COMP4651 Project Report

High-Performance S&P 500 Data Project: Parallel Distributed Analytics and Machine Learning Model Optimization on Cloud

by

LEE Jimin, SHIN Yunju, KIM Hyunju

Group 7

Department of Computer Science

The Hong Kong University of Science and Technology

2024-2025

Date of submission: November 30, 2024

Table of Contents

1 Introduction

1.1 Overview

In the financial sector, particularly within stock market analysis, the ability to accurately forecast stock prices is a critical asset these days. Understanding market trends allows investment firms and analysts to optimize their trading strategies and allocate resources effectively. Our project, "High-Performance S&P 500 Data Project: Parallel Distributed Analytics and Machine Learning Model Optimization on Cloud," seeks to harness the power of historical data from the S&P 500 to create a sophisticated forecasting model on cloud for predicting future stock prices. The distributed data will be analyzed and leveraged for serverless machine learning techniques on cloud infrastructure. This approach aims to enhance the precision of our predictions, improve resource utilization, and even enable scalable insights. By developing a robust analytical framework, our project aspires to provide actionable insights that empower stakeholders to make informed investment decisions in the fast-paced stock market environment.

1.2 Objectives

Our objective is to use historical data from the S&P 500 to develop a forecasting model for predicting future stock prices. We aim to implement distributed storage solutions and leverage distributed training techniques for machine learning within a cloud computing environment. This approach will enable us to efficiently process large datasets, enhance model performance, and ensure scalability.

2 Methodology

2.1 Cloud Computing Platforms & Tools

2.1.1 AWS RDS with MySQL Database

AWS Relational Database Service (RDS) was employed for persistent and scalable storage of market data. MySQL was chosen as the database engine due to its compatibility with structured query workloads and seamless integration with the AWS ecosystem. The data collector fetches historical stock price data for S&P 500 companies over the past 7 days, automatically collecting data at 8:00 am every 7 days based on Hong Kong Time (HKT). The stored data will be stored in AWS RDS and distributed with the MYSQL database to be used for training a machine learning model.

2.1.2 Amazon S3 - Storage Platform

Amazon S3 is used for storing large amounts of data. We created an S3 bucket to store training and testing models from multiple instances. We can access AWS S3 buckets on each instance by connecting buckets utilizing boto3.client library with the provided ACCESS_KEY, SECREAT_KEY, SELECTED_REGION and

AWS_BUCKET_NAME

2.1.3 Databricks(Spark) - Visualization Tool

Databricks(Spark) is used as our visualization to have better data visualization as it supports a lot of use functions such as SQL context , and etc.

2.2 Data & Pre-Processing

2.2.1 Fetching Data from Database

```
connection = pymysql.connect(
    host='project-db.cbegtabjn5eh.us-east-1.rds.amazonaws.com',
    user='admin',
    password='comp4651',
    database='stock_price',
    cursorclass=pymysql.cursors.DictCursor
)
```

Stock price market data has been fetched from the MySQL database in AWS RDS utilizing pymysql.

2.2.2 Normalization

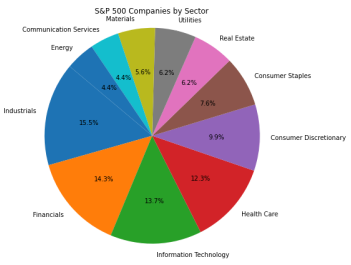
Normalization is implied for data preprocessing pipeline to ensure that input features are scaled to a uniform range, typically between 0 and 1. We utilize the MinMaxScaler from the sklearn.preprocessing library for this purpose. We extract key attributes from the market data, including 'Adj Close', 'Close', 'High', 'Low', 'Open', and 'Volume'. After normalization, we generate sequences from the scaled data, which serve as inputs for our machine learning model. This normalization process enhances model performance and stability and leads to more accurate predictions.

2.2.3 Cloud Computing

The training dataset contains S&P500 per minute data (390 minutes * 500 symbols = 195,000 rows per day -> 5,850,000 rows per month). Since the size of the dataset is extremely huge, the model training is performed with multiple instances to reduce time and increase efficiency. The trained model is then uploaded to Amazon S3 for visualizing the evaluation of the model in Databricks.

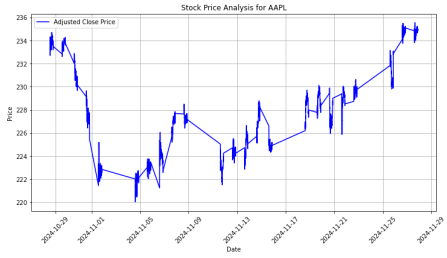
2.3 Visualization

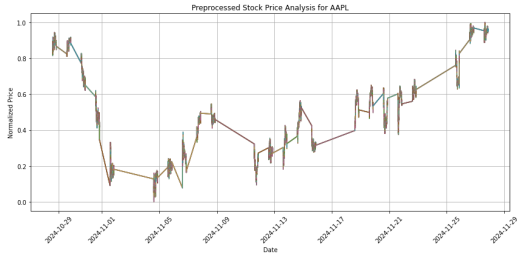
2.3.1 Sector Distribution



By leveraging Spark SQL, we aggregate the data by the GICS Sector column to compute the number of companies in each sector. The aggregated results are then converted to a Pandas DataFrame for visualization.

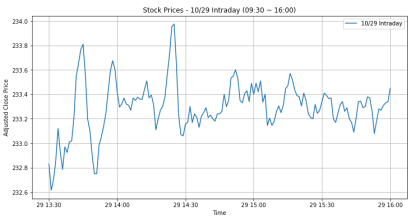
2.3.2 Normalization

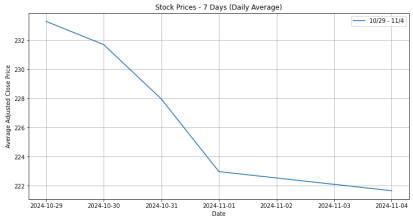


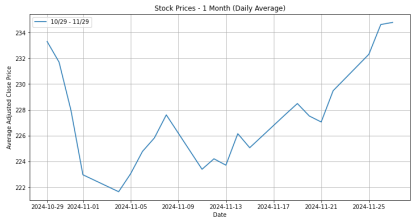


The raw dataset has been preprocessed utilizing MinMaxScaler, scaling the stock price data points within 0 to 1. The Spark Dataframe has been used to effectively visualize the effect of preprocessing.

2.3.3 Date







To further explore the datasets, we generated daily, weekly, and monthly graphs. The daily graph visualizes the continuous stock price data points recorded from 9:30 AM to 4:30 PM, providing a detailed view of intraday price movements. In contrast, the weekly and monthly graphs depict the average adjusted closing prices for each day, offering a broader perspective on price trends over time. This multi-faceted approach allows for a comprehensive analysis of stock performance, highlighting both short-term fluctuations and long-term trends.

2.4 Machine Learning Pipeline

2.4.1 Fetching & Preprocessing Data

The data is fetched from a database using SQL query filtered by the selected ticker_id and it is normalized using MinMaxScaler and transformed into time series sequences of a fixed length, with each sequence serving as input (X) and the next timestep's Adj Close as the target (y).

2.4.2 Hyperparameter Optimization

** Model configuration and Hyperparameter Tuning*

batch_size	**32**, 64, 18
hidden_size	**1**, 2
learning_rate	0.001, **0.01**
batch_size	16, **32**

We first retrieved the best-performing model from the grid search using grid_search.best_estimator_, to determine the hyperparameter to be used for further evaluation or predictions.

2.4.3 Distribute Data Across Multiple Instances

The data is divided among multiple worker instances using the parrition_data function. Each worker processes a subset of the data to enhance computational efficiency.

2.4.4 Train / Test Split

The dataset is split into training (80%) and testing (20%) subsets using train_test_split. The data is then reshaped to match the LSTM input format ([samples, timesteps, features]).

2.4.5 Distributed and Parallel Training of Model

An LSTM model is constructed with the following architecture:

- A single **LSTM layer** with 32 units and ReLU activation.
- A **Dense layer** with 1 unit to output the predicted value.

Use Three instances to build and train models. One instance is used for the master instance which distributes the data, sends the data into the worker instances and finally gets the result from the worker instances. Other two instances are worker instances which get partition data from the worker, train the model based on the partition data and send the result, containing training time and Mean Squared Error(MSE).

2.4.6 Evaluate and Save Result in S3 Bucket

The model's performance is evaluated on the test set using MSE and the evaluation results (MSE and training time) exported as a JSON file and model files are saved in an **S3 bucket** for persistence and accessibility.

3 Result

Multiple Instances Training				Single Instance Training
	Worker 0	Worker 1	Total	Total
Training Speed(sec)	10.29	9.27	10.29	14.98
Mean Square Error	0.0002	0.0002	0.0002	0.0001

4 Evaluation

The use of multiple instances allowed us to efficiently handle the vast dataset, which consists of millions of rows of stock price data.

4.1 Performance Assessment

By distributing the data across multiple worker instances, we were able to achieve several advantages:

4.1.1 Reduced Training Time

The total training time for the distributed model was 10.29 seconds, compared to 14.98 seconds for the single-instance model. This demonstrates a significant reduction in training time, largely attributed to the parallel processing capabilities of cloud computing.

4.1.2 Model Performance

The MSE for the distributed approach was 0.0002, while the single-instance model achieved an MSE of 0.0001. Although the MSE for the single instance was slightly better, the trade-off for reduced training time and increased scalability justifies the use of multiple instances.

4.1.3 Resource Utilization and Scalability

The distributed approach maximized resource utilization across instances, allowing us to make full use of the available computational power. This not only improved efficiency but also lowered operational costs by optimizing resource allocation. The cloud infrastructure allowed us to easily scale our resources according to the size of the data. This scalability is crucial when dealing with fluctuating data volumes, as it ensures that our system remains responsive and efficient.

4.2 Further Improvement

The higher MSE in the distributed approach can be attributed to the relatively lower number of epochs utilized during training, as well as the challenges associated with fully leveraging the dataset. Given the extensive size of the dataset, training time constraints limited our ability to explore more epochs and optimize the model further. In future applications, the distributed framework is expected to perform effectively, as it allows for rapid adjustments and iterations.

5 Conclusion

In conclusion, our evaluation indicates that the use of parallel distributed analytics on cloud infrastructure significantly enhances the performance of machine learning models for financial predictions. The combination of efficient data handling, improved model accuracy, and scalable resources positions our framework as a robust solution for tackling complex financial forecasting challenges. This project not only highlights the advantages of distributed computing but also sets a foundation for future enhancements in machine learning applications within the financial sector.