CST3130 Advanced Web Development with Big Data

Coursework 2: Data Visualization Website

Final Submission

Dereck Lam Hon Wah

M00826933

Date of Submission: 17.04.24

Lab Tutor: Waseemah Moedeen

Table of Contents

Table of Figures

Introduction

This report depicts the development of "KryptoVizyon," a cryptocurrency cloud-based data visualization website, which displays the numerical data, market predictions and sentiment analysis of five cryptocurrencies (BTC, ETH, BNB, SOL, and DOGE), that uses the latest third-party web services.

The website was developed in the **AWS Academy Learner Lab [70080].**

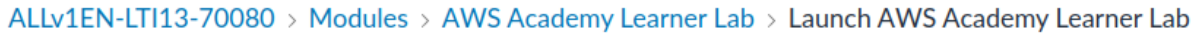

Figure 1 AWS Class

The frontend built using Vue 3 and Typescript is hosted publicly on the cloud using Amazon S3 (https://cst3130-dl661-testbucket-1.s3.amazonaws.com/index.html). Numerical data for machine learning implemented using AWS Sage maker and text data for sentiment analysis using Text Processing API have been downloaded from "Crypto Compare" and "News API" respectively, and then stored in DynamoDB, a cloud database using Typescript. The backend of the website also runs in the cloud through serverless technology, specifically Lambda functions using Nodejs for handling of database and event triggers for data processing between the different AWS cloud services. Finally, WebSocket using AWS API Gateway, is employed to handle efficient data transmission between the client and the cloud serverless backend.

Serverless Architecture

The serverless architecture for data processing of numerical and text is shown in Figure 1:

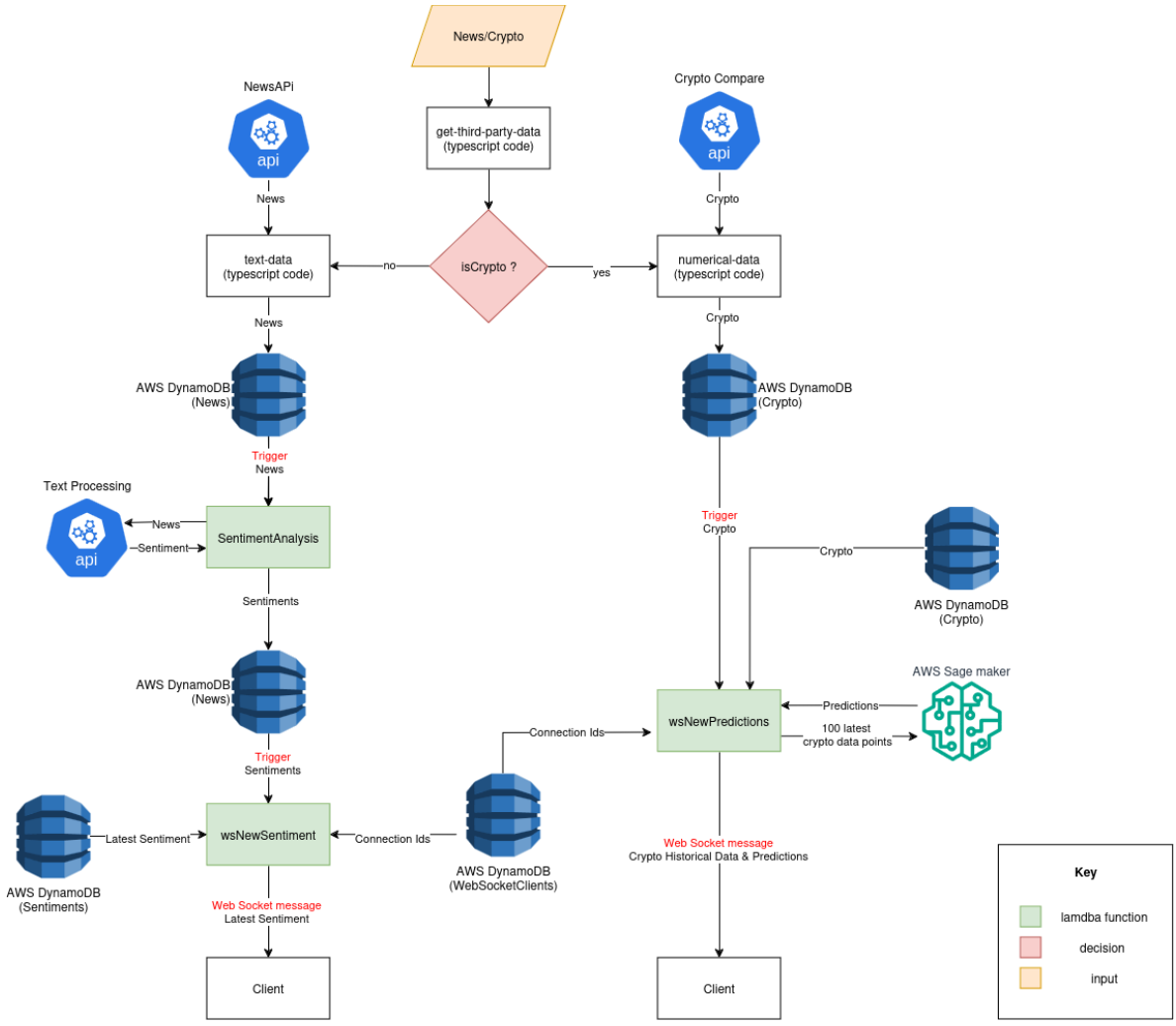

Figure 2 Serverless Architecture

Data Download and Upload

The above serverless architecture starts by retrieving a total of 2500 historical numerical data from Crypto Compare (https://www.cryptocompare.com/) and 500 text data from News API (https://newsapi.org/) for five cryptocurrencies (BTC, ETH, BNB, SOL, and DOGE).

Then these extracted data are stored into two individual DynamoDB tables; News and Crypto for the text data and numerical data, respectively. For both tables, the partition key is an id of type string and its sort key is a timestamp of type number and additionally, we created a global secondary index symbol-timestamp-index to support more efficient querying in the database for data processing.

We also created a Sentiment and WebSocketClients table to store the sentiment analysis of each cryptocurrency and connected client ids, respectively. The sentiment table will have its partition key as an id of type string and its sort key as timestamp of type number, and a global secondary index symbol-timestamp-index whereas, the WebSocketClients table will only have a ConnectionId of type string as partition key.

Sentiment Analysis

The sentiment analysis workflow will only extract the three latest entries of a cryptocurrency from the News table and process them using the Text Processing API (www.text-processing.com), a third-party web service for natural language processing. Its return value format is as follow:


```
{
        "label": "pos",
        "probability": {
                "pos": 0.85,
                "neg": 0.15,
                "neutral": 0.4
        }
}
```


Figure 3 Text Processing API Response

Then a positive, negative, and neutral average on the return values of these three entries is calculated and stored in the Sentiments table.

Machine Learning

To predict the future numeric data of the cryptocurrency, we will train and deploy a model using the AWS Sage Maker console, specifically a Deep AR algorithm, designed to work with time series data. Firstly, we will extract the 500 datapoints of one cryptocurrency from the Crypto table and split them into two data sets in JSON Lines format for training; train and test. The train dataset will represent 400 datapoints, the 500 datapoints with the last 100 datapoints removed and the test dataset with the complete 500 datapoints, also called the validation dataset). The two datasets require two fields; start with the timestamp of the first datapoint in YYYY-MM-DD HH:MM:SS format and target with an array of the datapoints. Then they are uploaded to an S3 bucket with a folder to hold the training results.

Name	Type	Last modified	Size	Storage class
BNB-test.json	json	April 9, 2024, 10:56:32 (UTC+04:00)	3.4 KB	Standard
BNB-train.json	json	April 9, 2024, 10:56:46 (UTC+04:00)	2.7 KB	Standard
BTC-test.json	json	April 9, 2024, 10:58:43 (UTC+04:00)	4.4 KB	Standard
BTC-train.json	json	April 9, 2024, 10:58:52 (UTC+04:00)	3.5 KB	Standard
DOGE-test.json	json	April 9, 2024, 10:55:44 (UTC+04:00)	3.4 KB	Standard
DOGE-train.json	json	April 9, 2024, 10:55:53 (UTC+04:00)	2.7 KB	Standard
ETH-test.json	json	April 9, 2024, 10:55:05 (UTC+04:00)	3.9 KB	Standard
ETH-train.json	json	April 9, 2024, 10:55:14 (UTC+04:00)	3.1 KB	Standard
results/	Folder	-	-	-
SOL-test.json	json	April 9, 2024, 10:54:15 (UTC+04:00)	3.4 KB	Standard
SOL-train.json	json	April 9, 2024, 10:54:27 (UTC+04:00)	2.7 KB	Standard
synthetic-test.json	json	April 5, 2024, 21:22:20 (UTC+04:00)	8.9 KB	Standard
synthetic-train.json	json	April 5, 2024, 21:22:44 (UTC+04:00)	7.2 KB	Standard

Figure 4 S3 Training Data

Then, we will use Amazon SageMaker to train models with the uploaded datasets for each cryptocurrency. We first need to create a training job with the appropriate Time Series Forecast – Deep AR algorithm and set the hyperparameters to fine tune its performance on our data.

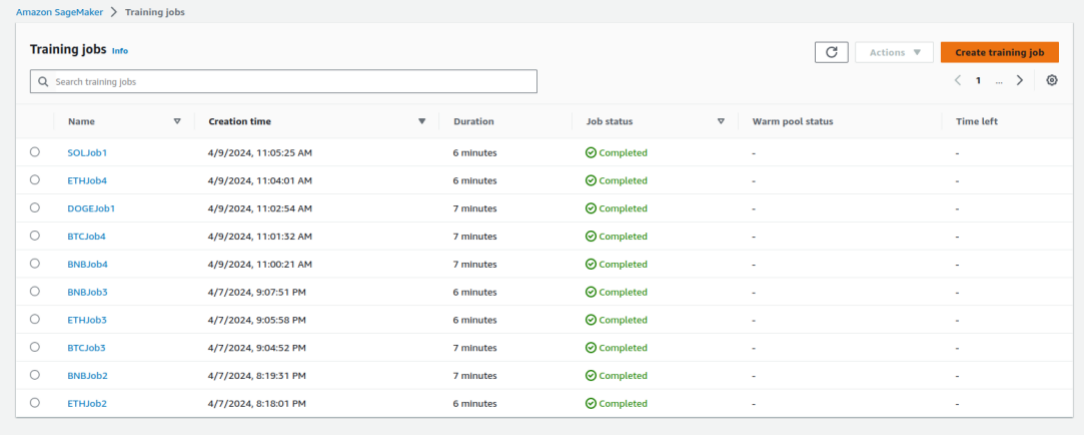

Figure 5 AWS Sage maker Training Jobs

For the training of synthetic and real data, we used below configuration and achieved satisfactory results:

Figure 6 AWS Sage maker Model Hyperparameters

When the training jobs are finished, we create the models from its result to generate the prediction.

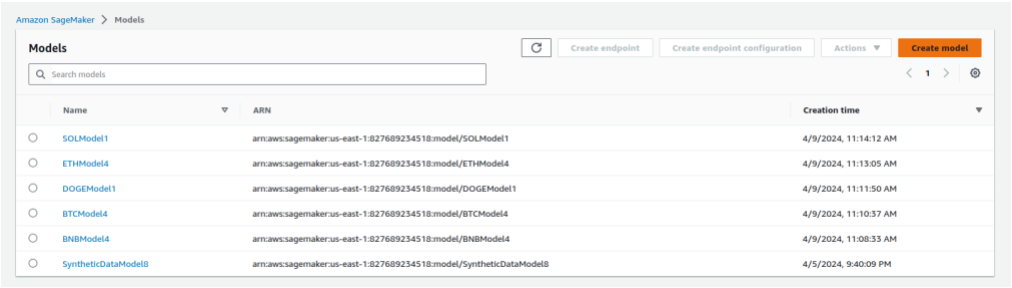

Figure 7 AWS Sage maker models

After the model creation, we then create a serverless endpoint to query the model to get prediction about new data.

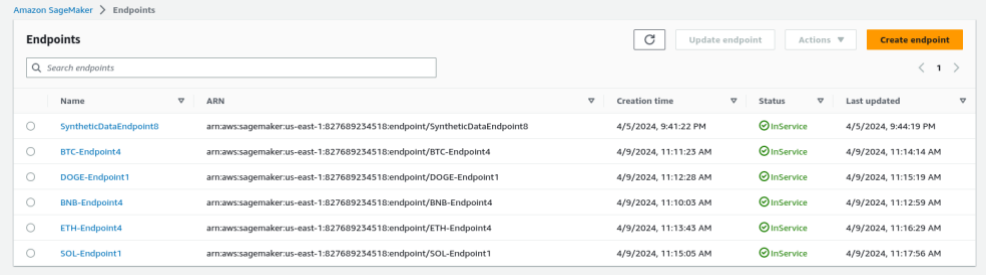

Figure 8 AWS Sage maker Endpoints

Data Visualization

Following the creation of an endpoint for each cryptocurrency and synthetic data, we created a lambda function to see and test each result on a graph using plotly.

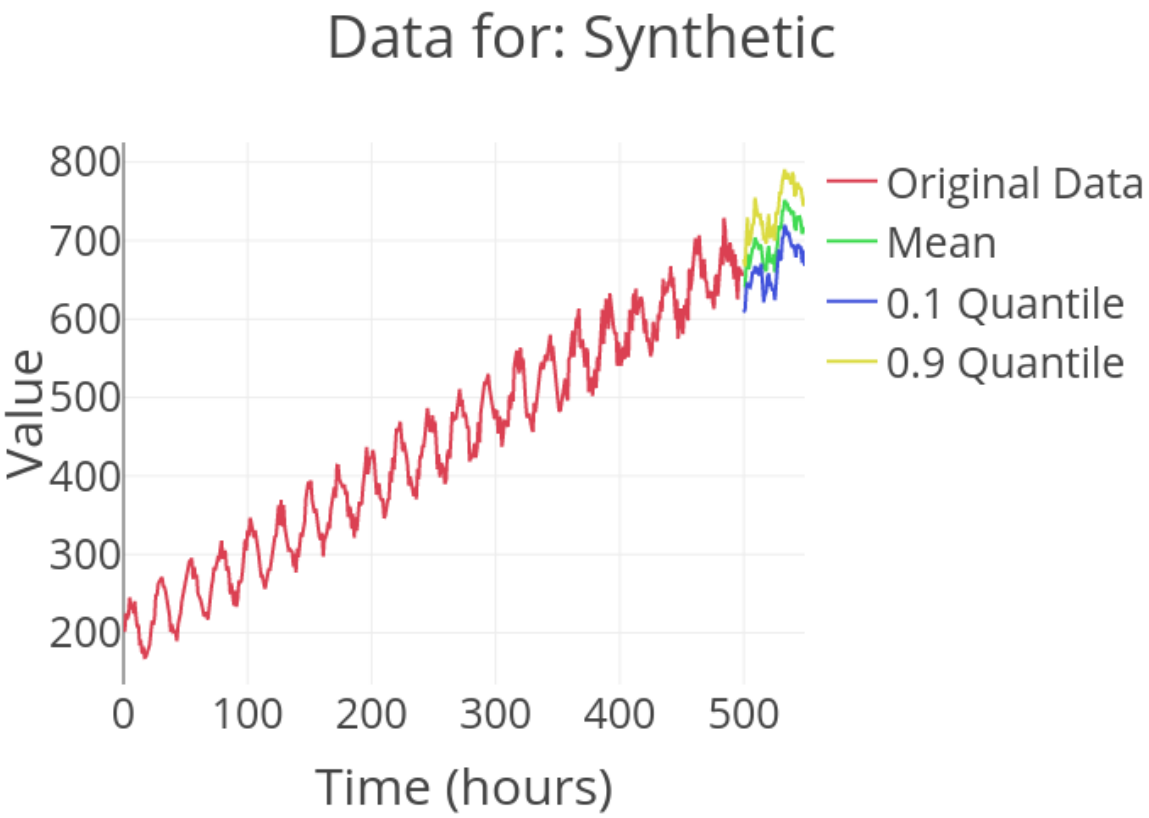

Figure 9 Data Visualisation Synthetic Data

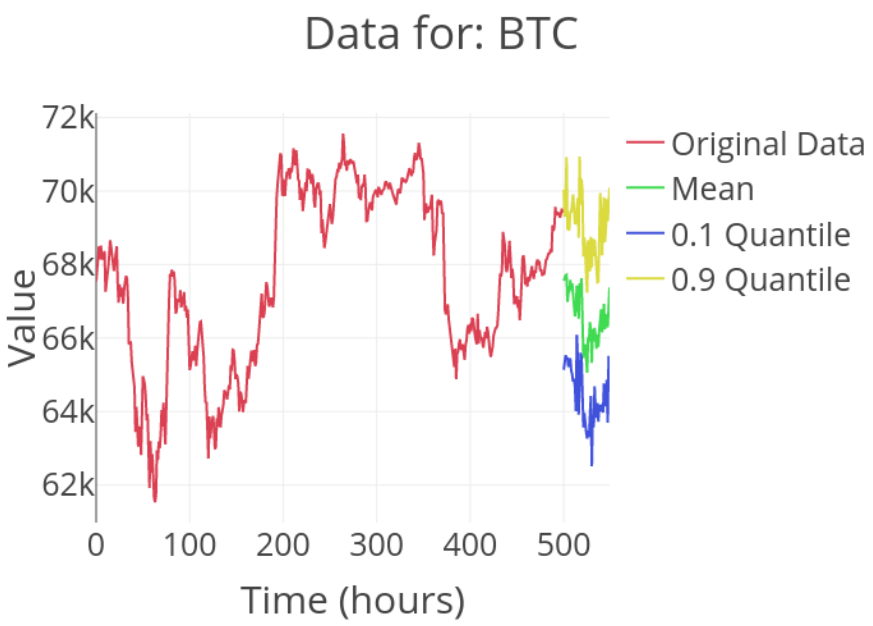

Figure 10 Data Visualisation BTC

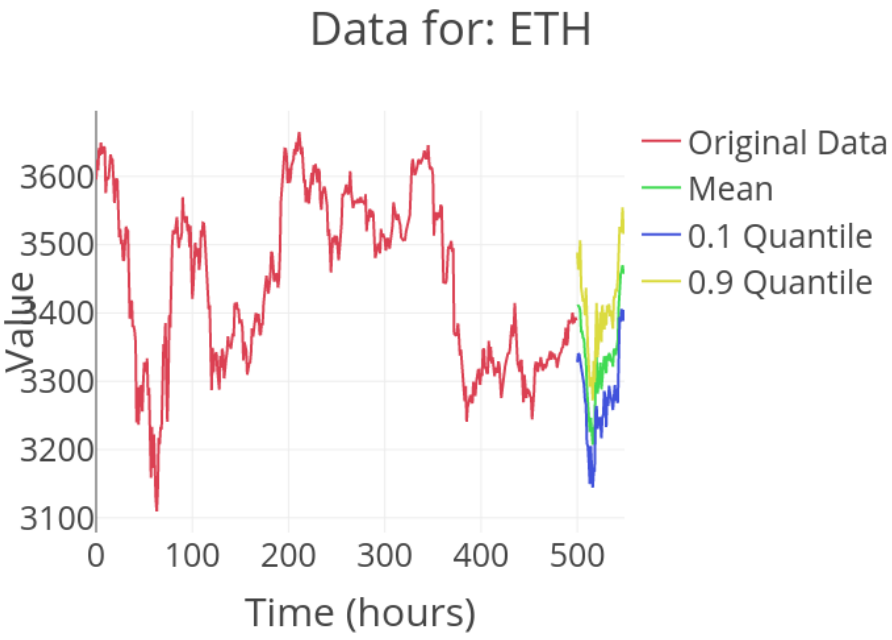

Figure 11 Data Visualisation ETH

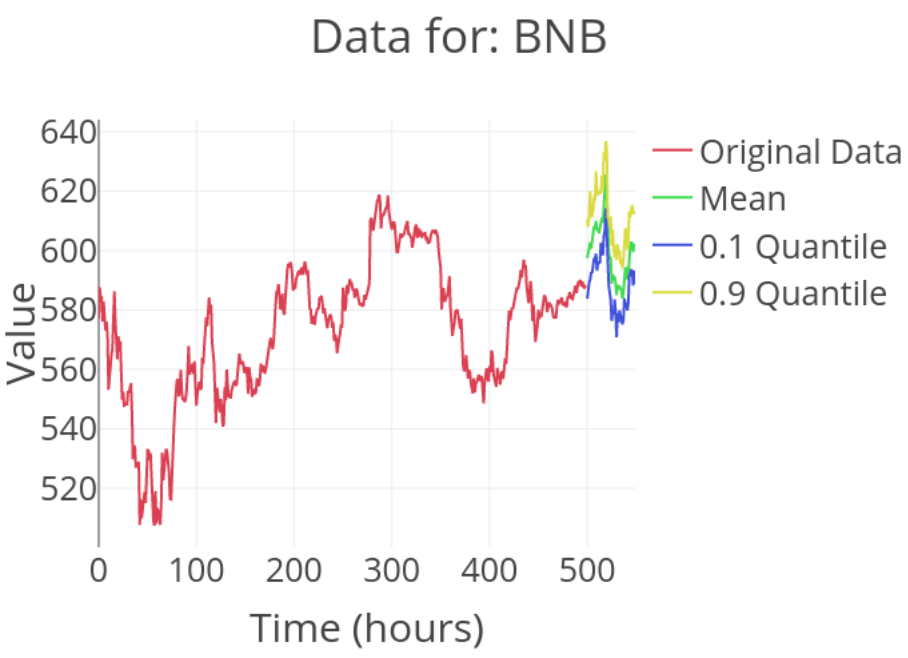

Figure 12 Data Visualisation BNB

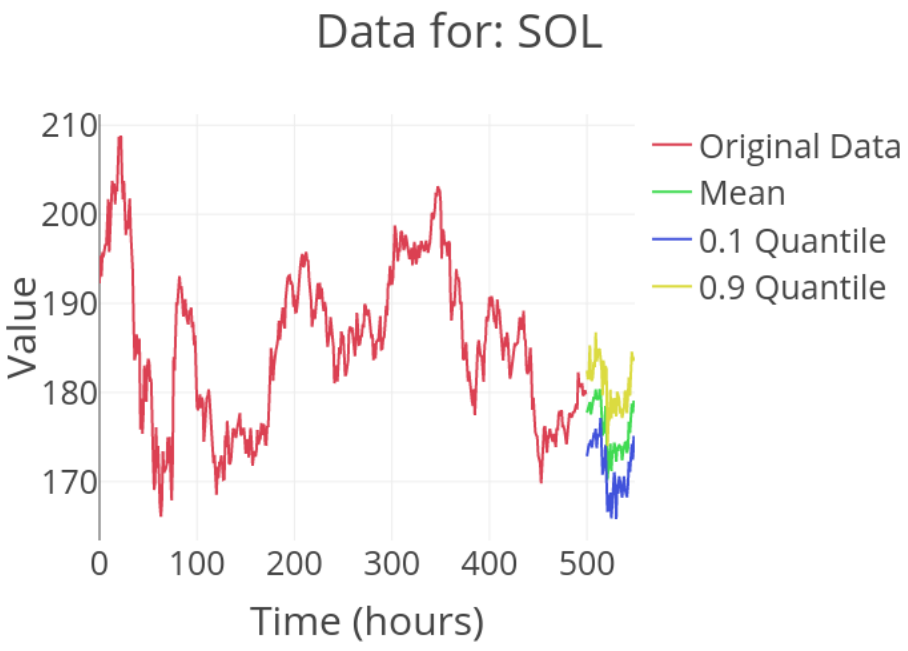

Figure 13 Data Visualisation SOL

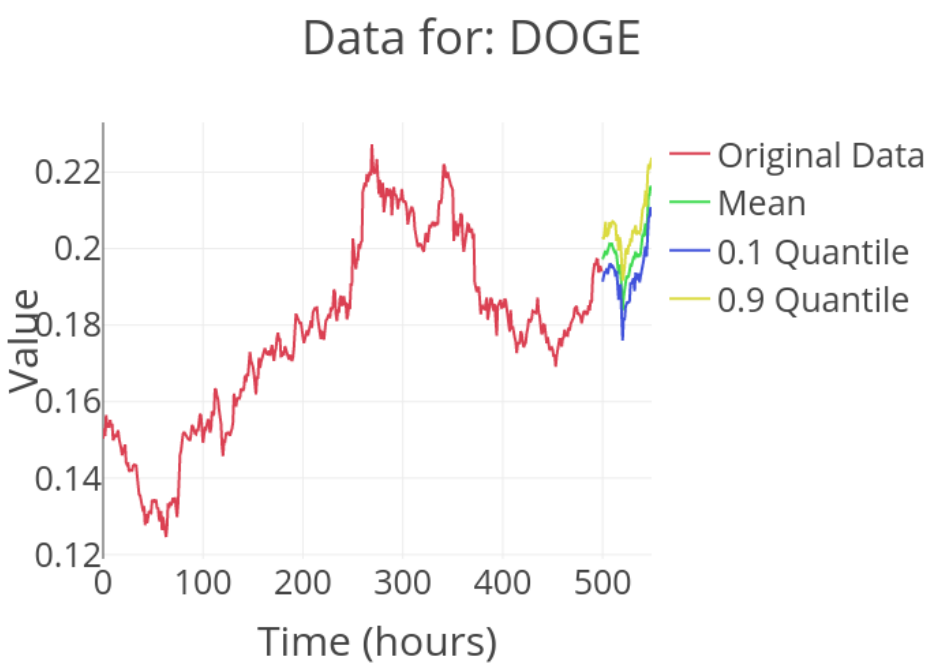

Figure 14 Data Visualisation DOGE

Other Cloud Services

Lambda Functions

In this coursework, Lambda functions serve as the backend to handle interactions with other AWS web services like DynamoDB, API Gateway and Sage maker. Below table shows the different lambda functions created their uses and triggers.

Name	Trigger	Use
wsDisconnect	API Gateway – $disconnect	When a client disconnects, it will delete the connection id from WebSocketClients table.
wsNewPredictions	DynamoDB - table Crypto	When Crypto table is edited, it broadcasts new predictions to all connected clients.
wsNewSentiment	DynamoDB – table Sentiment	When Sentiment table is edited, it broadcasts new sentiments to all connected clients.
wsInitialData	API Gateway – initialData	When a new client connects, it will retrieve sentiments and historical data from DynamoDB tables and get predictions for AWS sage maker.
SentimentAnalysis	DynamoDB – table News	When the News table is edited, it will perform and insert sentiment analysis using Text Processing API in the Sentiments table.

wsConnect	API Gateway – $connect	When a client connects, it will insert the connection id in WebSocketClients table.
wsDefault	API Gateway – $default	Return response with error message to client
plotData	None	Return a plotly URL to show on a graph the result of the AWS sage maker models for synthetic and real data.

Amazon S3

It is a simple storage service where a key is associated with an object. Objects are organized into buckets. The frontend of the website was uploaded and hosted from an Amazon S3 bucket, configured to be publicly accessible.

Link: https://cst3130-dl661-testbucket-1.s3.amazonaws.com/index.html

Moreover, the training datasets and result of the machine learning model were uploaded on a private bucket for AWS sage maker to access.

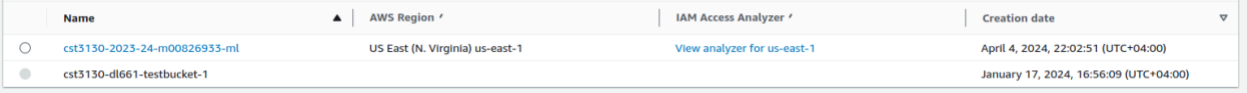

Figure 15 S3 Buckets

API Gateway

Web Socket is an alternative to HTTP for real time connection between client and server with no issues with cross-origin communication and connection with clients remain open. Modern browsers support client-side WebSocket functionality.

On AWS API Gateway, WebSocket functionality is also available, and it is possible to attach WebSocket events as triggers to lambda functions for data processing and data broadcast to connected clients. Below are the WebSocket routes and their use:

Name	**Use**
$connect	When a client connects
$default	When an error is caught
$disconnect	When a client disconnects
initialData	Message to initiate retrieval of information to be sent back to client.

Frontend

Screenshots

Figure 16 Frontend

Features

The frontend is a single page website developed using Vue 3 and Typescript.

The top section contains a list of graphs highlighting the five different cryptocurrencies historical data in green and the price predictions in blue. It is possible to zoom in and download the graph in SVG, PNG or JPEG.

The bottom section displays a selected cryptocurrency candlestick chart for its historical price data and a pie chart for its sentiment analysis. A dropdown menu in the right corner allows the user to select a cryptocurrency, which will dynamically update the candlestick and pie charts to display the newly selected cryptocurrency data.

Any changes in the News or Crypto tables will be broadcasted by AWS API Gateway to all the connected clients, which will dynamically update all the necessary charts.