## 1. Introduction

Recently, the cryptocurrency market has garnered significant attention, with Bitcoin emerging as a leading digital asset reaching almost $100k after 2024 US election. This project leverages the comprehensive and renewing public market data provided by Binance, one of the largest cryptocurrency exchanges, to analyze Bitcoin's price movements and trading patterns. The project utilizes Amazon Web Services (AWS) for data storage and preprocessing, model training and testing on the distributed framework - Spark. The project aims to explore the possibility of strategic market analysis on cloud platforms and to uncover insights into Bitcoin's price, volatility, and market trends.

## 2. Methodology

To examine the historical data of Bitcoin, we apply comprehensive feature engineering techniques including technical indicators (Moving Averages, RSI and Bollinger Bands) to incorporate the domain knowledge into the data. This approach could improve the machine learning (ML) models' performance and offer more insight into the data.

Moreover, we will use two ML models to analyze the data and make predictions on future Bitcoin price. Firstly, the Linear Regression model will be used. It is because speed and explainability are important for investors to make informed decisions in the financial world. Hence, linear Regression is a good starting point in implementing data analysis in financial data due to its simplicity and interpretability. Another model will be the Random Forest Model. Unlike the Linear Regression model, Random Forest can capture the non-linear relationship between the features and the regression target. It would also reduce the risk of overfitting since it will combine the results of multiple decision trees. More importantly, this algorithm can be easily parallelized, which makes it a good model to test the parallelization ability of Spark.

The back testing framework evaluates our trading strategies using the most recent 2 years Bitcoin data. This simulates real-world trading conditions and helps assess the models' predictive performance and strategy profitability. Performance metrics including standard financial indicators such as account value, trade histogram, alongside traditional ML metrics like RMSE, R-Squared are used to evaluate our strategies and prediction models.

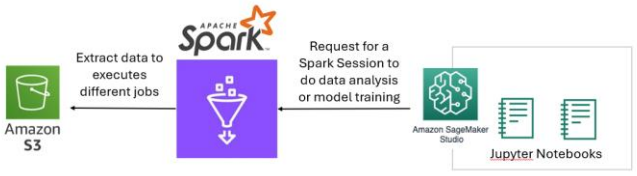
The implementation leverages Apache Spark's distributed computing capabilities to handle the computational demands of data preprocessing, machine learning and backtesting. Spark's DataFrame API is used for efficient data manipulation, while MLlib provides the distributed implementations of our machine learning algorithms. We optimize performance through strategic data partitioning and RDD caching of frequently accessed datasets. The parallel processing capabilities of Spark are particularly beneficial for the Random Forest model, where multiple decision trees are trained simultaneously, and for the feature engineering pipeline.

## 3. Implementation

### 3.1 Environment Setup

Firstly, the original dataset and preprocessed dataset were stored in the s3 bucket in .parquet file, which allows efficient compression and faster query performance. The bucket also contains a policy allowing all the rights to all the teammates' accounts. This sharing facilitates collaboration among the team. Next, we will run our codes inside the JupyterLab Integrated Development Environment (IDE) inside the SageMaker Studio. The space settings of JupyterLab are with instance of ml.t3.medium, image of SageMaker Distribution 2.1.0 and storage of 5GB. Lastly, glueSpark will be used as the notebooks' kernel. This allows our team to use the Glue interactive session to interact with Spark inside the notebooks. For each session, we will assign 5 workers of type G.1X which have 1 DPU (4 vCPUs, 16 GB of memory) and 84GB disk (approximately 34GB free). This will offer a good balance between cost-effectiveness and speedy performance. In short, this approach will provide us with more convenient data analysis and model training experience.

Besides, most of the services and environments used are serverless. This could simplify the setup, reduce the operational overhead and enhance scalability since it would eliminate the need to maintain our own infrastructure. Hence, this enhances the convenience and development efficiency of our project.


Extract data to executes different jobs
Request for a Spark Session to do data analysis or model training
Amazon S3
Amazon SageMaker Studio
Jupyter Notebooks
Figure 1: Simplified Illustration of the Environment

## 3.2 Dataset

The Binance Dataset [1] includes various kinds of cryptocurrency market data. In this project, we will limit our scope to utilize the Bitcoin spot data from 1st August 2017 to 30th October 2024. Besides, the k-line market data is extracted with a trading interval of 1 hour, with features stated in Table 1 below.

| Features | Description |
|---|---|
| Open time | The timestamp when the trading period starts |
| Close time | The timestamp when the trading period ends |
| Open | The price at the beginning of the trading period |
| High | The highest price reached during the trading period |
| Low | The lowest price reached during the trading period |
| Close | The price at the end of the trading period |
| Volume | The total amount of Bitcoin traded during the trading period |
| Quote asset volume | The total amount of quote asset (USD) traded during the period |
| Number of trades | The total number of trades executed during the trading period |
| Taker buy base asset volume | The amount of base asset (Bitcoin) bought by takers during the period |
| Taker buy quote asset volume | The amount of quote asset (USD) bought by takers during the period |

Table 1: Features of the Bitcoin spot K-line market data

## 3.3 Data Preprocessing

There are two stages of data preprocessing. The first stage is to improve the structure and format of the downloaded raw data while the second stage is to implement feature engineering to add more features.

For the first stage, we refer to the GitHub repository “Guidance for Digital Assets on AWS” [2] to first download the raw data from the Binance website. Then, we transform the data into a structured format as a data frame. For the features “Open Time” and “Close Time”, we change them from timestamp to the dateTime type, which is more human readable. Besides, we also add a ticker column to indicate the trading pair (i.e. the pair of base assets as Bitcoin and quote asset as USD). Finally, we will convert the data into. parquet file with Snappy compression and upload them to the s3 bucket.

For the second stage, we do some research on the financial domain to investigate some important technical indicators. By utilizing the domain knowledge, we add the 10-day Simple Moving Average (SMA-240), 20-day Simple Moving Average (SMA-480), 14-day Relative Strength Index (RSI) and Bollinger Bands with period of 20 days which include the Middle Band, the Upper Band, the Lower Band and the standard deviation (STD). We expect these new features will allow us to explore more possibilities of the AI models and gain more insight into the dataset.

## 3.3 Linear Regression

We will use the close price in the next trading period as the regression target and the selected features (Open, High, Low, Volume, Quote asset volume, Taker buy base asset volume, Taker buy quote asset volume, 10-day Simple Moving Average, 20-day Simple Moving Average, 14-day Relative Strength Index, the Middle Band, the Upper Band, the Lower Band and the standard deviation) to train the Linear Regression model provided by the Spark ML package [3].

Before we train the Linear Regression model, we plot some scatter plots of different features against our regression target: the close price in the next trading period to see if linear relationships exist among them. With Figure 2, we can observe that the features Open, High, Low, SMA-240, SMA-480, Middle Band, the Upper Band, the Lower Band illustrate a clear linear relationship with the regression target. Although the other features do not show a clear linear relationship with the next close price, our team still include them in the model construction to test the ability of Spark - the distributed framework on handling a large feature space.

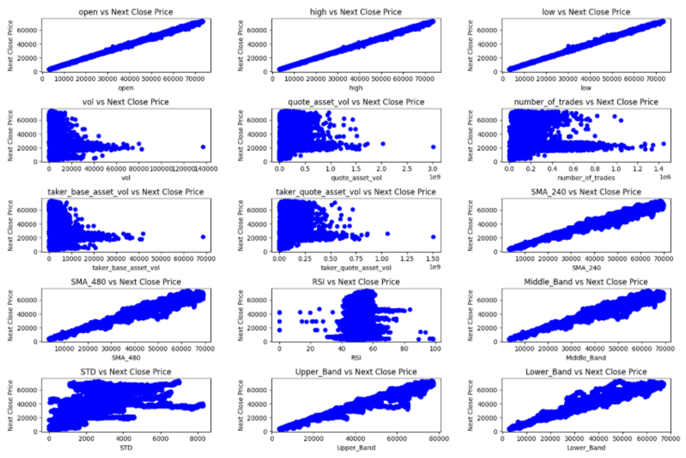


Figure 2: Scatter Plots of different features against next close price

To train and test the Linear Regression model, we will first randomly split the data as 80% training data and 20% test data. After the model construction on the training data to find the coefficients and intercept parameters, we will test the model by using it to do prediction on the test data. Finally, the Linear Regression model achieved a Root Mean Square Error (RMSE) of 688.489 and R-Squared score of 0.9988. This indicates a high accuracy and performance of the model in predicting the next close price.

## 3.3 Random Forest

To train the Random Forest model, we will choose the same group of selected features used in linear regression model construction and set the next close price as the regression target. The model "RandomForestRegressor" provided by the Spark ML package [3] will used with numTrees set to be 100.

Similarly, we will first randomly split the data as 80% training data and 20% test data. After fitting the training data into the model and getting the random Forest model supported by 100 decision trees, we will test the model using the test data. The Random Forest model gives an RMSE of 719.848 and R-Squared score of 0.9987. This illustrates that the Random Forest model can predict the close price in the next trading period accurately, despite a slightly poorer performance compared to the Linear Regression model.

Table 2 below summaries the results of the two models to demonstrate a better comparison while Figure 3 illustrates the visualization of the prediction of both model and the actual next close price.

| Model | RMSE | R-Squared |
|---|---|---|
| Linear Regression | 688.489 | 0.9988 |
| Random Forest | 719.848 | 0.9987 |

Table 2: Comparison of the performance of Linear Regression and Random Forest

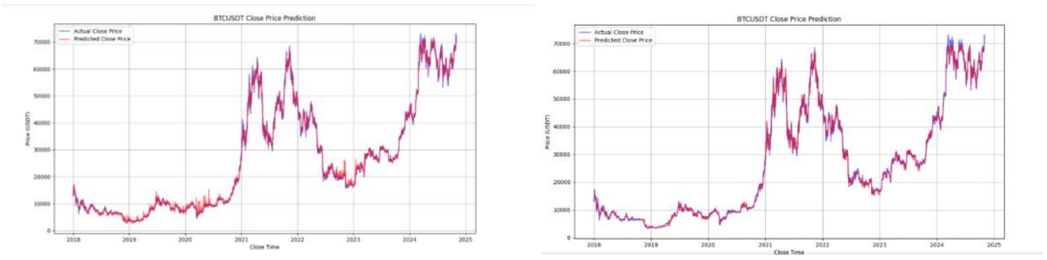


Figure 3: Visualization of the prediction of Linear Regression (left)
and Random Forest (Right) and the actual next close price

Apart from the prediction performance, Random Forest offers more insight on the data by finding the feature importance. The model shows that the features Open, High, Low illustrates the highest importance over the other features with importance of 0.187067, 0.369678 and 0.252303 respectively while SMA_240, SMA_480, Middle Band, Upper Band and Lower Band demonstrate a middle level of importance of 0.065437, 0.070826, 0.031623, 0.019123, 0.003282 respectively. The other features show the importance of value close to 0.

## 3.4 Back Testing

We will carry out back testing on four different strategies, with two being traditional trading strategies and the remaining two using the prediction from Linear Regression and Random Forest model respectively to make investment decisions.

The subset of Bitcoin market data from January 2023 to October 2024 will be used to evaluate the strategy. The back testing procedure is as follows. Firstly, we will pass the subset of data to each strategy, which will return a series of Buy and Sell signals at different timings. Next, with these signals, we will apply a simple method. When Buy signals are received, we will use all our cash to buy Bitcoin if the remaining cash balance is positive. When Sell signals are received, we will sell all the Bitcoin to the market. Finally, we will summarize the strategy results by computing the total number of trades, the overall profit or loss, the number of winning trades and losing trades, the ratio of winning trades over losing trades (Win/Loss ratio) and the average profit or loss per trade. Also note that all the signals are generated using Spark to enhance efficiency and act as another experiment on the Spark capacity.

### 3.4.1 Simple Moving Average Strategy

The Simple Moving Average Strategy is a traditional and effective method for identifying potential trading opportunities. In our implementation, we will first compute the short-term and long-term moving averages with periods of 10 days and 20 days respectively by taking the average of previous n close price, in which n is equal to the period. Then, a buy signal is generated when the short-term moving average is higher than the long-term moving average and a sell signal is generated vice versa.

Finally, this strategy gives a performance of a total profit of USD 97359.18, the average profit per trade is USD 5408.84 and the Win/Loss ratio of 0.8. The total number of trades is 18, the number of winning trades is 8 while the number of losing trades is 10.

Figure 4 shows the graph of account value (cash + current market value of the Bitcoin) over time, the histogram of profit and loss and the graph of timing of Buy and Sell signals over time.

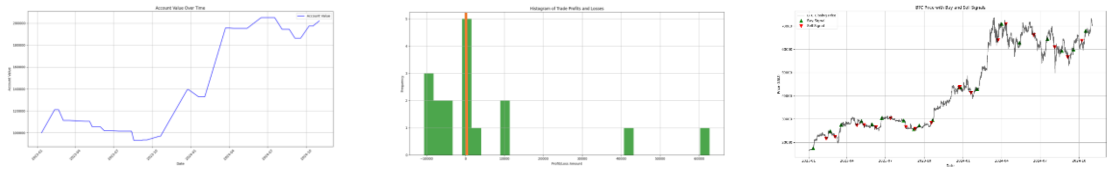


Figure 4: Visual demonstration on the performance of Simple Moving Average Strategy

### 3.4.2 Volume Weighted Average Price Strategy

The Volume Weighted Average Price Strategy is another traditional method that is famous among investors. In our implementation, we will first compute the Volume Weighted Average Price (VWAP) which is the weighted average of the close price with volume as the weights. Then, if the current close price goes above the VWAP, a Buy signal is generated, otherwise, there will be a Sell signal.

As a result, this strategy gives a performance of a total profit of USD 25324.98, the average profit per trade is USD 3165.62 and the Win/Loss ratio of 0.6. The total number of trades is 8, the number of winning trades is 3 while the number of losing trades is 5.

Figure 5 shows the graph of account value over time, the histogram of profit and loss and the graph of timing of Buy and Sell signals over time.

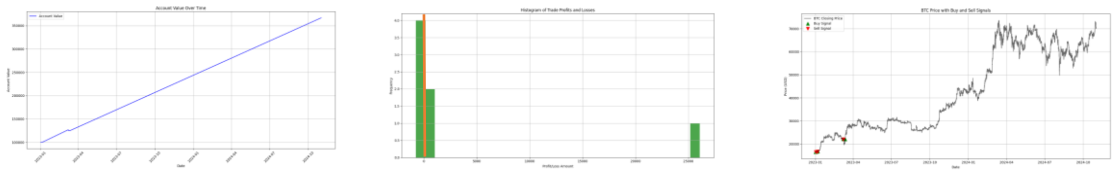


Figure 5: Visual demonstration on the performance of Volume Weighted Average Price Strategy

### 3.4.3 Linear Regression Strategy

The Linear regression Strategy is a simply trading strategy that takes advantage of the prediction results of the trained Linear Regression model. If the prediction is higher than the current close price, it represents a rising trend, and a buy signal is generated. Otherwise, a sell signal is generated.

Hence, this strategy gives a performance of a total profit of USD 94243.64, the average profit per trade is USD 485.79 and the Win/Loss ratio of 1.62. The total number of trades is 194, the number of winning trades is 120 while the number of losing trades is 74.

Figure 6 shows the graph of account value over time, the histogram of profit and loss and the graph of timing of Buy and Sell signals over time.

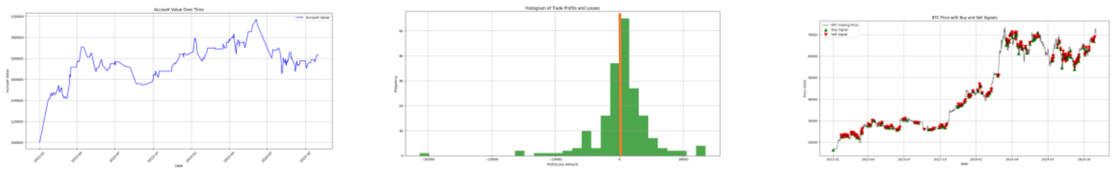


Figure 6: Visual demonstration on the performance of Linear regression Strategy

### 3.4.4 Random Forest Strategy

The Random Forest Strategy is like the Linear Regression strategy. Both strategies make use of the prediction results and apply the same signal generation rules.

Therefore, this strategy gives a performance of a total profit of USD 89070.86, the average profit per trade is USD 315.85 and the Win/Loss ratio of 1.24. The total number of trades is 282, the number of winning trades is 156 while the number of losing trades is 126.

Figure 7 shows the graph of account value over time, the histogram of profit and loss and the graph of timing of Buy and Sell signals over time.

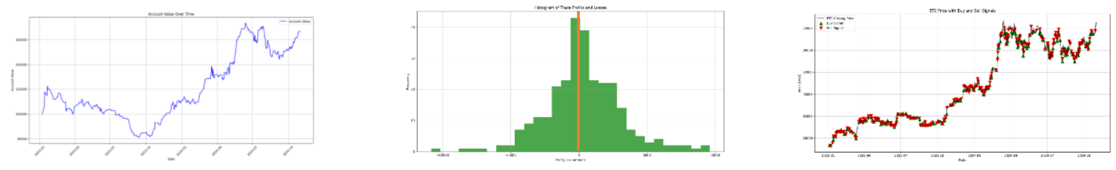


Figure 7: Visual demonstration on the performance of Random Forest Strategy

## 4. Conclusion

### 4.1 Insight From datasets

We first analysis bitcoin historical data using ML approaches like linear regression and random forest, to explore the coefficients between price, features and financial indicators. We found that both linear regression and random forest have some capability in predicting future closing prices. Therefore, we further developed strategies using these two ML models and compared the performances with baseline simple strategies like long-short SMA cross and VWAP strategies using our own back tester. Our strategies outperform VWAP strategy with positive results. Since we are using the last two years as back testing data which has only a general upward trend, in the future dividing data into windows of period would allow better zoom in view of the impact of trends on our strategies. While the dataset is still updating periodically, our work can also be extended to cover more market circumstances and better fine tuning our strategies.

### 4.2 Insight on Distributed Frameworks (Spark)

The implementation of Apache Spark for Bitcoin data analysis demonstrated significant advantages in handling large-scale financial data processing with different time intervals. The distributed framework showed efficiency in parallel processing of computationally intensive tasks such as feature engineering and model training. In our task of processing and back testing thousands of thousand data row, Spark's distributed computing capability enabled faster processing (average of few seconds) compared to traditional single-node approaches (average of few minutes). The distributed framework like spark shows that it is applicable in handling multiple trading strategies and actual implications in the high frequency crypto trading.

**Reference**

[1] Binance. (2024). *Binance Public Data* [Source Code]. GitHub. Retrieved November 29, 2024, from https://github.com/binance/binance-public-data

[2] Steffmann, O. (2024). *Guidance for Digital Assets on AWS* [Source code]. GitHub. Retrieved November 29, 2024, from https://github.com/aws-solutions-library-samples/guidance-for-digital-assets-on-aws

[3] Apache Software Foundation. (2024). *PySpark Documentation*. Retrieved November 29, 2024, from https://spark.apache.org/docs/latest/api/python/index.html