Automated Stock Price Forecasting System using Snowflake, Airflow, and yfinance API

Alisha Kartik
Department of Applied Data Science
San Jose State University
Student ID: 017518813

Rajarajeswari Premanand
Department of Applied Data Science
San Jose State University
Student ID: 017776993

***Abstract*—This project introduces a predictive analytics system designed to forecast the stock prices of Wells Fargo & Co. (WFC), Bank of America Corp. (BAC), and JPMorgan Chase & Co. (JPM) over the next seven days. Historical stock data is systematically gathered through the yfinance API, managed within a Snowflake database, and maintained by an automated pipeline using Apache Airflow. Integrated machine learning models are employed to provide accurate and continuous predictions.**

I. Introduction

The goal of this project is to create a sophisticated data warehouse solution that leverages the yfinance API to collect, process, and store stock market data for JPMorgan Chase & Co. (JPM), Bank of America Corp. (BAC), and Wells Fargo & Co. (WFC). A robust Extract, Transform, Load (ETL) pipeline is connected to a predictive analytics system to forecast stock prices for the upcoming seven days.

Snowflake, an advanced cloud-based database platform that offers safe and effective storage capabilities, is at the heart of this solution. Through the organization of comprehensive stock-related metrics, including daily open, high, low, close, and trade volume, into a structured and easily accessible manner, Snowflake guarantees data integrity and expedites retrieval.

Apache Airflow is used to automate data workflows by methodically managing data loading, transformation, and extraction processes. The scheduling feature of Airflow ensures the consistency and dependability of the data by drastically reducing manual labor and possible human error. The combination of Airflow's automation capabilities with Snowflake's robust storage and retrieval features guarantees smooth, real-time access to vital data, enabling investors to make strategic decisions and supplying a solid basis for precise stock price forecasts.

A. Requirements

To develop an Automated Stock Price Forecasting System, the following technical and infrastructural requirements must be met:

1) Data Sources and APIs:

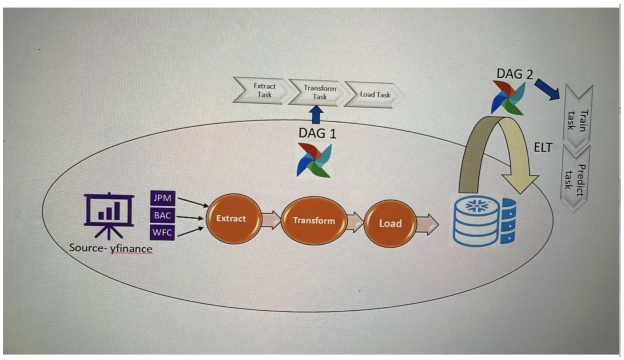

Fig. 1. System Diagram: Data Pipeline for Stock Price Forecasting

- yfinance API: Retrieves historical stock price data for JPM, BAC, and WFC.
- Snowflake Database: Stores and processes stock price data efficiently.

2) Infrastructure and Tools:

- Apache Airflow: Manages ETL pipeline automation and DAG execution.
- Docker: Containerizes Airflow for efficient workflow execution.

3) Machine Learning Framework:

- Snowflake ML Functions: Used for forecasting stock prices.
- SNOWFLAKE.ML.FORECAST: Assists in data processing and visualization.

4) Storage and Processing:

- Snowflake Schemas:
  - Raw_Data: Stores historical stock price data.

5) Security and Access Control:

- Airflow Connections & Variables: Securely handle API keys and Snowflake credentials.
- Role-Based Access in Snowflake: Restricts data modification and ensures security.

II. Data Description

Three stocks—JPMorgan Chase & Co. (JPM), Bank of America Corp. (BAC), and Wells Fargo & Co. (WFC)—had their historical stock price data collected using the yfinance API. The data was organized and stored in a Snowflake table, with the database schema explicitly defined as follows:

A. Variable Descriptions

- **Symbol** (STRING)

  Description: The STRING data type is used for stock symbols (e.g., JPM, BAC, WFC). It supports varying alphanumeric lengths, making querying and filtering efficient. This field is part of the PRIMARY KEY when combined with Date.
- **Date** (TIMESTAMP)

  Description: The TIMESTAMP data type records each stock price entry's precise date and time. It is marked as a component of the primary key to ensure data consistency and prevent duplication.
- **Open** (NUMBER)

  Description: The NUMBER data type is used to represent the stock's opening price. This allows for decimal values, ensuring accurate representation of stock prices down to cents.
- **High** (NUMBER)

  Description: The NUMBER data type is used to store the highest price of the stock during a trading session. This ensures precise tracking of the maximum price fluctuations in a day.
- **Low** (NUMBER)

  Description: The NUMBER data type is also used for the lowest price, allowing for the representation of decimal values. This is important for accurately capturing the lowest price point during the trading period, which can be crucial for financial analysis.
- **Close** (NUMBER)

  Description: The NUMBER data type represents the closing stock price at the end of the trading session. This value is critical for historical price analysis and forecasting.
- **Volume** (NUMBER)

  Description: The INTEGER data type is used to store the total number of shares traded during the session. Since trade volumes are whole numbers, using an integer ensures precise representation.

III. Functional Analysis

The following functional components make up the Stock Price Prediction system:

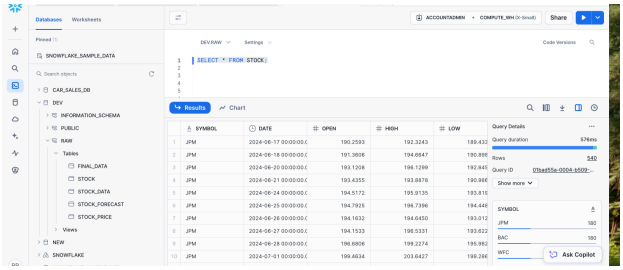

Fig. 2. Input Data

A. Data Collection

- The yfinance API gathers historical daily stock price data for JPM, BAC, and WFC.
- Extracted variables include volume, date, close, high, low, and open, stored in structured formats.
- An automated Airflow job retrieves and retains only the latest 180 days of stock price data.

B. Airflow Connections and Variables

- Secure Airflow variables manage sensitive credentials for yfinance API keys and Snowflake login.
- Airflow UI configures connections to the Snowflake database.

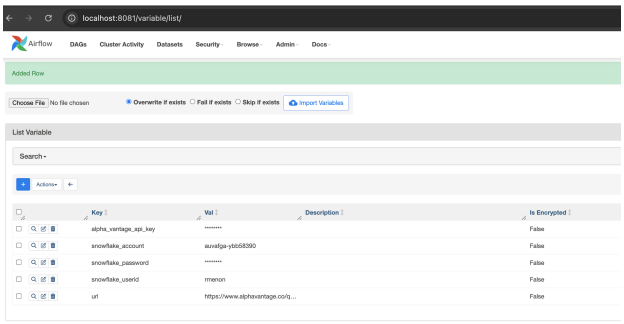

Fig. 3. Airflow Variables

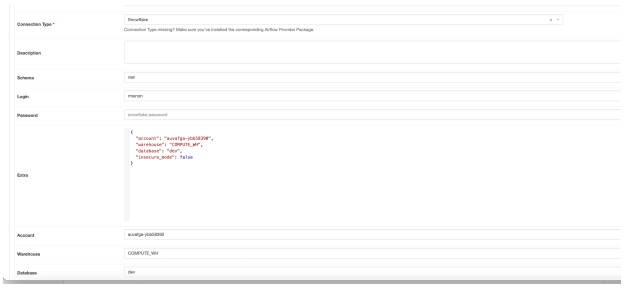

Fig. 4. Airflow Connections

C. Data Storage

- The Snowflake database contains two schemas:
  - Raw_Data: Stores raw stock data.

  - Analytics: Maintains machine learning models and processed data.
- An analytics schema for keeping track of machine learning models, processed data, and prediction results.
- The stock_prices table in the Raw_Data schema loads and maintains stock price data that is restricted to the last 180 days.
- To ensure data accuracy and consistency, an Apache Airflow DAG (Directed Acyclic Graph) is set up to automatically refresh the Snowflake database with the stock prices for the previous 180 days every day.

D. Data Processing

- Airflow tasks automatically train machine learning models inside Snowflake.
- The training dataset includes date, stock symbol, and closing price.
- The `SNOWFLAKE.ML.FORECAST` function is used for model training.
- Using the most recent dataset, this training process is automatically executed daily to make sure the machine learning model takes into account the most recent developments.

E. Machine Learning Prediction

- Functionality: Produce precise closing stock price projections for the upcoming seven days.
- Using the trained machine learning model, an Airflow job is created to predict the daily closing prices of JPM, BAC, and WFC stocks over a rolling seven-day period.
- Predictions are methodically kept in the stock_forecast database of Snowflake's Analytics architecture.
- In addition, the Analytics schema's final_stock_data database contains a consolidated historical record that combines current stock prices with anticipated future prices. Based on the most recent data, these forecasting jobs are executed every day and regularly produce updated projections.

F. Airflow DAGs

The Airflow DAG automates the following tasks daily:

1) Data Extraction: Use the yfinance API to retrieve historical daily stock prices for JPM, BAC, and WFC.
2) Data Loading: Fill the Snowflake stock_prices table with the most recent 180 days' worth of stock price data.
3) Machine Learning Model Training: Utilizing the most recent historical stock data, update and retrain the predictive model.
4) Stock Price Forecasting: Using the most recent machine learning model, forecast stock prices for the upcoming seven days.
5) This automatic DAG guarantees daily updates, preserving the accuracy and freshness of the data while giving investors timely predicted insights for strategic decision-making.

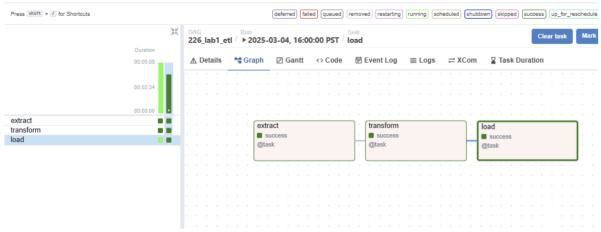

Fig. 5. ETL DAG Execution

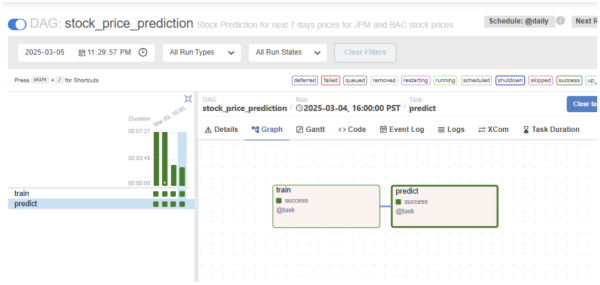

Fig. 6. Prediction DAG Execution

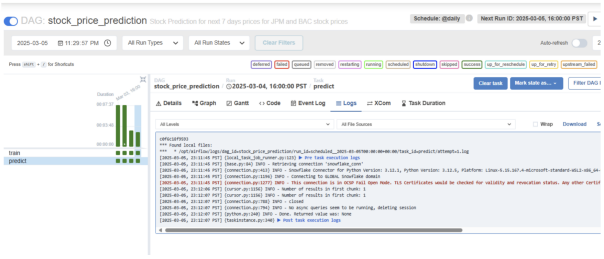

Fig. 7. Airflow DAG Logs

IV. Code and Results

The full code repository is available at: https://github.com/al-rv/226_pair9.

A. Observations and Conclusions

- **Accuracy for Recent Historical Data:**
  Since `FINAL_DATA` shows both `ACTUAL` and `FORECAST` for past dates, you can compare how close the predicted closing prices are to the real ones. When `ACTUAL` values lie within the `LOWER_BOUND` and `UPPER_BOUND`, it indicates that the model's confidence intervals are capturing the true values.
- **Short-Term Focus:**
  The model is currently set to predict a 7-day horizon. While this is helpful for near-term trading strategies, it may not capture longer-term trends or macroeconomic factors.

- **Rolling Predictions:**
  Because the data is refreshed daily and the pipeline reruns, each day's new 7-day forecast reflects the latest historical data. This allows you to monitor whether the forecasts improve over time.
- **Confidence Intervals:**
  The `LOWER_BOUND` and `UPPER_BOUND` columns provide insight into the uncertainty of each forecast. Narrower bounds suggest higher model confidence, whereas wider bounds indicate expected volatility.
- **Data Limitations:**
  Only the most recent 180 days of data are retained. This limits the ability to detect extended historical patterns. Additionally, external events (e.g., major financial news, policy changes) are not factored in, potentially reducing forecast accuracy in volatile market conditions.
- **Practical Next Steps:**
  - *Compare Actual vs. Forecast*: Systematically evaluate how often the actual closing price falls within the confidence interval for recently passed dates.
  - *Track Error Metrics*: Automate the calculation of metrics such as MAPE, RMSE, or MAE over the last N days to see if forecasts are improving.
  - *Retain More Historical Data*: Consider storing data older than 180 days to strengthen model training and potentially capture longer-term trends.
  - *Enhance With External Data*: Integrate news sentiment or economic indicators for improved robustness and accuracy, especially if used for real trading decisions.

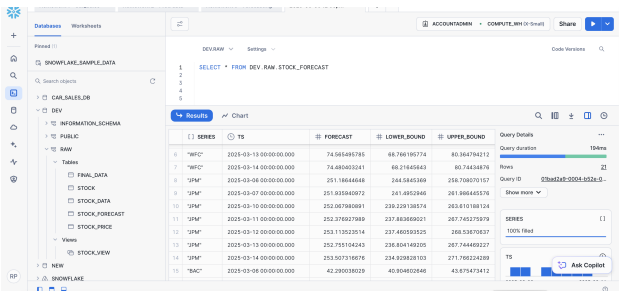

Fig. 8. Forecast Data

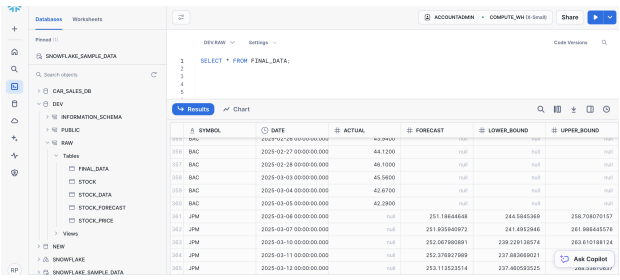

Fig. 9. Final Data - Actual vs Predicted

V. Limitations of This System

The current stock price forecasting system has several limitations that impact its effectiveness and scalability. The following constraints have been identified:

A. Full Refresh Only

At the moment, the system only stores the most recent 180 days of historical data and conducts a full refresh of the data every day. This indicates that every operation methodically deletes data that is more than 180 days old. As a result, historical information beyond this period is not available. The current codebase would need to be improved to support incremental refreshes, keeping older records with new ones, in order to store and analyze historical data over extended time periods.

B. Google Cloud Platform (GCP) Cost Constraints

The current deployment of Apache Airflow in the Google Cloud Platform (GCP) environment involves considerable costs due to its resource-intensive nature. GCP allocates multiple servers by default for executing Airflow DAGs daily, making the operational expenses impractically high for regular, long-term use. Transitioning the workflow execution environment to Docker containers or a similar cost-effective solution would significantly reduce operational costs while maintaining performance.

C. Short-Term Focus

The predictive system is specifically optimized for short-term forecasts, providing insights limited to a seven-day future window. This short-term perspective may fail to identify or reflect critical long-term market trends, fundamental changes within companies, or broader economic shifts. For a comprehensive investment strategy, incorporating models capable of analyzing and predicting long-term trends is advisable.

D. Lack of External Data Considerations

The current forecast model is based only on past pricing data, which means it cannot take into consideration outside variables like geopolitical events, abrupt financial news, policy changes, or economic indicators. These outside factors have a significant effect on stock prices and may compromise the accuracy of forecasts. Forecasts would be more accurate and robust under volatile market situations if sentiment analysis or external data sources were integrated.

References

[1] K. Lee, *SJSU Data-226 SP25 - train_predict.py*, GitHub. Available: https://github.com/keeyong/sjsu-data226-SP25/blob/main/week6/train_predict.py.

[2] K. Lee, *SJSU Data-226 SP25 - Demo_ETL_Dag_Snowflakes.py*, GitHub. Available: https://github.com/keeyong/sjsu-data226-SP25/blob/main/week6/Demo_ETL_Dag_Snowflakes.py.

[3] Apache Airflow Documentation. Available: https://airflow.apache.org/docs/apache-airflow/stable/index.html.
[4] yfinance Documentation. Available: https://pypi.org/project/yfinance/.