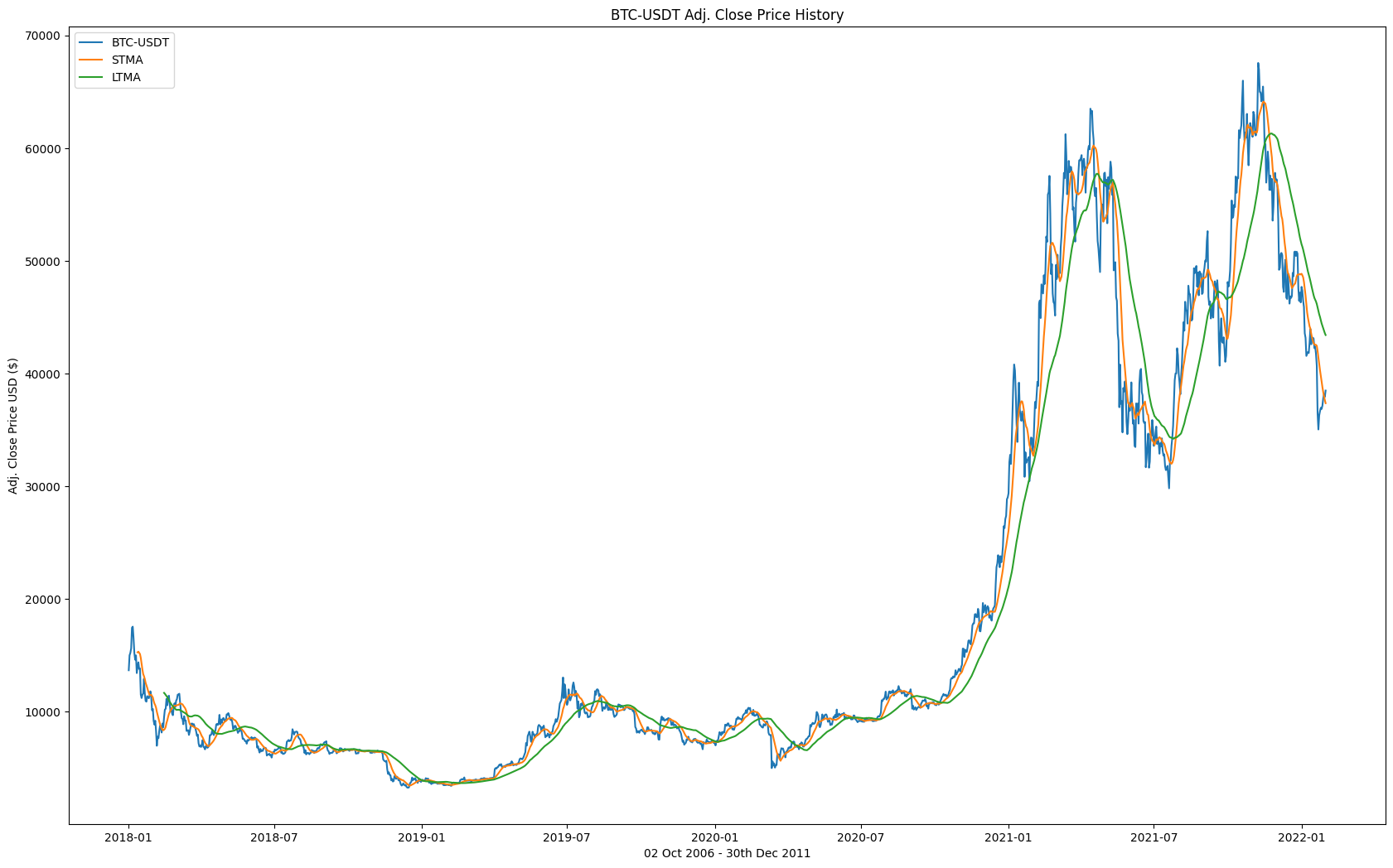
# **IIT-KGP DATA SCIENCE HACKATHON: FINAL REPORT**

Our best solution to the problem statement was divided into three main approaches. In the first, we implemented a purely technical analysis-based strategy to test the efficacy of our later assumptions, which would eventually be the backbone of our final algorithm. We then moved on to machine learning-based approaches while simultaneously trying to implement different technical analysis strategies in tandem with the ML algorithm.

## **Approach 1: Golden Crossover Strategy**

### Approach Overview:

A classic technical analysis approach using simple moving averages (SMAs) to generate buy and sell signals based on the crossover of short-term and long-term SMAs. The time periods for the short and long-term moving averages were 12 days and 45 days respectively. A bullish signal was given by the upward crossing of the 12-day moving average over the 45-day moving average, and a bearish signal was given by the opposite crossover.



### What Was Good:

Simplicity: The strategy was straightforward and easily implementable. The computational resources dedicated to this approach were significantly lower compared to both the other approaches.

### What Was Bad:

Lagging Indicator: With the SMA being a lagging indicator, there were several cases of delayed signals, causing missed opportunities or late exits from trades.

Whipsaws: In volatile markets, SMAs can generate false signals due to short-term price fluctuations, leading to whipsaws (buy/sell signals quickly reversing).

Lack of risk management: Our implementation of the algorithm eventually did not end up having a reliable risk-management system. When an ATR-based stoploss was implemented, it adversely affected the returns.

### What We Learned:

Importance of Confirmation: Using additional indicators or confirmation signals alongside SMAs helped reduce false signals and improve the strategy's accuracy.

Adaptability: SMA strategies have not historically performed up to mark in certain markets and tend to have niche markets where they show periods of outperformance.

### Insights into the Market:

Trend Identification: While the model’s returns were unsatisfactory, the entry points designated by the algorithm turned out to be hotspots of profitability if not for premature or delayed exits.



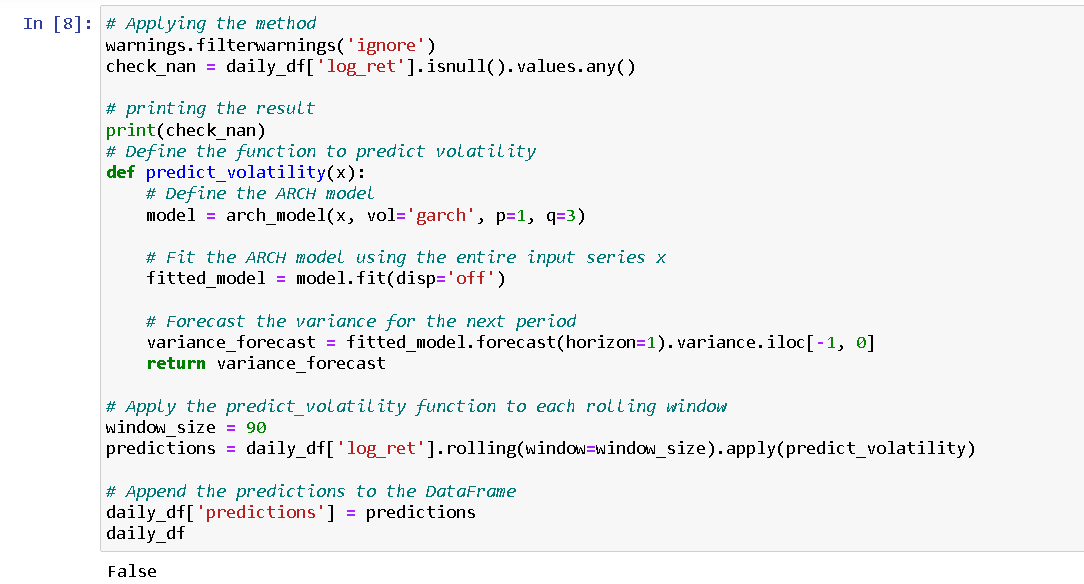
Volatility Consideration: Whipsaws in the strategy indicate that market volatility can significantly impact its performance.

Need for Dynamic Strategies: While the strategy showed small periods of acceptable performance during the four-year period, it lacked adaptability to newer market features in later years as the price of Bitcoin increased.

## **Approach 2: GARCH Unsupervised Learning Model**

### Approach Overview:

Learning from the shortcomings of our previous approach, and recognising the need to take volatility into account when making trade decisions, we decided to make use of a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model - a statistical method used to analyse and forecast volatility in financial markets. Unlike the golden crossover strategy, which focuses on price trends, the GARCH model focussed on volatility, and was aimed at managing our risk better. This was used in tandem with a MACD (Moving Average Convergence Divergence)-based strategy, where a bullish signal was given by the MACD line crossing over the signal line (under the zero line) while a bearish signal was given by the MACD line crossing under the signal line (above the zero line).

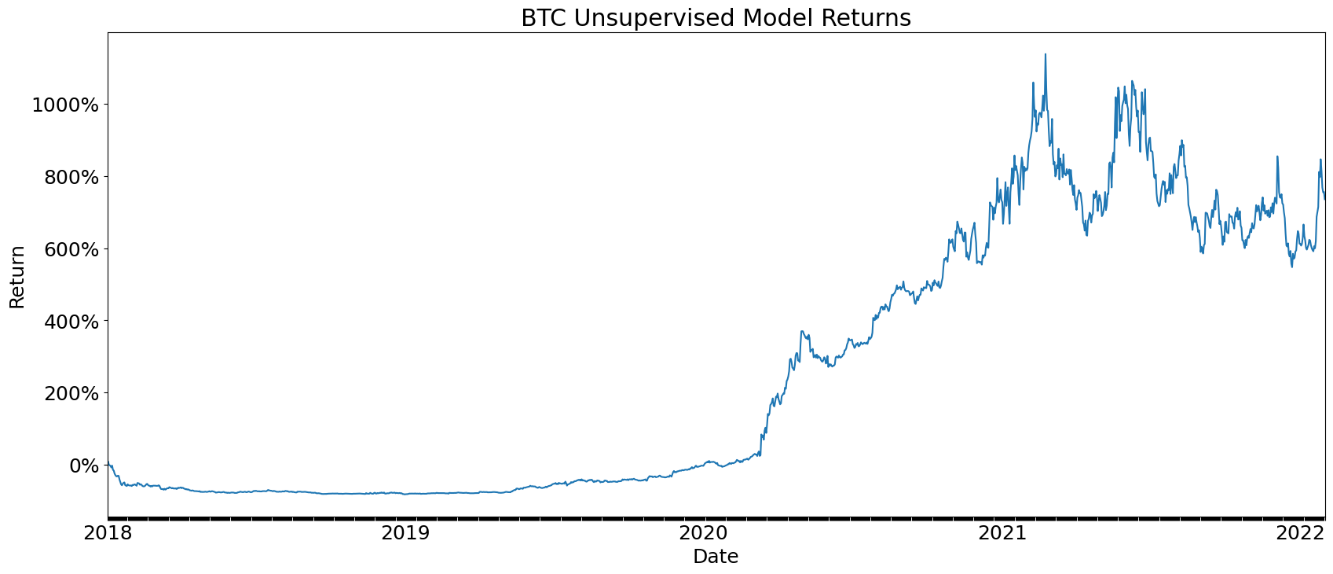


### What Was Good:

Volatility Forecasting: The GARCH model's ability to forecast volatility, proved to be essential for risk management and position sizing.

Statistical Rigor: GARCH is based on statistical principles, making it a robust framework for modelling financial time series data.

Trend Identification: Playing into the strengths of our previous strategy, we implemented a MACD-based signal generation algorithm to ride the trends as best we could.



Increased returns: In comparison to the last model, the returns were significantly better. The return eventually came out to be around 800% over the four years.

### What Was Bad:

Assumption of Normality: GARCH assumes that the underlying data follows a normal distribution, which may not always hold true for financial markets.

Limited Predictive Power: Though the GARCH algorithm could predict volatility with reasonable accuracy, it was not always indicative of price movement.

Over-reliance on trend continuation: Although the MACD indicator has historically shown significant aptitude for trend identification, signal generation based on MACD created some problems in the later stage of the four-year period when trends were more inconsistent.

### What We Learned:

Focus on Risk Management: Just through the implementation of a volatility-based risk management system, the returns skyrocketed.

Consideration of Volatility Clustering: GARCH recognizes that volatility tends to cluster, with periods of high volatility followed by periods of low volatility.

Importance of handling consolidation: The major reason for the (relative) lack of success of the MACD-based indicator was the periods of consolidation in the market which we were unable to fully take advantage of.

### Insights into the Market:

Volatility Dynamics: The GARCH model highlighted the dynamic nature of volatility in the BTC/USDT market.

Risk Awareness: By incorporating volatility forecasts from GARCH, we were able to make more informed decisions regarding position sizing and risk exposure.

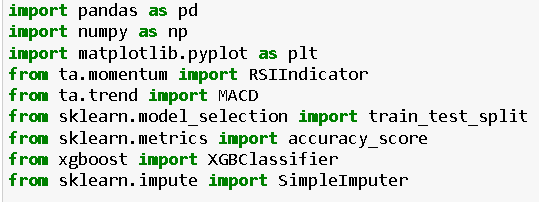
Trend formation: The appearance of the same external drivers for trend-based price action that occurred in the past are not always indicative of similar behaviour in the future.

## **Approach 3: XGBoost Model + SMA + RSI**

### Approach Overview:

For our final approach, taking into consideration everything we had learned from our past attempts, we settled on using the XGBoost model as a classifier, to check the outperformance potential of our planned trades. The feature we focussed the model training on was the Relative Strength Index (RSI), calculated over a 14-day period. The XGBoost algorithm is faster, runs with simpler computations and easily handles larger datasets, making it our ideal algorithm.

We used the XGBoost algorithm to classify trading signals into positions. The trading signals themselves were generated by a moving average crossover strategy, with a bullish signal being the crossing of the short-term moving average over the long-term moving average and a bearish signal being the reverse. The time period we selected for the moving averages were 24 days and 59 days for the short and long-term moving averages respectively. Decreasing the test size of the training data we fed into our model enabled us to maintain our higher returns while simultaneously decreasing both the number of estimators in the tuning of hyperparameters of our XGBoost model.



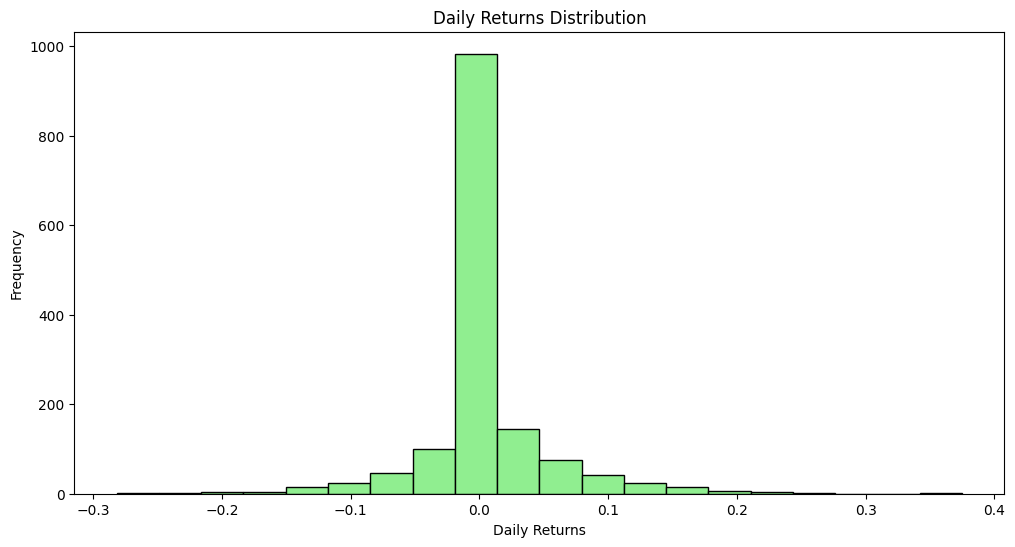
### What Was Good:

Predictive Accuracy: XGBoost proved to be highly accurate in making predictions and classifying the trading signals.

Feature Importance: XGBoost provides insights into feature importance, helping identify which factors are most influential in predicting price movements. Iin our case, RSI turned out to be a major turning point in our strategy. Initially, the features we focussed on were the moving averages themselves, but when we moved to momentum-based indicators like RSI, our returns began to skyrocket.

Flexibility: XGBoost was less prone to overfitting compared to other algorithms, as was demonstrated by its increased outperformance when tasked with generating returns using outlier data not from the specified time period (February 2022 data).

Performance Metrics: The model delivered satisfactory performance metrics across the board (Sharpe ratio, Sortino ratio, etc.).



### What Was Bad:

Complexity: Like other machine learning algorithms, XGBoost was more complex to understand and implement, requiring expertise in both machine learning and finance.

Data Requirements: XGBoost's performance is heavily dependent on the quality and relevance of the input data, which can be a challenge in financial markets. We circumvented this somewhat using sklearn’s SimpleImputer to present finer training data to the model.

### What We Learned:

Machine Learning in Trading: The use of XGBoost highlights the increasing role of machine learning in trading strategies, especially in handling large datasets and complex patterns.

Feature Engineering: The importance of feature engineering in generating meaningful input for machine learning models, as the quality of features significantly impacts model performance (again, thank you, sklearn).

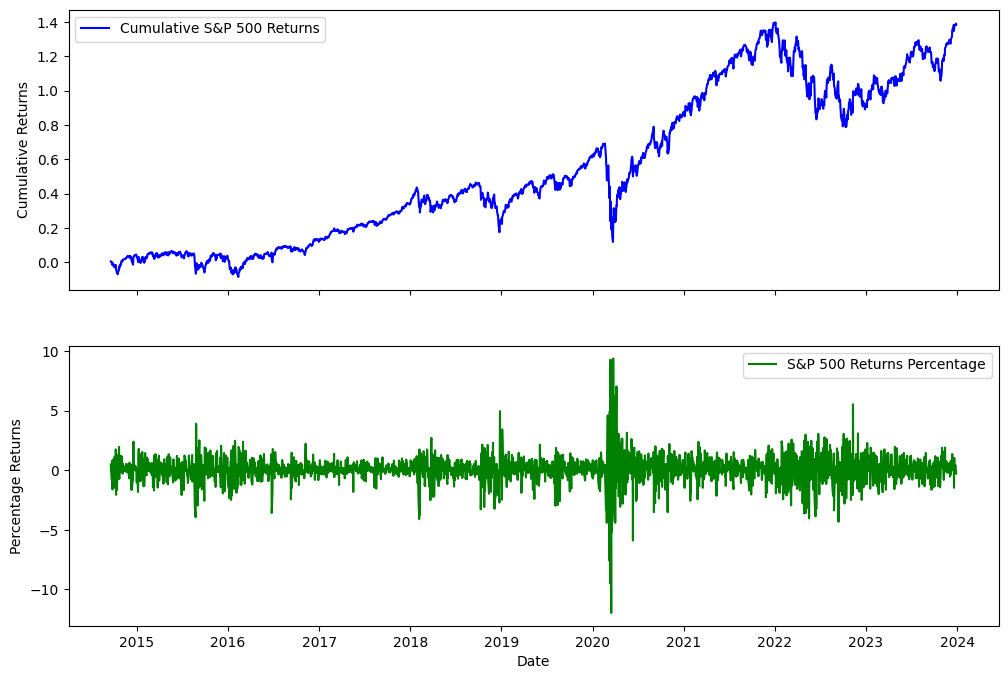
Performance Evaluation: The use of performance metrics like Sharpe ratio and drawdown to evaluate trading strategies provided a quantitative framework for assessing model effectiveness.

### Insights into the Market:

Complex Relationships: The improved performance the XGBoost model delivered after increasing the number of estimators (in its hyperparameters) suggests the dependence of Bitcoin’s price in a wide range of factors in the market.

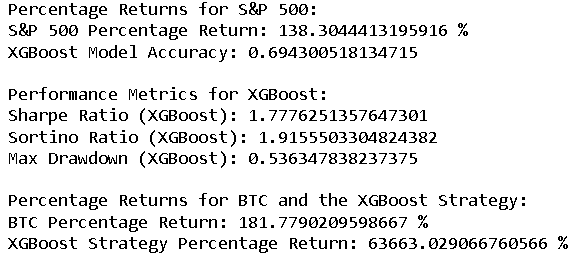
Risk-Return Trade-off: The model's performance metrics offer insights into the risk-return trade-off associated with the trading strategy, providing valuable information for risk management.

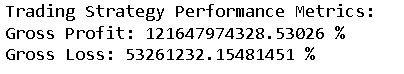
BTC and Momentum: As mentioned earlier, the introduction of momentum-based technical indicators into our overarching strategy resulted in huge success, indicating the susceptibility of Bitcoin’s price to changes in momentum.

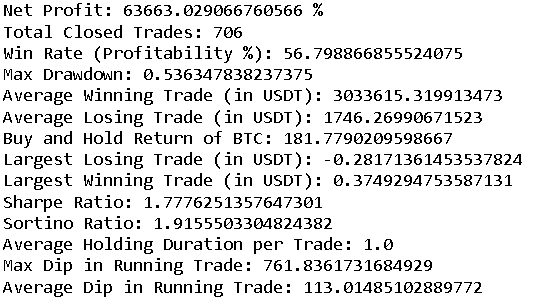


## **THE RESULTS.**

The final metrics. The conclusion of this saga. The commencement of this trilogy. Are we, in fact, the next Medallion Fund? Or did we, like many others, crash and burn whilst trying to find our way in the deep, dark forest of cryptocurrency?



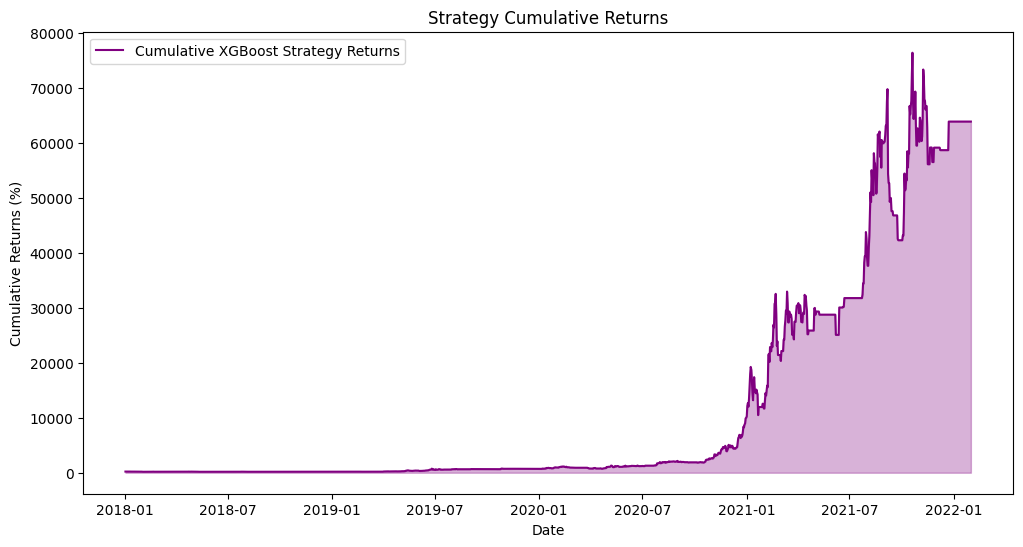




NOTE: the cumulative profit was calculated assuming NO LOSSES during any time in the trading period, allowing the cumulative effect of continuous positive returns to add up very, very quickly. This resulted in the exponentially large number that is in no way reflective of the actual realised profit, or even the **gross profit**. The same goes for the **gross loss**. However, being given no amount for initial investment, and trying to not lose generality, we stuck to our approach of dealing in only percentages. Consequently, the **average winning trade** and **average losing trade** metrics are off as well. (The **net profit** is still very accurate.)

Let’s do a little comparison here.

XGB STRATEGY PERCENTAGE RETURNS: 63663.029066760566 %



If we invested $100,000 in the S&P500 in 2018, we would have $238,000 at the end of the investment period.

Jim Simons, with his impossible return of 66% year over year, would have $759,333.

And with this algorithm, we would have $63,663,029.