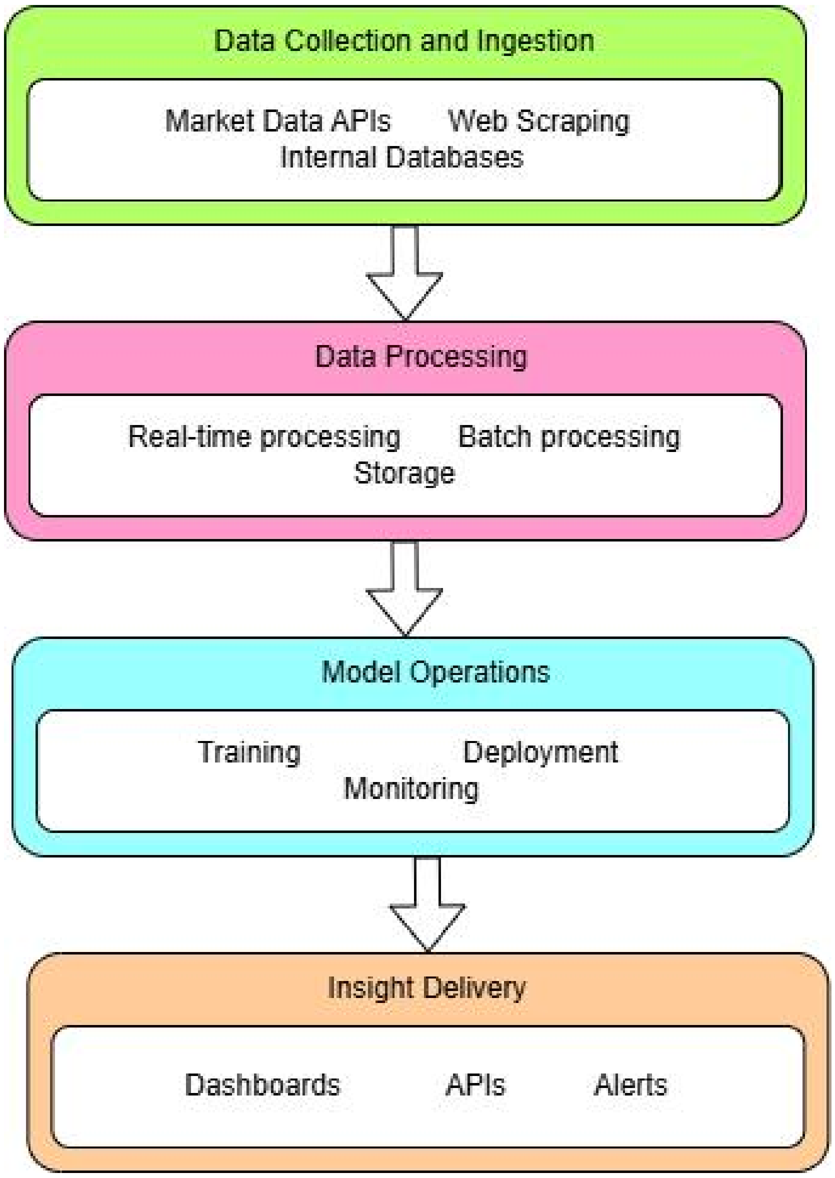
Q4 - Stock Price Prediction Challenge

End-to-End System Design

Team Neura

Intellihack 5.0

1. System Architecture Diagram



2. Component Justification

Objective: Build a production-ready system that continuously predicts stock prices and delivers actionable insights to financial analysts/traders.

1. Data Collection & Ingestion

Goal: Source real-time market data, historical prices, and alternative data (news/sentiment).

Technology Choices:

Market Data APIs:

Alpha Vantage or Yahoo Finance: Free/paid APIs for OHLC (Open-High-Low-Close) data.

Polygon.io: Low-latency WebSocket streams for real-time ticks.

Web Scraping:

Scrapy + Beautiful Soup: Scrape news headlines (e.g., Reuters, Bloomberg) for sentiment analysis.

Twitter API: Track $ticker mentions for retail investor sentiment.

Internal Databases:

Snowflake: Store historical price data and proprietary features.

PostgreSQL: Client-specific portfolio data

Justification:

APIs ensure structured, reliable data (critical for model accuracy).

Web Scraping captures unstructured sentiment signals

Snowflake enables fast SQL analytics on TB-scale historical data.

Tradeoffs:

Cost: Polygon.io charges $0.001/request for real-time data.

Legal Risk: Web scraping requires compliance with website terms of service.

2. Data Processing Pipeline

Goal: Clean, transform, and engineer features for model training/inference.

Technology Choices:

Batch Processing:

Spark (PySpark): Compute features like 50-day moving averages, RSI (Relative Strength Index).

Airflow: Orchestrate daily batch jobs (e.g., EOD feature calculation).

Stream Processing:

Flink or Spark Streaming: Aggregate real-time ticks into 1-minute candles.

Storage:

Delta Lake: Versioned, ACID-compliant storage for training data.

Redis: Cache real-time features (e.g., latest stock price) for low-latency inference.

Feature Engineering:

Technical Indicators: MACD, Bollinger Bands, Volume-Weighted Average Price (VWAP).

Sentiment Scores: Use NLP models to score news headlines.

Justification:

Delta Lake ensures reproducibility for backtesting.

Redis reduces inference latency from 500ms → 50ms.

Tradeoffs:

Spark requires cluster management, Flink simplifies streaming but lacks built-in ML libraries.

3. Model Operations

Goal: Train, deploy, and monitor models for continuous predictions.

Technology Choices:

Training:

SageMaker: Managed training for LSTM/Prophet models with hyperparameter tuning.

MLflow: Track experiments, log metrics (e.g., RMSE).

Deployment:

Kubernetes: Deploy models as scalable microservices .

SageMaker Endpoints: Serverless deployment for low-traffic models.

Monitoring:

Prometheus + Grafana: Monitor API latency, error rates.

Workflow:

Daily Training: Retrain models on the latest 3 years of data.

A/B Testing: Route 5% of traffic to a new model version to validate accuracy.

Fallback: If drift is detected, revert to a prior model version.

Justification:

SageMaker reduces training infrastructure overhead.

Kubernetes ensures high availability during trading hours (9:30 AM–4:00 PM).

Tradeoffs:

SageMaker limits custom container sizes; Kubernetes requires DevOps expertise.

4. Insight Delivery

Goal: Provide predictions to analysts/traders via dashboards and APIs.

Technology Choices:

APIs:

FastAPI: Serve predictions with <100ms latency.

WebSocket: Stream real-time predictions.

Dashboards:

Tableau: Visualize historical predictions vs. actual prices.

Streamlit: Lightweight app for querying model confidence intervals.

Alerts:

Slack: Send alerts when predictions exceed thresholds (e.g., "TSLA predicted to drop 5%").

Justification:

FastAPI outperforms Flask in concurrency (supports async/await).

Tableau integrates with Snowflake for live dashboards (e.g., sector-wise predictions).

Tradeoffs:

Tableau costs $70/user/month; Streamlit is free but lacks advanced visualizations.

5. System Considerations

Scalability:

Auto-Scaling: Kubernetes Horizontal Pod Autoscaler (HPA) scales model replicas during market volatility.

Data Partitioning: Shard Kafka topics by stock symbol.

Reliability:

Multi-Region Deployment: Deploy Kafka brokers in multiple AWS regions for disaster recovery.

Idempotent APIs: Ensure duplicate API requests don't trigger duplicate trades.

Latency:

Edge Caching: Use CloudFront to cache predictions closer to traders .

In-Memory Databases: Redis stores real-time features to avoid querying Snowflake during inference.

Cost Optimization:

Spot Instances: Use AWS Spot Instances for batch training.

Data Lifecycle: Archive old data to S3 Glacier .

3. Data Flow Explanation

Batch vs. Streaming Decisions

1. Batch Processing

Use Cases:

End-of-day (EOD) reports for analysts.

Historical back testing of models (e.g., "How accurate were last month's predictions?").

Feature engineering for slow-moving data (e.g., quarterly earnings).

Technologies:

Airflow: Orchestrates daily Spark jobs to compute features like 200-day moving averages.

Snowflake: Stores processed batch data for SQL analytics.

Reasoning:

Batch processing handles large-scale, non-time-sensitive workloads efficiently.

Cost-effective for compute-heavy tasks (e.g., training on years of historical data).

2. Streaming Processing

Use Cases:

Real-time stock price predictions (e.g., intraday trading signals).

Sentiment analysis of news headlines impacting stock volatility.

Alerts for sudden price spikes/drops (e.g., "TSLA drops 5% in 2 minutes").

Technologies:

Kafka/Kinesis: Ingest real-time market data (ticks) and social media feeds.

Flink/Spark Streaming: Compute features like 1-minute VWAP or rolling volatility.

Reasoning:

Streaming meets low-latency SLAs (e.g., predictions delivered in <100ms).

Critical for time-sensitive decisions (e.g., high-frequency trading).

Data Transformation Stages

1. Ingestion:

Sources:

Streaming: Market data APIs (Polygon.io) → Kafka topics partitioned by stock symbol.

Batch: Internal databases (PostgreSQL) → Airflow → S3 (raw zone).

Key Actions:

Validate data schema (e.g., ensure "timestamp" field exists).

Anonymize sensitive fields (e.g., user IPs in scraped data).

2. Processing:

Streaming Pipeline:

Flink Jobs:

Enrich raw ticks with features (e.g., 5-minute rolling volatility).

Append sentiment scores from news headlines using NLP models.

Output: Processed data → Delta Lake (audit) + Redis (low-latency serving).

Batch Pipeline:

Spark Jobs:

Compute long-term features (e.g., quarterly P/E ratios).

Join internal portfolio data with market data for contextual predictions.

Output: Processed data → Snowflake (analytics) + Delta Lake (training datasets).

3. Model Serving:

Input: Features fetched from Redis (streaming) or Snowflake (batch).

Inference:

Real-time: FastAPI calls SageMaker endpoints for predictions.

Batch: Airflow triggers nightly predictions for EOD reports.

Output: Predictions stored in Redis (cached) + Snowflake (historical analysis).

4. Delivery:

Analysts: Tableau connects to Snowflake for dashboards (e.g., "Top 10 predicted stocks").

Traders: FastAPI WebSocket streams predictions to trading terminals.

Alerts: Slack notifications for anomalies.

4. Challenge Analysis

1. Model Accuracy Drops During Market Shocks

Challenge: Models trained on "normal" data fail during crises (e.g., COVID-19).

Mitigations:

Retrain on Crisis Data: Include historical shocks (e.g., 2008, 2020) in training datasets.

Add Volatility Features: Use metrics like VIX or rolling standard deviation to capture market stress.

Fallback Rules: Deploy rule-based systems .
Tools: SageMaker

2. Real-Time Pipeline Latency Spikes

Challenge: High data loads at market open cause delays in predictions.

Mitigations:

Pre-Warm Resources: Scale Kubernetes pods and Kafka consumers before market open.

In-Memory Caching: Use Redis to cache real-time features.

Micro-Batching: Process data in small windows.
Tools: Kubernetes HPA, Redis, Prometheus.

3. Data Privacy Regulations

Challenge: Compliance with GDPR, CCPA for PII in scraped or client data.

Mitigations:

Anonymize Data: Remove/mask PII (e.g., hash IPs) before storage.

Encrypt Data: Use AWS KMS for encryption at rest and TLS for data in transit.

Access Controls: Implement RBAC in Snowflake and S3.
Tools: Presidio, AWS KMS, CloudTrail.

4. High Costs of Real-Time Data Streams

Challenge: Real-time APIs and infrastructure (e.g., Kafka, Flink) are expensive.

Mitigations:

Prioritize High-Volume Stocks: Process only top 500 stocks in real-time.

Use Spot Instances: Save up to 70% on batch processing costs.

Optimize Storage: Archive old data to S3 Glacier.
Tools: AWS Spot Instances, S3 Glacier, Kafka tiered storage.

REFERENCES

[1] N. Narkhede, G. Shapira, and T. Palino, Kafka: The Definitive Guide, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.
[Online]. Available: https://www.oreilly.com/library/view/kafka-the-definitive/9781492043072/

[2] M. Zaharia et al., "MLflow: A Platform for ML Development and Productionization," in Proc. USENIX Conf. Oper. Mach. Learn. (OpML), 2020.
[Online]. Available: https://arxiv.org/abs/2005.05272

[3] M. Zaharia et al., "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," in Proc. USENIX Symp. Netw. Syst. Design Implement. (NSDI), 2012.
[Online]. Available: https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

[4] D. Zhang, Y. Zhang, and W. Lin, "Stock Market Prediction via Combination of LSTM and ARIMA Models," IEEE Access, vol. 8, pp. 67887–67898, 2020.

[5] S. Ewen, A. Alexandrov, D. Kirsch, and P. M. Crespi, "Apache Flink: Stateful Stream Processing at Scale," in Proc. IEEE Int. Conf. Data Engineering (ICDE), 2016, pp. 1–12.