UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING
STUDENT BRANCH

IntelliHack 5.0

TEAM HYPER TUNERS

TASK 04

Stock Price Prediction

End-to-End System Design

Moving from an analytical model to a production-ready system involves comprehensive considerations. In this section, we detail an end-to-end system architecture designed for continuous stock price prediction using the selected LSTM model. The system design addresses data collection, data processing, model deployment, monitoring, and insight delivery to end-users.

System Overview

Our proposed system architecture includes the following key components:

- **Data Collection and Ingestion**
- **Data Processing Pipeline**
- **Model Operations (Training, Evaluation, and Deployment)**
- **Insight Delivery and Presentation**
- **System Considerations (Scalability, Reliability, Latency, and Cost)**

This comprehensive approach ensures ongoing value delivery and efficient performance in a real-world financial analysis environment.

Data Collection & Ingestion

To ensure accurate and up-to-date predictions, our system collects stock price data from reliable financial data providers such as Yahoo Finance or Alpha Vantage via API integration. The system supports **real-time data ingestion** for intraday predictions and **batch data updates** for historical trend analysis.

- **Data Sources:** REST APIs from financial providers, CSV files, or direct feeds from stock exchanges.
- **Update Frequency:** Hourly updates for short-term forecasting, daily updates for long-term predictions.
- **Storage:** Raw data is stored in a **cloud database** (AWS RDS/PostgreSQL) for structured access, while historical records are archived in a **data lake (AWS S3/Google Cloud Storage)** for long-term storage.

Data Processing Pipeline

Once collected, stock price data undergoes rigorous preprocessing to ensure quality and consistency before being used for model training and inference.

- **Data Cleaning:** Missing values are handled using forward-fill techniques, and any inconsistencies are flagged for review.
- **Feature Engineering:** Additional features such as **7-day and 30-day moving averages**, daily percentage price change, and volatility indicators are computed.
- **Normalization:** The processed dataset is scaled using **Min-Max Normalization**, ensuring the LSTM model receives properly scaled inputs for stable training.

This pipeline is automated using **Apache Airflow** to schedule and monitor data processing tasks efficiently.

Model Operations (Training, Evaluation, Deployment)

The LSTM model undergoes continuous improvement through scheduled retraining and performance monitoring.

- **Training Strategy:** The model is retrained **weekly** using the most recent stock price data to adapt to changing market conditions.
- **Evaluation Metrics:** Model performance is measured using **Root Mean Square Error (RMSE)** and **directional accuracy** to ensure consistency.
- **Deployment Method:** The trained model is packaged as a **REST API** using **FastAPI** and deployed in a **Docker container** hosted on **AWS Lambda** for efficient real-time predictions.
- **Model Monitoring:** A monitoring system logs prediction performance and detects drift in stock market trends.

Insight Delivery & Presentation

The final predictions and market insights are presented in an intuitive and accessible manner for traders, analysts, and financial institutions.

- **End Users:** Financial analysts and trading bots use these predictions for decision-making.
- **Delivery Channels:** Predictions are made available through:
    - A **web-based dashboard** (using **Plotly/Dash**).
    - An **API endpoint** that financial systems can query for real-time predictions.
    - **Email alerts** for significant market movements detected by the LSTM model.
- **Visualization Tools:** Interactive charts built with **Matplotlib, Plotly, and Tableau** display historical trends, predictions, and confidence intervals.

System Considerations

To ensure smooth operation and high-performance predictions, we designed the system with the following considerations:

- **Scalability:** The architecture is cloud-native, using **AWS Lambda** and **Kubernetes** to scale up dynamically as demand increases.
- **Reliability:** The system incorporates **redundant storage solutions** (S3 backups) and **failover mechanisms** to minimize downtime.
- **Latency:** Predictions are served in real-time (<500ms per request) by using **optimized LSTM inference on GPU-backed cloud servers**.
- **Cost Optimization:** By leveraging **serverless computing (AWS Lambda) and batch processing** for retraining, costs are minimized while maintaining high availability.

Challenges & Mitigation Strategies

While designing the system, several challenges were identified, along with strategies to mitigate them:

1. **Data Drift & Market Changes**
    - *Challenge:* Stock market trends evolve, causing models trained on past data to lose relevance.
    - *Mitigation:* Implement **adaptive retraining** where the model is fine-tuned weekly with the latest data.
2. **Model Interpretability**
    - *Challenge:* LSTM predictions lack direct interpretability.
    - *Mitigation:* Incorporate **SHAP (SHapley Additive Explanations)** to analyze feature importance in predictions.
3. **Handling Market Volatility**
    - *Challenge:* The model struggles with sudden market crashes or rallies.
    - *Mitigation:* Introduce **anomaly detection layers** to detect market shocks and adjust predictions accordingly.

System Architecture Diagram