Stock Price Prediction System Architecture

J.A.N.S Jayasooriya

March 9, 2025

Abstract

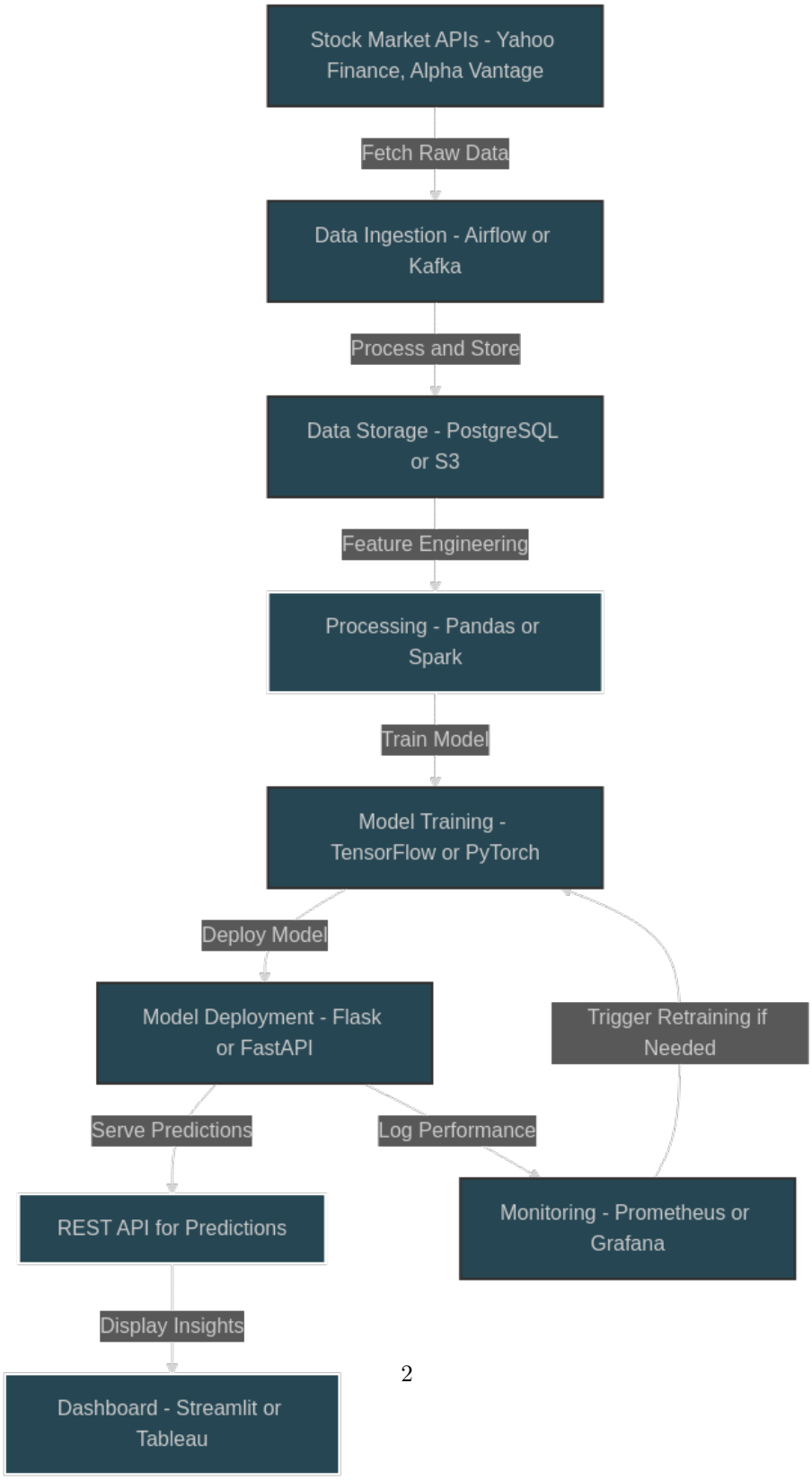
This report presents an end-to-end system design for deploying a stock price prediction model in a real-world financial analysis environment.

1 Introduction

In today's financial markets, predicting stock prices is crucial for investors and analysts. This report presents a system that automates stock price prediction using machine learning. The system collects real time market data, processes it, and trains a model to forecast stock prices. Predictions are delivered through an API and a dashboard for easy access by financial analysts. The report explains the system's architecture, data flow, model operations, and challenges in implementation.

2 System Architecture

The stock price prediction system is designed to handle data ingestion, processing, model training, deployment, and insight delivery in an efficient and scalable manner. The architecture consists of the following key components:


Stock Market APIs - Yahoo Finance, Alpha Vantage
Fetch Raw Data
Data Ingestion - Airflow or Kafka
Process and Store
Data Storage - PostgreSQL or S3
Feature Engineering
Processing - Pandas or Spark
Train Model
Model Training - TensorFlow or PyTorch
Deploy Model
Model Deployment - Flask or FastAPI
Trigger Retraining if Needed
Serve Predictions
Log Performance
REST API for Predictions
Monitoring - Prometheus or Grafana
Display Insights
Dashboard - Streamlit or Tableau

figureSystem Architecture for Stock Price Prediction

- **Data Collection and Ingestion:** Stock market data is sourced from APIs such as Yahoo Finance and Alpha Vantage. The data is ingested using Apache Kafka for real-time streaming or Apache Airflow for scheduled batch processing.
- **Data Processing and Storage:** The collected data undergoes preprocessing and feature engineering using Pandas or Apache Spark. Processed data is then stored in a structured database like PostgreSQL and raw data is stored in AWS S3.
- **Model Training and Deployment:** A machine learning model is trained using TensorFlow or PyTorch to predict stock prices. Once trained, the model is deployed as a REST API using Flask or FastAPI, making predictions accessible to other applications.
- **Prediction and Insight Delivery:** Predictions are served through the API and displayed on a dashboard using visualization tools like Streamlit or Tableau, allowing financial analysts to make informed decisions.
- **Monitoring and Retraining:** The system is monitored using Prometheus and Grafana to track model performance. If accuracy declines, an automated retraining process is triggered to keep predictions reliable.

This architecture ensures that the system is scalable, efficient, and capable of providing accurate stock price predictions in real-time or batch mode.

3 Data Collection and Ingestion

Stock market data is collected from reliable sources such as Yahoo Finance, Alpha Vantage, or Quandl. These APIs provide historical stock prices, including opening, closing, high, and low prices, along with trading volume. The system will ingest data in real-time or in scheduled batches using two primary methods:

- **Real-time Ingestion:** For real-time stock data, Apache Kafka is used to stream stock prices continuously. Kafka ensures that the system can handle high volumes of data and provides fault tolerance, which is critical for accurate prediction during trading hours.
- **Batch Processing:** For non-real-time data, Apache Airflow is used to schedule and automate batch processing tasks. This allows for daily or weekly data collection and ensures that the model can be retrained with the latest market trends.

Data is stored in both structured and unstructured storage systems:

- **Structured Storage:** Processed and preprocessed data is stored in PostgreSQL for easy querying and analysis.
- **Unstructured Storage:** Raw data is stored in AWS S3 for large-scale, cost-effective storage.

4 Data Processing Pipeline

The data processing pipeline consists of three main stages:

1. **Data Cleaning and Preprocessing:** The first stage focuses on handling missing values, removing outliers, and ensuring data consistency. Missing values are filled using forward/backward filling or interpolation, while outliers are detected using statistical methods and removed if necessary.
2. **Feature Engineering:** We extract meaningful features from the raw stock data, such as moving averages, momentum indicators, and volatility measures. Technical indicators like Relative Strength Index (RSI) and Bollinger Bands are also included to enhance the predictive power of the model.
3. **Data Storage:** After processing, the cleaned and feature-engineered data is stored in a PostgreSQL database. This allows for efficient querying and historical data analysis. Raw data is stored in AWS S3 for archival purposes.

5 Model Operations

The model operations process ensures the continuous training, evaluation, deployment, and monitoring of the stock prediction model:

- **Model Training:** We use TensorFlow or PyTorch to train a machine learning model, such as a Long Short-Term Memory (LSTM) network or a Random Forest model. The model is trained using historical data to predict stock prices for the next 5 trading days.
- **Model Evaluation:** The model is evaluated using several metrics, such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and directional accuracy. These metrics help ensure that the model performs well in predicting stock price trends.
- **Model Deployment:** Once trained, the model is deployed as a RESTful API using Flask or FastAPI. This allows the model to serve predictions via HTTP requests, making it easy for analysts and brokers to interact with the system.

- **Model Monitoring and Retraining:** The system uses Prometheus to monitor the model's prediction performance in real-time. If the model's accuracy drops below a certain threshold, it triggers an automated retraining process using updated data to ensure continuous high-quality predictions.

6 Prediction and Insight Delivery

Predictions and insights are delivered to end users, such as financial analysts and brokers, through the following mechanisms:

- **API Interface:** A RESTful API allows analysts to retrieve predictions for specific stocks. The API can be accessed using tools like Postman or integrated into other systems for real-time data retrieval.
- **Dashboard:** A web-based dashboard built using visualization tools like Streamlit or Tableau presents stock predictions, market trends, and model performance. The dashboard provides an easy-to-understand interface for users to make informed decisions.

The dashboard offers features such as:

- Visualization of predicted stock prices over the next 5 days.
- Historical trend analysis to help analysts understand the model's prediction accuracy.
- Alerts and recommendations based on model predictions and real-time stock performance.

7 System Considerations

In designing this system, several important considerations are addressed:

- **Scalability:** The system is designed to scale horizontally, allowing it to handle large amounts of data and high-frequency predictions. Cloud-based services like AWS Lambda and EC2 can be used to scale the system dynamically based on demand.
- **Reliability:** To ensure reliability, the system employs redundancy techniques, such as database replication and failover mechanisms. Kafka and Airflow provide fault tolerance during data collection and processing.
- **Latency:** Real-time predictions require low latency, which is achieved through optimized APIs and efficient model inference. Caching strategies, such as using Redis, are implemented to reduce API response times.
- **Cost:** To manage costs, the system leverages cloud-based services like AWS S3 for storage and EC2 for computation. Reserved instances are used to reduce cloud infrastructure costs.

8 Challenges and Mitigation Strategies

The design faces several challenges:

- **Challenge 1: Data Quality and Availability:** Financial market data can be noisy or incomplete, which can impact model accuracy.
  Mitigation Strategy: Multiple data sources are used to ensure completeness and consistency, and advanced data cleaning techniques are employed to handle missing data and outliers.

- **Challenge 2: Model Drift:** Over time, the model may lose accuracy due to market changes.
  Mitigation Strategy: Regular model retraining with new data and continuous performance monitoring using Prometheus will help detect and mitigate model drift.

- **Challenge 3: Real-time Processing:** Ensuring the system can process and make predictions in real-time can be challenging.
  Mitigation Strategy: Stream processing tools like Apache Kafka and efficient model inference techniques, such as model quantization, can reduce latency.

9 Conclusion

This report outlines the design of a robust stock price prediction system that automates data collection, model training, prediction generation, and insight delivery. By incorporating scalable and reliable cloud technologies, the system ensures that financial analysts have access to accurate predictions in real-time. The challenges identified, such as data quality and model drift, can be addressed through continuous monitoring and regular system updates. Future improvements may include enhancing model accuracy with additional features and experimenting with different machine learning techniques.