CS611 Machine Learn Engineering

Project Report: Stock Price Prediction

A logo for a university

Description automatically generated

By:

Derek CHO Jun Hao

Krishna Kumar VENKATESAN

Melvin ONG You Le

NG Ming Bing

TUN Wai

19th May 2024

Table of Contents

[Business Problem 4](#_Toc170757746)

[Dataset 4](#_Toc170757747)

[System Architecture 4](#_Toc170757748)

[IAM Management 4](#_Toc170757749)

[Data Ingestion 4](#_Toc170757750)

[Data Update Pre and Post Inference 4](#_Toc170757751)

[ML Pipeline Stock Price Prediction 4](#_Toc170757752)

[Data Preprocessing 4](#_Toc170757753)

[Training 4](#_Toc170757754)

[Process 4](#_Toc170757755)

[Model Evaluation 5](#_Toc170757756)

[Model Registration 5](#_Toc170757757)

[Pipeline Execution 5](#_Toc170757758)

[ML Pipeline Sentiment Analysis 6](#_Toc170757759)

[Preprocessing 6](#_Toc170757760)

[Training 6](#_Toc170757761)

[Deployment 7](#_Toc170757762)

[Front End 7](#_Toc170757763)

[API Gateway 8](#_Toc170757764)

[Monitoring 8](#_Toc170757765)

[System Manager 8](#_Toc170757766)

[Cloudwatch 9](#_Toc170757767)

[Cost Management 9](#_Toc170757768)

[Future Works 10](#_Toc170757769)

[Model Updating and CI/CD 10](#_Toc170757770)

[Scaling of Services 10](#_Toc170757771)

[Appendices 11](#_Toc170757772)

[Data Dictionary 11](#_Toc170757773)

[Finance Data 11](#_Toc170757774)

[News Data 11](#_Toc170757775)

[Deployment Guide 11](#_Toc170757776)

[S3 Bucket 11](#_Toc170757777)

[EC2 Instance 11](#_Toc170757778)

[Lambda Functions 12](#_Toc170757779)

[EventBridge Rules 16](#_Toc170757780)

[ML Pipeline 22](#_Toc170757781)

[API Gateway 22](#_Toc170757782)

[Frontend Deployment 26](#_Toc170757783)

# Business Problem

While Mobile Trading Platforms have changed how retail investors buy and sell stocks, most still struggle to make smart trades. The main issue is that the average person doesn't have the time or know-how to analyse all the latest news and figure out whether stock prices go up or down. They may also not fully understand how the markets work, or the impact different events can have.

We want to solve this problem with a tool that predicts how stock prices will react to news stories. It would use existing metrics that are publicly available alongside analysis of the latest headlines and news articles to forecast if a stock is likely to go up or down. This way, retail investors can make better choices on whether to buy or sell a stock with just a rudimentary understanding of how markets work.

# Dataset

We decided to scrape from Yahoo Finance and News to get the relevant information as we found that it was the least challenging dataset to perform scraping. Other sites have either stricter scraping protection or more complex user interface which makes it harder to scrape. The table below shows the datasets that we retrieved.

|  |  |  |
| --- | --- | --- |
| **Dataset Name** | **Purpose** | **Source** |
| Yahoo Finance | For stock prices and news articles | Yahoo Finance |
| News Article Archive | Retrieve the body of the news articles for sentiment analysis of the company’s performance | Yahoo News |

Refer to Appendix for the data dictionary of the Finance and News data we have extracted.

# System Architecture

A diagram of a software company

Description automatically generated

## IAM Management

We managed the project with 1 root user and 4 other IAM users under a common user group called ‘MLE-Project-Users’ to allow for multiple team members to work currently on the project.

In addition, we also had to attach roles with specific permission to allow them to run their tasks. For example, Lambda functions that are involved in ETL pipeline will require S3 full access. Refer to the appendix for the role permissions for the lambda functions.

## Data Ingestion

We have implemented an ETL pipeline to ingest the 2 data sources into S3. Below is the process involved.

1. EventBridge Rules triggers 2 Lambda Function to perform web scraping daily in sequence
   1. Finance data web scraping
   2. Finance news web scraping
2. Lambda Function will trigger the web scraping on an EC2 instance as Selenium is not supported on the lambda function itself.
3. The EC2 will be triggered to execute the web scraping script saved in the instance prior using command line.
4. After web scraping is completed, the data will be saved in S3 bucket.
5. Instance is shut down after scraping is completed.
6. Systems Manager can be used to monitor status of the run commands.

## Data Update Pre and Post Inference

1. After the Data Scraping job is done, lambda function transfers data from the individual scraped files to the consolidated file (stock\_price\_consolidated and news\_data\_overall).
2. The Sentiment Analysis will draw data from the news\_data\_overall file and do the inference for new cases, Output will be saved inside S3 as a new file for each day
3. Lambda Job will call the Sagemaker Endpoint to perform inference for all new data. The inference result is then updated back in the overall inference file.

## ML Pipeline Stock Price Prediction

This project aims to predict the 7th-day closing price of stocks using an automated ML workflow on Amazon SageMaker. The workflow includes data preprocessing, model training, evaluation, and registration.

### Data Preprocessing

* **Processor**: SKLearnProcessor
* **Script**: preprocess.py
* **Steps**:
  1. Read and clean data (e.g., removing the 'Date' column and handling missing values).
  2. Encode categorical variables.
  3. Split data into training, validation, and test sets.
  4. Save processed data and scaler model to S3.

### Training

* **Estimator**: XGBoost
* **Hyperparameters:**
  + objective='reg:squarederror',
  + num\_round=50,
  + max\_depth=4,
  + eta=0.2,
  + gamma=4,
  + reg\_lambda=10,
  + min\_child\_weight=6,
  + subsample=0.7,

Process:

* 1. Use the Estimator to define the XGBoost model.
  2. Train the model with preprocessed training data.
  3. Validate the model using the validation set to ensure it generalizes well.

### Model Evaluation

* **Processor**: ScriptProcessor
* **Script**: evaluate.py
* **Metrics**:
  + - RMSE (Root Mean Squared Error)
* **Steps**:
  + - Load the trained model and test data.
    - Perform predictions on the test set.
    - Calculate evaluation metrics (RMSE, MAE, R2 Score).
    - Save evaluation results to S3.

### Model Registration

* **Criteria**: Model is registered if RMSE is below a specified threshold, RSME threshold is Currently 10.
* **Process**:
  1. Define a ConditionStep to check if RMSE meets the criteria.
  2. If criteria are met, register the model in the SageMaker Model Registry.
  3. Store model metrics for future reference.

### Pipeline Execution

* **Components**: The pipeline includes preprocessing, training, evaluation, and conditional registration steps.
* **Benefits**: Using SageMaker Pipelines automates the workflow, ensures reproducibility, and simplifies model management.

A screenshot of a computer

Description automatically generated

By structuring the pipeline in this way, we ensure an efficient and scalable process for training and deploying machine learning models in a production environment.

## ML Pipeline Sentiment Analysis

The Sentiment Analysis component of our stock price prediction system aims to analyze the sentiment of news articles related to specific stocks. This analysis helps to gauge market sentiment and predict stock price movements based on the tone of the news.

### Preprocessing

Data Ingestion:

We retrieve news data from an S3 bucket. The file news\_data\_overall.csv contains the necessary news articles, which we load into a pandas DataFrame.

Data Cleaning and Preparation:

We combine entries with the same URL to ensure unique news articles.

Each article is grouped and aggregated by its URL, combining stocks mentioned in each article into a single entry.

Stock Mapping:

We map stock symbols to their respective company names for clarity in sentiment analysis.

A dictionary maps symbols META, AAPL, GOOG, AMZN, MSFT, NVDA, and AMD to their full company names.

Sentiment Analysis String Creation: For each news article, we create a sentiment string that includes the relevant stocks and their context within the article. This enabled us to avoid the problem of articles mentioning several different stocks at once, which other pre-trained transformer models such as BERT or VADER were unable to adjust for.

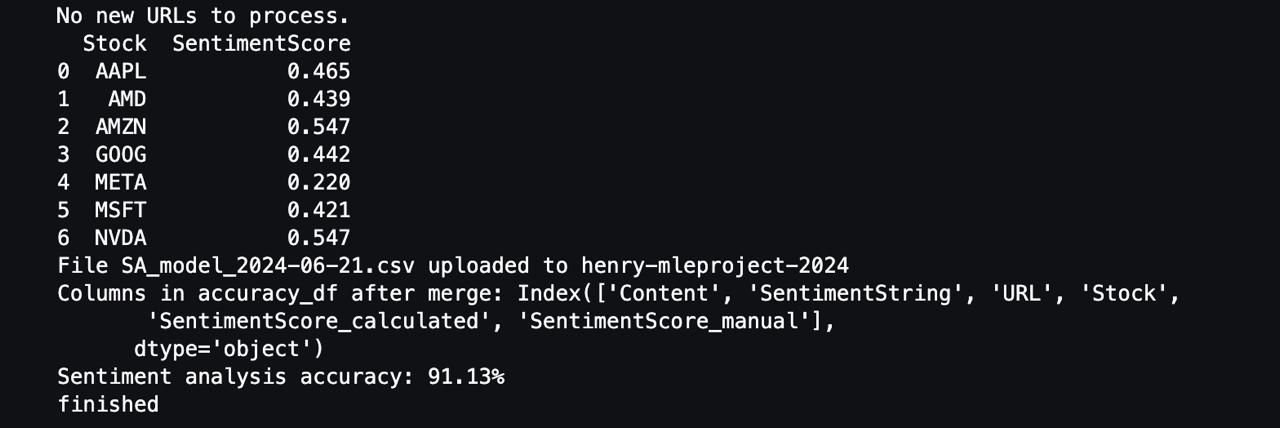
### Training

Sentiment Analysis Model: We utilize Ollama 3, an open-source large language model (LLM) with 7 billion parameters just released by Meta, to perform sentiment analysis on the news articles. In accordance with LLM prompt engineering strategy, the same prompt is utilized for all inputs to ensure consistency and accuracy across articles.

The sentiment strings are fed into the model to predict the sentiment with a set output (A range of -1 to 1, with -1 being most negative and 1 being most positive) for each article. The exact prompt used was:

“Analyze the sentiment of the following text and provide only a decimal number between -1 and 1, where -1 is extremely negative and 1 is extremely positive. Please remember I just want a fraction as output and nothing else - \n\n\n"

Each article takes roughly 70-80 seconds using the ml.t3.xlarge computing, the process of which was computationally expensive and also time-consuming.



Evalution: Manual evaluation of Accuracy for Sentiment Analysis was done due to the nature of sentiment analysis leading to a lack of ground truth, where a sample (15, changeable) of manually evaluated articles is compared to the model’s Llama3 output. As seen in the Figure above, a satisfactory Accuracy score of 91.13% was obtained.

### Deployment

Model Hosting: The sentiment analysis model is hosted on Amazon SageMaker.

The model endpoint allows for real-time sentiment predictions as new articles are ingested. As new articles are released and ingested through the pipeline, the sentiment analysis model had to append date to file names and append timestamp for every article for which sentiment has been evaluated. The model’s code had to be rewritten and analysed multiple rounds as due to limitations, a new csv file could not be provided for every new batch of articles; new entries were appended to the existing csv file instead.

## Front End

We utilized ReactJS (HTML, CSS, JavaScript) and Bootstrap 5 to develop our responsive web application, ORLANDO.

Link: <https://staging.dumx7u9xgh3it.amplifyapp.com/>

For the login page, we integrated Google OAuth for user authentication. Upon successful login, the user's username and profile picture are retrieved from the Google OAuth console and displayed on the dashboard. Once logged into the dashboard, users can select a stock from the 'Stock Name' dropdown menu and choose a 'Start Date' and 'End Date' to view stock prices for the specified date range.

The dashboard includes cards that display the Open price, Close price, Low price, High price, and News sentiment derived from our sentiment analysis of the relevant stock news. Users can view stock prices along with a prediction of the next seven days' prices in a chart created using Chart JS.

This chart provides the stock's closing prices for the selected date range, with the green line representing actual prices and the red line representing predicted prices. Users can hover over the data points to view the specific prices. At the bottom of the dashboard, a news ticker is included, allowing users to click and view the latest news related to the selected stock. This feature provides users with a comprehensive view of the stock, enabling them to make informed decisions.

A screenshot of a computer

Description automatically generated

A screen shot of a graph

Description automatically generated

React JS compiles the web app code into a build folder, which is then deployed using AWS Amplify. Amplify generates a web URL for the deployed code, allowing users to access the web app.

## API Gateway

We created 3 API in API Gateway:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **API** | **Request Type** | **Description** | **Lambda Function Called** | **Lambda Function Description** |
| Yahoo\_News\_API | POST | To retrieve the stock headlines and hyperlink to display at the bottom of the frontend interface | Yahoo\_News\_S3 | Get News article stored in S3 |
| Yahoo\_Sentiment | POST | To retrieve and display news sentiment result on the card | Yahoo\_Sentiment\_Extract | Get inferred sentiment based on the news article |
| Yahoo\_Finance\_Data | POST | To retrieve both historical and predicted price of stocks | Yahoo\_Finance\_S3 | Get historical and predicted price that was inferred and stored in S3 |

# Monitoring

## System Manager

System manager session manager is used to manage and monitor the status of the EC2 instance during data ingestion. Run command can be used to check the history of commands executed on the EC2 instance. This is particularly useful when troubleshooting errors during the scheduled ingestion if an error happens when the EC2 instance is executing the web scraping script using commands. This will help us identify if the ingestion error is caused by the EC2 or by an error in the lambda function.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Cloudwatch

We used Cloudwatch Logs will monitor the training pipeline for any errors in any of the pipeline stages, from pre-processing and evaluation. It can also be used to monitor the lambda functions that was triggered during the data ingestion. This allows us to troubleshoot errors that is logged during the pipelines.

We also used Cloudwatch Alarm to monitor the status of our EC2 instance activity. We created an alarm to automatically shutdown the instance if its CPU usage is below 3% on average for 5 minutes. This is because in an event of an error in the data ingestion, the shutdown command in Lambda function will not be triggered and will cause the instance to remain on indefinitely. Below is an example of the alarm being triggered due to a failed job.

A screenshot of a computer

Description automatically generated

## Cost Management

Anomaly detection is used to manage the cost of our project. Below shows the cost incurred for the month of June which is below the budget threshold of $25 and 40%. If the budget had exceeded by 40%, the root user will be notified by email. In addition, the root user will also be notified if the cost incurred is 100% more than the previous month.

A screenshot of a computer

Description automatically generated

# Future Works

## Model Updating and CI/CD

One of the things that needs to be set up is a human feedback system that will help us determine the ground truth of news sentiment. Currently, there is no ground truth on the sentiment, which can be provided by the users themselves. Once we have the ground truth, we can make the analysis more accurate, as we will have a reference on how better to tune the model.

The other thing that both ML systems lack is an auto-retraining function. Currently, while the accuracy of the model can be monitored, retraining it will still require a person to change the parameters and retrain the model. We hope to include some form of hyperparameter tuning in the future if we have the computing power to do so.

We also thought of integrating AWS Code Pipeline into our existing architecture for automation of building, testing and deployment of our model code. This will allow for faster release cycles of future versions of models and reduces the risk of human error in deployment.

## Scaling of Services

EC2 instance used in the ETL pipeline uses t2.micro instance. Currently we are scraping ‘big tech’ stock data. However, if we proceed to expand to ingest stocks of other categories, attaching EC2 instance to auto scaling group will be recommended.

Inference time for news sentiment is long (71 seconds per new article inference). This is due to the budget constraint of our project where we do not use a GPU instance for our model training. In a live production environment, we will need to upgrade the instance one which has GPU.

If the amount of data increases, we will also have to consider using a Relational Database Service to store our data. Currently, our data is stored in csv format in an S3 bucket due to the small size of the data involved. Once we expand our stock offerings or increase the scraping frequency, we will need an RDS to handle and categorize all the data.

Another reason to consider having a RDS with a proper backend hosted on an additional EC2 instance will be if we enhance our web application to do user-based analytics or including more metrics to the finance dataset which involves both technical and fundamental analysis which involves more complex state management.

# Appendices

## Data Dictionary

### Finance Data

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Description** | **Sample Data** |
| Stock | Stock Name of the Company | APPL |
| Date | Date when the stock price is gathered | 18/6/2024 |
| Open | Opening stock price | 217.59 |
| High | Highest stock price reached of the day | 219.63 |
| Low | Lowest stock price reached of the day | 213 |
| Close | Closing stock price | 214.29 |
| Adj Close | Closing Stock Price Adjusted for Dividends etc. | 214.29 |
| Volume | Volume traded | 7994300 |
| Close\_7\_Days | Closing 7-day price | 214.29 |

### News Data

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Description** | **Sample Data** |
| Title | Title of article | Microsoft's AI chatbot will 'recall' everything you do on a PC |
| Date | Date of article | 20/5/2024 21:55 |
| Sources | Author of the article and last updated date of article | MATT O'BRIEN, MANUEL VALDESUpdated 20 May 2024 at 2:55 pmÂ·3-min read |
| Content | Full article text | ‘Content’ |
| URL | URL of the news article | https://sg.finance.yahoo.com/news/microsofts-ai-chatbot-recall-everything-185736872.html |
| Stock | Stock Name of the Company | APPL |

## Deployment Guide

### S3 Bucket

Create default bucket with any bucket name. Enable bucket versioning.

Take note of the bucket names as you will need them later. Recommended Setup is one bucket for Codes for EC2 to retrieve, The other bucket for the webscrape data.

If you need sample data, please put the news\_data\_overall.csv and stock\_price\_consolidated.csv file inside the webscrape bucket. Please also drop in inference\_finance\_consolidated.csv into the same bucket

### EC2 Instance

* + 1. Put the Two Codes below into the code-bucket-mle-cs611 S3 Bucket.
  1. Yahoo\_Finance\_Data\_Webscrape.py
  2. Yahoo\_Finance\_News\_Webscrape\_EC2.py
     1. Open EC2, Select Launch Instance, choose a Name, and leave all other settings as Default, then click launch instance

A screenshot of a computer

Description automatically generated

* + 1. Start the instance and run the commands in the EC2\_Install\_Webscrape.txt. Refer to the text file for more information.

### Lambda Functions

#### Role Creation

2 roles to be created. 1 role specific for web scraping the other role for the generic lambda functions

Webscrape Role

A screenshot of a computer

Description automatically generated

Role name: webscraper-role (or any name you choose)

Permissions:

1. AWSLambdaBasicExecutionRole (Customer Managed)
2. StartStopEC2 (Customer inline)

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"ec2:StartInstances",

"ec2:StopInstances"

],

"Resource": "arn:aws:ec2:ap-southeast-1:357080086340:instance/i-0242760bdfc72d666"

},

{

"Sid": "VisualEditor1",

"Effect": "Allow",

"Action": [

"ssm:SendCommand",

"ec2:DescribeInstances",

"ssmmessages:\*",

"ssm:GetCommandInvocation"

],

"Resource": [

"\*",

"arn:aws:s3:::webscrape-bucket-mle611"

]

},

{

"Sid": "VisualEditor2",

"Effect": "Allow",

"Action": "ssm:SendCommand",

"Resource": "arn:aws:ec2:ap-southeast-1:357080086340:instance/i-0242760bdfc72d666"

}

]

}

Generic Lambda Role

Role Name: Any

Permissions:

A screenshot of a computer

Description automatically generated

putObjectS3 is optional as there is AmazonS3FullAccess

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"s3:PutObject",

"s3:PutObjectAcl"

],

"Resource": "arn:aws:s3:::webscrape-bucket-mle611/\*"

}

]

}

#### Functions

There should be a total of 8 Lambda functions to be created.

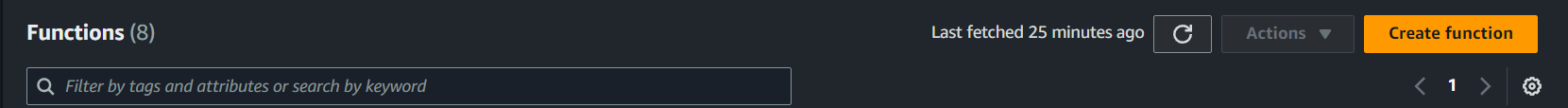
|  |  |
| --- | --- |
| **Name** | **Purpose** |
| Yahoo-News-Webscrape | News Webscraping |
| Yahoo-Finance-Webscrape | Finance Webscraping |
| Yahoo-Finance-S3 | Extraction of financial data from S3 |
| Yahoo-News-S3 | Extraction of news data from S3 |
| Yahoo-News-Update-S3 | Concatenation job for News data in S3 |
| Yahoo-Finance-Update-S3 | Concatenation job for Finance data in S3 |
| Finance-Inference-Update-S3 | Inference for finance price prediction |
| Yahoo-Sentiment-Extract | Extraction of sentiment analysis from S3 |

A screenshot of a computer

Description automatically generated

The setup for the above 8 is the same

1. Select Create Function.



1. Select author from scratch, type in the function name, Select runtime as Python 3.12, and change the default execution to the Generic Lambda Role set above, the click create function

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

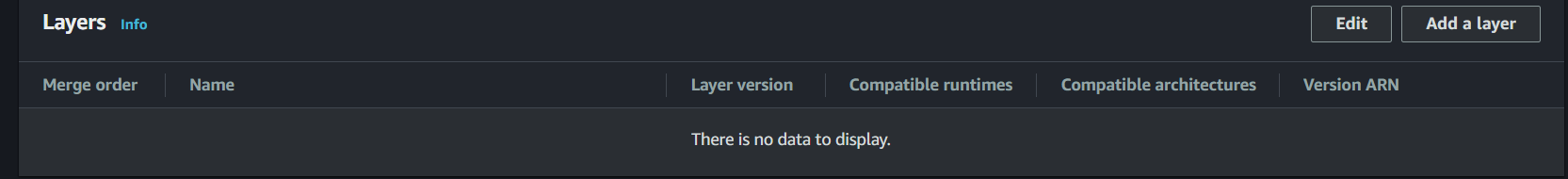
Description automatically generated

1. Once you created it, upload the respective zip file.

A screenshot of a computer

Description automatically generated

1. Go down to layers and click “Add a layer”



1. Select the layers as per below and click add.

A screenshot of a computer

Description automatically generated

1. There might be a need to increase the timeout, Memory and Storage. These can be found under the “Configurations” Tab.

A screenshot of a computer

Description automatically generated

### EventBridge Rules

To create 3 EventBridge Rules. Rules to be created instead of schedule as only rules can be set as the trigger for lLambda functions.

**Finance**

* + 1. Navigate to Rules in EventBridge and create rule

A screenshot of a computer

Description automatically generated

* + 1. Define rule detail as per the screenshot. Flexible time window set to 5min and Timeframe can be left blank.

A screenshot of a computer

Description automatically generated

* + 1. You can use any time to initiate the trigger. The example below shows the cron expression we used.

A screenshot of a schedule

Description automatically generated

* + 1. In the targets settings, select the finance data webscraping lambda function created previously.

A screenshot of a computer

Description automatically generated

* + 1. Proceed back to the Finance webscraping lambda function and add trigger

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**News**

Repeat the steps as in the finance rule. However, ensure the cron schedule is different from the finance rule and that the lambda function target is the news function. Also ensure that the target input is define as per the screenshot below. This is to select the stocks we want to ingest

A screenshot of a computer

Description automatically generated

**Update\_S3**

Repeat the steps as in the finance rule. However, ensure the cron schedule is different from the finance rule and that the lambda function target is the update lambda function.

Also ensure to include the payload for the following targets as per the screenshot below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Inference\_Update\_S3**

Repeat the steps as in the finance rule. However, ensure the cron schedule is different from the finance rule and that the lambda function target is the inference lambda function.

1. Follow the target settings below, with payload

{"sm\_finance\_endpoint\_name": "price-prediction-endpoint-2024-06-30-11-53-50",

"s3\_source\_bucket\_name": "webscrape-bucket-mle611",

"source\_file": "stock\_price\_consolidated.csv",

"s3\_dest\_bucket\_name": "webscrape-bucket-mle611",

"dest\_file": "Inference\_Finance\_Consoliated.csv"}

A screenshot of a computer program

Description automatically generated

### ML Pipeline

#### Stock\_Price\_Prediction

1. Ensure that the raw stock\_price\_consolidated.csv file is available in the bucket

A screen shot of a computer

Description automatically generated

1. Please Upload the “Stock\_Price\_Prediction\_Final.ipynb” file into sagemaker.
2. Use Kernel Data Science 3.0 to run the notebook. The endpoint will be deployed on sagemaker.

#### Sentiment\_Analysis

1. Run `pip install transformers` on AWS sagemaker terminal
2. Run `curl -fsSL https://ollama.com/install.sh | sh` on AWS sagemaker terminal
3. Run `ollama serve` on AWS sagemaker terminal
4. Please Upload ollama\_Final.ipynb into sagemaker and run it, the outputs will be found in S3

### API Gateway

There are 3 APIs here that needs to be set up

* Yahoo\_News\_API
* Yahoo\_Sentiment
* Yahoo-Finance-Data

All 3 APIs above follow the same steps for setup, if there is any difference it will be pointed out.

1. Navigate to API Gateway and Click Create API

A screenshot of a computer

Description automatically generated

1. Click on “Build” on Rest API

A screen shot of a computer

Description automatically generated

1. Key in the API name and Click Create API

A screenshot of a computer

Description automatically generated

1. Once Done, Click on Create Method

A screenshot of a computer

Description automatically generated

1. Select POST as Method Type, and Select the relevant lambda functions, then click create Method
   1. Yahoo\_News\_API – Yahoo\_News\_S3
   2. Yahoo\_Sentiment – Yahoo\_Sentiment\_Extract
   3. Yahoo-Finance-Data – Yahoo\_Finance\_S3

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black rectangular object with a white line

Description automatically generated

1. Go to Enable CORS, and enable the below setting, then click save.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Click on Deploy API, select new Stage, choose a name for the new stage, then click deploy.

A screenshot of a computer

Description automatically generated

A screenshot of a stage setting

Description automatically generated

### Frontend Deployment

Requirements:

1. Node.js and npm must be installed.

To deploy the react application code to AWS Amplify, the process starts by building the application using the “npm run build” command, which creates a production-ready version of the app in the “build” folder. Navigate to the project directory at “CS611\_MLE\_Project\frontend” to run this command. Next, log in to the AWS Amplify Console and create a new Amplify app by choosing to deploy without a Git provider. Then, drag and drop the build folder into the provided area on the console. Once uploaded, Amplify automatically starts deploying the application, and the deployment process can be monitored through the console. After the deployment is complete, Amplify provides a URL where the live application can be accessed.

**Steps to Deploy React Code to AWS Amplify:**

1. Open the terminal and navigate to the React project directory:

cd CS611\_MLE\_Project\frontend

2. Run the following command to create a production build of the application

npm run build

This command generates a “build” folder containing the optimized production-ready files.

A screenshot of a computer screen

Description automatically generated

3. Open AWS console, and go to AWS Amplify service. Click “Create new app” and select “Deploy without Git” and click next.

A screenshot of a computer

Description automatically generated

4. Open the build folder and zip the contents.

A screenshot of a computer

Description automatically generated

5. Drag and drop the zipped folder into the designated area.

A screenshot of a computer

Description automatically generated

6. Click Save and deployed.

A screenshot of a computer

Description automatically generated

7. A domain URL is generated, allowing access to the web app.

A screenshot of a computer

Description automatically generated