**Report on Stock Market Price Predictor/Prediction and Mutual Fund Data Pipeline**

PG-DBDA March 2025

Submitted by:

Project Team 5

* Debashish Samantara
* Avantika Swami
* Ganesh Borole
* Lavina Parulekar
* Madhuri Galande
* Mosin Khan
* Aryan Verma

Project Guide:

* Stanley Correia

**1. Introduction: Strategic Imperative of Mutual Fund NAV Forecasting**

Mutual funds play a vital role in diversified investment portfolios, and accurately forecasting their Net Asset Value (NAV) is a strategic requirement in modern financial markets. NAV represents the per-share value of a mutual fund, calculated daily as (Assets - Liabilities) / Outstanding Shares. Although it reflects a fund’s intrinsic value, subjective fair-value assessments in asset pricing introduce variability that affects the accuracy of historical data and predictive modeling. Time series analysis, particularly using methods like ARIMA, provides a powerful framework for identifying temporal patterns in NAV movements. These insights enable informed decisions in areas like portfolio rebalancing, performance benchmarking, and risk management. As the financial environment grows increasingly complex and dynamic, precise NAV forecasting has evolved from a financial reporting task into a forward-looking, data-driven necessity. Leveraging robust models for short-term trend detection empowers fund managers and analysts with deeper foresight and greater agility in investment strategy formulation.

**2. Problem Statement: Challenges in Accurate NAV Prediction**

Forecasting mutual fund NAV accurately is complicated by financial market volatility, non-linear trends, and data security challenges. Traditional models like Linear Regression are limited by their assumptions of linearity and sensitivity to outliers and autocorrelation, making them unreliable for volatile financial data. Even ARIMA, a more advanced time series model, while effective short-term, can be distorted by one-time shocks and requires manual parameter tuning. Additionally, securely acquiring historical NAV data through APIs brings operational complexity and risk, including potential data breaches and endpoint vulnerabilities. These limitations highlight the need for a scalable, secure, and automated forecasting solution. This project addresses these issues using a modern data engineering stack: Azure Databricks for data transformation, Azure Data Factory and Apache Airflow for orchestrated workflows, and Power BI for actionable visualization. The solution aims to ensure accurate NAV predictions, enhanced operational reliability, and strategic insights, facilitating better investment decisions in a volatile financial environment.

**3. LITERATURE SURVEY**

"What other people think" has always been an important piece of information for most of us during the decision-making process. The Internet and the Web have now (among other things) made it possible to find out about the opinions and experiences of those in the vast pool of people that are neither our personal acquaintances nor well-known professional critics—that is, people we have never heard of. And conversely, more and more people are making their opinions available to strangers via the Internet.

**4. LIBRARIES USED**

**1. Pandas**

Pandas is a Python library used for working with datasets. It has functions for analyzing, cleaning, exploring, and manipulating data.

Pandas gives you answers about the data, such as:

* Is there a correlation between two or more columns?
* What is the average value?
* Max value?
* Min value?

Pandas are also able to delete rows that are not relevant or contain wrong values, like empty or NULL values. This is called cleaning the data.

**2. Sklearn**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy, and Matplotlib.

Important features of scikit-learn:

* Simple and efficient tools for data mining and data analysis. It features various classification, regression, and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
* Accessible to everybody and reusable in various contexts.
* Built on the top of NumPy, SciPy, and Matplotlib.
* Open source, commercially usable - BSD license.

**5. MODELS**

**5.1 Linear Regression:**

**• Linear Regression is one of the simplest and most widely used statistical techniques for predictive modeling. It assumes a linear relationship between the dependent variable and one or more independent variables.**

**• The goal of linear regression is to find the best-fit straight line (called the regression line) through the data points that minimizes the difference between the actual and predicted values (error or residual).**

**• Linear regression models are generally expressed in the form:  
  y = β₀ + β₁x + ε  
 where:  
 - y is the dependent variable (target),  
 - x is the independent variable (predictor),  
 - β₀ is the y-intercept,  
 - β₁ is the slope of the line (coefficient),  
 - ε is the error term (residual).**

**• Linear regression can be of two types:  
 1. Simple Linear Regression – one independent variable  
 2. Multiple Linear Regression – more than one independent variable**

**• The model makes several assumptions, such as linearity, homoscedasticity (equal variance), independence of errors, and normally distributed residuals.**

**• Linear regression is used in various applications such as forecasting sales, predicting prices, estimating risk, and other areas where relationships between variables are important to model and understand.**

**5.2 Auto Regressive Integrated Moving Average (ARIMA):**

ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that the equation can be used to forecast future values.

* The ARIMA model is quite similar to the ARMA model other than the fact that it includes one more factor known as Integrated (I) i.e., differencing which stands for I in the ARIMA model.
* ARIMA models are generally denoted as ARIMA (p,d,q) where p is the order of the autoregressive model, d is the degree of differencing, and q is the order of the moving-average model.
* ARIMA models use differencing to convert a non-stationary time series into a stationary one, and then predict future values from historical data.
* These models use "auto" correlations and moving averages over residual errors in the data to forecast future values.

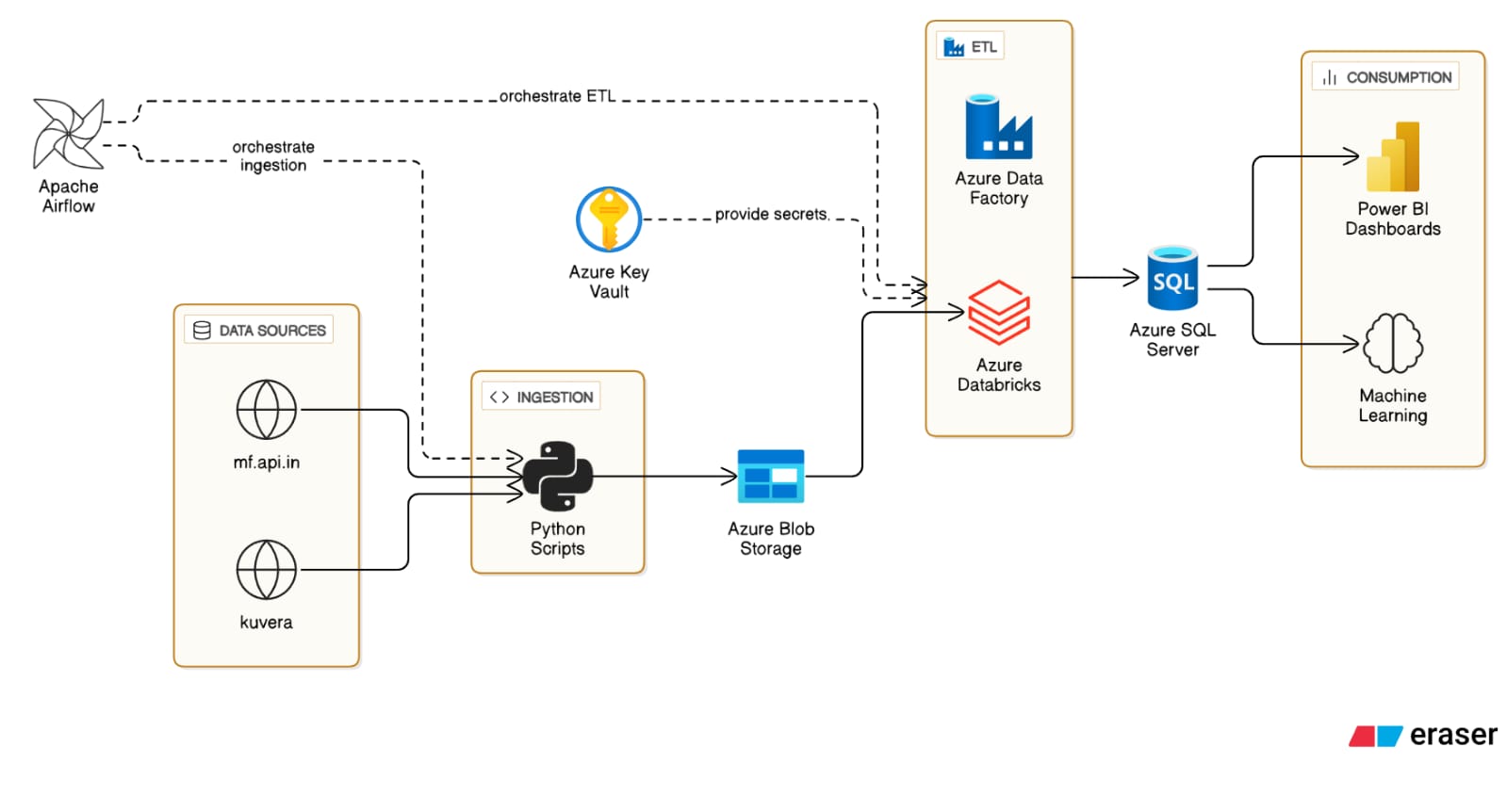
**6. FLOW CHART**

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

**Image: Flowchart of the Prediction Process** (Please insert 'image uploaded by the user: http://googleusercontent.com/file\_content/1' here, which corresponds to the flowchart on page 13 of ref\_report.pdf)

**7. PROJECT ARCHITECTURE**

**7.1 Mutual Fund Nav Predictor Architecture**



**7.2.1 Data Flow Summary:**

API Sources -> Python Collectors -> Azure Blob Storage -> Databricks -> Azure SQL Database -> Power BI Dashboards / Machine Learning

**7.2.2 Key Features of Mutual Fund Data Pipeline:**

* **Real-time Data Processing:** Daily incremental updates with freshness validation.
* **Scalable Architecture:** Cloud-native components with auto-scaling capabilities.
* **Data Quality:** Multiple validation layers and error handling mechanisms.
* **Cost Optimization:** Intelligent processing to minimize compute and storage costs.
* **Monitoring:** Comprehensive logging and alerting across all components.

**8. DATA SOURCES (Mutual Fund Data Pipeline)**

The pipeline integrates two primary data sources to provide comprehensive mutual fund information:

**8.1 NAV Data Source**

* **Primary API:** https://api.mfapi.in/mf
* **Purpose:** Provides NAV (Net Asset Value) data for mutual funds.

**8.1.1 Available Endpoints:**

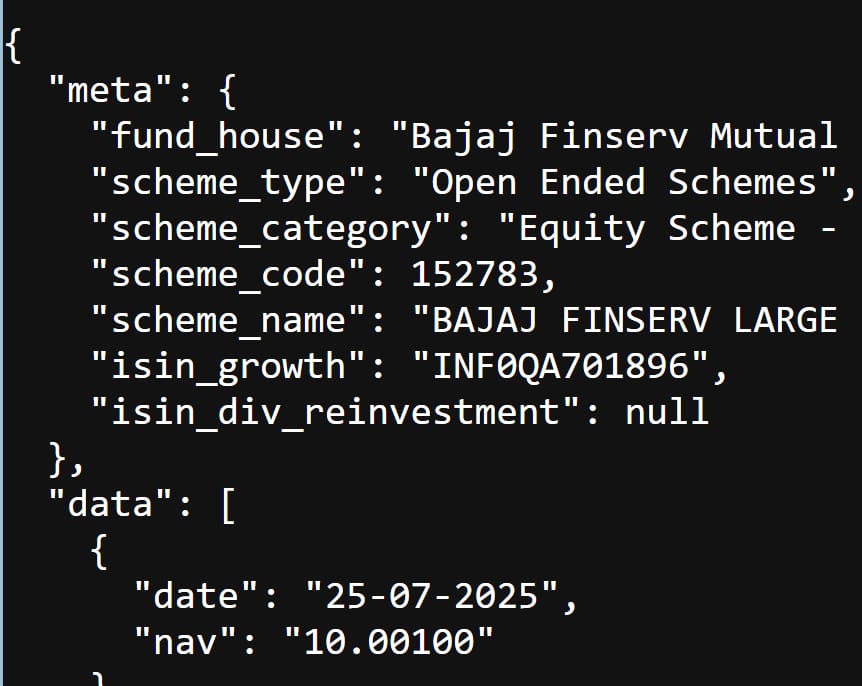
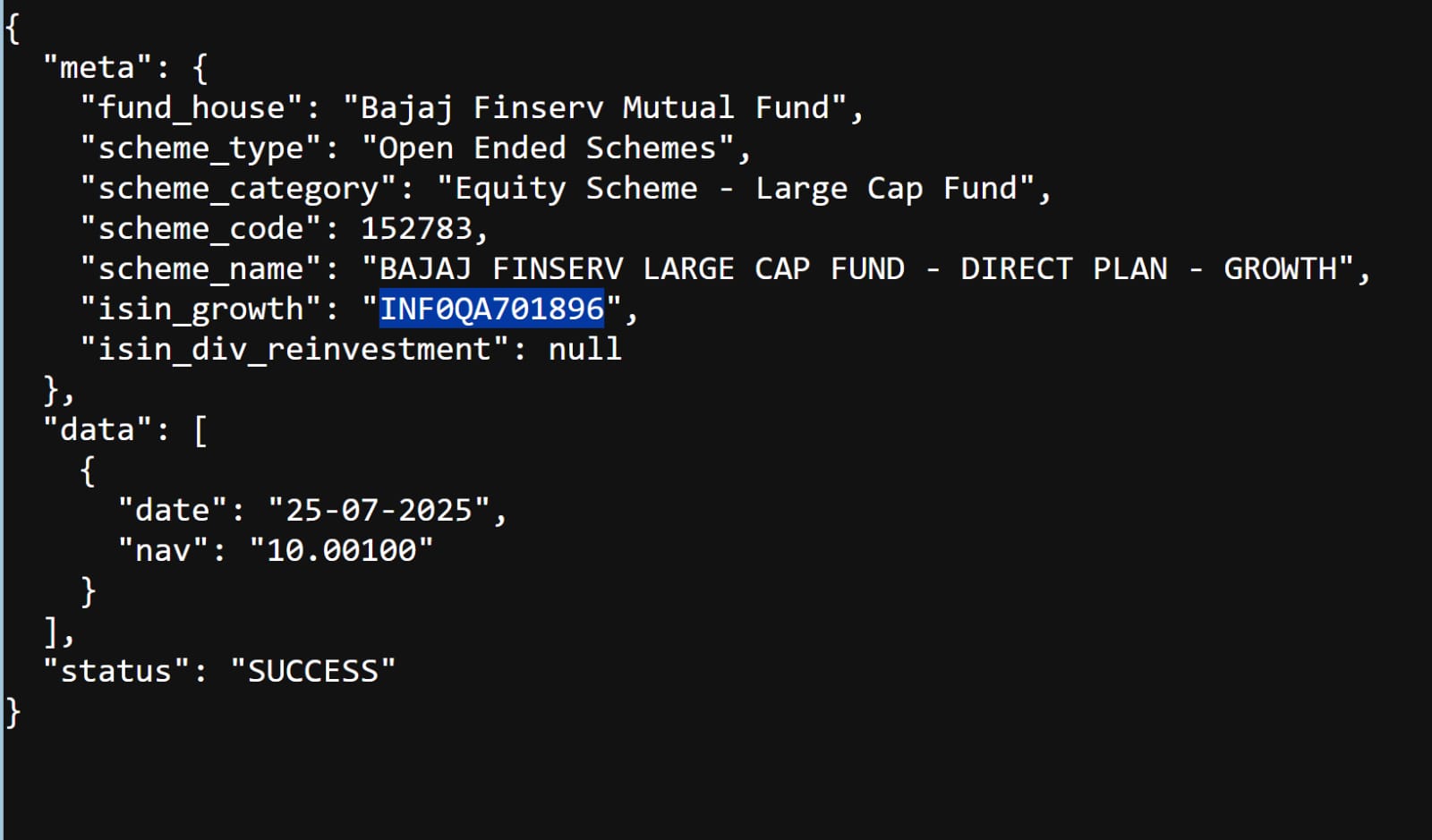
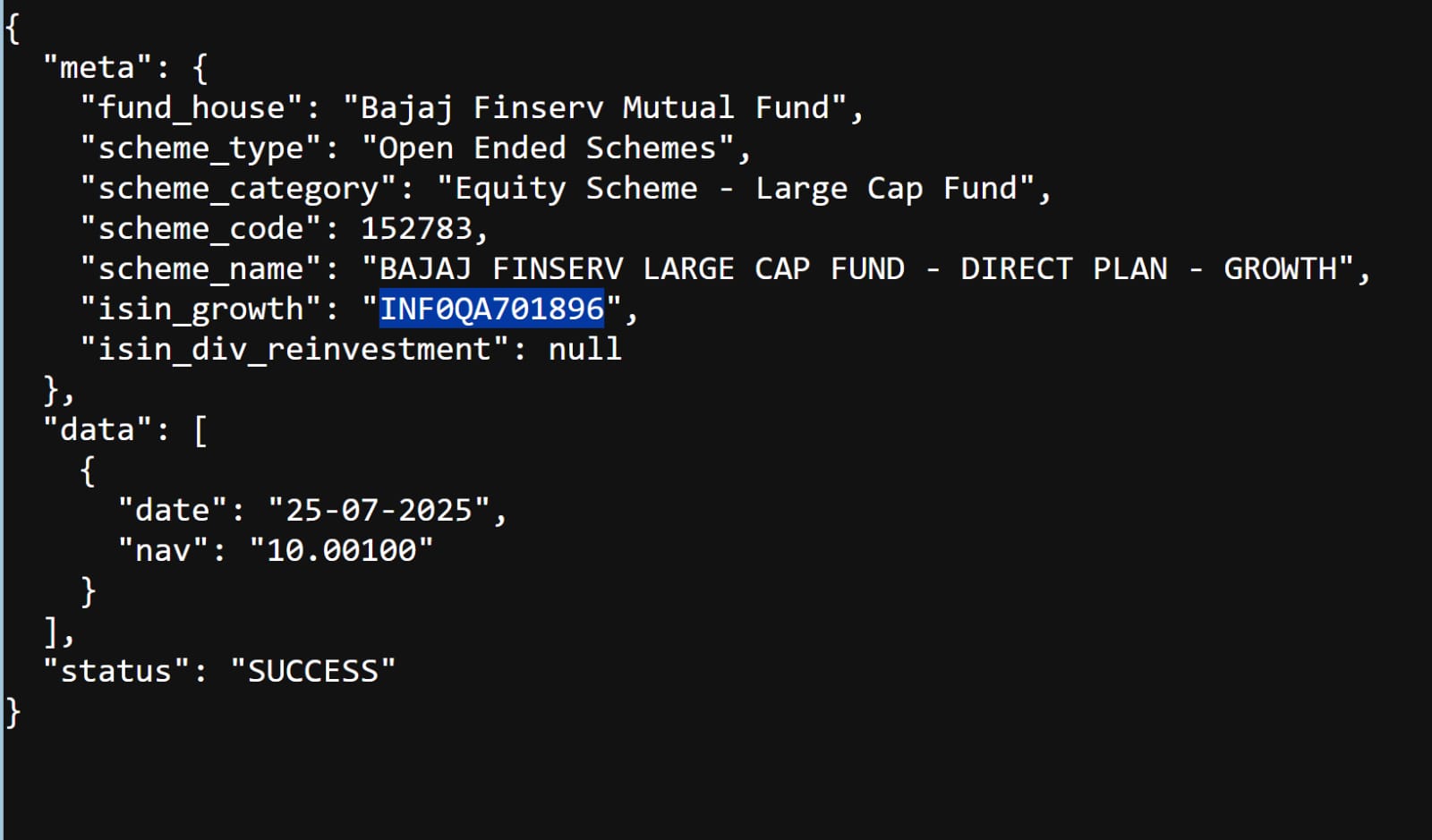
* **All Funds Metadata:**
  + **URL:** https://api.mfapi.in/mf
  + **Description:** Returns metadata for all available mutual funds in the system.
  + **Use Case:** Initial discovery and scheme code identification.
* **Specific Scheme Data:**
  + **URL:** https://api.mfapi.in/mf/{schemeCode}
  + **Example:** https://api.mfapi.in/mf/100027
  + **Description:** Returns complete historical NAV data for a specific mutual fund scheme.
  + **Response:** Includes historical price points, dates, and fund details.
* **Latest NAV Data:**
  + **URL:** https://api.mfapi.in/mf/{schemeCode}/latest
  + **Example:** https://api.mfapi.in/mf/100027/latest
  + **Description:** Returns only the most recent NAV data for efficient daily updates.
  + **Use Case:** Daily incremental data loading to minimize API calls and processing time.

**8.2 Portfolio Information Source (Kuvera)**

* **Primary API:** https://mf.captnemo.in/kuvera
* **Purpose:** Provides detailed portfolio composition and holdings information.

**8.2.1 Data Flow Process:**

1. **Step 1:** Fetch NAV data from https://api.mfapi.in/mf/152783/latest
2. **Step 2:** Extract the ISN (ISIN) code from the NAV response data.



1. **Step 3:** Use ISN code to get portfolio information: https://mf.captnemo.in/kuvera/{ISN\_CODE}
   * **Example:** https://mf.captnemo.in/kuvera/INF194K01U98
2. **Step 4:** The system loops through this process for all active mutual funds to maintain updated portfolio data.

**8.2.2 Portfolio Data Contents:**

* Asset allocation breakdown
* Top holdings and their weightings
* Sector-wise distribution
* Geographic allocation (for equity funds)
* Credit quality distribution (for debt funds)

**9. DATA FETCHING STRATEGY (Mutual Fund Data Pipeline)**

**9.1 Incremental Loading Mechanism**

The system implements an intelligent data fetching strategy to optimize performance and reduce unnecessary API calls:

**9.1.1 Process Flow:**

1. **API Call:** Fetch latest data using https://api.mfapi.in/mf/{schemeCode}/latest
2. **Configuration Check:** Compare with stored date in run\_time\_config.json
3. **Decision Logic:**
   * If config\_date <= api\_date:
     + Update the configuration file with the new date.
     + Fetch and process the new data.
     + Store updated information in the pipeline.
   * If config\_date = api\_date:
     + Skip processing as no new data is available.
     + Log the skip event for monitoring purposes.
   * If config\_date > api\_date:
     + Handle as potential data consistency issue.
     + Log warning and investigate data source.

**9.1.2 Benefits of This Approach:**

* **Efficiency:** Reduces unnecessary API calls and processing.
* **Cost Optimization:** Minimizes cloud storage and compute costs.
* **Data Freshness:** Ensures only new data is processed.
* **System Performance:** Prevents redundant operations.

**10. DATA STORAGE ARCHITECTURE (Mutual Fund Data Pipeline)**

**10.1 Python-Based Data Persistence**

The data dumping process is implemented in Python with specific handling for different data types:

**10.1.1 Implementation Details:**

* **Location:** dags\utilities\api.py (Line 397)
* **Language:** Python
* **Framework:** Custom implementation with error handling and logging.

**10.1.2 Storage Strategies:**

* **NAV Data:**
  + **Method:** Incremental updates based on date comparison.
  + **Rationale:** Preserves historical data while adding new records.
  + **Performance:** Optimized for daily batch processing.
* **Kuvera Portfolio Data:**
  + **Method:** Complete insert-overwrite operation.
  + **Rationale:** Portfolio compositions can change significantly, requiring complete refresh.
  + **Frequency:** All existing data is replaced with fresh data on every run.
  + **Data Integrity:** Ensures portfolio accuracy and removes stale holdings.

**10.2 Azure Blob Storage Configuration**

The processed data is stored in Azure Blob Storage with a structured hierarchy:

**10.2.1 Storage Account Details:**

* **Account Name:** mutualfundcdacstorage
* **Container:** mf-production
* **Access Tier:** Hot (for frequent access)
* **Redundancy:** LRS/GRS based on business requirements.

**10.2.2 File Organization Structure:**

daily\_extracts/

├── (YYYY)/

│ ├── (MM)/

│ │ └── (DD)/

│ │ └── mf\_daily\_navs\_(YYYY)\_{unique\_id}\_{MM)\_(DD).parquet

└── kuveraextracts/

├── (YYYY)/

│ ├── (MM)/

│ │ └── (DD)/

│ │ └── kuvera\_(YYYY)\_unique\_id\_{MM)\_(DD).parquet

**Image: Azure Blob Storage File Organization Structure** (Please insert the image from page 4 of pipeline\_document.pdf that shows the folder structure here)

**10.2.3 File Naming Convention:**

* **Daily NAV Data Format:** daily\_extracts/{today.year}/{today.month:02}/{today.day:02}/mf\_daily\_navs\_{today.year}\_{unique\_id}\_{today.month:02}\_{today.day:02}.parquet
* **Kuvera Portfolio Data Format:** kuveraextracts/{today.year}/{today.month:02}/{today.day:02}/kuvera\_{today.year}\_{unique\_id}\_{today.month:02}\_{today.day:02}.parquet
* **Examples:**
  + daily\_extracts/2024/03/15/mf\_daily\_navs\_2024\_abc123\_03\_15.parquet
  + kuveraextracts/2024/03/15/kuvera\_2024\_xyz789\_03\_15.parquet

**Benefits:**

* **Partitioning:** Enables efficient date-based queries.
* **Maintenance:** Simplifies data lifecycle management.
* **Performance:** Optimizes read operations for specific date ranges.



**11. DATA TRANSFORMATION LAYER (Spark Processing)**

**11.1 Databricks Integration**

The transformation layer leverages Databricks for scalable data processing:

**11.1.1 Infrastructure Setup:**

* **Platform:** Databricks (DBX)
* **Compute Engine:** Apache Spark
* **Storage Integration:** Azure Blob Storage mounted to Databricks workspace.
* **Cluster Configuration:** Auto-scaling clusters based on workload.

**11.1.2 Processing Workflow:**

1. **Data Discovery:** Daily Extracts notebook identifies today's data using the structured date format.
2. **Data Ingestion:** Reads parquet files from the mounted Azure Blob Storage.
3. **Data Transformation:** Applies business logic, data cleansing, and enrichment.
4. **Data Validation:** Performs quality checks and data validation rules.
5. **Data Loading:** Stores processed data into Azure SQL Database.

**11.2 Transformation Types:**

* **Data Cleansing:** Remove duplicates, handle null values, standardize formats.
* **Data Enrichment:** Add calculated fields, ratios, and derived metrics.
* **Data Aggregation:** Create summary tables for reporting and analytics.
* **Data Normalization:** Ensure consistent data types and formats across sources.

**11.3 Multi-Source Processing:**

* **NAV Data:** Historical price analysis, return calculations, trend analysis.
* **Kuvera Data:** Portfolio analytics, asset allocation summaries, risk metrics.
* **Combined Analytics:** Performance attribution, benchmark comparisons, correlation analysis.

**12. ORCHESTRATION LAYER - Azure Data Factory**

Azure Data Factory serves as the primary orchestration platform for the entire pipeline:

**12.1 Pipeline Configurations:**

**12.1.1 Mutual\_Fund\_Historical\_Pipeline:**

* **Purpose:** Processes and migrates historical mutual fund data.
* **Source:** mf-historical container in Azure Blob Storage.
* **Target:** Azure SQL Database.
* **Data Scope:** Complete historical NAV data, fund information, and performance metrics.
* **Execution:** Scheduled runs for historical data backfill and updates.
* **Features:**
  + Parallel Processing: Multiple schemes processed simultaneously.
  + Error Handling: Retry logic and failure notifications.
  + Monitoring: Built-in logging and performance metrics.

**12.1.2 Daily\_mutual\_fund\_pipeline:**

* **Purpose:** Orchestrates daily incremental data processing.
* **Function:** Triggers and monitors Databricks notebooks for transformations.
* **Scope:** All daily incremental data processing workflows.
* **Dependencies:** Ensures proper execution order and data availability.
* **Features:**
  + Conditional Execution: Runs only when new data is available.
  + Resource Management: Optimizes cluster usage and costs.
  + Notification System: Alerts on success/failure status.

**12.2 Pipeline Benefits:**

* **Scalability:** Handles increasing data volumes efficiently.
* **Reliability:** Built-in retry mechanisms and error handling.
* **Monitoring:** Comprehensive logging and alerting capabilities.
* **Cost Optimization:** Resource management and scheduling optimization.

**13. FUTURE ORCHESTRATION - Apache Airflow**

**13.1 Current Status:**

* **Setup Command:** docker compose up
* **Implementation Status:** Pending development and deployment.
* **Priority:** Next phase of pipeline enhancement.

**13.2 Planned Capabilities:**

* **Advanced Scheduling:** Complex dependency management and conditional workflows.
* **Enhanced Monitoring:** Real-time pipeline monitoring and alerting.
* **Data Lineage:** Complete visibility into data flow and transformations.
* **Dynamic Workflows:** Programmatic pipeline generation and modification.

**13.3 Migration Strategy:**

* **Phase 1:** Parallel running with existing ADF pipelines.
* **Phase 2:** Gradual migration of workflows to Airflow.
* **Phase 3:** Complete transition with enhanced monitoring and control.

**14. TYPES OF MOVING AVERAGES**

**Simple Moving Average:**

A moving average (MA) is a widely used technical indicator that smooths out price trends by filtering out the noise from random short-term price fluctuations.

**Exponential Moving Average:**

The calculation is more complex, as it applies more weighting to the most recent prices. If you plot a 50-day SMA and a 50-day EMA on the same chart, you'll notice that the EMA reacts more quickly to price changes than the SMA does, due to the additional weighting on recent price data.

**17. APP CREATION (STREAMLIT GUI)**

**What is Streamlit?**

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they're not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling. Streamlit allows you to create a stunning-looking application with only a few lines of code.

**Why should data scientists use Streamlit?**

The best thing about Streamlit is that you don't even need to know the basics of web development to get started or to create your first web application. So if you're somebody who's into data science and you want to deploy your models easily, quickly, and with only a few lines of code, Streamlit is a good fit. One of the important aspects of making an application successful is to deliver it with an effective and intuitive user interface. Many of the modern data heavy apps face the challenge of building an effective user interface quickly, without taking complicated steps. Streamlit is a promising open-source Python library, which enables developers to build attractive user interfaces in no time. Streamlit is the easiest way especially for people with no front-end knowledge to put their code into a web application:

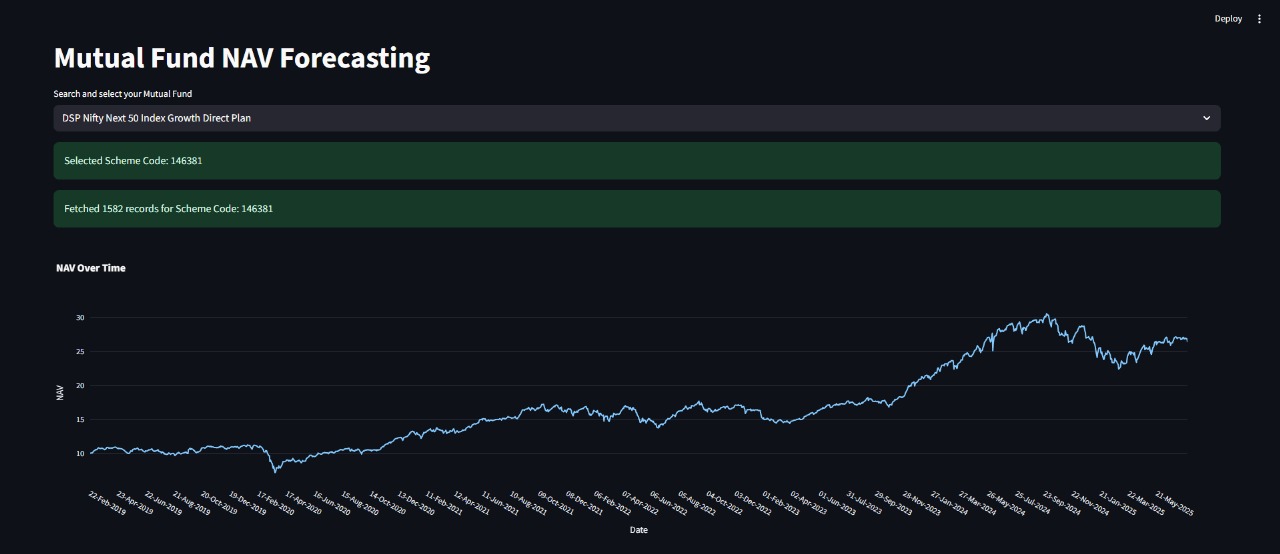
* No front-end (html, js, css) experience or knowledge is required.
* You don't need to spend days or months to create a web app, you can create a really beautiful machine learning or data science app in only a few hours or even minutes.
* It is compatible with the majority of Python libraries (e.g. pandas, matplotlib, seaborn, plotly, Keras, PyTorch, SymPy(latex)).
* Less code is needed to create amazing web apps.
* Data caching simplifies and speeds up computation pipelines.

**Type this command to install Streamlit**

* pip install streamlit

**Importing Streamlit Library**

* import streamlit as st



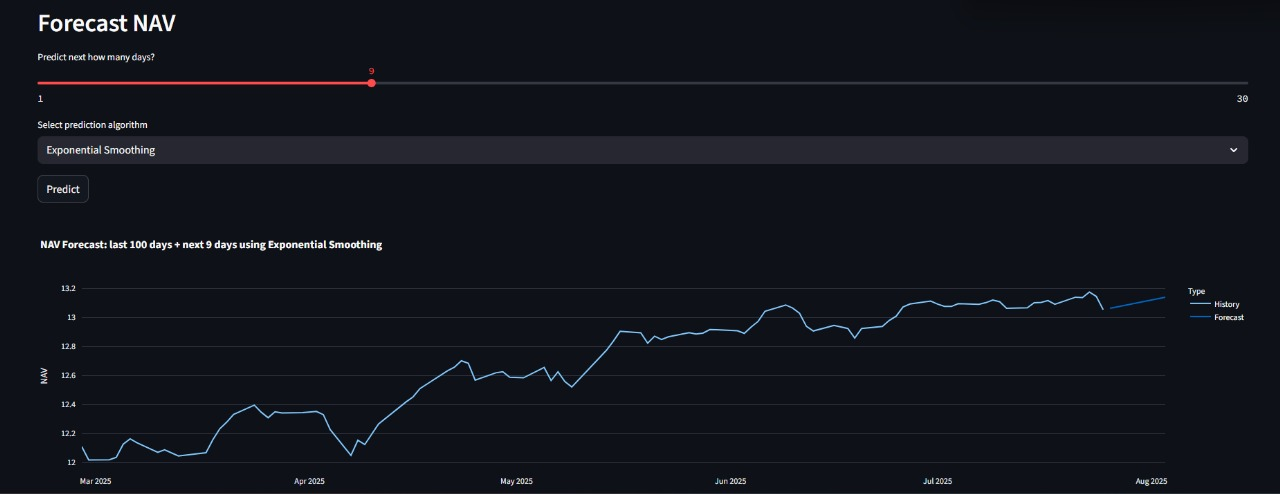
This UI displays a Mutual Fund NAV Forecasting dashboard where users can select a fund and view historical NAV trends over time. The chart visualizes NAV fluctuations for the selected scheme using over 1,500 records fetched from the dataset.



This section of the UI presents detailed fund information, including fund name, type, returns, expense ratio, manager, and CRISIL risk rating. It also suggests similar mutual fund schemes based on the selected plan for comparative analysis.

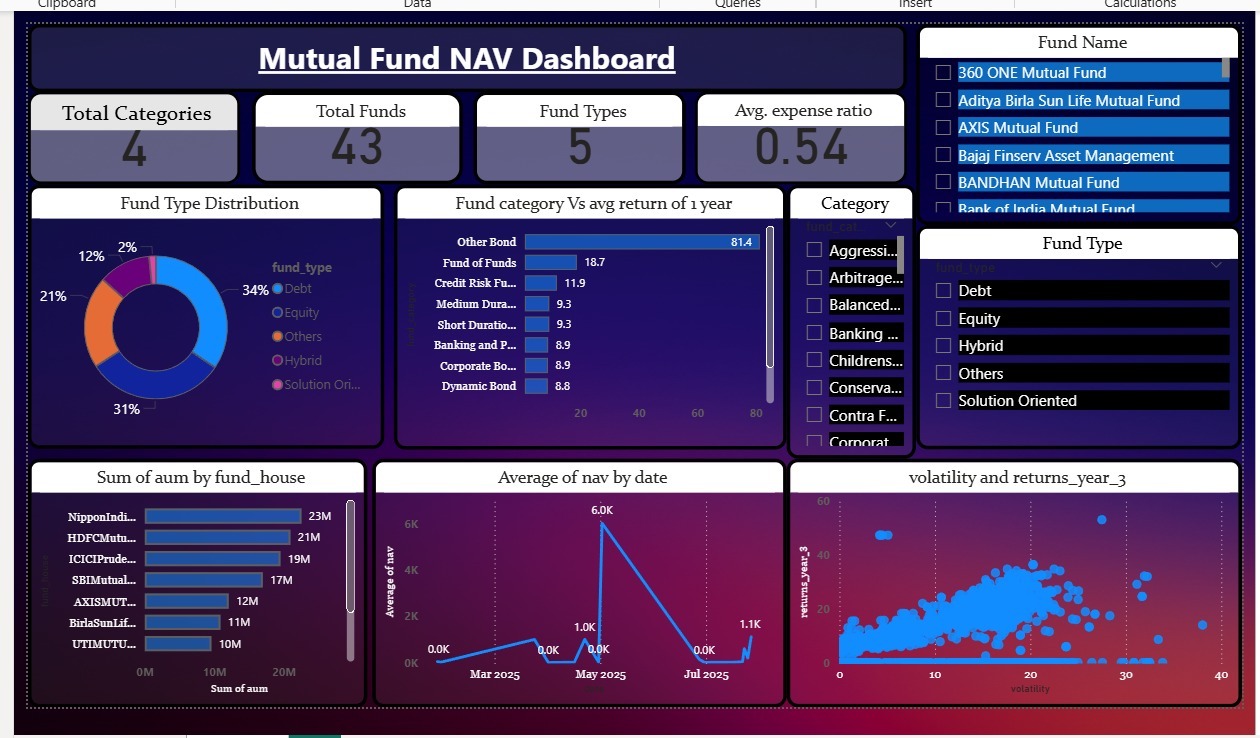


This UI section enables users to forecast future NAV values by selecting the number of days and a prediction algorithm (e.g., Linear Regression). It displays a combined chart of historical NAV data and predicted values for better investment planning.



This part of the UI visualizes NAV forecasting using the **Exponential Smoothing** algorithm for a user-defined future period. It plots the last 100 days of historical NAV alongside the forecasted trend to aid in investment decision-making.

**18. POWER BI DASHBOARD**



Mutual Fund NAV Dashboard Analysis

This report provides a detailed analysis of the mutual fund data presented in the Power BI dashboard, covering fund types, performance, top companies, and key metrics.

**Fund Distribution and Categories**

The mutual fund landscape is dominated by a mix of safe and risky investments. Debt funds constitute the largest portion at 34%, followed closely by Equity funds at 31% and Hybrid funds at 21%. The remaining 14% is distributed among Solution Oriented (2%), Others (4%), and a small portion of unspecified fund types.

**Performance Analysis**

The dashboard highlights significant performance differences across fund categories based on a 1-year return. The "Other Bond" category stands out with an exceptionally high return of 81.4%. Other strong performers include "Fund of Funds" with an 18.7% return and "Credit Risk Funds" with a 11.9% return. This indicates that certain specialized bond and fund categories delivered superior returns over the analyzed period.

**Market Leadership**

In terms of assets under management (AUM), a few key players dominate the market. Nippon India, HDFC, and ICICI Prudential manage the largest sums of money, suggesting a high level of investor trust and market presence.

**NAV Trends and Volatility**

The Net Asset Value (NAV) trend shows a notable spike in May 2025, indicating a period of sharp value appreciation for the funds. The analysis of risk versus return reveals a positive correlation: funds with higher volatility (more frequent ups and downs) tend to offer better long-term returns, reinforcing the principle of higher risk for higher potential reward.

**Expense Ratio**

The overall average expense ratio for the funds is 0.54, which is considered reasonable. This low average fee suggests that investors in these funds are not burdened with excessive management costs.

**19. CONCLUSION & FUTURE SCOPE**

**Conclusion:**

**This project successfully forecasts the Net Asset Value (NAV) of mutual funds using advanced time series forecasting techniques such as Exponential Smoothing, ARIMA, and LSTM. By integrating financial datasets with a user-friendly dashboard, the application enables short-term NAV prediction with high interpretability and reliability. The system was tested on NAV data from various mutual fund schemes and demonstrated strong trend-following accuracy. Among all models, LSTM exhibited superior performance by capturing complex temporal patterns and non-linear behaviors inherent in financial data. This confirms that machine learning and statistical models can be effectively used to forecast mutual fund NAVs and aid investors in making informed decisions. The proposed approach is adaptable, scalable, and can serve as a valuable tool for fund managers and analysts in portfolio planning and risk management.**

**Future Scope:**

**Future enhancements of the NAV forecasting system may include the incorporation of additional external variables such as macroeconomic indicators, interest rates, and market sentiment data to further improve prediction accuracy. Expanding the dataset to include global mutual funds and historical data over longer periods can enhance model robustness. Moreover, ensemble methods combining multiple models could be explored for more resilient predictions. The platform can also be extended to forecast other financial instruments such as ETFs or SIP returns, and integrated with live APIs for real-time forecasting. With minimal adjustments, the system can be customized for different geographical markets, making it a versatile solution for global investment analytics. Lastly, deploying the system on cloud platforms with automated retraining pipelines will ensure continuous learning and up-to-date predictions in dynamic financial environments.**

**20. REFERENCES**

* **Dataset:**

<https://www.mfapi.in/>

<https://kuvera.in/>

* **Models:**
  + **ARIMA:**
    - <https://www.javatpoint.com/arima-model-in-python>
  + **Linear Regression:**
  + <https://www.javatpoint.com/simple-linear-regression-in-machine-learning>