Big Data
CS-GY 5513 | Spring 2023
Final Project Report

Stock Price Prediction

Submitted By

Yashika Khurana (yk2773)
Raj Oza (ro2151)

NYU

Acknowledgement

We would like to express our sincere gratitude to Professor Juan Rodriguez for their lucid teaching methodology that helped us in understanding the technologies used in this project. Their expertise, guidance and constant support were invaluable in shaping our methodology in this project. We would also like to thank the teaching assistants for their help in achieving our objectives. We look forward to continuing our work in the domain of big data to achieve effective, scalable and efficient solutions to real world problems.

Contents

Introduction

Stock price prediction is crucial for making investment decisions, managing risk, strategising business decisions and forecasting the economy. Investors use stock predictions to assess the potential future performance of a company's stock and decide whether to buy, hold, or sell it. Accurate predictions can help investors make informed decisions and avoid losses.

Stock predictions also help investors manage risk. By understanding potential future market trends and stock movements, investors can make better decisions about diversifying their portfolios, setting stop-loss orders, and hedging against potential losses. Companies can use stock predictions to make strategic business decisions. For instance, if a company predicts that its stock will perform well in the future, it may choose to issue more shares or engage in mergers and acquisitions. On the other hand, if a company predicts that its stock will decline, it may choose to focus on reducing costs or selling off underperforming assets.

Stock predictions can also be used as an indicator of broader economic trends. By tracking stock market performance, economists can make predictions about overall economic growth, inflation, and employment. This stock prediction is a quintessential task to assist in making informed decisions, reducing risk and thus increasing the potential for success.

The aim of this project is to create a stock prediction engine by leveraging big data technology. We firmly believe that by combining the power of data analytics, big data and machine learning, we can achieve such a solution that can be scaled to use for regularly updated time series data, that can help thrive in today’s dynamic financial market.

In this project report, we present our data extraction methodology, data preprocessing, exploratory data analysis, machine learning model and most importantly, the complete big data pipeline we developed to create a scalable solution for stock prediction.

Methodology

Data Acquisition

The stock price data has been acquired from the Alpha Vantage API. The Alpha Vantage is a powerful and versatile tool for developers and investors who need to access financial data for their applications and analysis. Its easy-to-use interface, customizable options, and real-time data capabilities make it a popular choice among developers in the financial industry. It supports multiple programming languages. However, for this project, we used Python. It offers flexible and customizable options for retrieving data, allowing developers to specify parameters such as time interval, output size, and technical indicators. For the purpose of this project, we used the TIME_SERIES_DAILY_ADJUSTED data to achieve data going back up to 20 years for the following 6 major companies:

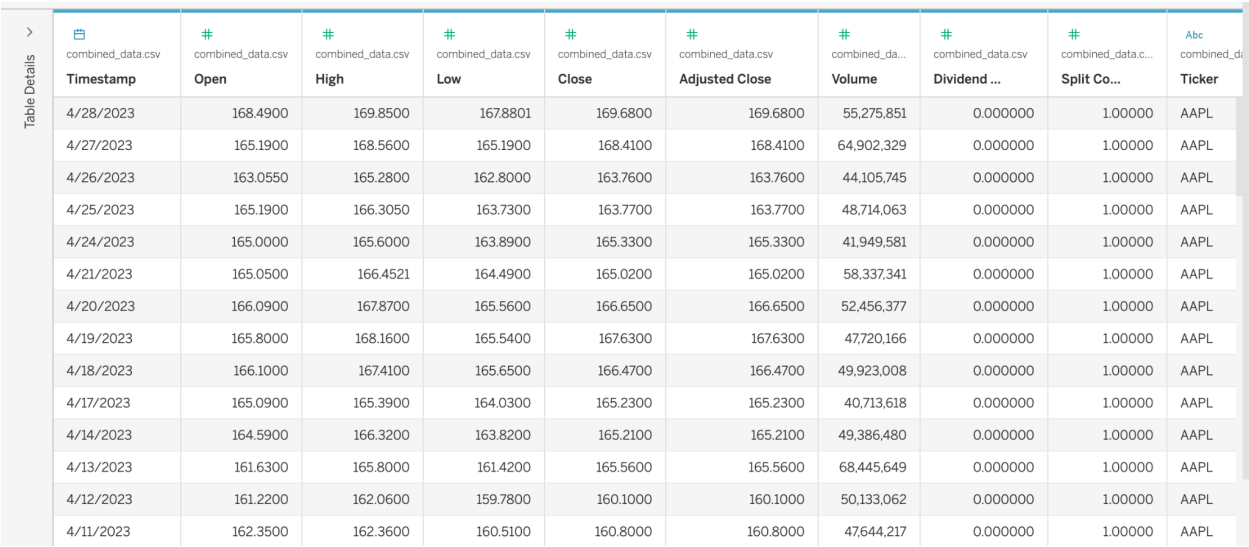
- ➢ IBM
- ➢ Google
- ➢ Meta
- ➢ Tesla
- ➢ Microsoft
- ➢ Apple

The data acquisition scripts are present in the attached code base.
The data contains the following features:

Features	Description
Timestamp	Represents the date of the stock market trade.
Open	Represents the price of the stock at the beginning of the trading day.

Low	Represents the lowest price of the stock during the trading day.
High	Represents the highest price of the stock during the trading day.
Close	Represents the price of the stock at the end of the trading day.
Adjusted Close	Represents the closing price of the stock adjusted for any corporate actions that may have affected the price, such as stock splits or dividends.
Volume	Represents the number of shares that were traded during the trading day.
Dividend Amount	Represents the amount of dividend paid per share for the trading day.
Split Coefficient	Represents the ratio by which the stock was split.

We combined the data for all 6 companies and added another column to denote the ticker value. A fragment of the data is shown below.



Timestamp	Open	High	Low	Close	Adjusted Close	Volume	Dividend ...	Split Co...	Ticker
4/28/2023	168.4900	169.8500	167.8801	169.6800	169.6800	55,275,851	0.000000	1.00000	AAPL
4/27/2023	165.1900	168.5600	165.1900	168.4100	168.4100	64,902,329	0.000000	1.00000	AAPL
4/26/2023	163.0550	165.2800	162.8000	163.7600	163.7600	44,105,745	0.000000	1.00000	AAPL
4/25/2023	165.1900	166.3050	163.7300	163.7700	163.7700	48,714,063	0.000000	1.00000	AAPL
4/24/2023	165.0000	165.6000	163.8900	165.3300	165.3300	41,949,581	0.000000	1.00000	AAPL
4/21/2023	165.0500	166.4521	164.4900	165.0200	165.0200	58,337,341	0.000000	1.00000	AAPL
4/20/2023	166.0900	167.8700	165.5600	166.6500	166.6500	52,456,377	0.000000	1.00000	AAPL
4/19/2023	165.8000	168.1600	165.5400	167.6300	167.6300	47,720,166	0.000000	1.00000	AAPL
4/18/2023	166.1000	167.4100	165.6500	166.4700	166.4700	49,923,008	0.000000	1.00000	AAPL
4/17/2023	165.0900	165.3900	164.0300	165.2300	165.2300	40,713,618	0.000000	1.00000	AAPL
4/14/2023	164.5900	166.3200	163.8200	165.2100	165.2100	49,386,480	0.000000	1.00000	AAPL
4/13/2023	161.6300	165.8000	161.4200	165.5600	165.5600	68,445,649	0.000000	1.00000	AAPL
4/12/2023	161.2200	162.0600	159.7800	160.1000	160.1000	50,133,062	0.000000	1.00000	AAPL
4/11/2023	162.3500	162.3600	160.5100	160.8000	160.8000	47,644,217	0.000000	1.00000	AAPL

Figure 1: A fragment of dataset

Data Preprocessing & Feature Engineering

We conducted a thorough literature review to understand which features contribute towards determining accurate stock predictions. Henceforth, we performed feature engineering to define certain features.

We created a feature called daily_return which computes the percentage change in adjusted_close price from the previous row, which is the daily return. Then, we created a price_range feature which computes the difference between the high price of the stock during a trading day and the lowest price of a stock during the same day. Another feature added is the prev_close feature. The "prev_close" column is derived from the "adjusted_close" column, and represents the adjusted closing price of the stock on the previous day. The lag() function is used with a window specification created using the function to ensure that the lagged value is taken from the previous row in the DataFrame based on the ordering of the "timestamp" column. Then, we used PySpark's VectorAssembler to combine multiple columns in a DataFrame into a single feature vector column. In our case, the VectorAssembler is used to create a new column called "features", which is constructed by combining the values from the "open", "high", "low", "prev_close", "daily_return", and "price_range" columns. These columns represent different

features or variables that are thought to be relevant for predicting the stock price. The labels for prediction are determined by the “close” feature.Once the "features" column is created, it is used as input to a machine learning algorithm to train a predictive model. For pre-processing, we removed the null values and created a feature vector. The VectorAssembler is a convenient way to combine multiple columns into a single vector column, which is a common requirement for many machine learning algorithms. By including the "prev_close" and "daily_return" features in the "features" vector, the model will be able to take into account the stock's historical performance when making predictions. Similarly, the "price_range" feature provides information on the daily volatility of the stock, which may also be useful for predicting future price movements.

Please refer to the codebase to find the script used to achieve the aforementioned steps.

Machine Learning

For the Machine Learning component of our project, we used Spark ML.
Spark ML leverages distributed computing to process hefty datasets across multiple nodes in a cluster. This allows faster processing of large amounts of data as compared to traditional single node ML algorithms. Hence, this was our top choice. So, we performed a comparative analysis of various machine learning models such as linear regression, decision tree, random forests, gradient boosted tree regression, long-short term memory model and evaluated them on the basis of root mean square error and r2-score.
The best performance, however, was observed on the linear regression model. So, we deployed that in our system. We split our dataset into the 80:20 ratio for training and testing respectively, which resulted in 22764 training records and 5659 testing records. The model was trained for 5 iterations, with a regularization parameter of 0.3 and elasticNetParam of 0.8. The elastic net mixing parameter controls the balance between L1 (Lasso) and L2 (Ridge) regularization. A value of 1.0 corresponds to pure L1 regularization, while a value of 0.0 corresponds to pure L2 regularization.The LinearRegression model is then fit to the training data using

the fit() method. This trains the model to learn the relationship between the input features and the target variable using the training data. Once the model is trained, it is used to make predictions on test data by calling the transform() method. The RMSE score for the linear regression model is ~2.05 on the streaming test data. It performs fairly well and a plot is presented below to compare the predicted vs actual labels.

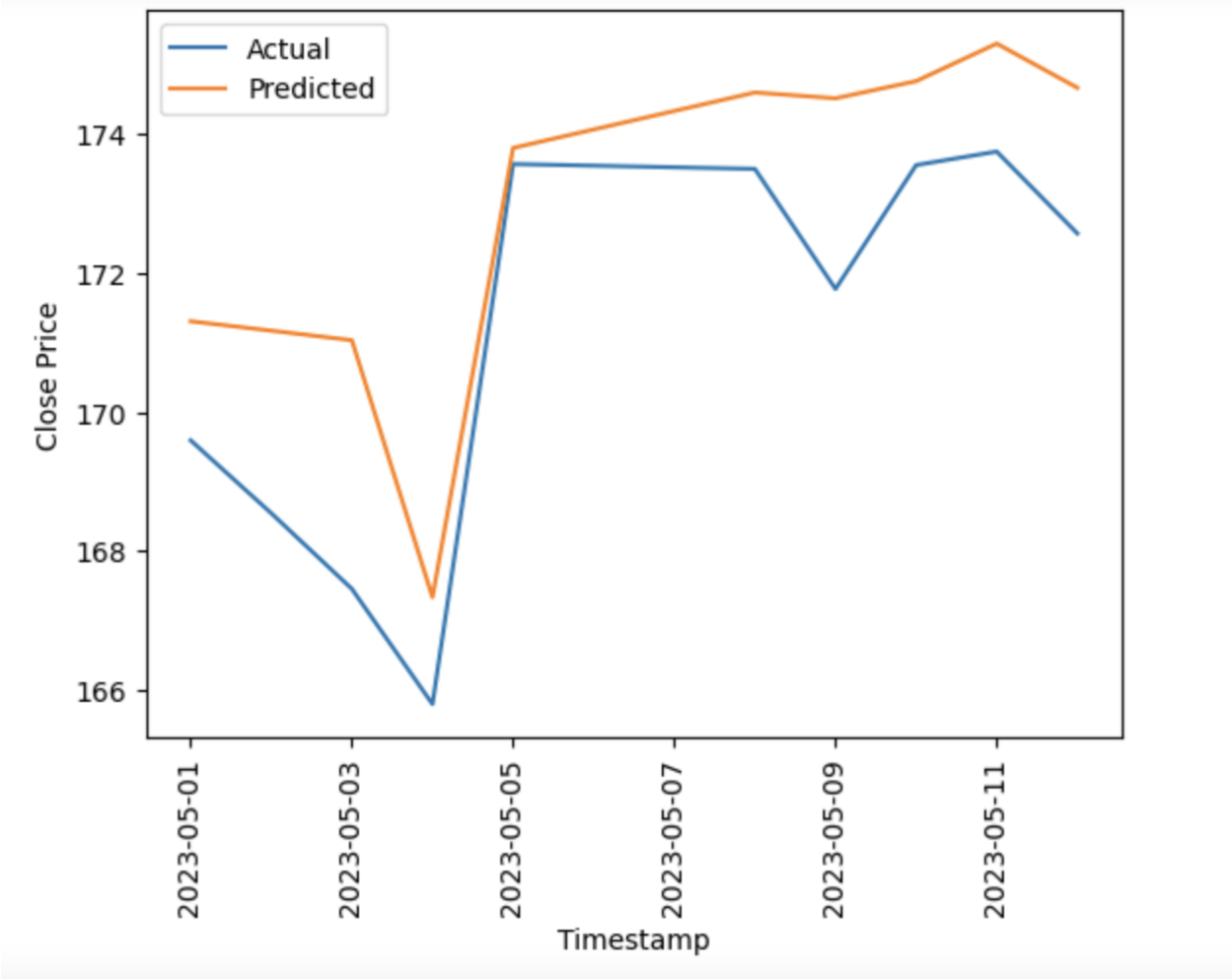


Figure 2: Actual vs Predicted values

Data Streaming

We trained our model using historical data fetched from the Alpha Vantage API. However, the streaming data which we use for generating predictions is implemented by using Kafka. Kafka is a popular distributed streaming platform

that is often used for data streaming because it provides a number of benefits for handling large volumes of real-time data. Kafka is designed to be highly scalable and can handle large volumes of data across multiple nodes. This makes it well-suited for handling the high data volumes that are common in stock prediction.This makes it well-suited for handling data streams that may be constantly growing and changing over time. It is designed for low-latency, real-time data processing. This is important for stock prediction, where fast and accurate processing of real-time data is critical for making informed trading decisions. Kafka is designed to be fault-tolerant, which means that it can continue to operate even if individual nodes or components fail. This makes it a reliable choice for handling real-time financial data, which can be critical for making trading decisions. We set up a Kafka producer that fetches data from the Alpha Vantage API and publishes it to a Kafka topic. This data is then consumed by our Spark ML application, which performs real-time predictions. Below is the pipeline that we constructed for our project.

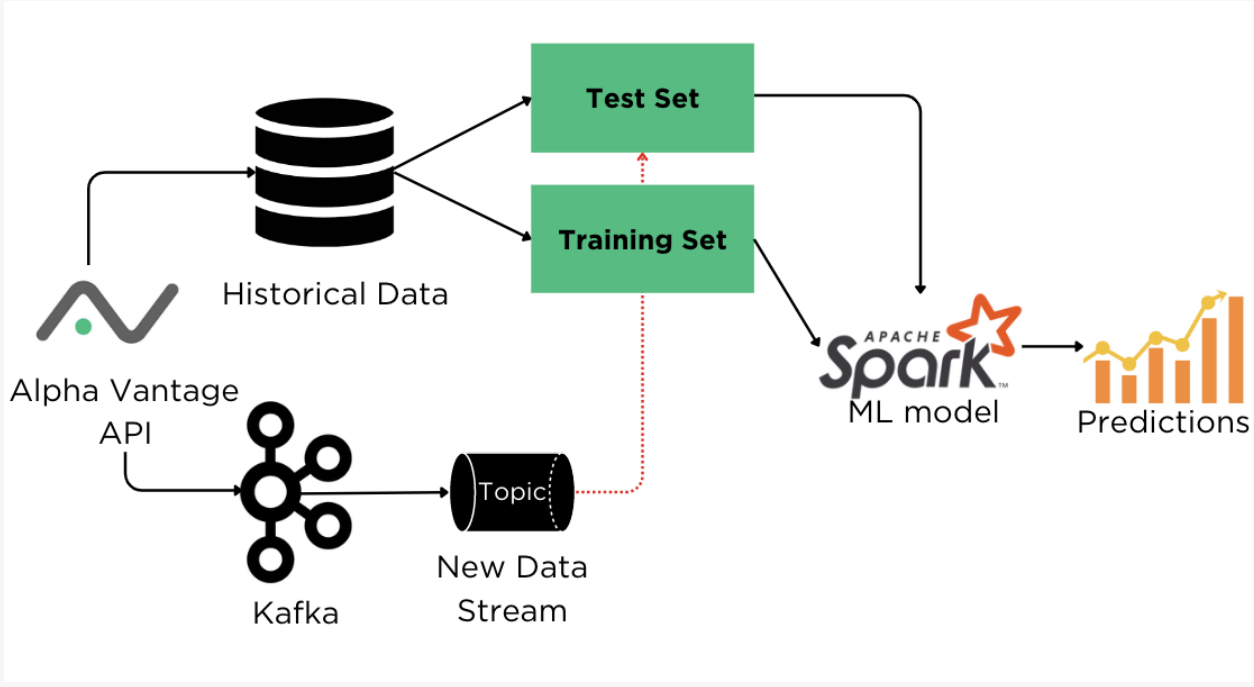


Figure 3: Pipeline

Kafka topics are used to categorize or label messages that are published by producers. Each topic is divided into partitions, which are stored on various nodes in the Kafka cluster. A partition is an ordered and unchangeable sequence of

messages that are assigned a unique identifier called an offset. The offset helps to keep track of which messages have been consumed by which consumer group. Producers publish messages to specific topics, while consumers subscribe to one or more topics to receive those messages. Kafka stores the published message in one of the partitions of the topic, determined by a partitioning algorithm that uses the message attributes to determine which partition it belongs to. Consumers can read messages from one or more partitions of a topic by forming a consumer group, which is a set of consumers that work together to consume messages from a topic. Only one consumer in a consumer group can consume a partition at a time, allowing multiple consumers to read messages from a topic in parallel.
The following figure illustrates the working explained above:

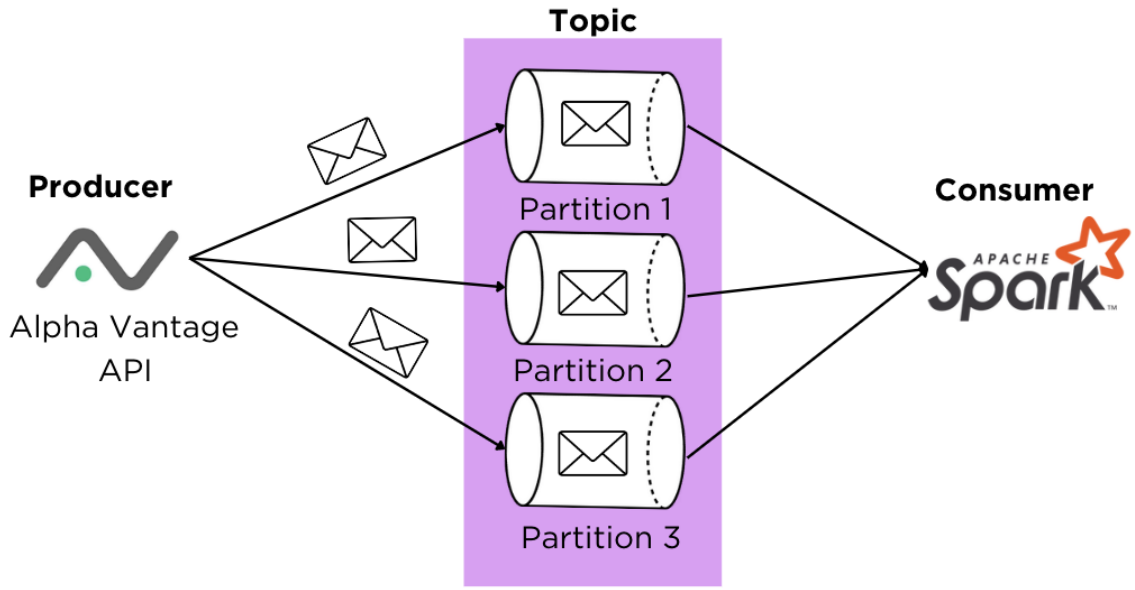


Figure 4: Kafka Streaming Architecture

Exploratory Data Analysis

We conducted exploratory data analysis to understand the trends prevalent in our data. For achieving this, we used Tableau due to its quick and easy exploration methods, and interactive visualizations.

Here are a few charts from our EDA:

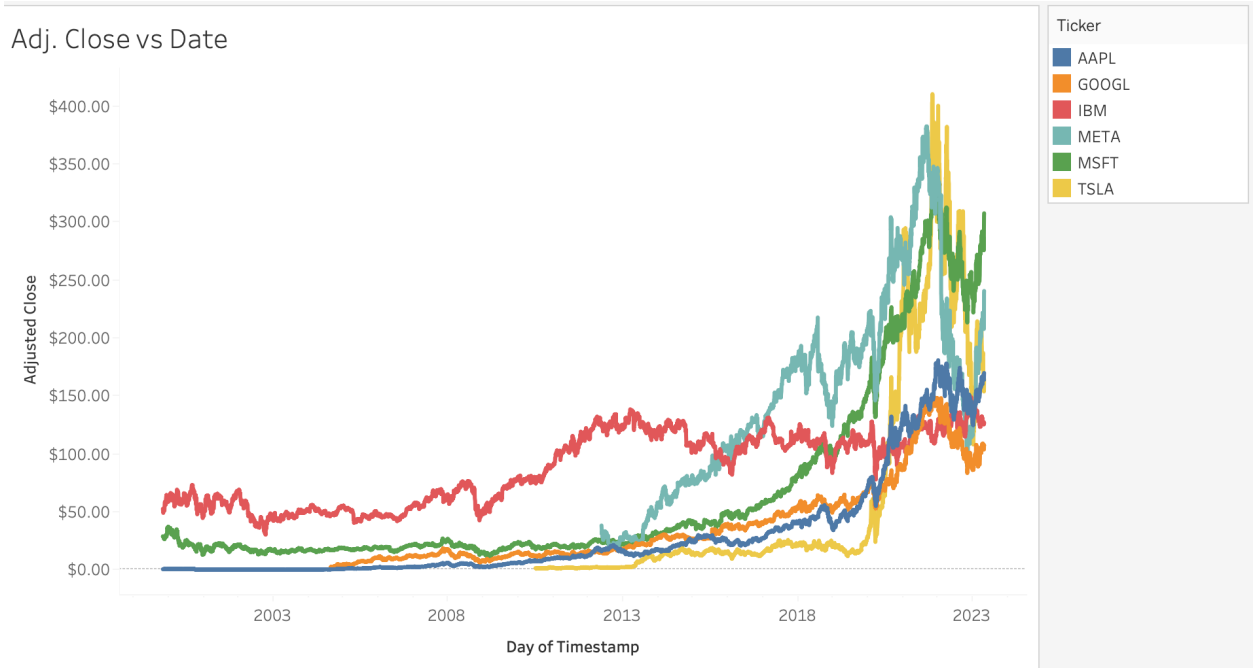


Figure 5: Adjusted Closing Price over the years

The plot above depicts the trends in adjusted closing price for all the companies, observed over the years. We observed a sharp incline in the adjusted closing price following the year 2013 this could be due to the global economic recovery that began in the aftermath of the 2008 financial crisis. The economic recovery was driven by a combination of factors, including low interest rates, so Central banks around the world lowered interest rates to stimulate economic growth, which made borrowing cheaper for companies and individuals. Another factor is quantitative easing, which involves purchasing large amounts of government bonds and other assets to inject liquidity into the financial system. Another important influence could be fiscal stimulus, so many governments around the world implemented tax cuts and infrastructure spending, to boost economic growth. As the global economy recovered, many companies saw improved financial performance, which led to higher stock prices. The rise of digital technology disrupted many industries and created new opportunities for growth, leading to increased investor demand for technology stocks. While the global financial crisis had a significant impact on the stock market as a whole, IBM was one of the few companies that was able to weather the storm and maintain its financial stability and growth prospects, which is also evident from its behavior in the graph.

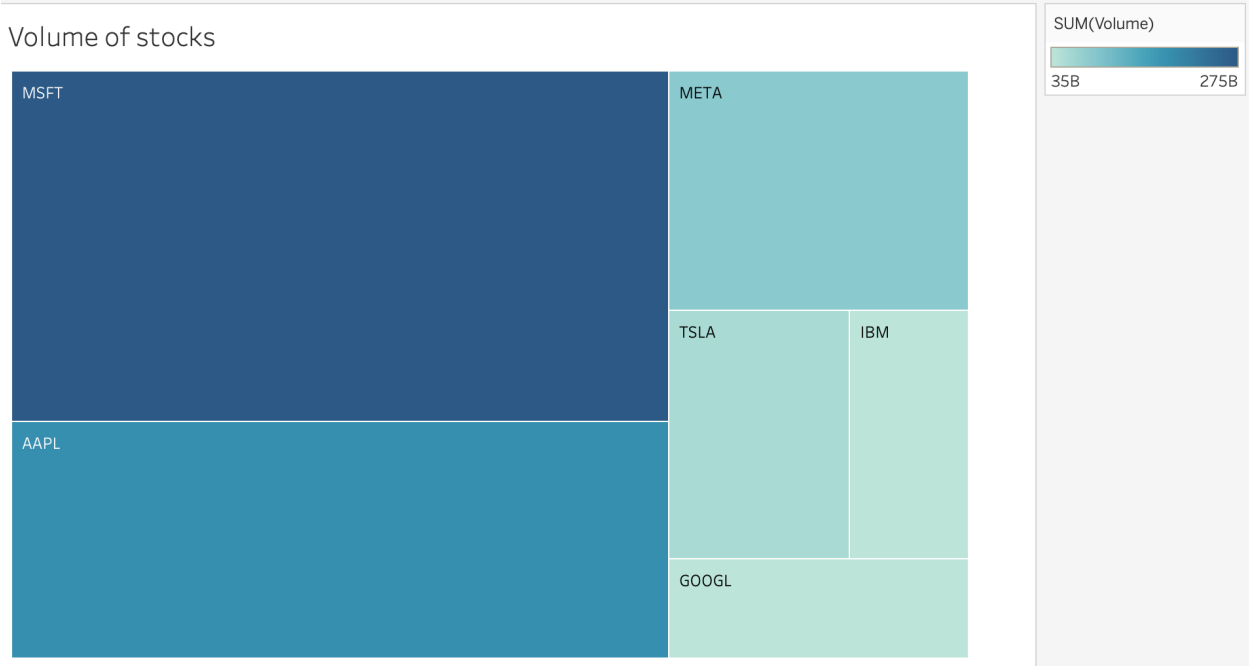


Figure 6: Volume of stocks traded per company

The treemap above indicates the total volume of stocks traded over the years by the 6 companies under consideration. Clearly, Microsoft and Apple are the major volume leaders. They are often among the most actively traded stocks on any given day.

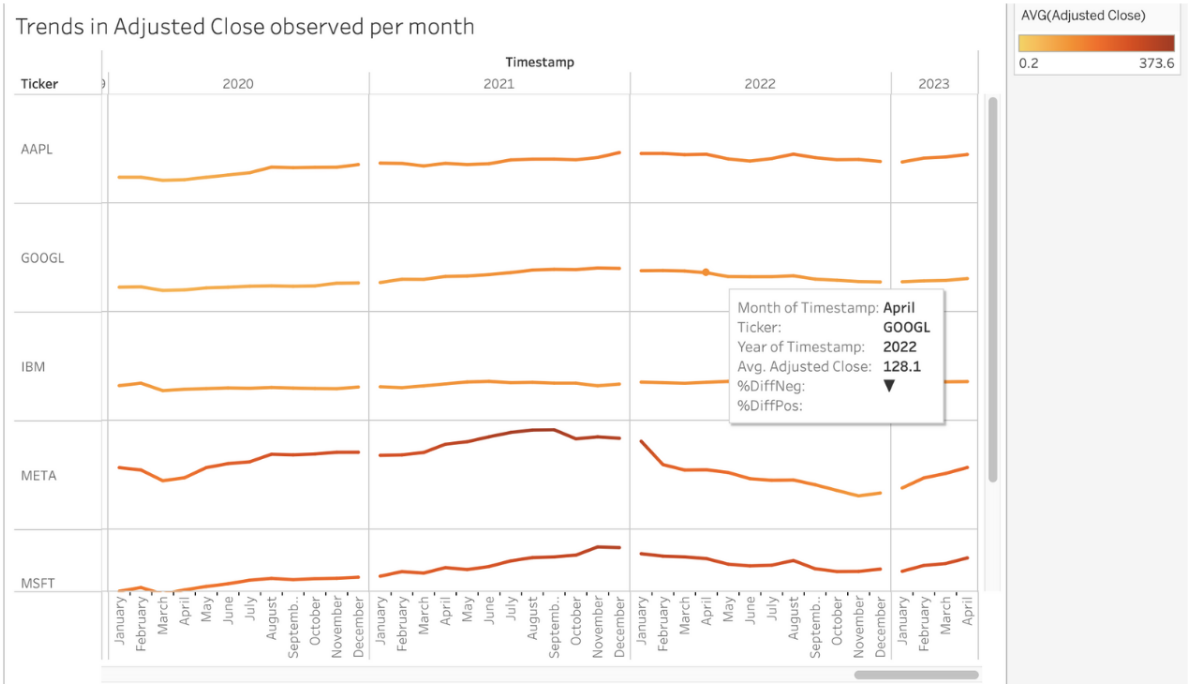


Figure 7: Percentage Change in Adjusted Closing Price

The plot above compares the percentage change in the average change in the adjusted closing price of a stock compared to the previous month over all the years. The arrow here is indicative of an increase or decrease, relatively. It represents the fluctuation and volatility in the prices over the years. Please refer to the attached Tableau workbook for accessing the complete view.

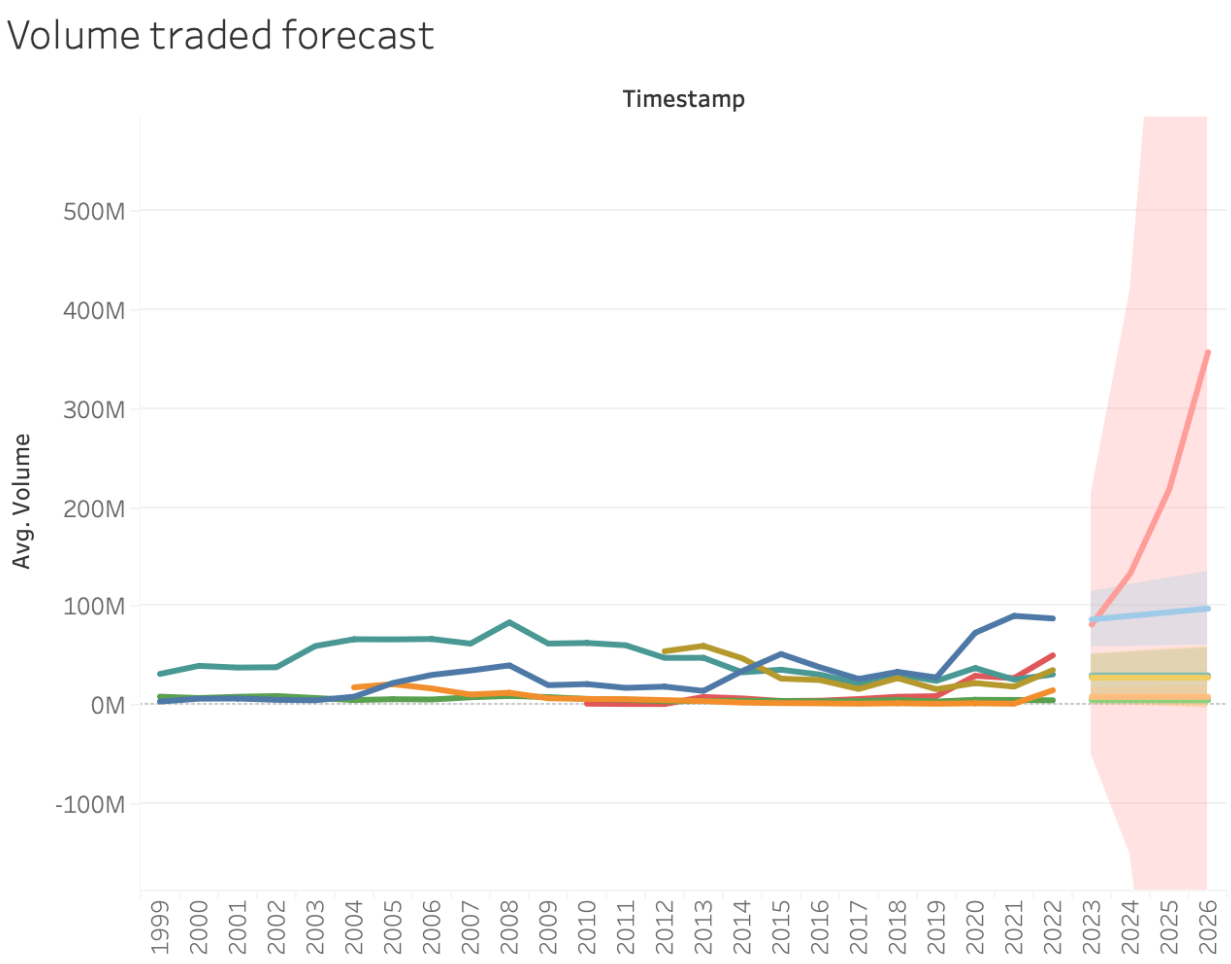


Figure 8 : Forecast of volume traded

This plot leverages the forecast feature in Tableau to identify the volume of stocks that will be traded in the next 3 years. Tableau uses simple exponential smoothing and Holt Winter exponential smoothing for forecasting, so it might not be the best indicator but it still provides a vague idea. Overall, forecasting the volume of stocks that will be traded in the coming years is an important activity for investors, traders, analysts, and companies, and it can provide valuable insights into market trends, investment opportunities, and potential risks.

Challenges

The project was an invaluable learning experience for us. We faced the following challenging situations:

We tried scraping the data from various sources but were limited by paywalls. So, we used Alpa Vantage API, as it provides a good quantity of data in its Basic plan.

We began by using Kafka along with AWS s3 buckets & Docker. However, we were constrained by limitations on the AWS platform so we used Kafka locally, as suggested by Prof. Rodriguez.

Also, we tried using LSTMs on our dataset. However, LSTMs usually require terabytes of data to perform well. So, we deployed a Linear Regression model.

Lessons Learned

This project provided us the opportunity to understand and apply various Big Data techniques and technologies. Post the project completion, we feel confident about the application of:

- ➢ ETL pipelining
- ➢ Real Time Data Ingestion using APIs
- ➢ Kafka deployment for data streaming
- ➢ PySpark for Machine Learning
- ➢ Tableau for Exploratory Data Analysis

Future Scope

We would love to further explore the application of Big Data in Stock prediction and finance. This project can be extended further and we are excited about the prospect of:

- Incorporating Twitter based sentiment analysis in our workflow. Twitter-based sentiment analysis can be useful in stock prediction as it provides an additional source of data that can be used to gauge public perception and sentiment towards a particular company or industry. By analyzing tweets related to a company or industry, sentiment analysis can provide insight into how the public perceives the company, its products, or its actions, which in turn can impact stock prices.

- We would like to provide detailed trading suggestions to users by forecasting on the data.

- Improving the Machine Learning component by improving our model by performing hyperparameter tuning or by trying other models.

References

1. Alpha Vantage API documentation
2. Kafka 3.4 documentation
3. Spark MLib Guide
4. Packt Chapter 9
5. Analytics Vidhya
6. Secution.io Blog
7. Stock Prediction with DL