**Stock Data Streaming and Analysis**

|  |  |  |
| --- | --- | --- |
|  | Electric Vehicle Stock |  |
| Emmitt Hasty  Data Science  University of Missouri-Kansas  City Missouri emhrv3@umsystem.edu Snehalatha Mallavarapu  Computer Science  University of Missouri-Kansas | Amulya Sara Computer Science University of Missouri-Kansas  City Missouri asc79@umsystem.edu  Naladala Sai Karthik  Computer Science  University of Missouri-Kansas | Gautham Sidhardh Dommeti  Computer Science University of Missouri-Kansas  City Missouri  gsddbc@umsystem.edu |

City City

|  |  |
| --- | --- |
| Missouri  smfyp@umsystem.edu | Missouri sndzm@umsystem.edu |

# ABSTRACT

Data analytics is the process of examining enormous amounts of data to find patterns, insights, and trends. Although data analytics technologies can be applied to a wide range of fields, including healthcare, politics, retail, banking, and government agencies, they are essential for survival in the competitive financial markets of the modern era. Data analytics are essential in finance to understand the ups and downs of capital markets. Great financial data science gives traders and investment advisors the assurance they need to decide for themselves whether to purchase, sell, or hold a certain security. They can use it to manage a portfolio according to short-, mid-, or long-term goals.This live streaming of data from a stock analysis is taken as source data. the data from Forex and Electric Vehicles trading API is streamed through Kafka. After streaming, the data stored in topics is analyzed. The analysis of data is visualized in Jupyter notebook for various use cases. Stocks for Electric vehicle brands are complicated, and insight can be gained from many things, as electric vehicles need special batteries, and valuable insight may be gained from looking into the stock of these companies that make the special batteries. These are companies like Amaraj, or HBL Power. As these stocks become more valuable it can become an indicator that the market for EV’s is increasing and will attribute to an increase in stock value for EV’s across the board in India. For this paper, we studied three company types, Auto Manufacturing, Battery, and Charging Point. In our study, we predict that increases in the volume of shares per day correlates to an increase in stock prices related to EV’s and can be useful for stock trading and investing purposes. Using Kafka, daily streaming data was ingested from AlphaVantage, showing stock data relevant to the daily prices. Then using Spark, the data is loaded into a Jupyter Notebook, with which we can do data analysis on. Plotting the volume of stocks moving day to day can then be used for insight into long term or short-term goals.

# INTRODUCTION

There's a Silent EV Revolution Coming in the world.

Over the world, electric vehicles are becoming more and more popular. The mass adoption of electric vehicles is being fueled by alarming emission rates and surging gasoline costs, making EV stocks the hottest trend.

It might be a business that manufactures lithium batteries, charging stations, associated components, and more. Due to the industry's youth and quick expansion, investing correctly in such businesses could result in a substantial return.

# TOOLS & TECHNOLOGIES

* Requests library,
* Pandas datareader library,  API : Tiingo API,Alpha Vantage API.
* data visualization tool: matplotlib,Seaborn
* Spark
* Kafka
* Pyspark
* AWS

# ARCHITECTURE

|  |
| --- |
|  |

Diagram

Description automatically generated

# KEYWORDS

EV: Electric Vehicle

## 1.1 KAFKA PRODUCER AND CONSUMER

Using Kafka, we have acquired daily stock reports detailing the price at market open, the high for the day, the low for the day, and the price at market close, as well as the time stamp of the day of the data. The data we gathered is the most recent 100 days, but the data can be altered to consider all data since 1999. The data is preprocessed to declare an index, as well as convert the date, which was the original index, to become a value of the data instead. Some minor changes are then made to the data for aesthetic purposes, and then the original columns are then removed from the data. This data is then loaded into the target cluster, to a topic declared ‘mm.’ Another topic, ‘mm1’ is also declared within the Kafka cluster, which is then used to load the data from the cluster into the target source,theJupyterNotebook.

Graphical user interface, text, application

Description automatically generated

**Figure 1.1: Source code from the Kafka Producer, to ingest the data and add it to the Kafka Cluster.**

## 1.2 SPARK PROCESSING

Spark is used to prepare the data for analyzing. Using readstream, Spark subscribes to the mm topic, then converts the entire data field to a string, and allowing the data to be structured. This is done by creating a data model type, and then casting the type onto the data. Once the data is prepared, it can then be sent back to the Kafka broker using writestream, and prepared for the Notebook.

Graphical user interface, text, application

Description automatically generated

**Figure 1.2: Code of the Spark Processing**

## 1.3 JUPYTER NOTEBOOK

Once the data is sent to the Notebook, it can be loaded into a dataframe, and then analysis can begin. The data is loaded from the Kafka cluster, and then cleaned to prepare a plot of the volume vs data. For convenience’s sake, only the latest 30 days are plotted to have a legible graph. This allows the data to be analyzed for trends in volume change, which is beneficial knowledge due to its effect on stock price. Using matplotlib, and kafka-python allows for the gathering of data from the topic and moving it to the Jupyter Notebook.

Graphical user interface, chart, line chart

Description automatically generated

**Figure 1.3: The plot that comes from the notebook. This is a small selection of the data, only spanning six days but showing good trend perspective.**

# 2 INITIATION

Zookeeper is used as a cluster manager and must be initiated before the system starts. Starting it will begin the Kafka session, allowing for clusters to be maintained. After that, Kafka itself must then be initiated. Once Kafka is opened, the topics can then be created, mm and mm1, respectively. These topics are within the brokers, which are within the clusters. Running the mm file will then complete the process to ingest the data to the cluster, and then running mm1 will preprocess and send the data to the target source. The data is now ready for the notebook and can begin data analysis.

Text

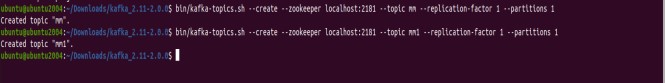
Description automatically generated

## Figure 2.1

Text

Description automatically generated

## Figure 2.2



## Figure 2.3

**Figure 2.1,2.2,2.3: These images show the code and processes of the Zookeeper initiation, the Kafka initiation, and then the topic declaration.**

Text

Description automatically generated

**Figure 2.4: Shows the code and processes of the Spark submittal process.**

**3.** **AWS ENHANCEMENTS**

In order to protect data security and storage concerns, Amazon Web Services were used in order to store the data and protect from security issues. The services used are:

* VPC
* EC2
* MSK
* Kinesis Firehose
* AWS Glue
* Lambda
* S3
* Athena

**3.1 VPC CREATION**

Firstly, we create a separate VPC for our project with minimum two public and private subnets, though we are deploying our application private subnet but to connect to internet public subnets are needed as well. The VPC is stared as ‘EV Project-vpc’ and now the private subnet must connect to internet for installing required libraries in our EC2 instance, a NAT gateway must be instantiated, shown in figure 3.1. Now we need to change route table of our private subnets to allow traffic through this NAT Gateway, shown in figure 3.2.

Graphical user interface, application

Description automatically generated

**Figure 3.1**

Graphical user interface, text, application, email

Description automatically generated

**Figure 3.2**

**3.2 LAUNCHING MSK CLUSTER**

Now we have created an MSK (Amazon Managed Streaming for Apache Kafka), which will stream the data through Kafka. To optimize performance only two brokers which will be running in each private subnet are used.

**3.3 KINESIS DATA FIREHOSE**

Let’s create a delivery stream as the Firehose receives the input directly not from Kinesis Data Stream we must select “Direct PUT” as the source, shown in figure 3.3, and as we are store the data in s3 our destination will be s3. In Kinesis Firehose transformation phase is optional you can avoid this if your application doesn’t require any transformation while streaming data. As I want to transform my data into parquet file format, which will be easy for spark to process, I have enabled ‘Enable read format conversion’ and selected ‘Apache Parquet’ as the output format, shown in figure 3.4. For this transformation we need to provide a schema through AWS Glue database. Two more important configurations are Buffer Size and Buffer interval. Kinesis Firehose will place the data in the destination in batches and the Buffer Size will indicate the size of each batch and Buffer Interval is the time interval at which firehose will place the data into destination, shown in figures 3.5 and 3.6.

Graphical user interface, application, email

Description automatically generated

**Figure 3.3**

Graphical user interface, text, application, email

Description automatically generated

**Figure 3.4**

Graphical user interface, text, application, email

Description automatically generated

**Figure 3.5**

Graphical user interface, text, application

Description automatically generated

**Figure 3.6**

**3.4 LAMBDAS**

We need two lambdas, one to get the data from the API and push it into Kafka producer and the other one to read the consume the data and push it to Kinesis Firehose

Producer Lambda:

This lambda function will get the data from API using requests, preprocess the data and publishes the data to Kafka producer.



Increase the timeout for Lambda to 2 mins, provide MSK, CloudWatch logs and VPC access & put in Private VPC (where MSK Brokers are running)

Have added Kafka python and requests packages through layers.

Add the MSK brokers as the environment variables, we get the brokers from MSK Cluster – View Client Information.

Consumer Lambda:

This lambda function will consume the data from Kafka topic and pushes the data into Kinesis Firehose.



Add MSK Cluster as the trigger with topic name that you are going to create and allow permissions to Kinesis Firehose write access, VPC access, MSK access to this Lambda

**3.5 EC2 INSTANCES**

We need an EC2 instance to create a Kafka topic for which we need to install Java and Kafka in Linux machine. So, we are creating two EC2 instance each in public and private subnet of our VPC. Similarly, we have launched one more EC2 instance in our private subnet with same configurations. We must establish the connection between EC2 instance launched in private subnet and MSK cluster by allowing traffic from each. For this all we must simply edit security groups of both and allow the traffic from each other, shown in figures 3.8 and 3.9. Next, we must SHH into private EC2 instance from public EC2 instance using PuTTY, PuTTygen and WinSCP and install Java and Kafka using the below commands.

* sudo yum install java-1.8.0-openjdk
* wget <https://archive.apache.org/dist/kafka/2.8.1/kafka_2.12-2.8.1.tgz>
* tar -xvf kafka\_2.12-2.8.1.tgz
* cd kafka\_2.12-2.8.1

Now we have to create a Kafka topic by using the below command.

* bin/kafka-topics.sh --create --topic mm --bootstrap-server b-1.kafkacluster.ix0h7i.c4.kafka.us-east-1.amazonaws.com:9092,b-2.kafkacluster.ix0h7i.c4.kafka.us-east-1.amazonaws.com:9092 --replication-factor 1 --partitions 2

If required, you can also run the consumer of same topic to see the live transfer of data by using the below command

* bin/kafka-console-consumer.sh --topic mm --bootstrap-server b 1.kafkacluster.ix0h7i.c4.kafka.us-east-1.amazonaws.com:9092,b-2.kafkacluster.ix0h7i.c4.kafka.us-east-1.amazonaws.com:9092

Replace the bootstrap servers in above commands with MSK Cluster bootstrap servers.

Now our setup is ready to stream the data and place it in s3 bucket using MSK and Kinesis Firehose and we can do this by running the Producer lambda function, shown in figure 3.10.

As the data is successfully placed in s3, now we will create Glue job which will read the files from the s3 directory, process by extracting few columns and place the results in different s3 bucket and create the Athena table in the mentioned database and stores the data.

Glue Script: 

Processed data is successfully stored in given s3 location with date as partitioned column.

Using Athena to query the data, we will receive the following results, shown in figure 3.11, given this command, Query - select \* from processed\_data limit 10;

Graphical user interface, text, application, email

Description automatically generated

**Figure 3.7: Creating EC2 in public subnet**

Graphical user interface, application, Word

Description automatically generated

**Figure 3.8**

Graphical user interface, application, Word

Description automatically generated

**Figure 3.9**

Text

Description automatically generated with medium confidence

**Figure 3.10**

Table

Description automatically generated

**Figure 3.11**

# 5. RESULTS

After the data has been visualized, it yields the following data, shown in figures 4.1, 4.2, and 4.3. The data shows that the stocks begin being shared increasingly in January, but has slowed in volume since January. This is likely due to an increase in price of the stocks, due to an outside factor such as increase in sales. It is important to note that because the stocks have slowed down does not necessarily mean it is a bad thing and that stock price has dropped considerably. Rather that the market has slowed due to less shares being bought and sold. This is not an indicator that the stocks are failing. A major indicator of stock failure would be high amounts of stock being sold, and less being bought.

Chart, line chart

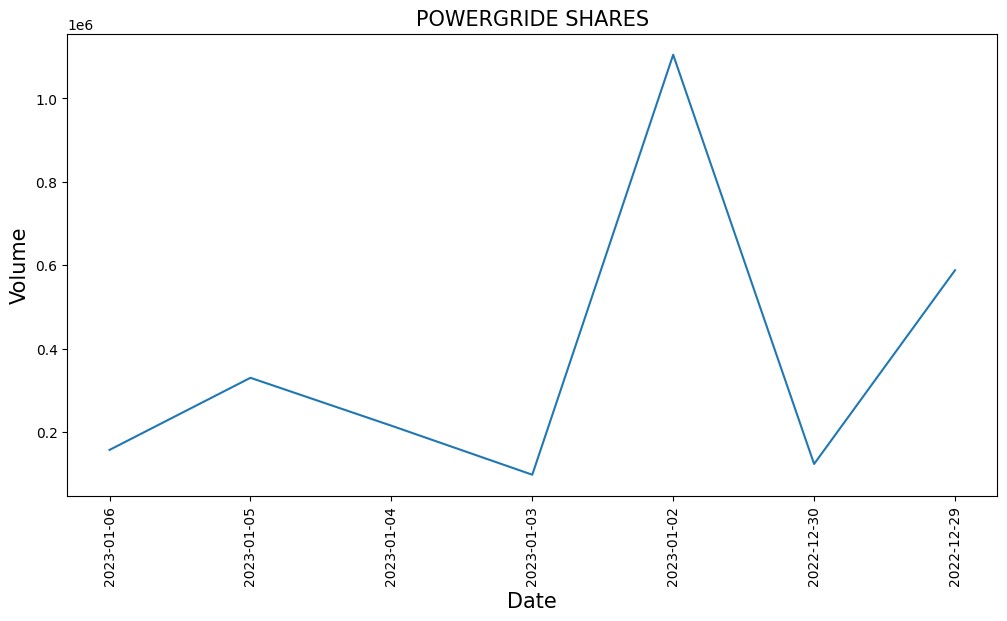
Description automatically generated

**Figure 4.1: Mahindra Shares**

Chart, line chart

Description automatically generated

**Figure 4.2: HBL POWER shares**



**Figure 4.3: POWERGRIDE Shares**

# 6. CONTRIBUTIONS

## Emmitt Hasty

* Streaming Data Pipeline
* Install and configured Zookeeper, Kafka

## Snehalatha Mallavarapu

* Install and configured Jupyter Notebook
* Spark Streaming Jobs

## Amulya Sara

* Streaming Data Pipeline
* Install and Configured Zookeeper, Kafka

**Sai Karthik Naladala**

* Spark Streaming Jobs
* Data Pre-processing
* Kafka Producer

## Gautham Sidhardh Dommeti

* Data Pre-processing
* Kafka Producer

# 7. REFERENCES

1. https://www.digitalocean.com/community/tutorials/how-to-set-up-jupyternotebook-with-python-3-on-ubuntu-20-04-and-connect-via-ssh-tunneling
2. https://sparkbyexamples.com/kafka/apache-kafka-cluster-setup/
3. https://www.alphavantage.co/documentation/