# S&P 500 Sector-Sensitivity Monitor

## An Automated ELT Pipeline for Detecting Structural Market Volatility

Disha Rawat
Student
disha.rawat@sjsu.edu

Haritha Gouttumukka
Student
nagasaiharitha.gottumukkala@sjsu.edu

Rahul Mohan Devi
Student
rahul.mohandevi@sjsu.edu

Ruchika Chaurasia
Student
ruchika.chaurasia@sjsu.edu

*Professor: Dr. Keeyong Han*
*Department of Applied Data Science, San José State University*
*DATA 226 – Data Warehouse and Pipeline (Section 23), Fall 2025*

*Abstract - This paper presents the design, implementation, and operationalization of an automated, cloud-native analytics pipeline for quantifying sector-level behavioral influence within the S&P 500 index. Traditional financial dashboards and market analytics tools disproportionately rely on sector weights or historical price charts, creating an "illusion of diversification," where the index is perceived as widely diversified despite being highly sensitive to the behavior of a few dominant sectors, especially Technology. To address this analytical blind spot, we develop a warehouse-native ELT system that computes covariance-based rolling betas, log-returns, weighted risk contributions, sentiment indicators, and machine learning–based forecasts using Snowflake Cortex.*

*The system integrates five major technologies—Apache Airflow, Snowflake, dbt, Snowflake Cortex, and Preset—to automate ingestion, feature engineering, model training, prediction, and reporting. We implement four decoupled Airflow DAGs that orchestrate raw data ingestion, dbt transformations, ML forecast generation, and downstream serving-layer preparation. The resulting pipeline processes six years of historical market data, performing incremental extraction from the source API while utilizing a truncate-insert strategy into the Staging layer to guarantee idempotency, and supports fully automated dashboard updates with no manual intervention.*

*The final analytical outputs include sector volatility heatmaps, a strategic risk–reward opportunity matrix, multi-sector regime indicators, and a volatility forecast cone. The results reveal previously hidden correlations, regime shifts, and structural changes within market behavior, demonstrating that rolling betas and risk-contribution metrics capture market dynamics far more accurately than traditional sector-weight analyses. The paper concludes with engineering lessons, system limitations, and future enhancements including dbt snapshots, versioned sector metadata, and multi-factor modeling. The system employs a fully containerized development environment using Docker, enabling consistent reproducibility and seamless multi-developer collaboration throughout the project lifecycle.*

*Index Terms— Snowflake, Apache Airflow, dbt, Snowflake Cortex, ELT Pipeline, Rolling Beta, Covariance, Financial Analytics, Market Regime Detection, Preset Dashboard.*

## INTRODUCTION

### *I. Background and Motivation*

The S&P 500 index is widely interpreted as a highly diversified portfolio representing the U.S. economy. However, historical evidence consistently shows that a small subset of sectors—notably Technology—drive a disproportionate share of total index movement. This results in a structural imbalance where market participants perceive diversification, but underlying beta behavior suggests significant concentration risk.

Traditional dashboards focus on sector weights (market-cap distributions), daily price changes, or simple correlation matrices. Yet these methods ignore **rolling sensitivity**, **dynamic covariance**, and **risk contribution**, leaving investors and analysts blind to shifts in behavioral influence. As a result, they may overestimate diversification and underestimate systemic risk.

From a data engineering standpoint, market analysis pipelines are often fragmented, using disparate scripts, multiple transformation engines, ad-hoc notebooks, and manually refreshed reports. This leads to operational brittleness, inconsistent metrics, and latency between data generation and insight delivery.

To avoid environment drift and ensure team-wide consistency, all development was performed within Docker containers, with dbt models, ingestion logic, and ML workflows executed against a shared Snowflake environment. This centralized, containerized workflow served as a unified source of truth for all computations.

### *II. Problem Statement*

There is a lack of a unified, real-time system that:

1) Reliably ingests historical and daily OHLCV data.
2) Computes statistically robust features such as log-returns and rolling covariance.
3) Calculates rolling betas and sector influence metrics.
4) Trains predictive sensitivity models directly inside the data warehouse.
5) Generates analytical marts and dashboards without manual intervention.
6) Preserves data quality and ensures idempotency.

Thus, the problem addressed in this project is twofold:

- **Analytical Gap:** The inability of traditional tools to measure real behavioral drivers of index movement.
- **Engineering Gap:** The absence of an end-to-end automated ELT + ML pipeline that produces daily sector-sensitivity insights.

### *III. Objectives*

Our system is designed to achieve:

1) Measure True Behavioral Influence
   - Compute log-returns for statistical stationarity
   - Compute rolling covariance, rolling variance
   - Derive rolling betas relative to SPY
   - Quantify risk contribution using rolling beta × sector weight
2) Automation Through Four Decoupled DAGs
   - Daily ingestion
   - dbt-based feature engineering
   - Cortex training and forecasting
   - Downstream mart creation
3) Fully Warehouse-Native Compute
   - No data leaves Snowflake
   - Eliminates data movement overhead, improving reliability and performance
4) Daily Dashboard Updates
   - Preset dashboards refresh automatically after all DAGs complete

5) Ensuring Reliability
   - Idempotent truncate-insert patterns
   - Sensor-based dependencies
   - dbt test suites for data quality

## *IV. DataSet Description*

### *A. Data Sources*

We use two forms of data as required by project specifications:

1) Historical Dataset
   - 6 years of OHLCV data for 11 S&P 500 sector ETFs: XLK, XLE, XLF, XLY, XLI, XLV, XLU, XLP, XLB, XLRE, XLC
   - Benchmark index: SPY
   - Retrieved using the yfinance Python library.
2) Real-Time Dataset
   - Daily OHLCV updates are fetched automatically via Airflow at 02:30 UTC.
   - Ensures that dashboards and ML forecasts remain current.

### *B. Data Importance*

Sector ETFs serve as tradable proxies for the aggregated performance of underlying constituents. Because their volatility and covariance properties vary over time, rolling calculations capture temporal dynamics that are not observable in static price charts.

## *V. System Architecture*

The architecture consists of five tightly integrated components:

- Yahoo Finance (data source)
- Apache Airflow (orchestration)
- Snowflake (warehouse + ML compute)
- dbt (SQL transformations)
- Preset (visual analytics)

### *A. High-Level Architecture*

The architecture follows a modular, scalable ELT paradigm emphasizing:

- Compute immutability
- Zero data movement
- Logical separation of ingestion, transformation, and serving

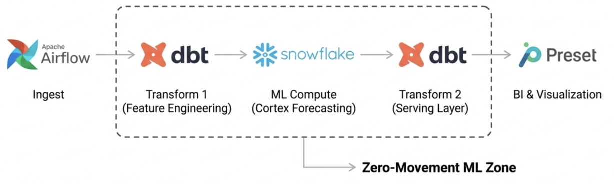

Fig. 1. End-to-End ELT Architecture

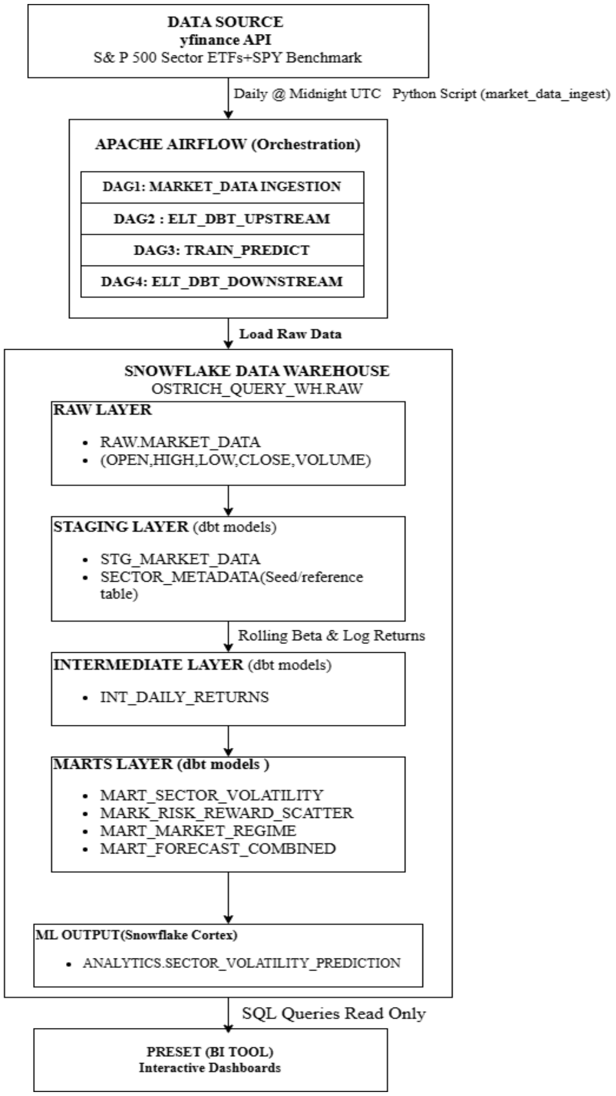

*Fig. 2. Decoupled Orchestration Logic*

### *B. Airflow Pipeline Implementation*

Our system includes four decoupled DAGs connected only through ExternalTaskSensors, preventing cascading failures and enabling partial recomputation.

1) DAG 1 — Daily Ingestion
   - Downloads OHLCV data for all sector ETFs + SPY
   - Executes truncate-insert for idempotency
   - Validates non-null values, correct date ranges, and duplicate primary keys

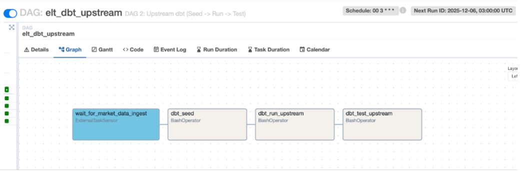

Fig. 3. DAG 1: Market Data Ingestion

2) DAG 2 — Upstream Transformations
   - Runs dbt seed for sector metadata
   - Executes staging models to clean and standardize data
   - Builds intermediate models such as log-returns
   - Computes volatility metrics used by all downstream marts

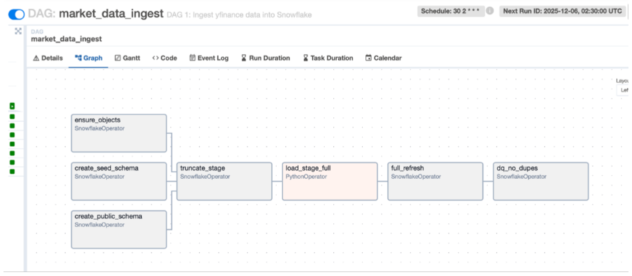

Fig. 4. DAG 2: Upstream Transformations

3) DAG 3 — ML Training and Prediction
   - Runs Snowflake Cortex's FORECAST model
   - Uses 730 days of historical rolling betas
   - Produces 30-day future forecasts with confidence intervals
   - Stores predictions in ANALYTICS.SECTOR_VOLATILITY_PREDICTION

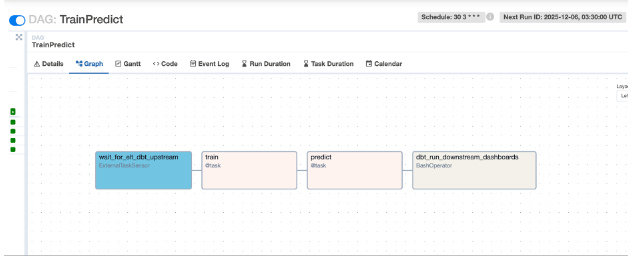

Fig. 5. DAG 3: ML Training & Prediction

4) DAG 4 — Downstream Serving
   - Merges historical rolling betas with forecasted values
   - Materializes final marts consumed by Preset
   - Ensures dashboards refresh only after all dependencies succeed

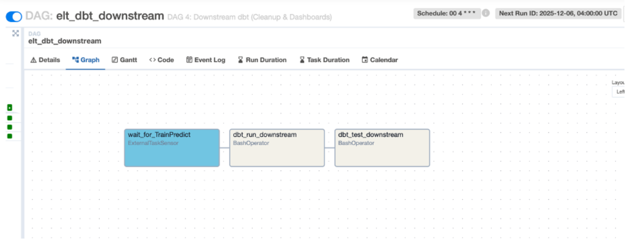

Fig. 6. DAG 4: Downstream Serving

### *C. Docker-Based Containerized Development Environment*

To ensure consistent results across all team members, the entire development environment was containerized using Docker. The container included:

- Python + yfinance extraction scripts
- dbt Core with Snowflake adapter
- Airflow services
- Dependency-managed Python environment
- Shared credentials using Airflow Connections

This unified environment enabled:

1) **Consistent Dependency Versions**
2) **Reproducible Pipelines Across Machines**
3) **Isolation Between Development and Production**
4) **Shared Execution Against a Single Snowflake Instance**
5) **Seamless Team Collaboration and Version Control**

## *VI. Data Model And Transformation Logic*

### *A. Medallion Architecture Overview*

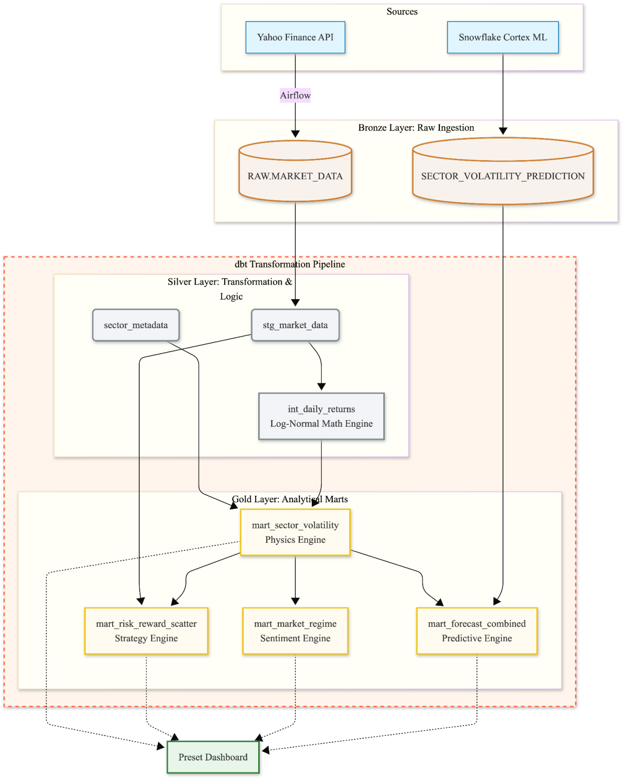

Fig. 7. Snowflake Medallion Schema

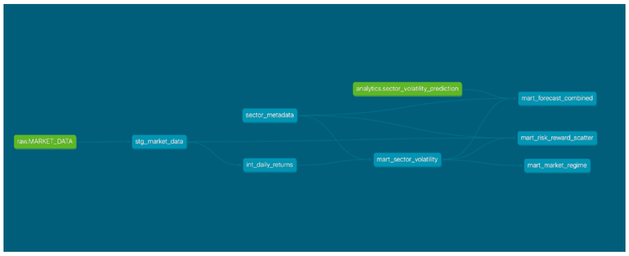

Fig. 8. dbt Data Lineage

### *B. Data Dictionary & Schema Definitions*:

| Table (Layer) | Description | Key Columns |
|---|---|---|
| RAW_MARKET_DATA (Bronze) | **Raw JSON payloads ingested from yFinance.** | **payload, ingest_ts** |
| STG_MARKET_DATA (Silver) | **Deduplicated OHLCV with Log-Return calc.** | **ticker, date, close, log_return** |
| SECTOR_META DATA (Silver) | **Static reference mapping ETFs to Sector names.** | **ticker, sector, weight** |
| MART_SECTOR_VOL (Gold) | **Rolling Beta & Normal Returns derived from Staging.** | **date, ticker, beta, normal_ret** |
| ANALYTICS_PRED (Gold) | **Cortex ML output with confidence intervals.** | **date, ticker, forecast, bounds** |

### *C. Mathematical Logic*

1) Log-Normal Returns

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right)$$

Preferred over simple returns for:

- Stationarity
- Normality assumptions
- Covariance stability

2) Rolling Covariance and Beta

$$\beta_t = \frac{\mathrm{Cov}(r_{\text{sector}}, r_{\text{SPY}})}{\mathrm{Var}(r_{\text{SPY}})}$$

3) Risk Contribution

WeightedImpact = βt × SectorWeight

4) Total Return (for Opportunity Matrix)

$$R_{90} = \frac{P_t - P_{t-90}}{P_{t-90}}$$

5) Market Regime Indicator

$$\text{Spread} = \beta_{\text{offensive}} - \beta_{\text{defensive}}$$

6) Forecast Stitched Series

$$\text{BetaFinal} = \text{COALESCE}(\beta_{\text{actual}}, \beta_{\text{forecast}})$$

## *VII. Machine Learning With Snowflake Cortex*

### *A. Model Architecture*

Cortex's **FORECAST** function applies a gradient boosting–based time series model optimized for Snowflake-native compute.

**Inputs:**

- 730 days of rolling beta history
- Date as time column
- Sector as series column

**Outputs:**

- Forecasted beta
- 95% lower bound
- 95% upper bound

### *II. Training Advantages*

- No data movement
- No dependency on external notebooks
- Native scaling based on warehouse size
- Automated hyperparameter optimization

## *VIII. Dashboard Visualizations And Insights*

### *A. Sector Sensitivity Heatmap*

Shows dynamic volatility cycles.
Interpretation:

- **Deep red** = high sensitivity
- **Blue** = decoupled / low influence

### *B. Risk–Reward Opportunity Matrix*

Plots:

- X-axis: 90-day rolling beta
- Y-axis: 90-day return Bubble size: relative sector weight

Quadrants reveal:

- High Beta, Low Return → Inefficient risk
- Low Beta, High Return → Defensive outperformers

### *C. Market Regime Indicator*

Tracks spread between cyclical vs. defensive sectors.

Offensive: XLK (Tech), XLY (Cons. Disc), XLC (Comm).
Defensive: XLU (Utilities), XLP (Staples), XLV (Health).

### *D. Forecast Cone*

Shows expected sensitivity path over 30 days.

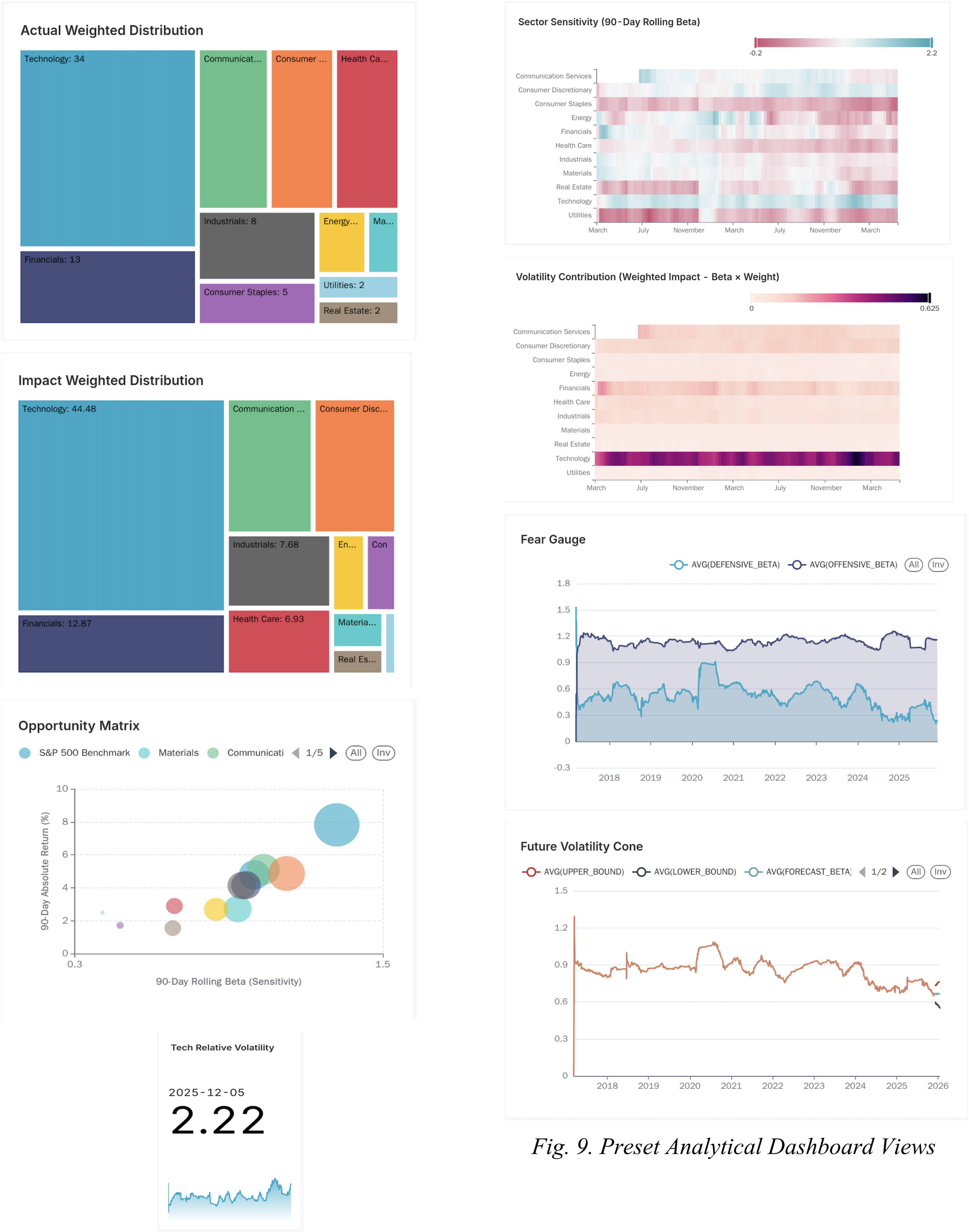

*Fig. 9. Preset Analytical Dashboard Views*

## *IX. Forecast Insights & Regime Validation*

### A. AutoML Approach & Evaluation Strategy

Unlike traditional supervised learning pipelines that require manual feature selection, training splits, and hyperparameter tuning, our system utilizes **Snowflake Cortex**, a fully managed AutoML service. As a result, standard performance evaluation metrics (such as RMSE or MAPE) were not explicitly calculated during the pipeline execution, as the model training and optimization processes are abstracted within the Snowflake compute layer.

Instead, validation focused on **directional regime consistency**. Given the project's core problem statement—the "Illusion of Safety" driven by sector concentration—the primary success metric was the model's ability to plausibly forecast the persistence of high-volatility regimes.

### B. Findings: The "Sticky Risk" of Technology

The 30-day volatility forecast (Fig. 9) serves as a critical validation of our hypothesis regarding market overvaluation.

- **Observation:** The Technology sector (XLK) currently exhibits a Rolling Beta consistently above 1.5, indicating it is significantly more volatile than the broader index.
- **Forecast Insight:** The Cortex model predicts **sustained high sensitivity** for the next 30 days, showing no "drastic drop" or mean reversion to safety.
- **Implication:** This flat/stable high-risk forecast confirms that the S&P 500 remains structurally dependent on Technology's volatility. The model correctly identifies that the current "high-beta" regime is stable, signaling to investors that despite potential overvaluation, the market dynamics are not yet shifting toward a defensive rotation. The lack of a predicted drop validates the "Illusion of Safety," as the index continues to rely on its most volatile components for momentum.

## *X. Lessons Learned*

1) **Static vs. Dynamic Sector Definitions:** Given that the 11 S&P 500 GICS sectors are structurally stable, we opted for **hardcoded ETF symbols** in our dbt seed files rather than dynamic variable scraping. This reduced pipeline complexity and failure points without compromising accuracy, as sector reclassifications are rare events.
2) dbt snapshots prevent historical overwrite—important during sector splits.
3) Cortex requires careful selection of training window—too long smooths out crucial patterns.
4) Airflow Sensors prevent downstream corruption ensuring safe execution order.
5) **Impact of Corporate Actions:** While our initial design utilized log-normal returns for statistical stability, a **stock split in the sectors** immediately prior to presentation required a runtime fallback to **simple returns**. This demonstrated the critical need for automated split-adjustment logic in financial pipelines, as standard log-transforms can amplify data anomalies during such events.

## *XI. Future Work*

1) Add **multi-factor ML models** (e.g., PCA, macroeconomic inputs).
2) Introduce **intraday ingestion** for higher-frequency analytics.
3) Expand dashboards with **regime classification ML**.
4) Implement **alerting mechanisms** using Airflow callbacks or Slack integrations.
5) Version-controlled **sector weights** using SCD Type-2 logic.

### *XII. Conclusion*

This project successfully demonstrates a modern, automated, warehouse-native ELT and ML system capable of quantifying real-time sector influence within the S&P 500. By combining Airflow, Snowflake, dbt, and Cortex, the system automates ingestion, feature engineering, sensitivity measurement, and forecasting. The dashboards provide actionable insights into market behavior, expose hidden dependencies, and clarify the true drivers of index volatility.

The approach meaningfully advances traditional market analysis by shifting focus from static sector weights to dynamic behavioral influence, offering analysts a clearer understanding of regime cycles, concentration risk, and forward-looking volatility patterns.

## GITHUB

**https://github.com/rahulmohannett/DAT226_GP_Sector_Sensitity_Monitor.git**

## REFERENCES

[1] Snowflake Inc., "Cortex ML Functions: Time-Series Forecasting," Snowflake Documentation, 2025. [Online]. Available: https://docs.snowflake.com/en/user-guide/snowflake-cortex/ml-functions/forecast.

[2] Yahoo Finance, "Yahoo Finance API Documentation," *PyPI*, 2024. [Online]. Available: https://pypi.org/project/yfinance/.

[3] Apache Software Foundation, "Apache Airflow Documentation," v2.10, 2024. [Online]. Available: https://airflow.apache.org/docs/.

[4] dbt Labs, "dbt Core Documentation," 2024. [Online]. Available: https://docs.getdbt.com/.

[5] Preset, "Preset Cloud & Apache Superset Visualization Guide," 2024. [Online]. Available: https://docs.preset.io/.