MM/DD/YYYY.
- Detected and printed duplicate (date, headline) pairs, then dropped duplicates keeping only one occurrence per pair.
- Tagged the entire JSON dataset with source = "harvard" and saved to financial_headlines_jsons.csv.

To build a unified master corpus, we then:

- Loaded financial_headlines_csvs.csv (CSV sources) and financial_headlines_jsons.csv (JSON sources).
- Ensured both had the same columns: [date, headline, description, source].
- Concatenated them into a combined DataFrame, dropped rows with missing date or headline, and parsed date as datetime.
- Sorted by date, reformatted date as MM/DD/YYYY, and saved the final corpus as financial_headlines_master.csv.
- Verified that Harvard/JSON stories were preserved by comparing (date, headline) pairs between the JSON subset and the master file and listing any dropped pairs for auditing.

2.3.4 Fed News Sentiment, CNN Fear & Greed, and VIX

Beyond FinBERT-based sentiment, we incorporated three additional macro sentiment and risk proxies:

- Fed news sentiment (DS340 “Fed sentiment” dataset):
    - Loaded fed_news_sentiment_data.xlsx and discovered that the default sheet contained metadata rather than usable data.
    - Listed sheet names, identified 'Data' as the relevant sheet, and reloaded only that sheet.
    - Validated structure using df.info(), df.isnull().sum(), df.duplicated().sum(), and df.describe().
    - Confirmed that the dataset contained ~16,731 rows with two clean columns (date and News Sentiment), no missing values, and no duplicates.
    - Saved the cleaned data to fed_news_sentiment_data.csv for later merging.

- CNN Fear & Greed index:
    - Loaded historical values from fear-greed-2011-2023.csv with columns [Date, Fear Greed].
    - Parsed Date via pd.to_datetime(format='mixed') and set it as the index.
    - Constructed a continuous daily date range from the earliest available date to a specified END_DATE and inserted placeholder rows with Fear Greed = 0 for missing dates.
    - For each record, converted the millisecond timestamp to a datetime and overwrote placeholder values at that date with the actual Fear & Greed reading.
    - Saved the result as all_fng_csv.csv, then:
        - Reloaded it, normalized timestamps to midnight, and grouped by date to collapse duplicates.
        - Computed weekday and removed weekend rows (Saturday/Sunday) where Fear Greed remained 0 (pure placeholders).
        - Dropped the helper column and saved the cleaned series as all_fng_csv_cleaned.csv.
- VIX (VIXCLS implied volatility):
    - Loaded VIXCLS.csv from FRED, inspected structure via df.head() and df.info().
    - Parsed observation_date as datetime and renamed it to date.
    - Converted the VIXCLS field to numeric and applied linear interpolation to fill missing values, creating a continuous daily implied volatility series.

2.3.5 Merging Sentiment Sources into a Unified Daily Feature Table

To support downstream correlation analysis and LSTM modeling, we aligned and merged all daily sentiment and risk series into a single table:

- Loaded and prepared:
    - df_vix from VIXCLS.csv (renaming observation_date → date).
    - df_fng from all_fng_csv_cleaned.csv (converting Date → date).
    - df_fed from fed_news_sentiment_data.csv.
    - df_finbert_master from finbert_master_sentiment_v2.csv.
    - df_finbert_csvs from finbert_csvs_sentiment.csv.
- Converted all date columns to datetime and standardized the merge key as date.
- Renamed the core FinBERT sentiment columns to distinguish variants:
    - In df_finbert_master, renamed sentiment → sentiment_master.
    - In df_finbert_csvs, renamed sentiment → sentiment_csvs.
- Performed a sequence of outer joins on date, starting with df_vix and merging in df_fng, df_fed, df_finbert_master, and df_finbert_csvs.
- Inspected the resulting merged_df via head(), info(), and describe() to validate:
    - Date coverage and row count.
    - Correct data types for all numeric series.
    - Reasonable ranges for VIX, Fear & Greed, Fed sentiment, and FinBERT metrics.

To make sentiment fields easier to reference in the modeling pipeline, we then standardized FinBERT naming as two generations of sentiment features:

- FinBERT v1 (from the master corpus) was mapped as:
    - bullish_x → finbert_v1_bullish
    - bearish_x → finbert_v1_bearish

    - neutral_x → finbert_v1_neutral
    - sentiment_index_x → finbert_v1_index
    - sentiment_scaled_x → finbert_v1_scaled
    - sentiment_smoothed_7d_x → finbert_v1_smooth7
    - sentiment_smoothed_30d_x → finbert_v1_smooth30
- FinBERT v2 (from CSV-only corpus) was mapped as:
    - bullish_y → finbert_v2_bullish
    - bearish_y → finbert_v2_bearish
    - neutral_y → finbert_v2_neutral
    - sentiment_index_y → finbert_v2_index
    - sentiment_scaled_y → finbert_v2_scaled
    - sentiment_smoothed_7d_y → finbert_v2_smooth7
    - sentiment_smoothed_30d_y → finbert_v2_smooth30

Finally, we constructed a sentiment_features_daily table containing:

- Date, VIXCLS, Fear Greed, News Sentiment, All FinBERT v1 features, All FinBERT v2 features

This table was sorted by date, saved as sentiment_features_daily.csv, and used as the primary sentiment input to the LSTM feature set (either directly, or after being aligned to daily anchor points in the multi-frequency QQQ dataset).

3. Feature Engineering

To uncover potential nonlinear structure in market behavior, we engineered four major classes of features—returns, volume, volatility, and sentiment—each built on a unified anchor-based framework. Every feature is computed at the daily, hourly, and minute levels while respecting the temporal resolution of each segment.

3.1 Anchor Framework

Because our dataset merges daily, hourly, and minute data, each frequency carries its own "dimension" of time. Minute data contains every minute inside each hour and day, while hourly data contains all hours inside a day. Without explicit boundaries, rolling windows would overlap across these dimensions, for example, a 5-day return could accidentally incorporate minute-level fluctuations, or a 5-hour volatility window could spill across multiple days.

To prevent this, we defined an **anchor system** that marks where each natural boundary occurs:

- Daily anchors (is_day_close) identify the end of each trading day within daily, hourly, and minute data
- Hourly anchors (is_hour_close) identify the end of each hourly bar within hourly and minute data
- Minute anchors (is_minute_anchor) identify every minute

Rolling features that are computed on timelines exhibiting different frequencies are computed only at their corresponding anchors. For example daily features that bleed into hourly or minute data will use entries where is_day_close = True for feature calculation.

3.2 Encapsulated Feature-Engineering System

To systematically construct rolling features across daily, hourly, and minute data, we implemented an encapsulated feature system based on two core functions: one for returns and one for volume. Both functions share the same design:

1. Choose an anchor timeline based on the desired source frequency:
    a. source_freq = "daily" → use actual daily rows and intraday day-closes
    b. source_freq = "hourly" → use hourly bars and intraday hour-closes
    c. source_freq = "minute" → use every minute
2. Compute the feature only at anchor timestamps using a rolling window over anchors, not raw rows.
3. Forward-fill the value between anchors, so every row inherits the most recent anchor-level feature value without mixing frequencies.

Rolling returns

The *add_rolling_return_via_anchors* function computes a trailing return of length k at each anchor t as

$$RollingReturns_k(t) = \frac{P_t}{P_{t-k}} - 1$$

where both P_t and P_{t-k} come from anchor points. After this value is calculated at anchor *t*, it is forward-filled to all rows between *t* and the next anchor. This is to ensure that:

- Daily rolling returns are updated only at day boundaries, even inside minute data, Hourly rolling returns are updated only at hour boundaries, Minute rolling returns behave like standard row-by-row rolling features.

We applied this function across multiple windows and frequencies using:

- Minute windows: 1, 5, 15, 30 bars, Hourly windows: 1, 3 bars, Daily windows: 1, 5, 21, 63, 126, 252 bars

producing features such as *rolling_return_5@minute, rolling_return_3@hourly, and rolling_return_63@daily*.

Volume features

The add_volume_feature_via_anchors function uses the same anchor logic but applies it to the volume column. It supports two types of volume features:

- Rolling sum over the last *k* anchors: $VolumeSum_k(t) = \sum_{j=0}^{k-1} V_{t-j}$
- Rate of change (ROC) versus the anchor *k* steps back: $VolumeROC_k(t) = \frac{V_t}{V_{t-k}} - 1$

In practice, we found the ROC-style features aligned best with return scaling and used those in our modeling (e.g., *rolling_volume_roc_5@daily, rolling_volume_roc_3@hourly*). As with returns, values are computed only at anchors and then forward-filled until the next anchor, preserving a clean separation between daily, hourly, and minute timescales.

Extension to volatility and sentiment

Volatility features follow the same pattern conceptually, but instead of comparing levels, they compute the standard deviation or realized magnitude of returns across anchor windows.

Sentiment features are added as a complementary "context" channel. Using FinBERT-classified financial news, we aggregate sentiment over 7-day and 30-day rolling windows, producing smoothed measures of recent positive/negative tone that evolve more slowly than raw price changes.

3.3 Sentiment Feature Construction from FinBERT and Macro Indicators

The sentiment channel described above is operationalized by combining the daily sentiment_features_daily table with the anchor-based QQQ price dataset. The construction proceeds in two main steps:

- First, we treat sentiment_features_daily as the master calendar for market "mood," containing:
    - FinBERT v1 and v2 daily probabilities (bullish, bearish, neutral) and derived indexes (sentiment_index, sentiment_scaled, smoothed 7- and 30-day values), Fed News Sentiment values from the dedicated Excel dataset, CNN Fear & Greed index and VIX (VIXCLS) as additional proxies for risk-on/risk-off conditions.
- Second, we merge these daily sentiment features into the multi-frequency QQQ dataset on the date dimension and then use the anchor framework to propagate them across intraday rows in a time-consistent way:
    - At each trading day's close (is_day_close = True), the sentiment values from that calendar date become the "daily sentiment snapshot" for that day. These daily sentiment snapshots are forward-filled into all intraday rows belonging to the same trading day. This ensures that minute- and hour-level features see the correct daily sentiment context without leaking information across days.

Within this integrated dataset, we then construct the specific sentiment features used by the LSTM:

- Raw and smoothed FinBERT indexes:
    - finbert_v1_index and finbert_v1_scaled capture the net bullish-bearish tone from the broader news universe, finbert_v1_smooth7 and finbert_v1_smooth30 provide short- and medium-horizon sentiment trends.
- Alternative FinBERT view from CSV-only feeds (v2), which acts as a robustness check and can be optionally included as an additional channel.
- Macro sentiment variables:
    - Fear Greed (scaled to [0, 1]) to indicate crowd risk appetite, VIXCLS (option-implied volatility) normalized and, when needed, smoothed to reduce idiosyncratic spikes, News Sentiment from the Fed news dataset, which provides a central-bank communication signal.

All sentiment variables are standardized (mean 0, variance 1) before modeling, so that the LSTM can weight them on the same scale as return, volume, and volatility features. This design allows the model to learn, for example, whether certain combinations of elevated FinBERT bullishness and low VIX precede sustained bullish regimes, or whether sharp deteriorations in sentiment coincide with the onset of bearish segments.

4. Historical Trade Segmentation (Target Construction)

4.1 What Trade Segmentation Is and Why We Use It

Trade segmentation is the process of retrospectively identifying all meaningful "trades" or directional episodes that occurred in the historical price series. Instead of viewing the market as a continuous stream of daily fluctuations, segmentation breaks the timeline into discrete periods during which price movement followed a coherent pattern—such as a sustained rise, a sustained decline, or a period of indecision. In practice, segmentation is a way of labeling the past with the movements we wish a model could recognize in real time.

The market is noisy most of the time, and training a model directly on raw daily returns would force it to learn mostly noise. By identifying and labeling the periods where meaningful movement actually occurred, we allow the model to learn from the parts of history where structure is strongest. Segmentation tells the model, "These are the situations you should pay attention to—this is what a real trend looks like," while ignoring large amounts of irrelevant noise.

Framed this way, trade segmentation serves two purposes:

1. It defines the events we want the model to detect.
2. It provides clean, interpretable targets for supervised learning.

Thus, trade segmentation is not about predicting exact returns. It is about teaching the model to recognize *when the market is entering a directional phase*, so that later sections can evaluate whether the LSTM successfully detected the structure surrounding those moments.

4.2 Constructing Trade Segments from Daily Price Data

To create meaningful training targets for the model, we convert the historical daily price series into directional segments—coherent bullish or bearish runs that the model should learn to anticipate.

4.2.1 Segment Construction (Direction → Runs → Regimes)

We begin by isolating daily bars (true daily rows or day-close intraday rows) from 2000-2024. For each bar, we compute a simple directional label:

- +1 if the bar is up, -1 if the bar is down, 0 if flat

We then scan the time series once and group consecutive bars with the same direction into segments. Each segment receives:

- a unique segment_id, a segment_len (number of bars), a segment_dir (+1 or -1)

Finally, segments are classified using a minimum-run threshold:

- Bullish (Regime 1): +1 direction and ≥3 bars, Bearish (Regime 2): -1 direction and ≥3 bars, Choppy (Regime 0): short, flat, or noisy movements

This creates a clean, interpretable target label for every day in the dataset.

4.2.2 Segment-Level Metrics

For each segment, we compute a compact set of metrics that describe its quality, shape, volatility, and participation. These metrics are useful both analytically (characterizing past trends) and for training (showing the model what "real" trends look like).

Return and Momentum Metrics

- total_return - net percent change across the segment, avg_daily_return - mean daily return, momentum_efficiency - ratio of net movement to total absolute movement (clean vs. noisy trend), max_excursion - max run-up (bullish) or max drawdown (bearish) relative to segment start

Purpose: distinguishes clean trends, noisy trends, and failed breakouts.

Trend Shape Metrics
Using a similar linear regression $price_t \sim a + bt$

- trend_slope - direction and speed of the move, trend_r2 - how well the segment fits a straight line

Purpose: identifies whether the trend was smooth (high R^2) or erratic.

Volume Metrics

- volume_mean, volume_std, volume_trend_slope - whether volume increased or faded across the segment

Purpose: rising volume often confirms trend conviction.

Volatility Metrics

- volatility_mean, volatility_std, volatility_trend_slope

Purpose: captures the uncertainty dynamics within bull vs. bear episodes.

4.2.3 Final Dataset and Role in the Model

All metrics are collected into a segment-level table (segments_df) and joined back to daily rows via segment_id and regime. This gives us:

- Clean target labels (bull, bear, choppy)
- Rich contextual descriptors of each trend
- A map of which feature configurations precede successful trends

Ultimately, this segmentation transforms raw price data into a supervised learning problem:

> *Given technical, volatility, sentiment, and volume features, can the model learn to detect conditions that historically led to sustained directional moves?*

5. LSTM Modeling Framework

5.1 Model Search: Lookback Windows and Prediction Horizons

Before selecting the final model, we conducted a systematic search over several combinations of lookback windows and prediction horizons. We tested models with 40, 80, and 120 day look back windows and with 5, 10, 20, and 30day prediction horizons. Across all tested architectures, we found that the highest correlation between predicted and realized returns, most monotonic conditional-mean curve, and cleanest structure when aligning predicted values with true turning points was a 120-day lookback window and a 20-day forecast horizon. Summary over the top 3 (lookback, horizon) pairs:

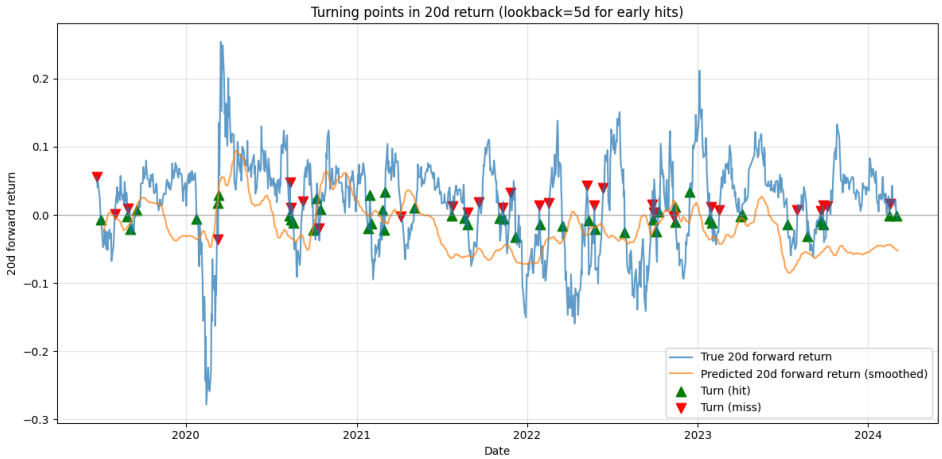
	lookback	horizon	mse	mae	corr	spread
10	120	20	0.004260	0.052483	0.288935	0.060022
2	40	20	0.010209	0.080665	0.263344	0.039749
3	40	30	0.007457	0.065740	0.188720	0.051730

5.2 Train/Test Split

To evaluate whether the LSTM captures meaningful market structure, we trained the model on historical NASDAQ data from 2000 through 2018 and evaluated its performance on an unseen test set covering 2019-2024. This strictly chronological split prevents information leakage and ensures that all results reflect genuine out-of-sample behavior.

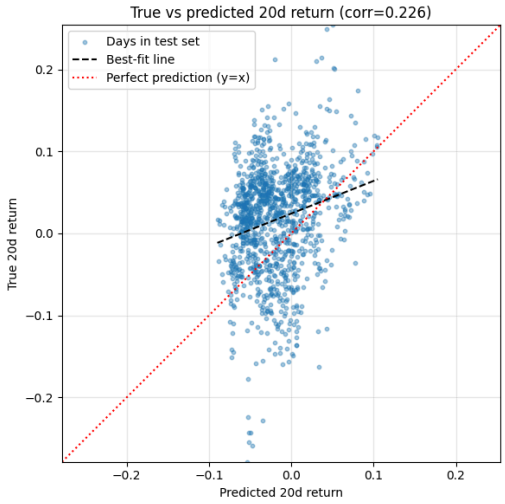
5.3 Results

Figure 1: Turning-Point Alignment (Sign Flips & Lead Time



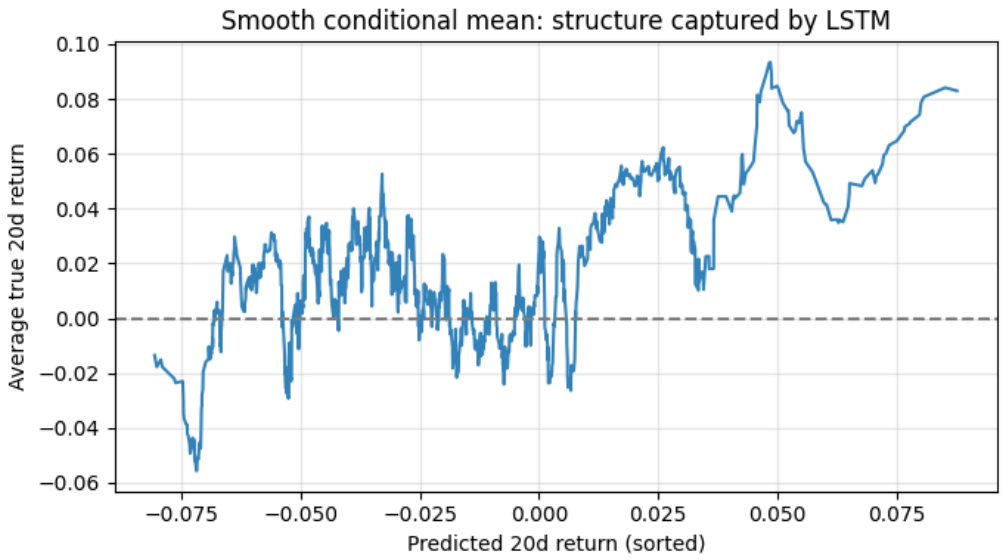
There are 76 turning points and our model anticipated 61.84% of the turns. On average, when it hits, the signal appeared about 4.47 days earlier. This suggests the model captures transitional dynamics, meaning its predictions change direction shortly before the market does.

Figure 2: Scatterplot of Predicted vs. True 20-Day Returns



The scatter plot demonstrates that although the LSTM does not accurately predict the exact magnitude of future returns (as reflected in a shallow slope and relatively high MSE), it does learn a weak but consistent positive association between predicted and realized outcomes. In financial settings, even modest structural alignment can indicate the presence of real underlying market structure.

Figure 3: Conditional-Mean Plot (Monotonicity Test)



The monotonic relationship between predicted-return quintiles and realized returns provides the clearest evidence of learned structure. The model assigns higher predicted values to periods that subsequently exhibit higher average returns, meaning the LSTM is able to rank market conditions by their directional pressure even if its point forecasts are conservative.

6 Implications for Market Structure and Trading (Conclusion)

Although the LSTM does not forecast return magnitudes with high precision, its behavior provides clear evidence of underlying structure in market dynamics. The model produces conservative predictions centered near zero, yet still learns to rank future outcomes meaningfully: higher predicted-return quintiles consistently correspond to higher realized returns. In addition, the model anticipates regime shifts—its output changes direction several days before actual turning points with a hit rate above 60%. These results indicate that the LSTM identifies slow-moving directional pressure and characteristic pre-reversal patterns that are invisible to linear models. This suggests that financial markets, while noisy, exhibit persistent nonlinear structure that can be extracted when models are allowed to observe long sequences of historical context.

From a trading perspective, the current model is not intended as a standalone strategy, but rather as a diagnostic tool that reveals conditions under which the market tends to strengthen or weaken. Its output could function as a state variable indicating improving, deteriorating, or transitional environments, forming the basis for more sophisticated models or risk filters. Future work could refine this approach by calibrating the signal, combining it with volatility or liquidity measures, or testing architecture variations such as Transformers. Overall, the findings demonstrate that meaningful temporal structure exists in return behavior and that sequence-based neural networks are capable of detecting it, even when direct prediction remains difficult.