

Comprehensive Overview of Full-Stack Energy Market Edge Operations

Nicholas Cohen
nick@redrlabs.com
<https://redrlabs.com/>
July 2026

"If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle" (Sun Tzu).

I. INTRODUCTION

Modern financial forecasting systems operate over a profoundly diverse information landscape with a wide variety of data types and structures. This creates a massive dislocation between time, accuracy, and the formation of market prices derived from extensive parties.

Energy markets operate similarly, though it consists of physical dynamics and delivery systems, creating a volatile market shape. The underlying complexities include macroeconomic, microeconomic, fundamental, and interconnected relationships regarding the internal structure of energy commodities and sources. In addition to the complexities, weights of importance regarding influence are constantly dynamic.

Classical methods of analysis fall short of effectively transpiring this information on a consistent basis. The vast array of data composition combined with the consistently changing values creates the inability to truly define what is expected of the market.

Incorporating a constellation of methodologies, comprised of non-linear, linear, textual, and qualitative data can create a robust system of predicting and analyzing the duration, magnitude, and direction of energy price movements.

The following system provides a technical overview regarding the development of autonomous infrastructure that allows for the improved accuracy of market dynamics, and provides insight into the overlooked relationships and patterns that exist within the energy domain, creating new investment opportunities, ideas, and innovation.

Important Note. Claude Code and other large language models helped develop the syntax, infrastructure, and recommendations for machine learning components, including mathematical representations.

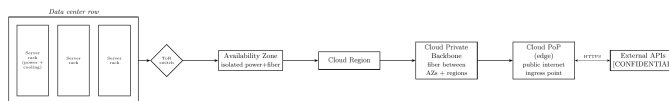


Fig. 1: Physical path from server rack to external API. Each physical rack holds several servers in which sharing a top-of-rack switch, with multiple independent power feeds and fiber paths, so a single failure becomes independent of another. The Availability Zone (AZ) component represents the location in which the servers are set upon, whilst connected to the private fiber backbone, allowing for internal traffic without external compromise. Outbound pulling of data via APIs exit through a channel called Point of Presence (PoP), essentially allowing a disconnect from the public network and the internal servers (this is not our expertise, and important to note some limitations in our understanding of how the core mechanics of this technology actually operate).

II. STACK

II-A Cloud Autonomy

What cloud autonomy means.

The firm operates with a fully autonomous structure. No manual execution exists regarding the data ingestion, analysis, and output other than the aspect of interface and potential to place positions (which is built out to be autonomous once turned on). The system works within itself, communicating with external APIs and data safely, whilst also running machine learning inference and reasoning layers with large language models through cloud-native design components.

This is feasible considering the components that the cloud provider offers regarding serverless functions, event buses, object storage buckets, container orchestration, and everything in between allowing for systematic cohesion.

The physical layer — what a data center actually is.

Regarding the specifics on how the cloud provider operates these without compromising security when communicating with external data sources includes using multiple core components.

The utilization of “cloud” technology becomes readily more reliable and available compared to owned hardware.

Event Bus Components.

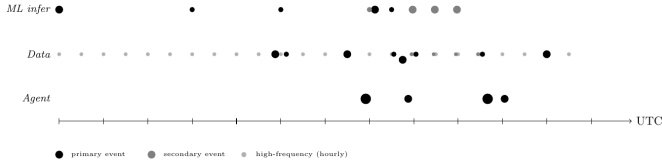


Fig. 2: EventBridges across a full trading day. Time in is UTC.

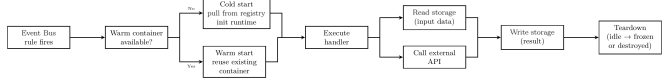


Fig. 3: Serverless function execution. Technical overview regarding the event bus relationship with the serverless function, where the system checks between container types available. The limit of functions running simultaneously is 10, staggered event scheduling works to avoid collision / overlap that can result in reaching the threshold. The teardown represents the shutting off of the specific innovation.

Providing context regarding the core components of this system. The event bus is a scheduling engine that allows for specific enabling of actions / calls without human action. Every action within the system refers back to a specific time and period set by these rules, which occurs autonomously.

Serverless Functions.

A serverless function provides actions without streaming servers, allowing for cheaper functionality with accurate results and limited latency (~10ms). The hardware is provided by the cloud provider, provisioning micro-VM (virtual machine). This loads the function code or container image at exceptional speeds, and limits the amount of compute consumed between runs, making it extremely cost effective.

Object Storage Components.

object storage acts as the database for the entire system. There is no relationed-based database, no cache, no message queuing system that exist within the system. Every serverless function reads specific data from the object storage and writes output back to different locations within. This makes the system stateless at the compute layer, meaning any serverless function can fail and be retried without corrupting external functions or objects.

The object storage component is organized into several distinct uses, regarding specific machine learning outputs, raw data inputs, and providing context into derived calculations, equations, etc. This component acts as the center of the system regarding any specific data.

The full autonomous loop.

The components regarding the event bus, serverless functions, and object storage are specific and integral parts into how this system operates fully autonomously. They each serve their purpose, combining into a single continuous loop that runs consistently and precisely to limit interruption.

Why autonomy matters.

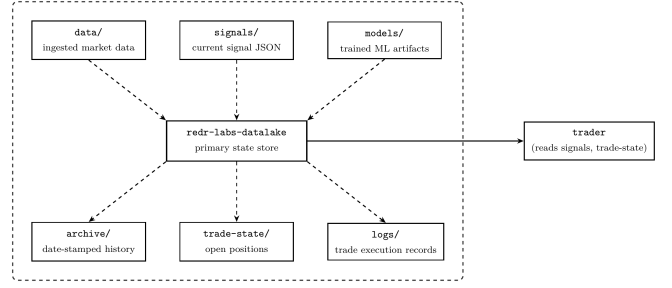


Fig. 4: Object storage data lake. Representation of the object storage database within the system, concurrent as the central position across retrieving, utilizing, storing output, and historical states for further reinforcement learning.

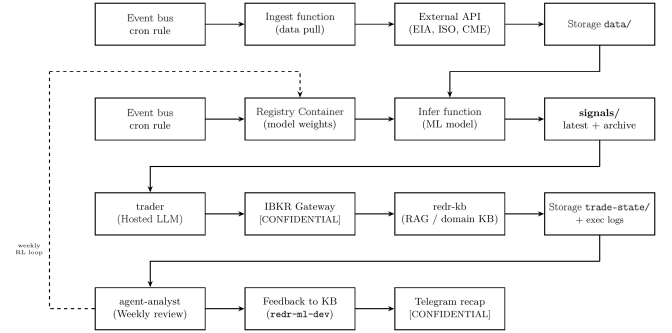


Fig. 5: Fully autonomous loop example from our present system.

Regarding the importance of autonomous utilization concludes with several advantages. Speed / reaction latency are far more advanced versus manual pulling, though the core advantage stems from consistent, tireless execution at zero marginal cost per invocation, whilst also enabling to scale by pulling from different data sources at the same time.

II-B Data Ingestion Architecture

Specific Detail.

The serverless functions issue an authenticated HTTP GET to the external data provider, which typically returns a JSON or CSV formatted data / dataset. The function parses and validates the response and provides a timestamped file to its assigned prefix or appends to a specific file for additional processing. The infrastructure runs over 50 serverless functions in total across data ingestion costing around ~\$0.00007 for the previous 30 days.

Daily Ingest Volume.

Let r_i be the row count per invocation of function i , f_i its daily fire-count, and $b \approx 120$ bytes the average row size for a typical energy time-series CSV. Total daily ingest volume: Assuming the data is depicted within a CSV, let r_i be the row count per invocation of function i , where f_i represents the daily firing cout, and $b \sim 120$ bytes (120 bytes is the average row size for a typical energy time-series CSV). Total daily ingest volume: $V_{\text{day}} = b \cdot \sum_i f_i \cdot r_i$

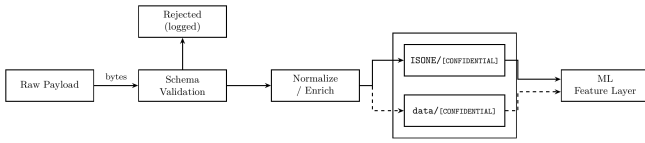


Fig. 6: Data flow representing serverless function payload being validated against expected schema which includes labels, non-null timestamps, and numerical v textual columns. Invalid payloads are rejected and logged without touching the object storage for protecting the current data.

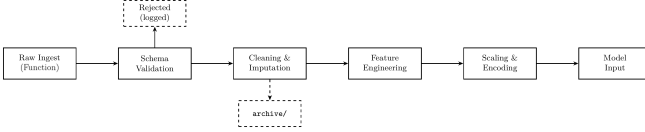


Fig. 7: Six-stage ETL pipeline. Raw data becomes ingested into validation. Once validated, cleaning occurs regarding data directly. Furthermore, feature construction, scaling / encoding, and the model goes within the input layer. Rejected payloads are logged without touching the object storage.

Depending on the day interval and payload, $r_i \in [24, 288]$ rows, which is representing a typical hourly or five-minute ISO data pull. Where, the volume per day or V_{day} falls within the range of [10 MB, 80 MB]. This is well within the object storage limitations regarding storage amounts and cost (\$0.023/GB storage cost at current scale). ($\in$ represents an element of, 24 for hourly, and 288 for 5min interval daily). That represents 288 data points per day from one Independent System Operator (ISOs) data type (DAM LMP 5min).

II-C Data Cleaning Processes and ETL Framework

Extract, Transform, Load (ETL) pipeline.

The ETL layer transformed raw API responses into the numerical feature types that are able to be readily utilized by ML models. It operates across six structural stages.

Schema validation and imputation post data ingestion.

Before any feature is computed, each ingestion of data is validated against a specific data schema, which is essentially what the data is expected to look like. This includes the field names, timestamps, numeric columns, labels, etc. Invalid payloads are rejected. Once verified valid, two specific classes of imputation / cleaning are applied: Forward Fill. Missing daily observations such as holiday closures during weekdays or API gaps, are filled from the previous valid value. This preserves the continuity of the rolling windows regarding machine learning platforms. Winsorization. For extreme outliers, which include values beyond the 1st and 99th percentile of the trailing 252-window, they are clipped to a boundary value prior to scaling. This prevents a single bad API response from blowing out a StandardScaler fit, or misrepresentation.

Feature construction.

Raw field values are combined and derived into different features that represent relationships. These features are then

consumed by the machine learning models. Specific derived values include:

Feature	Derivation	Consumed by
Thermal Generation %	[CONFIDENTIAL]	GMM, triple indicator
Natural Gas Generation Share %	[CONFIDENTIAL]	GMM, LSTM, MAS AE
Natural Gas Spread	[CONFIDENTIAL]	LSTM AE, spread model
Natural Gas Spread %	[CONFIDENTIAL]	LSTM AE
Change 1d and 5d Heating Degree Day Deviation	[CONFIDENTIAL]	GMM, LSTM
Heating Degree Day Z-score	[CONFIDENTIAL]	MAS AE
Petroleum Burn	[CONFIDENTIAL]	MAS AE
PADD1 Distillate	[CONFIDENTIAL]	MAS AE, spread model
Crack Spread	[CONFIDENTIAL]	Neural ODE (γ coeff)
Storage Deficit	[CONFIDENTIAL]	GMM, Neural ODE
Volatility Z-score	[CONFIDENTIAL]	Triple indicator

TABLE I: Derived feature construction used across the ISO machine learning pipeline.

StandardScaler.

StandardScaler, is applied to the continuous feature vectors prior to LSTM and GMM layer entry. The function maps each feature to zero mean and unit variance: $x_{\text{scaled}} = (x - \hat{\mu}) / \hat{\sigma}$ Where $\hat{\mu}$ and $\hat{\sigma}$ are estimated from the training window and serialized (.pkl file type) alongside model weights The rolling z-score is the inference-time analog of StandardScaler: $z_t = (x_t - \mu_{30d}) / \sigma_{30d}$. Where z_t represents the rolling z-score, x_t is the current observation, μ_{30d} is the 30 day rolling baseline mean, and σ_{30d} represents the 30 day rolling standard deviation. This is the direct mechanism in which the system makes today's observation comparable across time, providing context by derived values within history. The same raw LMP value is interpreted differently in summer and in winter once normalized to its recent context.

Why this matters for LLM autodetection.

A StandardScaler is fitted within a dataset with a distribution shift (e.g.m post-COVID demand collapse, or gas price spike) will produce systematically biased z-scores. The output will be dictionary meaningful but shifted.

OneHotEncoder.

OneHotEncoder is the methodology for representing categorical values as numerical values, without skewing them with bias due to invalid weight contexts. The variables that they are applied include:

Categorical variable	Levels	Used by
ISOs	ISONE, ERCOT, CAISO, NYISO, MISO, PJM	MAS 119-feature AE, VGAE
Fuel Types	gas, coal, nuclear, hydro, wind, solar, oil	GMM fuel-mix feature
Regimes	0, 1, 2 (learned GMM labels)	LSTM input, agent context
PADD Regions	PADD1, PADD2, PADD3, PADD4, PADD5	Petroleum feature layer
EDGAR Sector	E&P, midstream, refining, utility (10 sectors)	EDGAR signal aggregation

TABLE II: Categorical variables and their one-hot encoded levels.

The encoding represents a single categorical column into k binary columns, which allow for accurate training of machine learning models. Where the values are binary, no false weights are provided as anything greater than 1, has bias versus the value of 1 or 0.

Example if gas was the primary fuel type looks like the following:

fuel_type_gas	fuel_type_oil	fuel_type_wind	fuel_type_solar
1	0	0	0

TABLE III: One-hot encoding example if gas was the primary fuel type.

Whilst if solar was the primary fuel type:

fuel_type_gas	fuel_type_oil	fuel_type_wind	fuel_type_solar
0	0	0	1

TABLE IV: One-hot encoding example if solar was the primary fuel type.

Drop-first convention.

To avoid dummy variables, the first level of each category is dropped. The GMM and LSTM remain unaffected since they operate on the full feature vector without an intercept, but the LightGBM spread model uses the `drop=[first]` by convention. Seasonal cyclical encoding. The stacked LightGBM spread model encodes month and `[week_of_year]` as sine-cosine pairs rather than one-hot vectors. This preserves the cyclical continuity that one-hot encoding would perhaps break.

II-D Operations Infrastructure, Front-End Platform

Platform overview.

The Operations Infrastructure is the operator-facing web platform for the technical stack. It surfaces the entire ML and LLM pipeline as a usable interface, including live AI market analysis, autonomous trade execution controls, position monitoring, signal visualization, data exploration, and agent management. The platform is built on Next.js 16 with the App Router, deployed to Vercel, and maintained within a private GitHub repository. Every push to the main branch then triggers a Vertical Continuous Integration (CI) build. There is no separate build or deployment infrastructure, serverless

App	Function
Ops	AI analyst chat, trade controls, signal hub
Command	React Flow canvas, ESS project appraisal
My Agents	Agent config, app builder, dataset browser
Dashboard	P&L overview, weekly reports
Data	object storage browser, cross-signal correlation, CSV upload
Positions	Live IBKR positions, trade history
Charts	UNG, USO, BNO candlestick + spread charts
Scan	Multi-signal market scan
Cycle	Manual agent cycle trigger

TABLE V: Application shell and navigation.

[CONFIDENTIAL]

Fig. 8: Operations Infrastructure Chat Data Flow. The API route fires 40+ parallel object storage readings using a `Promise.all` function. This collects the appropriate information from each dataset and passes the result to `buildSystemPrompt.ts`. The prompt engineering file assembles a scorecard of several information into a single 15,000 token system prompt. The managed foundation model API connects to the hosted LLM (Claude Sonnet 4.6) and streams the response back. Where the browser is the front end that is secure.

API routes handle all communication between the cloud infrastructure and the Operations Infrastructure.

Application shell and navigation.

energyOS — the AI analyst interface.

energyOS is the primary interface. It is a streaming chat that assembles a prompt from 40+ parallel object storage reads, passes it to Claude Sonnet via the managed foundation model API, and renders the streamed markdown response in real time.

Authentication.

Operations Infrastructure uses a custom passphrase-based authentication system with over 20 characters. No OAuth provider is required, access is controlled by a list of passphrase strings configured as a Vercel environment variable and are never shown to any public facing data.

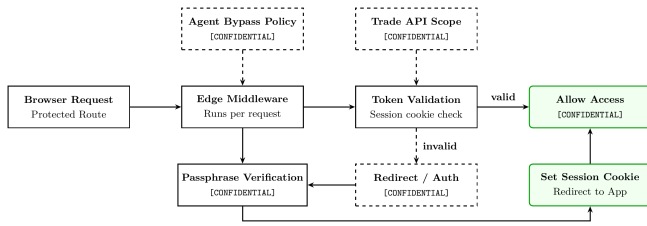


Fig. 9: Authentication flow. All protected routes pass through Next.js Edge middleware. Where the middleware checks the session cookie as a valid token is a base64 encoding of any passphrase in the environment variable. On sign-in, the managed key-value (NoSQL) database atomically increments a usage counter and enforces per-passphrase prompt limits. Middleware never returns 500, any specific unhandled error is redirected to prevent a crash from locking the front-end capabilities.

Library	Version	Used in	Purpose
mapbox-gl	3.22 / 7.1	MapDashboard	Base tile layer for geospatial views
react-map-gl	9.3.1	MapDashboard	Hexbin, scatter, arc layers over Mapbox for satellite signal viz
@deck.gl/layers	9.6 / 0.184	3D views	WebGL 3D scene graph, signal topology visualization
@react-three/fiber	3.8.0	Analytics, Hub	Standard bar, line, area charts for signal dashboards
three			
recharts			

TABLE VI: Main visualization libraries.

The deck.gl layer runs on top of Mapbox, rendering PADD3 refinery site information as a heatmap and satellite thermal signals. Three.js is used for signal topology view, which directly renders the cross-commodity graph as a 3D directed network.

Key dependencies and runtime.

Package	Version	Type	Role
next	16.1.6	prod	App Router, API routes, Edge middleware
react + react-dom	19.2.3	prod	UI rendering
typescript	5.x	dev	Type safety across all source files
tailwindcss	4.x	dev	Utility CSS, PostCSS pipeline
Object storage SDK	3.1011	prod	Object storage reads from API routes
Foundation model run-time SDK	3.1036	prod	Claude inference
Key-value store SDK	3.1034	prod	Session/usage tracking
Serverless invoke SDK	3.1047	prod	Invoke agent serverless functions
Telemetry & metrics SDK	3.1047	prod	Pipeline health metrics
Email service SDK	3.1055	prod	Email notifications
@vercel/analytics	2.0.1	prod	Page-level analytics
papaparse	5.5.3	prod	CSV parsing for user uploads
lucide-react	0.577	prod	Icon set
playwright	1.60	prod	Headless browser automation

TABLE VII: Key dependencies and runtime.

Data ingestion panel — user-uploaded signals.

Operations Infrastructure allows for the uploading of datasets that are analyzed in-browser and provided into chat context. This adds another layer of adaptability regarding signal research

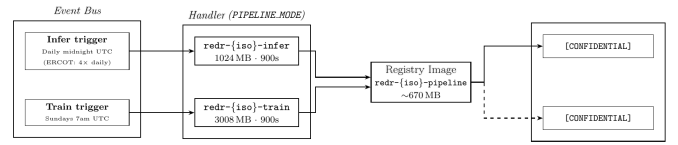


Fig. 10: Dual-mode pipeline architecture. A single container registry image serves both the inference and training serverless function for each ISO.

and learning. When data is uploaded via the panel, for example a csv, PapaParse parses the file in the browser, then the system auto-detects the data columns, trend region and residual uncertainty bands are computed, z-scores included. The LLM treats the uploaded data as main signals alongside the existing pipeline outputs.

Security Properties.

The front-end has specific security properties by design. These include the following: No Credentials in the Browser. All cloud SDK calls occur within the Next.js API routes, server-side. The browser only fetches from the Vercel endpoints. Three Distinct Authentication Surfaces. Separate secret functions to eliminate compromise across other endpoints. Atomic Use Limits. Managed key-value database conditional enforcement of passphrase usage caps with no race condition. Read-Only Identity and Access Management. Vercel’s cloud credentials are scoped to read operations on a specific location via ARNs, the front-end cannot write to any of the internal systems or invoke operations such as serverless functions or modify any cloud components. Private Repository. Source code, including all front-end deployments are within a private Github repository with no public fork.

III. COMPONENTS

III-A Independent System Operator Inference & Training Pipeline

Overview.

Regarding the ISO pipeline, it comprises several serverless functions, covering ISONE, ERCOT, CAISO, and NYISO. Each function runs from the same container registry image, ~[CONFIDENTIAL], built per ISO, with the operating mode controlled by an environment variable.

Input data.

Each function reads a consistent set of inputs from the object storage before running, whilst the container resolves the correct path based on the ISO identifier.

How the Machine Learning Pipeline Works.

Instead of predicting electricity prices using linear regression formulas, the pipeline detects grid stress regimes. The structured operational states in which the relationship between grid generation mix and natural gas prices becomes statistically exploitable / predictable.

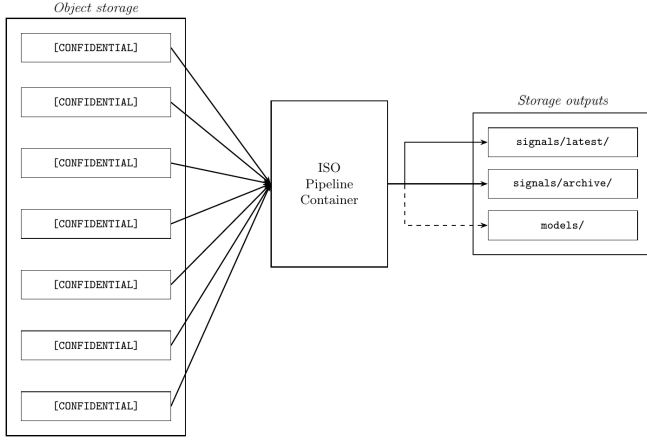


Fig. 11: Object storage input and output pipeline. The container reads seven input streams from the object storage.

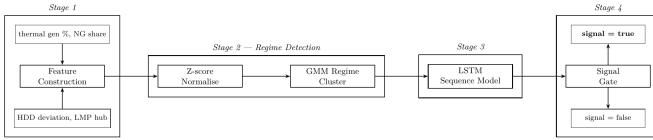


Fig. 12: ML pipeline stages. The process goes through four specific stages: feature construction, regime clustering, sequence modeling, and signal gate generation.

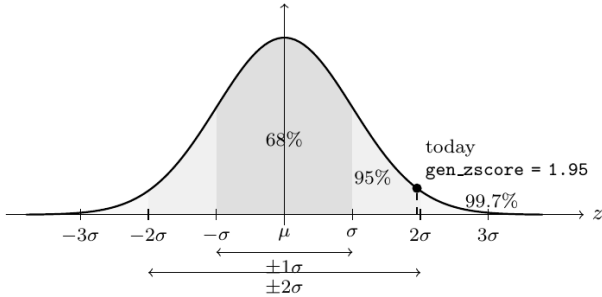


Fig. 13: Rolling z-score normal distribution. The bell-curve represents a normal distribution of a feature, over the trailing 30-day window. 1 standard deviations contain 68% of typical observations and 2 standard deviations cover 95%.

Step 1 – Z-score normalisation.

Prior to any model sees the data, each raw feature x becomes normalized against its own recent history using a rolling z-score: $z = (x - \mu_u) / \sigma_u$. μ_u and σ_u are the mean and standard deviation computed over the trailing $w = 30$ trading-day window. A z-score of 0, represents the central part of the normal distribution, exactly the recent average. A z-score of +2 represents two standard deviations above the mean, which would be unusual. Examples of this include unusually high thermal generation, or large natural gas electricity generation share. This is how the model can detect unusual activities that are occurring with the grid.

Step 2 — Gaussian Mixture Model regime detection.

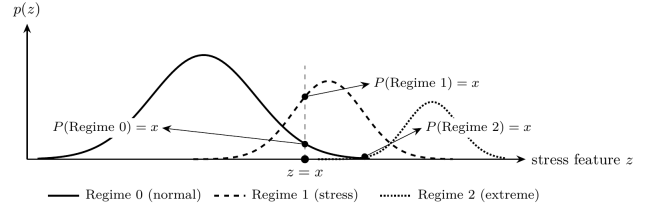


Fig. 14: GMM regime assignment. Three learned Gaussian components represent the three dominant grid states the ISO has historically occupied. For the observation of $z = 1.95$ on the stress feature axis, the GMM assigns the highest probability to Regime 1 (Grid Stress, 62%), a moderate probability to Regime 0 (Normal, 31%), and a small tail weight to Regime 2 (Extreme, 7%).

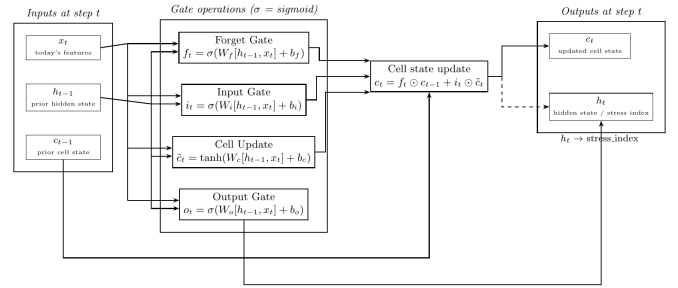


Fig. 15: LSTM cell at time step t . At each daily step, the cell receives the present normalized feature vector x and the prior hidden state represented as h_{-1} . The forget gate of f decides how much of the long-term cell memory c_{-1} to erase. The input gate represented as i and the cell update candidate together decided what new information is valuable to append into memory. The cell state $c = f \odot c_{-1} + i \odot \tilde{c}$ is the element-wise combination of retained old memory and new appended information. The output gate o controls how much of the updated cell state is exposed as the new hidden state $h = o \odot \tanh(c)$. In this pipeline h becomes the stress index compared against the learned stress thresholds in the signal gate, where all weight matrices of f , i , c and o are learned during retraining via backpropagation through time (BPTT) every week.

After normalisation utilizing z-scores, a Gaussian Mixture Model (GMM) assigns today's feature vector to one of K learned market regimes. Regime amounts are represented by the amount of K values. The GMM models the overall distribution of historical grid states as a weighted sum of K Gaussian components: $p(x) = \sum_k \pi_k \cdot \mathcal{N}(x | \mu_k, \Sigma_k)$. Each component k has a mixing weight π_k or how often the market is within that state, a mean μ_k represents what the regime looks like, and a covariance Σ_k shows the range or how spread out it is. At the inference time, the model computes the probability that today belongs to each regime and assigns the label of the most likely one.

Step 3 — LSTM sequence model.

Energy data is sequential, today's grid configuration depends on yesterday's generation dispatch and last week's EIA storage number, and temperature trends that are typical during the present season, etc. A standard feedforward network ignores temporal structure. An LSTM maintains a cell state c that acts as long-term memory, updated at each time step through the three learned gates (Forget, Input, and Output).

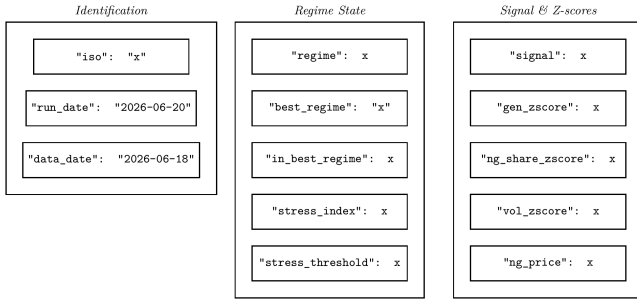


Fig. 16: Example Output. signal record structure for ISONE on June 20, 2026.

Signal output.

Both the inference that occurred along with the edge-signal functions write a single JSON file, overwriting the previous run on each execution:

III-B Machine Learning: Unsupervised & Signal Models

Function 1 — Unsupervised Machine Learning.

Overview.

The model runs three times daily and has no labeled training targets, representing the unsupervised aspect. It combines four distinct techniques into a single inference pass, these include: 1) LSTM autoencoder that learns what normal market structure looks like and flags deviations. 2) Causal entropy tracker that measures the amount and what are the active physical drivers. 3) A topological data analysis (TDA) module that monitors the shape of correlation across all datapoints. 4) a neural, ordinary differential equation that models price dynamics using physical coefficients. The main premise of these functions are to identify patterns of correlation and impact that exist within the data, allowing for detailed assessment and understanding of relationships in price action. Memory: 1769 MB Timeout: 600s

LSTM Autoencoder.

Regarding what an autoencoder is, it learns how to compress data to a low-dimensional representation and then reconstruct it. The model itself is trained exclusively on normal market data, therefore, at inference time with real conditions applied, anomalous inputs such as conditions that are unseen produce larger reconstruction error which provides an anomaly signal. This acts as a signal inference for unseen conditions or things the model potentially as not expected.

The autoencoder uses a Long-Short Term Memory (LSTM) to process a rolling window of [CONFIDENTIAL] across [CONFIDENTIAL] features. These are then compressed into a 16-dimensional vector, then the decoder reverses this, attempting to reconstruct the original sequence.

Regarding the specific nature of how a neuron works, each input represented by x_i is multiplied by a learned weight w_i , summed

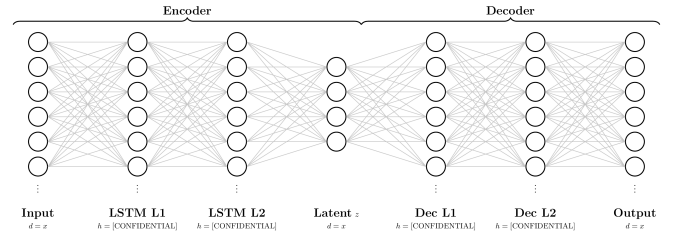


Fig. 17: LSTM autoencoder. The figure represents the autoencoder with the hidden layers. Each column acts as a layer whilst each circle represents a neuron. Grey lines are weighted connections between neurons, which each neuron in each layer connects to every neuron in the next layer. The encoder compresses, whilst the decoder mirrors the compression amounts.

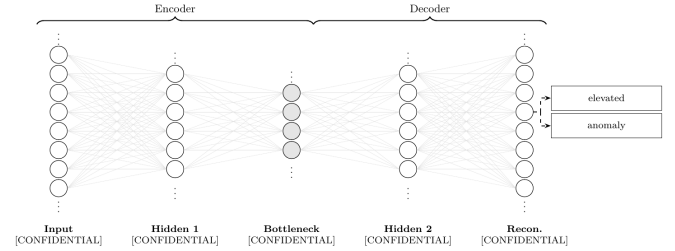


Fig. 18: MAS autoencoder. Structured by [CONFIDENTIAL] market compression, across massive data sets are fed simultaneously, every circle-to-circle connection is a learned weight. The massive scale of compression created limitations in accuracy, going from [CONFIDENTIAL] with the decoder creating bottlenecks that were missed or were inefficiency measuring states.

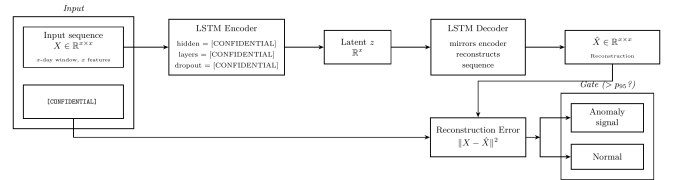


Fig. 19: Overview of Architecture. How it looks without the hidden layers shown. [CONFIDENTIAL] features over a [CONFIDENTIAL] sequence into a [CONFIDENTIAL]-dimensional latent vector z using a [CONFIDENTIAL]-layer LSTM with a hidden size of [CONFIDENTIAL]4. Reconstruction error $\|X - \hat{X}\|^2$ or the Euclidean distance of error. The smaller the distance, the less anomalous.

with a bias represented by b , then passed through a nonlinear activation function σ . Without each nonlinear component, every layer would collapse into a linear map which then stacks layers and adds nothing. The nonlinearity allows the deep networks to learn untraditional patterns. The weights and bias values are learned through a gradient descent methodology.

A second, larger autoencoder (mas_autoencoder.pt, 3.5 MB, trained May 17 2026) compresses [CONFIDENTIAL] simultaneous market features — spanning all five ISOs, petroleum, NG, weather, and satellite signals — to a [CONFIDENTIAL]-dimensional bottleneck. This has [CONFIDENTIAL] input features, named “MAS” or Market Aware System.

Function 2 — Causal Entropy.

Entropy is another word for uncertainty.

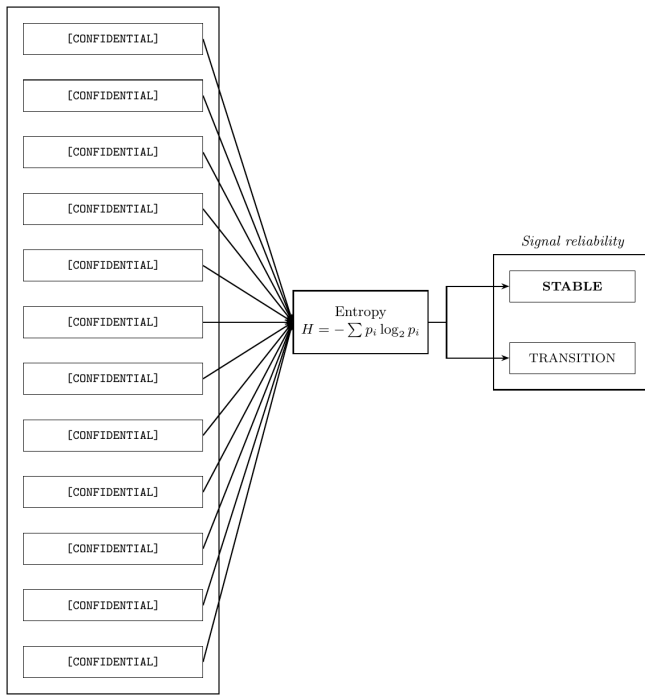


Fig. 20: Causal entropy tracker. [CONFIDENTIAL] driving NG pricing, [CONFIDENTIAL] driving HO pricing, and [CONFIDENTIAL] driving WTI pricing, each carrying a probability of p_i representing how strongly each driver is influencing price and showcasing potential cross-commodity correlation regarding price movements.

The model analyzes [CONFIDENTIAL] causal pathways between physical market drivers and the corresponding price points. Each pathway has an activation probability p_i which represents the strength of how that driver is influencing the price at a specific timestamp. Shannon entropy over these probabilities measures overall market clarity in price formulation. Shannon Entropy: H represents the entropy, p_i represents the probability of an outcome, $\log_2$ represents the measurement of entropy in units of bits, and sigma represents the sum of all possible outcomes in this system. $H = -\sum p_i \log_2 p_i$. The amount of low entropy reveals that one or two drivers are dominating market pricing, representing a clear, physical cause. Higher entropy represents that many drivers are actively impacting at the same time, which tells that the system is in a transitional / confused state.

Function 3 — Topological Data Analysis (TDA).

Literally creating a physical shape of the data.

The TDA measures the shape of the correlation matrix across all of the market variables, using tools from algebraic topology. The key aspects of technical framework include Betti numbers which β_0 counts the number of disconnected components in the correlation graph, and β_1 counts the number of independent cycles or loops within the graph. A cycle appearing in the correlation structure represents a set of assets mutually reinforcing components within a closed loop. This shows a classic signature of structural stress. When $\beta_1 > 0$ and

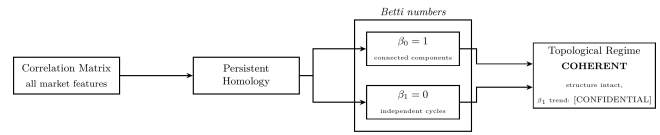


Fig. 21: TDA topological architecture. The correlation matrix goes through a persistent homology algorithm, which finds the topological features (connected components) across multiple distances. A rising β_1 historically precedes volatility and regime shifts by [CONFIDENTIAL], which provides a leading warning before price dislocations occur.

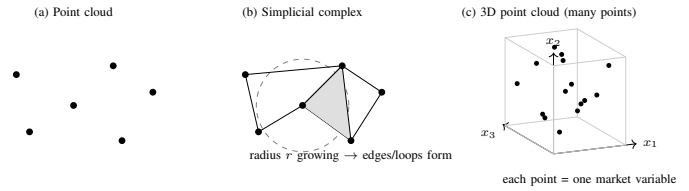


Fig. 22: Represents an illustration of three different aspects of data topology, including market position variable by pairwise correlation via distance to others, connection of radius growth, and correlation space of more than two dimensions.

persistence entropy is rising, the topology becomes more complex, which shows instability before it is represented in pricing. Complexity of topology is the measure of instability which can be used to predict pricing.

Component 4 — Neural Ordinary Differential Equations (ODE) (fair-value dynamics).

A neural network that conducts flow as opposed to consistent breaks / points of structure.

A neural ODE models actual price dynamics using a continuous differential equation rather than a discrete time step. For each commodity represented, the model learns physically interpretable coefficients that govern how price evolves / is formed over time. A Neural ODE models price dynamics as a continuous differential equation rather than a discrete time step. For each commodity (NG, HO, WTI) the model learns physically interpretable coefficients that govern how price evolves: $d(\log P)/dt$ represents instantaneous growth rate, which is equal to that of α weight for storage, β weight for weather, γ weight for the crack spread, δ representing the mean-version force that is pulling the price toward a learned value or speed of adjustment, and $\varepsilon_{\text{neural}}$ represents the neural network residual, complex adjustment.

$$d(\log P)/dt = \alpha \cdot \text{storage} + \beta \cdot \text{weather} + \gamma \cdot \text{crack} + \delta \cdot (\log P_{\text{fair}} - \log P) + \varepsilon_{\text{neural}}$$

A good analogy represents that of the classical terms (α , β , γ , and δ) as the predictable smooth components like a predictable highway of where the price should go based on fundamentals. Whilst the neural network residual $\varepsilon_{\text{neural}}$ acts like minor steering wheel adjustments for correcting wind, road texture, and traffic on the highway.

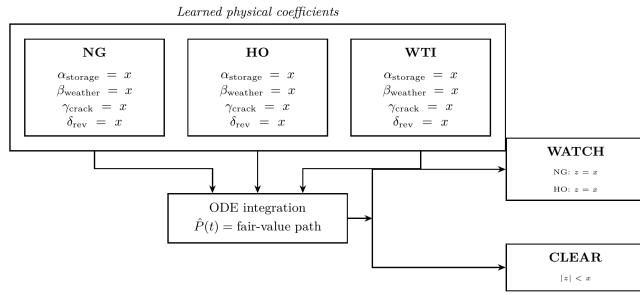


Fig. 23: June 20, 2026 Neural ODE coefficients and signals. Signals regarding anomalies, and driving force of pricing. Indicating aspects of fair value regarding the mean reversion signals.

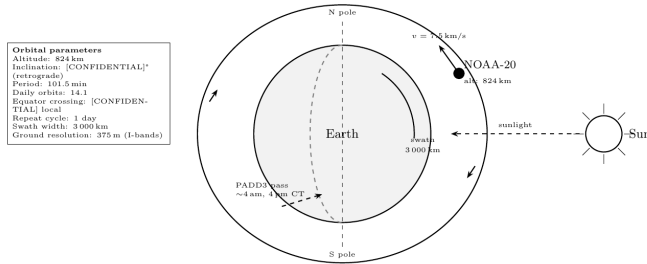


Fig. 24: NOAA-20 SSO. The orbit is in an inclination less than 90 degrees, so the orbital plane processes westward at .9856° per day. This matches Earth's revolution around the Sun, with a 101.5 minute orbital period, the satellite completes 14.1 orbits per day, and double global repeat coverage per day for the Gulf Coast / PADD3 latitude. The passes are consistent around the time of 04:00 and 16:00 CT, tagging nighttime passes with the label "N" to avoid solar heating contamination in data via thermal bands.

III-C Satellite Signal Models

Data.

Model Data.

Both models are driven by NASA's FIRMS (Fire Information for Resource Management System) API, which distributes thermal anomaly detections from the VIIRS instrument (Visible Infrared Imaging Radiometer Suite) aboard NOAA-20 (formerly JPSS-1). Refineries are dark, but they're process heat is not, and spotted with thermal infrared and surface temperature measurements.

NOAA-20 and the sun-synchronous polar orbit.

The NOAA-20 flies in a sun-synchronous, near-polar orbit (SSO). This represents a low-atmospheric orbital that allows for the satellite to pass by a specific location everyday at a consistent time. Regarding the equator, the satellite always crosses at the same time, 01:30 descending and 13:30 ascending. This guarantees a consistent solar illumination of geometry for every pass, which remains critical for measuring and removing solar-heat bias from surface temperature retrievals.

Electromagnetic spectrum and VIIRS detection bands.

The VIIRS instrument has 22 spectral bands that span 0.41

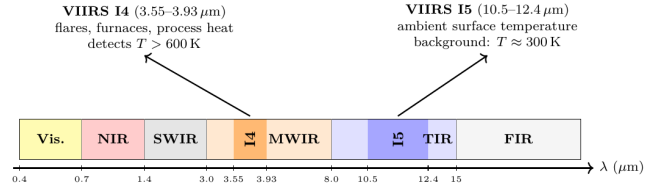


Fig. 25: VIIRS spectral bands. This represents where the bands fall regarding the μm ratio, where Lambda (λ) represents wavelength, Vis. represents visible, NIR represents near-infrared, SWIR represents short-wave infrared, MWIR represents mid-wave infrared, TIR represents thermal infrared, and FIR represents far infrared. VIIRS represents Visible Infrared Imaging Radiometer Suite by NASA.

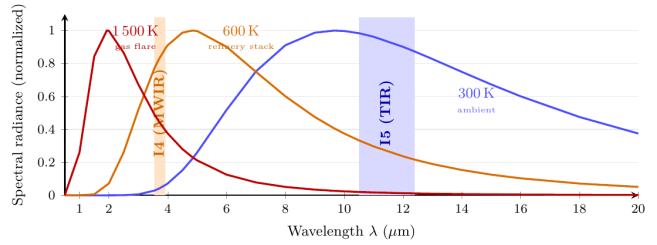


Fig. 26: Planck spectral radiance curves normalized for three temperatures. Each curve represented on these peaks at the same max of $2898/T \mu\text{m}$. At 300 K temperature, we see emissions peak at $9.7 \mu\text{m}$ in wavelength, whilst at 600 K, which represents refinery processing equipment in action, the peak shifts to $4.8 \mu\text{m}$. These separations of wavelengths are in the direct differences between the I4 and I5 bands.

to 12.5 micrometers (μm) or one-millionth of a meter. Two imagery bands are critical for energy infrastructure monitoring, they are the I4 and I5. I4 detects hot targets which include gas flares, furnaces, and processing heating radiating from the refineries above $\sim 600\text{K}$ and measuring between 3.55 through $3.93 \mu\text{m}$ which represents mid-wave infrared. I5 measures ambient surface temperature and is used to compute the background from which I4 datapoints are isolated. I5 acts as the control variable to provide a reference point for the I4 measurements. I5 measures around 10.5 to $12.4 \mu\text{m}$. Where $.7 \mu\text{m}$ is visible to the human eye and over $4,000\text{K}$.

Planck radiation: why hot sources glow in the infrared.

Every object that is above absolute zero emits thermal radiation, which this principle can be described within Planck's law as spectral power emitted per unit wavelength: $B(\lambda, T) = (2hc^2/\lambda^5) \cdot 1/(e^{hc/\lambda kT} - 1)$ h represents Planck's constant, c is the speed of light, k is Boltzmann's constant, T is temperature represented within Kelvin, and λ is wavelength. There is an aspect of temperature impact regarding peak shifts represented by Wien's displacement law where $\lambda_{\text{max}} = 2898 \mu\text{m} \cdot \text{K} / T$.

PADD3 geographic coverage.

Across the United States, there are five distinct locations of Petroleum Administration for Defense Districts (PADDs) used to track crude and refined goods. PADD3 serves as the hub for the production of these energy resources across the

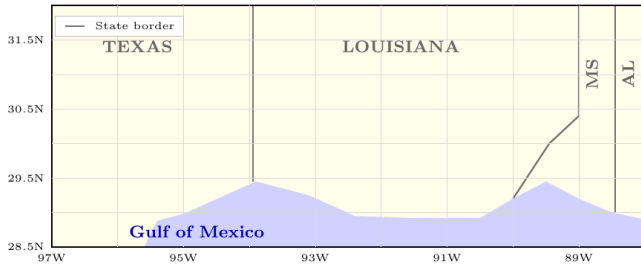


Fig. 27: PADD3 satellite coverage, 88°W–97°W. Numbered circles from 1 to 9 represent refinery / process clusters, whilst orange squares represent Strategic Petroleum Reserves (SPRs) and salt-cavern storage sites. A single NOAA-20 pass covers all 25 sites simultaneously via the 3,000km coverage.

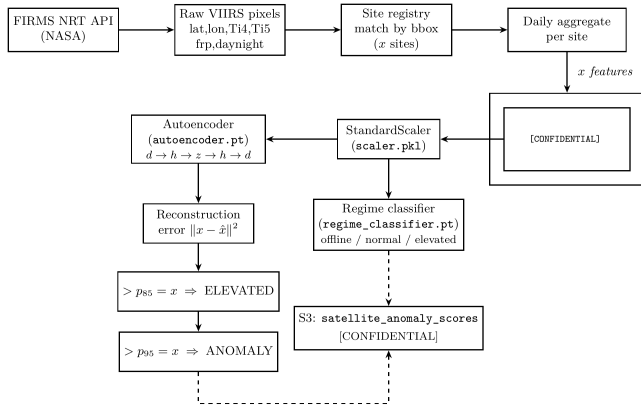


Fig. 28: Satellite Data and Machine Learning Pipeline. Daily aggregation computes the per-site Fire Radiative Power (FRP) statistics plus a 7-day momentum ratio and SWIR band-7 mean. The tabular autoencoder compresses 11 features into a 3d latent space; high reconstruction error shows anomalous activities. A separate regime classification system assigns each site-day to offline, normal, or elevated labeling systems. StandardScaler and OneHotEncoder are all methods of data cleaning for training models without having them skewed by numerical sizes of numbers or false representations of labeling.

United States. PADD3 is large, located in the Gulf Coast. The model monitors 25 sites across this area, spanning from Texas, Louisiana, Mississippi, and Alabama. Each site has a bounding box via coordinates, in which VIIRS pixels falling within the box are attributed to that site and aggregated to daily amounts.

Interpretation / Analysis.

Machine Learning via PADD3 Satellite.

Everyday, the serverless functions via the event buses call the FIRMS NRT (Near Real-Time) API with the prior day's data, and receives all of the VIIRS 14 detections within each site's bounding box, and aggregates them to per-site daily features, and then runs an inference through two models: A tabular autoencoder for anomaly detection and regime classification system for operating state labeling. These functions are similar to the ones shown in Figure 8.

PADD3 fire radiative power vs. heating oil price.

Refinery activity is a direct physical input into distillate supply,

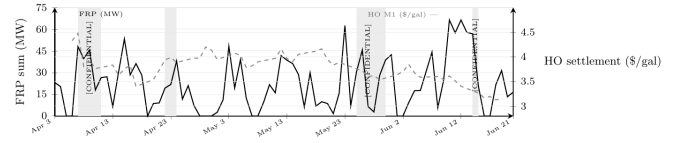


Fig. 29: PADD3 daily FRP Sum v Heating Oil / HO M1 (front-month) price from April through June 2026. Dotted Line = HO M1, Black Line = PADD3 FRP (~MW) The shaded bands (4) market periods when 10+ sites triggered the anomaly classification system (never seen before on training). The first major cluster occurred on April 7th, when 23 sites showed 48 MW (peak) via FRP. This was concurrent with a sharp HO sell-off from \$4.48 to \$3.76. Furthermore during this period, price regarding energy economics was heavily influenced by the situation occurring within the Middle East (macroeconomic impact).

when high FRP is represented, more heating oil and diesel is being produced. When the FRP collapses / is zero, the unit outages or planned maintenance may be tightening supply.

Refinery activity is a direct physical input to distillate supply. When throughput surges (high FRP, many detection events), more heating oil and diesel is being produced. When throughput collapses (zero FRP, no thermal signal), unit outages or planned maintenance may be tightening supply.

Synthetic aperture radar and InSAR ground deformation.

Whilst the VIIRS instrument remains effective for detecting surface thermal emissions, a second technique, Synthetic Aperture Radar Interferometry (InSar) measures the physical movement of a ground surface. The instrument measures the ground surface deformation above underground storage caverns with millimetre precision. A specific SAR satellite, the Sentinel-1, uses its C-band ($\lambda = 5.55\text{cm}$) to illuminate the ground with microwave pulses and records complex backscatter phasing. The difference between the phase of two passes separated by days to weeks gives the line-of-sight (LOS) displacement: $\Delta\phi = (4\pi/\lambda) \cdot d_{\text{LOS}}$ Where $\Delta\phi$ represents the difference, $(4\pi/\lambda)$ shows one 2π fringe = $\lambda/2 = 5.55/2 = 2.775\text{cm}$ LOS displacement and d_{LOS} represents $d_{\text{vertical}} \cdot \cos\theta + d_{\text{horizontal}} \cdot \sin\theta$ for satellite incidence angle θ is $\sim 35^\circ$

Deformation Metrics.

The salt domes of the Gulf Coast are massive bodies of halite (NaCl) that pierced overlying sediments over millions of years, their impermeability and mechanical strength make them ideal for underground storage. They are used to store gas, oil, or brine, by being injected under pressure and held indefinitely.

Peaker Plant Monitoring.

The ERCOT model monitors [CONFIDENTIAL] natural gas peaker and combined-cycle turbine plants across Texas, correlating their thermal signature with gas generation share via public data. When large peaker plants show elevated FRP, gas demand on the ERCOT grid is high, which tightens Henry Hub natural gas prices. The model output assigns specific dispatch labels. The natural gas, electricity-generation share provides an external supervision signal, allowing the model to distinguish

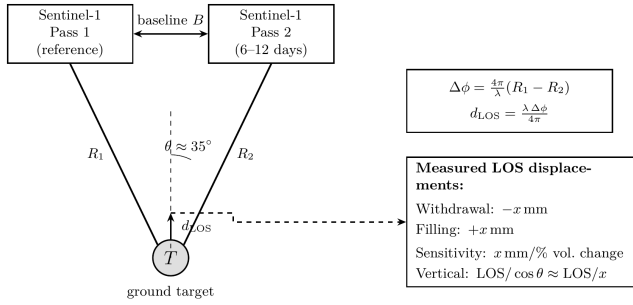


Fig. 30: InSAR two-pass geometrics and LOS displacement measurements. Two Sentinel-1 passes separated by 6 to 12 days record the same target at slightly different angles (baseline B), where the phase difference encodes the change in radar range between passes. Positive value means the ground moved towards the satellite providing a filling signal (decreasing supply in market), whilst negative means subsidence providing a withdrawal signal (increasing supply in market).

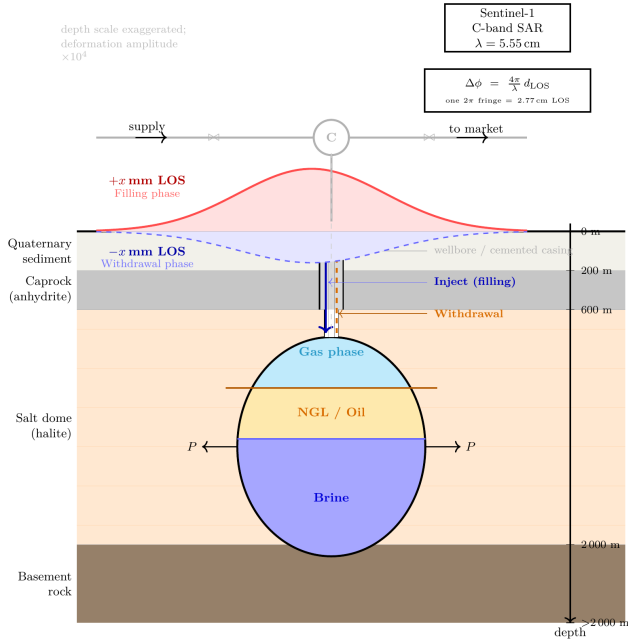


Fig. 31: Gulf Coast Salt Cavern. Represents the cross-section and pipeline on which surface deformation occurs. Caverns at Bryan Mound and Big Hill are roughly 70m in diameter, 300m tall at depths of 900 to 1,500m. Three fluid phases occur in the cavern, whilst pressurized NG in the gas phase (top), NGL/oil layer, and saturated brine at the storage floor. During filling, gas pressure rises and the surrounding salt deforms, propagating an uplift to the surface. The Sentinel-1 measures these sub-mm movements via InSAR phase difference at a theta of 35 degrees.

structural high-dispatch periods from anomalous plant restarts.

Compound Learning.

Both of the satellite models retrain autonomously at specific periods, regarding weights. This is to align with policy, macro, and subsects of catching decay. Each retraining cycle ingests all accumulated site-days and re-fits the autoencoder, updating the thresholds from the new error distribution. This is the same

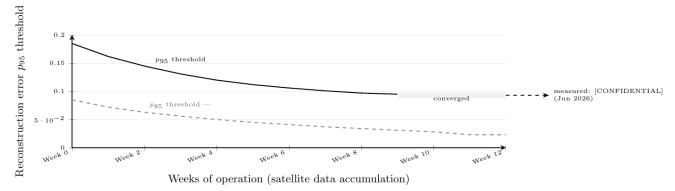


Fig. 32: Compound learning visual. Anomaly threshold tightening with accumulated data.

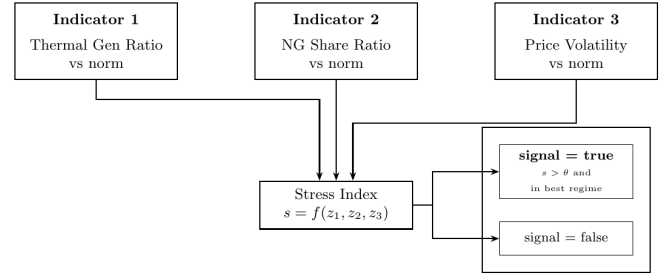


Fig. 33: Triple indicator system. Three z-scores each capture a different physical dimension of grid stress. Firstly, is the grid running unusually high on thermal generation, is the natural gas share of the fuel mix unusually high, separate from the overall thermal, and are the locational marginal prices (LMP) for [ISO]

function for the classifier as well. Early in development and creating real time predictions, the model has little history and anomaly thresholds are set from a historical distribution. As months of real time data accumulate in real time, the percentile thresholds become more accurate, where the false-positive rate drops, and can determine rare / genuine anomalies more effectively.

Triple Indicator ML System.

What it does.

This model runs daily and produces edge signals, which present the core per-Independ System Operator (ISO) trading signal used by a large language model (discussed later). It is called the triple indicator system because the stress index is developed from three independently computed z-scores, each measuring a different dimension of grid stress.

The three indicators.

elevated with volatility? The LMPs spiking consistently provides insight that the grid operator is struggling to balance supply.

Output record.

Market Spreads ML.

What it does.

Running daily on weekdays, the model predicts whether the natural gas–heating oil and natural gas–crude oil spreads are dislocated from their physical comparative fair values. It is a two-layer stacked model, with the first stage generating a

Identity & decision	Stress & regime	Z-scores & context
"iso": "x"	"stress_index": x	"gen_zscore": x
"run_date": "2026-06-20"	"stress_threshold": x	"ng_share_zscore": x
"data_date": "2026-06-18"	"regime": x	"vol_zscore": x
"signal": x	"best_regime": "x"	"ng_price": x
"reason": "OK"	"in_best_regime": x	"n_train_rows": x

Fig. 34: Signal Output Record Example for ISONE, June 20th, 2026.

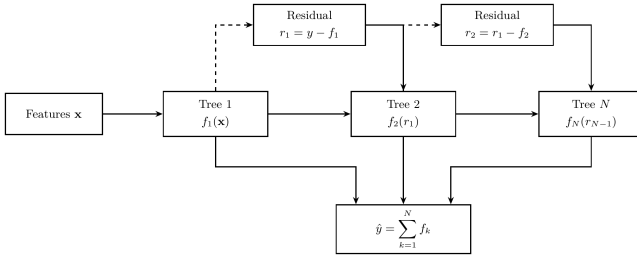


Fig. 35: LightGBM gradient boosting diagram. Each tree is fit to the residual, the prediction error of all of the prior trees. When Tree1 makes an initial prediction, Tree2 corrects what Tree1 got wrong. This occurs with TreeN where 1 through N-1 got wrong. The final output is $\hat{y}$ or the sum of all of the trees. The method also uses a leaf-wise tree growth, basically splitting the lead with the largest loss reduction first, making it faster and more accurate than depth-wise methods.

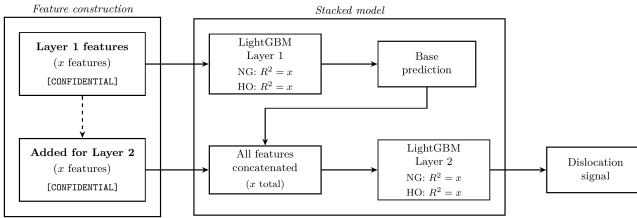


Fig. 36: Two-layer stacked model for natural gas-heating oil spread dislocation. Layer 1 uses fundamental physical features. Layer 2 receives Layer 1 features plus spread-specific features. Note: R2 is subject to be adjusted in the live model.

base prediction and the second stage refining it by adding spread-specific features such as lag, momentum, etc. in which the first stage is blind to. Both stages use LightGBM which are gradient boosted decision trees (unsupervised learning) trained on model artifacts that live with the object storage data lake.

LightGBM – Use Case.

LightGBM builds an ensemble of decision trees sequentially, each tree corrects the residual errors left by the previous stress, where the final prediction is the sum of all of the trees' outputs.

Construction of the Two-Layer.

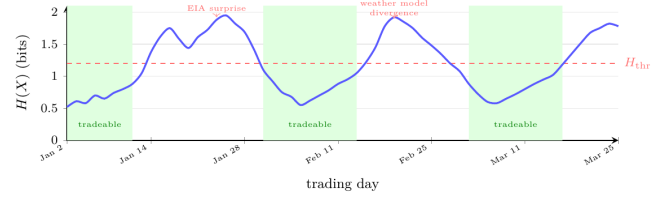


Fig. 37: Regime over Q1. Green shaded columns represent the threshold of entropy where agent88 generates signals.

III-D agent88 — Novel Signal Generation, Entropy, and Continuous Dynamics

What agent88 is.

Regarding the main ISO electricity models, these models are trained on structured time-series features that arrive on a fixed daily or weekly schedule. The agent88 provides a function via a container that targets the whitespace, signals which were not captured by scheduled ISO data pulls such as corporate hedging behavior, etc. Daily, entropy scans, novel feature synthesis, and signal emission functions are fired. These provide context regarding tradeable thresholds, pipeline nominations that arrived overnight, and textural rationale for reasoning layers.

Signal Quality.

Energy markets are often in high-entropy states, which are noisy, unstructured, and unpredictable due to the sheer multitude of external direction variables. The core of agent88 is that structured low-entropy physical regimes occasionally emerge from the typical noise, and are the only conditions worth trading in the public energy markets. Entropy provides the primary gate in which whether a signal generated is actually feasible to push the market in the predicted direction.

Entropy Over Time.

The gate is established and evaluated daily. Tradable windows are not predictable from the calendar, but rather emerge from physical conditions. High-entropy periods such as EIA surprise print, weather model divergence, and macro impact via geopolitical shocks are skipped entirely.

Mutual Feature Selection.

Mutual information measures how much knowing feature X reduces uncertainty about the target of Y. Where $I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$. Where H represents entropy. Features with $I(X;Y)$ below a floor are pruned during model training, whilst energy markets representing high-I features are physically determined. These include storage draws, generation fuel mixes, pipeline nominations, etc.

Entropy Loss for Regimes.

The training objective for all regime classification models is cross-entropy over the K-class output which is represented by $L = -\sum_i \sum_k y_{ik} \log \hat{p}_{ik}$, where $y_{ik}=1$. Where L represents loss,

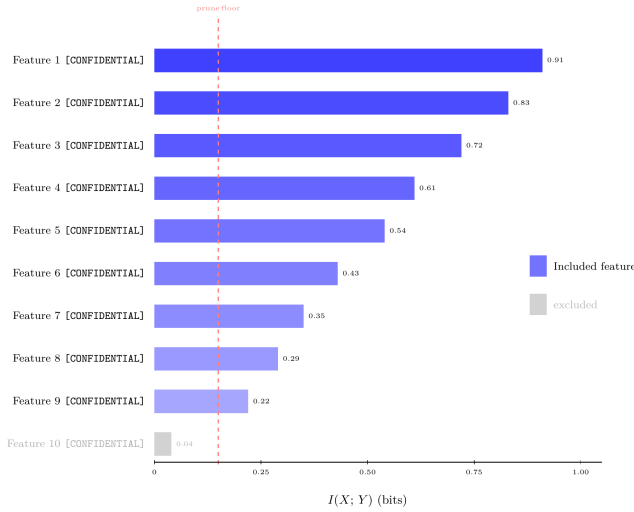


Fig. 38: Feature Mutual Info Ranking. Physical ISO and EIA features dominate predictive power, whilst EDGAR corporate hedging ratios, and other methodology carry significant $I(X; Y)$ above the prune floor, justifying their inclusion despite higher levels of latency.

Σ_i sum over all samples, Σ_k are the all possible regimes, y_{ik} is the onehot-encoder category represented as binary, 1s and 0s and $\hat{p}_{ik}$ represents predicted probability that i belongs to class k Where if $y_{ik} = 1$ and if sample i is truly within regime k and $\hat{p}_{ik}$ is the predicted probability, then minimizing L is equal to minimizing KL diverge between the predicted and true distributions.

Continuous Dynamics v Postings.

Discrete models such as LSTMs and transformers observe energy market state at calendar intervals, though the underlying physical steam is continuous and volatile. These include pipeline pressure, generator ramp rate, storage cavern temperatures, etc.

From discrete RNN to continuous ODE.

ODEs provide the bridge between the transition of energy observations, providing them to be continuous. A standard Recurrent Neural Network or RNN updates a hidden state discretely at each time step, whilst an ODE replaces this with a continuous differential equation.

Inter-report state estimation.

Between weekly EIA Natural Gas Storage reports, agent88 uses the neural ODE to maintain daily estimates of working gas in storage, using inputs to infer-report day + physical components.

EVOlutionary Model.

Gradient-based optimization finds good model weights but fails to effectively search for strategy parameters which includes discrete choices. The EVO model applies a genetic algorithm to this problem

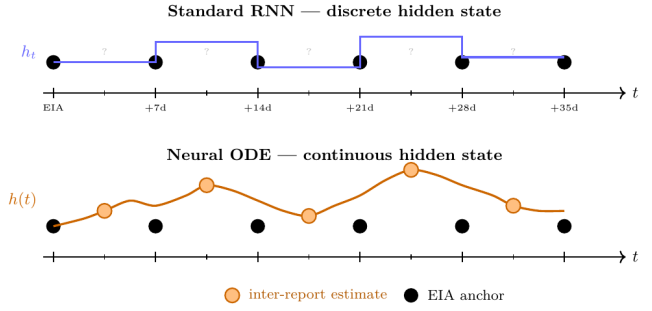


Fig. 39: RNNs v nODEs expressed. The top is the standard RNN as it produces a defined state only weekly at EIA report times. Whilst the bottom is the neural ODE integrating continuously providing daily probability estimates of storage states, which are anchored by the weekly EIA observations.

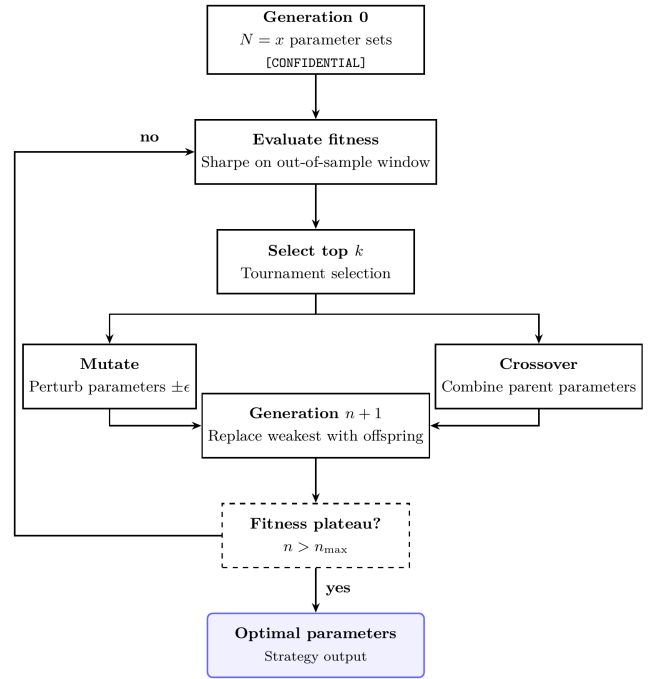


Fig. 40: EVO algorithm structure. A population of $N=50$ parameter sets is evaluated on an out-of-sample window using Sharpe ratios for assessments of output fit.

Convergence.

The algorithm typically converges within 60 to 80 generations, where early generators are dominated by random value variation and the crossover begins consolidating high-Sharpe parameter combinations after generation 20 and population median stabilizes above the initial baseline by generation 50.

Kolmogorov-Smirnov drift detection.

When training models based on historical distributions, it is important to implement some type of market shift analysis considering markets shift structurally. This includes policies by FERC or the DOE, new use of an LNG export terminal, etc. The Kolmogorov-Smirnov (KS) test is the primary mechanism

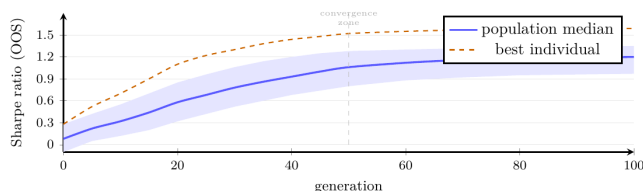


Fig. 41: EVO Fitness Convergence. The population median represented as the solid line and the best individual represented by the dotted line both improve rapidly through generation 40 then plateau.

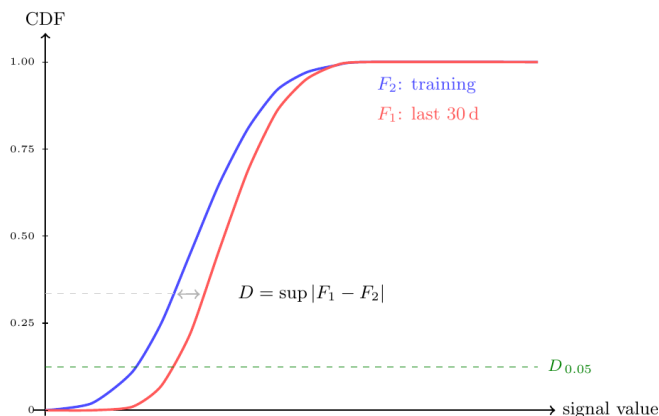


Fig. 42: KS Drift Detected and Expressed. CDF F_2 is the training function in blue and the CDF F_1 is the last 30 days function represented by the red. Regarding the green dashed line, or critical value of $D_{0.05}$, when D exceeds this line, the model is operating outside its training distribution. D represents drift, the larger the gap, the larger the drift.

for detecting this.

Two-Sample KS.

The KS statistic compares empirical Cumulative Distribution Functions (CDFs) of two samples: $D = \sup_x |F_1(x) - F_2(x)|$ where F_1 is the single CDF over the last 30 days and F_2 represents the training CDF. When D exceeds the critical value at $p < .05$, the distributions are significantly different, and the model shows drift. Probably known from the term “reject the null hypothesis.” Actions are put into place for retraining.

III-E Managed Foundation Model Autonomous LLM Layer, Three-LLM Architecture

Three LLM Architecture.

Most algorithmic trading systems are either pure quant, or pure LLM, where both fail at the boundary they ignore. This occurs because pure quant systems cannot assess whether a signal is physically coherent given the macro environment, whilst pure-LLM systems have no structured physical model to prevent hallucinated rationale + lack of quantitative capabilities because LLMs are inherently language prediction models. Combining both and creating a hybrid system proves to be the best approach. The machine learning pipeline / methodology previously discussed, with outputs landed within the object

storage. Combining that with the managed foundation model API component, using LLMs to read those structured outputs and provide critical thinking over them in natural language, grounded in early domain knowledge via RAG, proves to be exceptionally high-performing. The result is a written trade thesis that can be read, audited, discussed, before execution happens. Three distinct functions require three distinct communicative LLMs, because they operate on different information at different frequencies, with different risk.

Managed Foundation Model API.

The managed foundation model API is a managed API that exposes foundation models such as Claude, Llama, Nova, etc. without requiring any direct model hosting, GPU provisioning, or weight management.

Three properties are critical for a production priority information market prediction system:

Identity and access management-native security.

Serverless functions call the foundation model API via their identity and access management execution role, where no API keys are stored anywhere. No .env files, no hardcoded tokens. The call chain is an event bus trigger → serverless function → foundation model API.

No Data Egress to External Networks.

Foundation model API calls using Claude Sonnet 4.6 stay within the cloud provider’s regional boundaries. The input prompt which includes live position data, signal scores, portfolio state, and methodology never accesses the public internet. This is different from calling Anthropic’s API directly, where all inference traffic exits the cloud environment.

Audit Manager Integration.

Every foundation model API call is logged within the audit logging service, with the object storage bucket for further reinforcement training.

LLM 1 [Analyst].

The Analyst is the intellectual core of the system regarding macro synthesis and false signal detection. It receives a specific package of numerical outputs via the ML pipeline and must answer the question of if the signal is macro-consistent and physically coherent. Whilst this question cannot be answered by machine learning alone, a normal distribution modeling it can tell probable functions in specific regimes. This is combined directly with a consistent macro news feed with any public news headlines regarding energy, policy, and other impacting macros.

Macro filter.

The macro filter is the first pass, but it also checks whether any active macro conditions overrides the physical signal, regardless of how strong the ML confidence is. These include FOMC decision day, EIA WPSR release day, Geopolitical Supply,

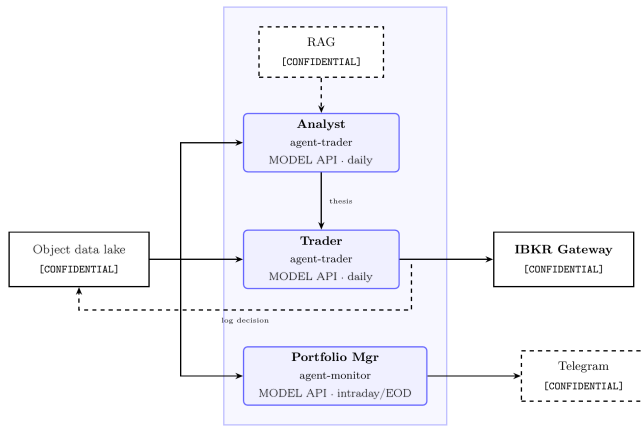


Fig. 43: Three-LLM Managed Foundation Model Architecture. The LLMs share a common state bus with the object storage data lake. The Analyst LLM synthesizes signals and macro context into direct trade thesis via RAG, then passes the thesis to the Trader LLM, makes the binary execute/hold/exit with decision and writes to both IBKR and the audit log. Whilst the PM LLM monitors the open position independently, enforcing profit and loss rules.

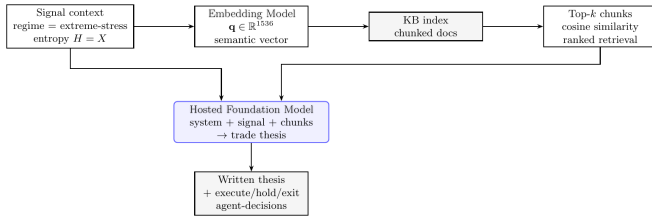


Fig. 44: RAG Pipeline Retrieval for Trade Thesis. Current signal context is encoding into a 1536-dimensional semantic vector by Amazon Nova Embeddings. The vector is matched against a chunked knowledge base via Sonnet’s context alongside the raw signal package. Claude critical thinks with the aspect of numerical signals, domain knowledge, and macro context into a written trade thesis that is grounded within retrieved text, not parameters or raw memory.

Weather Modeling, SPR release announced, Hurricane tracking over PADD3, etc.

RAG grounding.

Before anything is written regarding a trade thesis, it queries the Retrieval-Augmented Generation (RAG) system with a semantic vector derived from the present signal context. Knowledge base contains the aspect of strategy, energy physics, polices, methodology and texts. The retrieval step matters because LLMs hallucinate under specific domain pressure, where they will generate plausible-sounding energy physics that is wrong. RAG prevents hallucination with

retrieval, the model reasons over actual domain documents that are injected into context, not over pure knowledge baked during pre-training.

The output is a structured natural language document that is logged via JSON for reinforcement based on prediction versus what actually occurred.

LLM 2 [Execution].

The LLM 2 receives the thesis independent from the information that the LLM 1 utilized. This separate language model makes the final binary decision with a position to get placed. It has hard constraints and aspects of market data for feasibility such as liquidity, market prices, historical pricing, etc. Constraints include confidence thresholds, maximum contracts to open at a period, instruments to trade with, minimum risk/reward ratios, etc

IBKR Gateway (For Scaling).

Order execution reaches Interactive Brokers through a gateway container hosted on cloud compute instances (VMs). The gateway exposes an endpoint at a specific spot, where both LLMs have access to this point as an environment variable. Nothing is hardcoded, authentication is handled by the session on a completely protected internal virtual machine. When the trading system is active, the instance runs continuously during market hours and gateway maintains the session. Currently the IBKR Gateway is not being used, the infrastructure has been put into place and tested using port systems via Paper Trading, though Real Time Trading remains manual considering the risks of real capital.

LLM 3 [Portfolio Manager].

Once a position is open, LLM 1 and LLM 2 are out of scope regarding the nature of the position. The portfolio manager (PM) is the only LLM that has access to the live market for existing positions. It does not read signals or generate theses, it is trained to be an extremely calculated PM.

Exit conditions.

The PM manages three categories of exit:

Condition	Trigger	Action
Stop loss	Pre-set at entry, price-based	Immediate market exit
Thesis invalidation	Macro event changes direction	Exit at market; log reason
Hold period expiry	Entry date + hold_period days	Exit at next open
Profit target	Pre-set at entry, price-based	Limit order close
End of day reconciliation	Daily, after NYSE close	P&L snapshot to object storage

TABLE VIII: Portfolio manager exit conditions.

Reinforcement feedback loop.

LLM 1 performs a post-trade analysis that closes the learning loop. For each closed trade from the prior week, it reads the original analyst thesis, execution, daily P&L series, and the PM’s exit log. Furthermore, then it produces straightforward structured feedback based on the context. Though we are careful regarding autonomous retraining due to feedback loops without validation.

LLM Communication.

The three LLMs are not communicating directly, there is no

synchronization between foundation model API calls, instead the state flows through the object storage. This improves context during conversations and limits hallucinations.

Security via Identity and Access Management.

No foundation model API credential is stored in any system. Every serverless function that calls the foundation model API does through a specific identity and access management execution role which grants the permission to a specific model Amazon Resource Name (ARN).

III-F LLM Auto-Detection via Noisy Evidence from Cloud Infrastructure, and the $e + \hat{y} \rightarrow y$ Framework

Core concept.

The foundational insight regarding the project instruction is that LLMs can read numerical output and infer infrastructure, meaning if output + error/noise are readily available due to infrastructure setup mistakes, then the LLM can find the readily correct answer / what was meant to be achieved. Infrastructure flaws, in this section, are defined as minimal design failures regarding calculation skews or miscleaning of data implemented into the model. The three structures that are concurrent within our system include that of the following: Observable output. The result of $\hat{y}$, which includes that of z-scores, regime labels, reconstruction errors, ODE output, etc. is available to the LLM in structured form. Error / Noise is Observable. If the infrastructure makes mistakes, including misconfigured scalers, encoding inconsistencies, distribution shifts, missing data, etc. leave signatures are readable in the output itself to expose itself to the LLM. The LLM has Domain Knowledge / Context. The large language model understands the physical state of the world, it does not have the ability naturally to separate signals from artifacts based on pure numerical values without context.

The Idea.

Let $\hat{y}$ be the observed numerical output and ε be the observable noise/error signature due to infrastructure flaws. The LLM infers both the infrastructure I that produced $\hat{y}$ and the true intended signal where dk represents domain knowledge, then $y:(I,y) = \text{LLM}(\hat{y}, \varepsilon, dk)$ So counterintuitively, this is not a hallucination but rather structured reverse engineering. Where infrastructure mistakes are informative rather than merely cohesive to the system, where they expose the mechanism. A perfectly clean output with no visible error structure gives the LLM less to worth with than a noise output whose error pattern reveals how the pipeline was built and provides context. This design feature was not incidental, but rather the architectural reason for placing an agent that sits above the numerical pipeline rather than having a pipeline produce a final trading decision individually.

The $e + \hat{y} \rightarrow y$ framework in the pipeline.

Each numbered step shows where in the pipeline the LLM autodetection principle operates. Furthermore, the LLM 1 uses

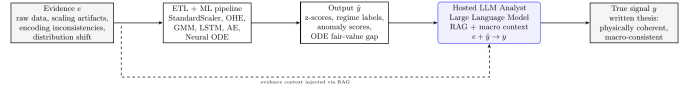


Fig. 45: Inference chain. The ETL and ML pipeline converts the raw evidence into structured numerical outputs represented as $\hat{y}$. Even if the evidence carries noise, which includes miscleaned data, distribution shift, encoded artifacts, and biased scaler fits, the LLM 1 received both the evidence and $\hat{y}$. From these two imperfect signals, it infers the true underlying physical state of y and writes the trade thesis.

its knowledge of both the evidence and output to infer the true underlying physical state.

Concrete examples of LLM noise correction:

ETL (evidence)	artifact	ML output ($\hat{y}$)	LLM inference (y)
StandardScaler trained on pre-2022 gas prices	OHE inconsistency: EDGAR sector mis-labeled	Anomalous z-score of +4.1 on storage	Scaler offset; true storage signal moderate
EIA WPSR revision (week t-1 restated)	NOAA station gap (KBOS offline)	EDGAR hedge signal suppressed for E&P	True hedging activity higher; signal understated
Distribution shift: post-hurricane PADD3 data	Neural ODE crack coefficient (γ) stale	Storage deficit z-score reversed sign	Prior week's number was wrong; trend intact
		HDD forward-filled for 3 days	HDD signal stale; elevated uncertainty
		Satellite AE reconstruction error > p95	Error is structural event, not model failure
		HO fair-value gap of -8.79σ flagged	Physical refinery margin compressed; gap real

TABLE IX: Concrete examples of LLM noise correction.

Why the LLM can perform this correction.

Three capabilities in addition to the LLM thinking abilities, make this system suited to the $e + \hat{y} \rightarrow y$ inference. Domain Knowledge via RAG. The knowledge base contains EIA methodology documents, ISO operating procedures, JP Morgan reviews / outlooks of expected electricity demand, and physics text of pipeline and barrel connections. The LLM 1 can retrieve the EIA's own description of how WPSR data revisions propagate or the ISONE operating manual's explanation of fuel dispatch order. This allows the LLM 1 to evaluate whether it is a given artifact in $\hat{y}$ as a data quality issue or a genuine physical event. Macro Context Awareness. Consistent news digest and macro event calendars provide the LLM 1 with external events that were and are occurring when the evidence was generated. A LMP z-score spike occurring at the same time as a ERCOT pricing decision has a different interpretation than a pipeline outage. The ML pipeline is blind to this, whilst the LLM layers are not. Falsifiable Thesis Generation. Rather than producing a raw number, the output is a written argument representing that if the predicted value was produced by corrupted evidence of a specific type, then y should look like this. This argument is then logged to `agent_decisions.json` and reviewed weekly by the reinforcement learning LLM which reads the original thesis, the actual outcome, and produces a structured review. The feedback then tightens both the ETL pipeline and the LLM's calibration of how to read noisy evidence.

Infrastructure Inference.

Every infrastructure decision leaves a numerical fingerprint:

Infrastructure property	Observable fingerprint in $\hat{y}$	What the LLM can infer
StandardScaler window too short	z-scores spike on moderate events	Scaler underestimates σ ; true deviation smaller
StandardScaler window too long / stale	z-scores systematically suppressed	Scaler μ reflects old regime; today's signal is stronger
Rolling window vs. expanding window	Seasonal z-score pattern (peaks every 12 months)	Expanding window; scaler has not adapted to seasonal level
OHE drop-first vs. drop-none	One ISO always reads near-zero signal	Reference level artifact; rescale relative to peer ISOs
OHE applied before/after merge	Feature vector length mismatch across runs	Encoding applied at different stage; columns shift
Forward-fill duration (1d vs. 5d)	Feature variance collapses for n consecutive days	Gap-fill active; date range affected by data outage
Retraining cadence (weekly vs. monthly)	Regime boundaries shift abruptly every 7 or 30 days	Retrain schedule readable from regime transition dates
Winsorization threshold	Hard cap visible at percentile-consistent value	Clip boundary; original value was higher / lower
Lag feature alignment	Signal leads price move by constant offset	Lag of k days baked in; adjust hold period accordingly
EIA preliminary vs. revised	Inverse signal flip the following week	Preliminary number was wrong; model retrained on revision

TABLE X: Infrastructure inference: numerical fingerprints and what the LLM can infer from them.

Why infrastructure mistakes are informative.

A clean, correctly-built pipeline produces output that is hard to diagnose from the outside: if everything worked, the numbers just look like good numbers. Infrastructure mistakes expose structure that would otherwise be invisible. The same logic applies across the pipeline. A gap-filled heating degree day series looks like a normal heating degree day series, but its variance drops to zero regarding the gap-filled. This would be a mistake, and each mistake, similar to these is a data point about how the infrastructure was built.

Sufficient Conditions for Infrastructure.

The LLM can successfully infer the pipeline and recover y when: Error is systematic, not purely random or invalid. Random noise averages out, systemic artifacts do not. A misconfigured scaler produces the same bias every day. Domain Provides a Physical Ground Truth. Energy markets provide ground truth to be publicly available via government releasing, such as EIA storage reports, ISO operating data, and weather station records. Structured Output. Unstructured text outputs hide their error structure, whilst numerical outputs expose it. The pipeline produces exclusively structured numerical outputs precisely because structured outputs are readily reviewable / transparent.

Noise Taxonomy and LLM recovery strategy.

Not all noise is the same. The LLM analyst handles structurally different noise types in structurally different ways:

Noise type	Origin in ETL	Signature in $\hat{y}$	LLM recovery
Distribution shift	Scaler fit on stale window	Systematic z-score bias	Cross-check against macro context; discount magnitude
Encoding error	Inconsistent OHE mapping	Category signal too weak / absent	Compare to prior-period EDGAR trend; flag uncertainty
Missing data	Forward-fill for n > 3 days	Stale feature values, low variation	HDD / NOAA gap flag in context; widen confidence range
Outlier contamination	Pre-winsorization API glitch	Single-day z-score spike	Check raw archive; confirm EIA / ISO corroboration
Model regime mismatch	GMM trained before structural break	Confidence score unexpectedly low	Entropy H rising; hold signal; flag for retrain
Measurement error	EIA WPSR preliminary vs. revised	Storage signal reverses next week	Analyst notes revision risk; PM monitors thesis validity

TABLE XI: Noise taxonomy and LLM recovery strategy.

This is the operational form of the $e + \hat{y} \rightarrow y$ framework where each noise type has a recognizable signature in the output, with a traceable origin within the ETL layer, and a specific recovery strategy. All of this is put into place just in case an underlying condition creates an unwarranted noise / error.

III-G Cost, Environmental Considerations & Conclusion

Cost

The total cost regarding the raw infrastructure external to the token framework via the managed foundation model API is less than \$35 a month, whilst there is a marginal cost regarding the amount of tokens that are utilized and under what model infrastructure is integrated. The current system uses Sonnet 4.6 by Anthropic via the platform, which can provide costs up to \$70 a month due to our strict token ceilings and caps on our infrastructure use.

The cost reduction is directly associated with environmental discipline which serves as a core focus and priority when developing infrastructural and componential relationships via the three aspects provided. Whilst this is true, there is no claim to a precise carbon number as that requires instance level CPU telemetry and marginal carbon intensity of the grid at the moment of execution. This varies continuously. The methodology for any representation of a number to be computed honestly at any period of time rather than an asserted one:

Regarding compute hours, for serverless functions, the cloud provider bills the exact invocation duration in GB seconds. This expresses the (billed duration) · (memory allocated) over the 50+ functions across the telemetry and metrics service billing. These logs give an exact, auditable, compute-hour figure for the entire pipeline.

Furthermore the aspect of energy is the process of converting compute-hours into kWh using a watts-per-vCPU method for the instance class that is being calculated. The methodology of the cloud provider and third-party tools provide coefficients per

instance family, where the provider’s carbon footprint reporting tool does the equivalent calculation natively.

Energy & Carbon Emissions

Month	Service	LBM (kg CO2e)	MBM (kg CO2e)
Jan	CDN	0.000	0.000
Jan	Object Storage	0.000	0.000
Jan	Serverless + Other	0.000	0.000
Feb	CDN	0.000	0.000
Feb	Object Storage	0.000	0.000
Feb	Serverless + Other	0.000	0.000
Mar	CDN	0.000	0.000
Mar	Object Storage	0.003	0.001
Mar	Serverless + Other	0.025	0.007
Apr	CDN	0.000	0.000
Apr	Object Storage	0.055	0.015
Apr	Serverless + Other	4.880	1.212
May	CDN	0.000	0.000
May	Object Storage	0.159	0.044
May	Serverless + Other	11.030	2.632

TABLE XII: Monthly energy and carbon emissions by cloud service, location-based method (LBM) and market-based method (MBM).

Analysis. For reference of scale regarding the CO2 emissions, 11 kilograms is roughly the same as driving a gas car around 27 miles.

Technical. Regarding the low-emissions for compute infrastructure correlates directly with how the framework operates. The architecture is event-driven, serverless compute, meaning functions only execute when triggered and occur within milliseconds. This is rather than servers running continuously regardless of load, which our methodology reduces the energy consumed.

Processor. The account uses x86 64 (also known as AMD64) across all 50 functions. The provider’s ARM64 processors deliver the same functionality / compute up to 60% lower energy consumption per the provider’s own published benchmarks, with the serverless runtime supporting ARM64 as a drop-in runtime target requiring limited application rewrite for Python workloads.

Futures and ETPs as Custody-Free Exposure

The second half of a clean fund’s footprint is not computational — it is what the fund chooses to own. This section argues that the choice of instrument, not just the choice of view, determines whether a fund’s positions add to real-world emissions.

A physical or delivery-based commodity contract requires someone to store, insure, and transport the underlying barrel, railcar, or cargo. Taking physical delivery, or trading instruments structured around guaranteed delivery, creates incremental demand for tankage, pipeline capacity, marine transport, and the energy those activities consume. This is true even for a fund expressing a bearish or hedging-motivated view: a short physical position can still require borrowing and moving real product to settle.

Conclusion

What this paper has described is three separate layers built on top of one another which includes: a serverless infrastructure stack that functions as the data ingestion layer, a machine learning layer that turns the data into actionable and recognizable states based on discovered states and patterns, and a structured three-LLM framework that acts as the reasoning layer behind the numerical functionality combined with the sentiment and macroeconomic analysis that estimates predictions structuring as opposed to raw price predictions. Each of these three layers is real, running, and auditable today, via the direct use case.

None of this is a specific claim that we have unlocked an innovative architectural feat that can predict energy markets directly, and it should not be read as one. What has been validated includes the functionality, architecture, and combination of components. What has not been validated yet includes a multi-season track record regarding the reasoning layer making unsupervised decisions with real capital behind them. That is the honest state of work as of this writing, though real capital is being deployed and manual position placement is the current protocol, the future outlook is accompanied by consistently scaling, reinforcing, and adding more edge components, including markets, contracts, and specific actions that are utilized.