Rule-based Algorithmic Trading using a Genetic Algorithm and Machine Learning Signals for the Cryptocurrency Market

Dylan Vassallo
dylan.vassallo.18@um.edu.mt

I. Abstract

Cryptocurrencies have attracted a lot of attention in recent years and the use of AI/ML models to assist or perform automated trading is on the rise. In this study, we use the approach of training three different classification models in particular XGBoost, Random Forest and Logistic Regression using technical indicators as features to test the inefficiencies of the cryptocurrency market to generate risk-adjusted returns. Furthermore, we try to combine these models into one trading strategy using a Genetic Algorithm. We analysed 15 minute data for the following pairs: BTC/USDT, ETH/USDT, XRP/USDT, LTC/USDT, EOS/USDT and XLM/USDT. Our results show that, AI/ML models can outperform simple strategies in particular the buy and hold strategy.

II. Introduction

Predicting markets can be quite difficult as markets can efficiently aggregate new information as stated in the Efficient market hypothesis. However, there are various studies which show that the market can be inefficient at times and this can be exploited to generate abnormal returns. Technical analysis is one of the methods used to try and test these inefficiencies. Numerous studies have investigated the profitability of using technical indicators, for instance [1] reported that technical analysis is more important to Foreign Exchange traders than fundamental analysis, as they make more profit when utilising such indicators. [2] showed that the profitability of the simple moving average technical trading rules have declined over time in the FX market. In their study they showed how this method was quite profitable in the 1970s and steadily decreased over time, and by the 1990s the profitability of using popular technical indicators was essentially 0. Another study [3] tested the SMA and RSI on the Singapore Stock Market. They reported that both indicators are profitable and also noted that the moving averages produces better results than the RSI indicator. There are mixed results on the profitability of technical indicators, where different strategies are back tested out of the thousand of strategies that exist. It is also worth noting that technical analysis may be more profitable in a certain market than another (or not at all), for example from prior research technical analysis is widely used in the FX market.

Another approach to exploit the inefficiencies of financial markets is to use AI/ML models. For instance, [4] made use of Reinforcement Learning to develop an automated FX trading system. In their study they reported that their method was able to make consistent gains and limiting large drawdowns when evaluated on the test dataset. [5] used Generative Adversarial Network with the Multi-Layer Perceptron (MLP) as the discriminator and the Long Short-Term Memory (LSTM) as the generator to forecast stock market prices. They reported that their model gave promising results when evaluated on the S&P500 daily dataset, when compared to other machine learning models.

A. Objectives

The main objectives of this study are as follows:

- Use technical indicators as features to train ML classifiers to predict an up or down move in the price (XGBoost, Random Forest and Logistic Regression)
- Evaluate the classifiers in terms of Accuracy and F1 score
- Apply a meta-heuristic optimisation (optimise on Sharpe Ratio) to find optimal trading rules when combining signals from the trained classifiers (Genetic Algorithm)
- Asses performance in terms of P/L and Annualised Sharpe ratio by running a trading simulation for all the proposed models by taking trading positions based on the predicted outcome
- Compare the proposed models against various other baseline models such as the buy and hold and standard technical indicator strategies (same indicators which were used as features)

With the popularity of cryptocurrencies in recent years, the growth in this market have been increasing exponentially [6], with the total market cap for all cryptocurrencies reaching 800 billion dollars in January 2018. With high returns and volatile prices we believe that using AI/ML models can aid in making good returns while taking less risk. In our study we tested out the following cryptocurrencies: Bitcoin [7], Ethereum [8], Ripple [9], Litecoin [10], EOS [11] and Stellar [12]. We decided to conduct our tests on major cryptocurrencies and since the specified cryptos are within the top ten cryptocurrencies ranked by market capitalisation and liquidity, we opted to select these assets out of the 2,286 cryptocurrencies (according to CoinMarketCap [13] on 1 June 2019). All the data used in this study is at 15-minute interval as we are interested to investigate intra-day trading strategies.

As stated our approach will make use of three different classifiers in particular XGBoost, Random Forest and Logistic Regression. All of which will try to predict a binary outcome, which is whether the price will go up or down. Based on these outcome we will take a long or short position. The reason for choosing classifiers rather than regressors is that they seem to obtain good results in prior research, even the models themselves were chosen based on their performance in prior research. Furthermore, to the best of our knowledge the XGBoost as a classifier has never been tested in the context of cryptocurrencies. We also intend to use a Genetic Algorithm to combine the signals generated by these models and find optimal trading rules based by optimising the Sharpe Ratio. From prior research this method seem to obtain good results when combining this method with standard technical indicators. One of the reasons for applying this technique with the ML signals, is that to the best of our knowledge there is no prior research investigated this approach in the context of cryptocurrencies.

Tree-based ensemble models were employed by various researchers to test the profitability of these models in the context of the cryptocurrency market. [14] utilised a Random Forest and LightGBM model to classify price movements using daily prices and obtained quite promising accuracy scores. [15] also used tree-based models, however an XGBoost regressor was utilised instead of a classification model and they tested their model using a total of 1,681 cryptocurrencies daily prices. [16] made use of Logistic Regression and Random forest to predict price movements on 10 second and 10 minute data. From their evaluation they obtained reasonable results, however their model was only evaluated on sensitivity (TPR), specificity (TNR), precision (PPV) and accuracy (ACC) rather than using more suited metrics for a trading strategy such as the Sharpe Ratio or P/L.

[17] combined an SVM model with a GA to find optimal trading strategies based on technical indicators. They used the SVM model to classify if the market is in bullish trend, bearish trend or no trend at all. Then the GA was employed to find optimal trading rules for each classified market state. [18] utilised a fuzzy logic system with GA to find optimal fuzzy rules when tested against different cryptocurrencies. Their intention for this study was to provide interpretable trading rules which are optimised on both the Sharpe ratio or the geometric mean.

From prior research tree-based ensemble models seems to obtain good results when evaluated as a trading strategies. However, we see a potential gap to investigate these models as if they were ”higher order indicators” meaning that these models will output a signal (based on their predicted outcome) and by combining these signals we take a trading position. Since the search space for all the possible rule-based combinations is quite high, we decided to investigate the use of Genetic Algorithm to find optimal trading rules based on the signal of each classifier. The classifier will be trained using technical indicators as features. Most of the work done in prior research evaluate each classifier as trading strategy on its own, and very little work has been done to combine these models in the context of the crypotcurrency market.

The main contributions of our work are:

- Analysing the performance of our proposed models in terms of P/L and Sharpe Ratio and how they compare against standard technical indicator strategies
- Investigating the use of Genetic Algorithm to combine the machine learning models into one trading strategy
- Analysing the performance in terms of P/L and Sharpe Ratio of the combined machine learning models strategy against using the individual models

III. Literature Review

In this section we will be discussing previous work done in algorithmic trading with respect to cryptocurrencies. We will highlight various AI/ML models which were previously investigated in this domain and provide critical analysis for this work. Some work done in other financial markets is also discussed in this section as it was deemed valuable for our study.

A. Forecasting Cryptocurrency Prices

Several studies have attempted to predict prices or classify price movements for the cryptocurrency market. For instance [14] utilised Random Forest (tree-based ensemble model), LightGBM (tree-based ensemble model) and SVM to forecast the price trend (falling, or not falling) for various cryptocurrencies. In their study they evaluated 42 different cryptocurrencies using daily prices and accuracy was used to measure the effectiveness for each model. As for features they used macroeconomic variables such as stock index, exchange rates, oil prices and so on, as there is prior research [19] that Bitcoin volatility is correlated to such variables. The study was intended to evaluate the accuracy when forecasting the price trend (classification) so no trading simulation or strategy was applied here. They evaluated the models on two different forecasting periods, whether the price will be falling/not falling in the next 2 weeks and the next 2 days. From their results they noted that their models are more suited to forecast medium term trends (2-week), with the highest accuracy score of 0.6 obtained by both the SVM and LightGBM models.

Similarly, to the previous study discussed [14], [15] also made use of a tree-based ensemble model however the proposed model was intended to predict the price itself (regressor rather than a classifier). In their study they utilised XGBoost (tree based ensemble method) and Long-Short-Term Memory (LSTM) recurrent neural network to forecast daily prices for 1,681 cryptocurrencies. Contrary to the previous study [14] the intention for this study was evaluate the models as trading strategies thus a trading simulation was used for evaluation. In their trading simulation they considered a trading fee on any position taken by their trading strategies and they even tested the models on various ranges of transaction fees (0.1%, 0.2%, 0.3%, 0.5%, 1%). In all the models they built investment portfolios based on their prediction and then these were evaluated using return on investment (ROI). They optimised the portfolio (number of currencies to include in a portfolio) using both the Sharpe ratio optimisation and the geometric mean optimisation. The best performing model in this study was the LSTM model.

B. Classification Models

[16] made use of the following classification models, Binomial Logistic Regression, Support Vector Machines (SVM) and Random Forest to try and capture/predict Bitcoin's price movement. They utilised two different datasets, the Bitcoin price data (10 second and 10 minute) and daily Bitcoin price data with an additional 26 features about the Bitcoin network and market. Some of the features in the second dataset included block size (network), hash rate (network), difficulty (network), trade volume (market) and market capitalisation (market). Out of the 26 features they manually selected 16 features which they deemed to be important from prior research. To evaluate their models they used sensitivity (TPR), specificity (TNR), precision (PPV) and accuracy (ACC). For the daily prices all the models were tested and the best performing model in terms of accuracy was the Logistic Regression (0.9879), followed by the Random Forest (0.9498) and lastly the SVM (0.2716). For the 10 second only the Logistic Regression was evaluated and achieved a poor accuracy score of 0.085. However the Logistic Regression obtained an accuracy of 0.539 on the 10 minute dataset, while the Random Forest obtained an accuracy score of 0.574. The SVM was not tested on these time intervals. The results obtained in this study [16] look promising in terms of classification performance however they did not provide any performance metrics for the trading strategy itself, for example cumulative returns or risk adjusted metrics such as Sharpe ratio or Sortino Ratio. Moreover, there is no explanation for how the trading strategy/simulation was evaluated and it seems that they just applied traditional evaluation methods for classification problems. If this was not the case there is no mention for the trading simulation or any other specifics (whether a transaction fee was applied with every trade). Our final thoughts is that the models tested seem to achieve good results in terms of predicting the price movement especially in daily movements however there is no real automated trading back testing being done even though their paper is coined as *'Automated Bitcoin Trading via Machine Learning Algorithms'*.

[20] implemented a decision tree algorithm with cross entropy impurity function to predict the range of the next day returns for Bitcoin. Unlike the previous study [16] they made use of technical indicators as features and applied an appropriate trading strategy back testing. A total of 124 indicators were used as features and some of the indicators included Exponential Moving Average, Simple Moving Average (SMA3, SMA5, SMA10, SMA20), Normalised Average True Range, Relative Strength Index (RSI5, RSI10, RSI14) and Williams %R. One interesting aspect in this research is that instead of trying to predict whether the price will move in an upward or downward direction, they tried to predict the range of the next day return. More specifically they constructed 21 different return intervals (categories), for example the next day return would be between -0.2% and 0.2%. In their trading simulation they considered both long and short positions. The strategy used in their proposed model is that if the decision tree predicts a negative range, they would short the asset and if the range is positive, they would take a long position. In the training phase the win and loss ratio for each range predicted was noted. Then in the testing phase the trading strategy would only take a position on the predicated range which had win ratios significantly higher than their loss ratio (signals from eight ranges) in the training phase. In their evaluation they used various performance metrics such as annualized return, win ratio, loss ratio win/loss ratio and annualized volatility. They claimed that their model outperformed the Buy and Hold strategy and managed to obtain an average annualised return of 842.024% (simple returns). The annualised results are impressive however in their trading simulation they ignored transaction costs which may have had a huge impact on the result.

C. Finding Optimal Trading Rules using AI/ML

There are various studies [21], [22], [18], [17], [23] which make use of AI/ML to find attractive trading rules which optimize the objective function (e.g. Sharpe Ratio, Profit or Accuracy). For instance, [18] made used of a Genetic Algorithm in combination with Mamdani-type fuzzy rule-based systems (FRBS), commonly known as genetic fuzzy systems (GFSs) to optimize/maximize the accuracy of the trading strategy by finding optimal fuzzy rules when testing on 15 different cryptocurrencies. Their model utilised 15-minute data and considered a transaction fee on every trade carried out. A constraint was also applied to the number of rules to be used, as one of the objectives was to provide interpretability hence the use of Fuzzy systems. Short positions were not considered in the trading strategy. It is worth noting that it would have been better to optimise on a risk adjusted metric rather than accuracy, even more when they suggested that their model could be used to aid traders, since it is quite interpretable, however most traders take into account risk when taking a trading position.

[17] combined support vector machines and genetic algorithms to generate technical rules in the forex market. They managed to obtain a return on investment of 83% when testing on the EUR/USD currency pair. The way they achieved this is by training the SVM model to identify/classify whether the market is a Bullish Market, Bearish Market or Sideway (No Trend) using technical indicators and prices as features. Once the model was trained, they employed the Genetic algorithm to optimize trading rules based on technical indicators and this was done for the three types of market states. Both short and long positions were considered in their strategy. They also considered a transaction cost of 0.02% in their trading simulation. Their results look very promising however it was only tested on one currency pair and this might only be efficient when dealing with EUR/USD only. Furthermore, the trading rules were optimised using the return on investment as a fitness function rather than a risk adjusted metric which would have tried to find optimal rules which are more robust to risk.

A different approach from the other two studies [18], [17], was implement by [23] were they used Genetic Programming to find optimal trading rules in the Cryptocurrency market rather than Genetic Algorithms.

D. Proposed Models

From the research discussed, we decided to implement the following machine learning models in our strategy XGBoost and Random Forest. These models were selected as tree-based ensemble models seems to obtain promising results when tested in the Cryptocurrency market [14], [15], [20], [16]. Apart from XGBoost and Random Forest, we also decided to test the Logistic Regression model too as it is a model which is completely different from the other two (GLM rather than tree-based) and according to the study conducted by [16] it seems to obtain comparable results when compared to Random Forest on 10 minute data. Although the tests conducted in this study [16] were evaluated on traditional machine learning metrics such as accuracy and precision rather metrics more suited to trading such as return on investment. The models selected will all be utilised as classifiers (RF and XGBoost can also be implemented as regressors) and we will also be testing the potential of XGBoost as a classifier in the context of trading cryptocurrency assets as up to our knowledge there is no prior research on the use of XGBoost as a classification model for trading cryptos. As for the time frequency we decided to go for 15-minute data as most of the studies conducted are on daily prices and from the research discussed some of the models selected seem to perform quite good on 10 minute and 15-minute data [16], [18]. Moreover, we decided to employ a Genetic Algorithm as a meta-heuristic optimisation to find optimal trading rules similarly to the studies conducted by [18] and [17]. The GA will optimise by trying to maximize the Sharpe ratio as it was properly done by [18], however we would be using the signals outputted by the machine learning models rather than combining technical indicators signals which was investigated by [17] in the context of the FX market. To our knowledge this way of finding optimal rules using a Genetic Algorithm based on the signals generated by the machine learning model is the first time being investigated in the context of the cryptocurrency market and this would be one of the main contributions of our research. All of the machine learning models will use readings from technical indicators as features and a detailed description and rationale for the chosen technical indicators is discussed in Section IV-B.

IV. Methodology

In this section we will provide a detailed description for our proposed model, baseline models, datasets, configurations and parameters, the evaluation method including the performance metrics used and finally a description for the trading algorithm/simulation.

A. Methodology Overview

The methodology consists of these different phases: split the dataset to three different parts (train/validation/test), use brute force to find the optimal parameters for each of the technical indicators using the in sample dataset, train the machine learning models using the technical indicators (with the optimal parameters found) as features on the training set, use brute force to find optimal hyperparameters for the ML models using the validation set, combine the signals generated from the ML models by using a meta-heuristic optimisation approach to find optimal trading rules (Genetic algorithm) and finally evaluate all the models including the Buy and Hold strategy on the test/out of sample dataset.

B. Models Description

1) Baseline Models: The first and simplest strategy which was used as a null/baseline model is the buy and hold strategy. Basically, this strategy takes a buy position at t_0 and holds the asset without taking any other trading positions hence the name buy and hold.

Apart from the buy and hold strategy we made use of the following 5 technical indicators: Exponential Moving Average (EMA), Simple Moving Average (SMA), Relative Strength Index (RSI), Williams Percent Range (Williams %R) and the Normalised Average True Range (NATR). All of which were used as features in the study conducted by [20]. Both the EMA and SMA are trend indicators, meaning that they try to capture a trend in some variable which is typically the closing price or the adjusted closing price. The RSI and the Williams %R are a type of momentum indicators which try to capture the rate of change, which is typically used to show the rate of change in prices. On the other hand, the NATR is a volatility indicator which tries to capture volatility over time. Typically, the trend and momentum indicators give us a sense of direction for where the price is heading, while the volatility indicator captures the spread of the price rather than the direction.

We used each indicator on its own as baseline models. Each indicator outputs a signal based on a given rule and then we act upon that signal by taking a trading position. Furthermore, we also combined and tested all indicators together as a single strategy. We defined the rules for any given strategy that make use of the standard/vanilla technical indicators (only) based on common strategies used in technical analysis. Further details for the trading strategies will be discussed in Section IV-E. From now on any formulas described will be in the context of our models, so for example in Formula 2 we show how to compute the SMA for the closing price, but this does not mean that SMA can only be computed for the closing price, in fact you can apply SMA to any of the OHLC variables.

The following are the baseline models which make use of technical indicators only:

- **EMA**
  The exponential moving average can be computed using Formula 1. Unlike the SMA this moving average gives higher weighting to recent prices. A common strategy used in this indicator is the crossover strategy, where a trader uses two EMA readings one with a short window and the other with a long window. When the short window crosses above the long window a long position is taken, and when the short window goes below the long window, a short position is taken.
- **SMA**
  The simple moving average indicator can be computed using Formula 2. Like the EMA indicator this indicator also smooths out the price and makes it easier to identify

$$a = \frac{2}{n+1}$$
$$ema_t = (1-a)ema_{t-1} + (a)in_t \quad (1)$$

1: The EMA takes n previous periods as a parameter. In first line the scaling factor is computed a. In the second line the ema_t is calculated for each data point in the window specified.

trends. The same crossover strategy as the EMA is employed when using this indicator which is go long when the short window SMA moves above the long window SMA and go short when the short window SMA goes below the long window SMA. Unlike the EMA this moving average gives equal weighting so no importance is given to the recent prices, however since the SMA will react slower than the EMA it may not exit a position due to a sudden move which may have resulted in bigger profits if we held the position, likewise this may be a disadvantage as due to reacting slowly we make a bigger loss. This is one of the reasons for testing both moving average indicators, furthermore this indicator is usually used as a null model when evaluating trading strategies [24], [25].

$$SMA_i = \frac{\sum_{k=i-n}^{i} close\ price_k}{n} \quad (2)$$

2: SMA: An unweighted mean for the previous n closing prices.

- **RSI**
  The relative strength index can be computed using Formula 3. This momentum oscillator gives a reading between 0 and 100. A traditional trading strategy used with this indicator is to go short when the asset is overbought (70) or go long when the asset is oversold (30).

$$up_t = \begin{cases} in_t - in_{t-1} & \text{if } in_t > in_{t-1} \\ 0 & \text{else} \end{cases}$$
$$down_t = \begin{cases} in_{t-1} - in_t & \text{if } in_t in_{t-1} \\ 0 & \text{else} \end{cases}$$
$$sup_t = \frac{n-1}{n} sup_{t-1} + \frac{1}{n} up_t \quad (3)$$
$$sdown_t = \frac{n-1}{n} sdown_{t-1} + \frac{1}{n} down_t$$
$$rsi_t = 100 - \frac{100}{1 + \frac{sup_t}{sdown_t}}$$

3: The RSI takes n periods as a parameter.

- **Williams%R**
  %R is calculated using Formula 4. This momentum oscillator technical indicator finds the relation of the close price with the highs and lows of the previous n days (window). The reading from this indicator is between 0 and -100, and a common strategy used with this indicator is to take a long position when the reading is below -80 which is referred to as oversold or take a short position when above -20 which is referred to as overbought, similarly to the RSI strategy. This indicator was also used by [22] and [21] when trying to find optimal trading rules when utilising evolutionary algorithms such as Genetic Algorithms.

$$\%R = \frac{high_{Ndays} - close_{today}}{high_{Ndays} - low_{Ndays}} \times -100 \quad (4)$$

4: Formula used to calculate Williams%R. A window must be specified which is denoted as n.

- **Combined Indicators + NATR** A combination of the above indicators was also used as a baseline model. The strategy/rules to combine each indicator together was specified by our intuition that it might work. When comparing the indicators with our proposed models this should not be given that much importance as the other indicators as there is no supporting literature on the strategy used to combine the indicators together. For future work this could be optimised in the same way we optimised our ML models to make a better assessment and comparison with our proposed model. Moreover, another technical indicator was added in this baseline model which captures volatility. The indicator added is the Normalised True Average Range (NATR) and can be computed using Formula 5. This indicator on its own does not give us a sense of direction but rather the spread of the price (volatility) thus it was used when combining the indicators to give a measure of volatility.

$$tr_t = maximum(high_t - low_t, |high_t - close_{t-1}|, |low_t - close_{t-1}|)$$
$$atr_t = tr_t \cdot \frac{1}{n} + atr_{t-1} \cdot \frac{n-1}{n} \quad (5)$$
$$natr_t = 100 \frac{atr_t}{close_t}$$

5: The first two lines calculate the Average True Return, by first finding the maximum from; high low, absolute high previous t close and absolute low previous t low. After doing so wilders smoothing is applied to yield the AVR at time t as shown in line two. Finally, the value is normalised by applying the last formula.

A more detailed description of the trading strategies used with each technical indicator is given in Section IV-E.

2) Proposed Models:

- **Random Forest**
  The Random Forest model [26] is an ensemble of decision trees and do not assume that the features follow a normal distribution, which may be the case in our proposed features (readings from technical indicators) and unlike the Logistic Regression it is not affected by the magnitude of the features.
  We use the Gini impurity [27] to quantify the uncertainty/purity at each node in a specific tree inside the Forest. Gini impurity at each node can be computed using the Formula 6. The formula takes the classification of a subset of the instances which in our case is an upward or downward direction in the cryptocurrency price and computes the probability of obtaining two different outputs from all the possible classes k. When

the subset of instances only contains the same class, it is pure.

$$I(t) = 1 - \sum_{j=1}^{k} p^2(j|t) \tag{6}$$

6: Gini impurity used to quantify the uncertainty/purity at each node in a specific decision tree.

A set of if statements are tested for each feature to find the condition which give out the highest information gain which is computed using Formula 7, and the subset is then split based on that condition. Information gain is computed by taking the $I(t)$ of the current node and subtracting it by the total number of samples in the right node over the total number of samples in the parent node $\frac{N_{tR}}{N_t}$ multiplied by the right node uncertainty $I(t_R)$ and subtracting again with weighted average of left node uncertainty $\frac{N_{tL}}{N_t} I(t_L)$ based on a specific decision.

$$I_g(t) = I(t) - \frac{N_{tR}}{N_t} I(t_R) - \frac{N_{tL}}{N_t} I(t_L) \tag{7}$$

7: At each node in the decision tree the information gain is checked to find the best split.

This procedure keeps going until a termination condition is met for example the max depth for a specific decision tree is met or until a pure node is reached. Now in Random forest we have a collection of these decision trees each performing this task in parallel, when the termination condition is met the mean predicated class probabilities of the trees is taking as the final classification. We then use that classification to take a long or short position whether the classification is that the price will move in an upward direction or downward direction.

- **XGBoost**
  The XGBoost algorithm [28] a type of gradient boosted trees are a lot similar to the previous discussed tree ensemble however instead of build the trees in parallel, the trees are built one at a time with each tree correcting errors of the previously trained tree. The optimisation function used in this model is called "cv" (cross validation) where it performs cross validation at each boosting iteration to find better performing trees. The model is trained in an additive manner as discussed. XGBoost can also make use of L1 and L2 regularization as it uses gradient descent (Formula 11) when optimising the loss function.
- **Logistic Regression**
  The Logistic Regression [29] model differs from the other two models as it is not a tree-based machine learning model however it also does not assume that the features follow a normal distribution. Similarly, to the other models this classifier was used to predict a binary outcome whether the price will go up which is encoded by 1 and whether the price will do down which is encoded by a 0. A position is then taken based on the predicted value.

In this model the Logistic Sigmoid function is used as a Hypothesis function to map the instances (sets of features at time t), with some given weights denoted as θ, to its estimated probability of the binary outcome (price move up or down) as shown in Formula 8.

$$\begin{aligned} h_\theta &= g(\theta^T x) \\ h_\theta &= \frac{1}{1 + e^{-g(\theta^T x)}} \end{aligned} \tag{8}$$

8: Logistic Sigmoid Function used as a Hypothesis function to map the features from the technical indicators to the estimated probability that the price will move in an upward or downward direction.

A decision boundary on the estimated probability is required to be able to classify the price direction and take a position, and this is done by checking if the estimated probability is greater or equal to some specific threshold and classifies it accordingly. This procedure is shown in Formula 9. It must be noted that the threshold is set to 0.5 in the formula, but this will be tuned/optimised in a later stage based on the Sharpe Ratio.

$$position = \begin{cases} \text{Long Position} & \sigma(\theta^T x) \geq 0.5; \theta^T x \geq 0 \\ \text{Short Position} & \sigma(\theta^T x) < 0.5; \theta^T x < 0 \end{cases} \tag{9}$$

9: This is the function used to take a position from the classified/predicted direction outputted by the Logistic Regression model.

This model is trained using gradient descent and the weights for each feature is adjusted to minimize the cost. The cost function $J(\theta)$ is shown in Formula 10 and the Gradient Descent function is shown in Formula 11. Ridge regularization is also used to penalize complex models as shown in the cost function.

$$\text{J}(\theta) = -\tfrac{1}{m} \sum_{i=1}^{m} y_i \log h_\theta(x_i) + (1 - y_i) \log(1 - h_\theta(x_i)) + \underbrace{\lambda \sum_{j=1}^{n} \theta_j^2}_{\text{L2}} \tag{10}$$

10: Cost function for the Logistic Regression model with Ridge/L2 regularization.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{11}$$

11: Gradient descent function which minimizes cost by adjusting the weights for each feature which in our case are the technical indicator readings.

- **Combining signals using Genetic Algorithm**
  As discussed, we utilised a Genetic Algorithm to combine the classifier and find optimal trading rules. The individuals/chromosomes were encoded using binary encoding. A better representation of the encoding used is shown in Table I.
  So basically we have an array of 8 with ones and zeros, where the first bit states if the rule should negate the

XGBoost	Operator	Random Forest	Operator	Logistic Regression
NEGATE	AND/OR/XOR	NEGATE	AND/OR/XOR	NEGATE
0,1	00,01,10,11	0,1	00,01,10,11	0,1
Example				
0	00	1	01	0
XGBoost	AND	not(RF)	OR	LogReg

TABLE I: The binary encoding used to specify rules in our Genetic Algortithm. The last two rows show an example of a particular encoding formation.

XGBoost, if set to 1 it will negate this signal, the second two bits is the operator joining both the XGBoost and RF where 00 means *'and'* operator, 01 and 11 mean an *'or'* operator and 10 is the *'xor'* operator. The next bit then check whether to negate the RF followed by another two bits for an operator, and the final bit check whether to negate the logistic regression. Apart from the encoding this algorithm is split into these different phases: A new population is created where each individual has its own trading rule, the fit for each individual is computed using the Sharpe Ratio, selection for the parent individual is done using the roulette selection wheel where an individual is more likely to be selected if its Sharpe Ratio is high, create offspring from parents, randomly do crossover (with a certain probability) where the genes from the parent are crossed using one point crossover, mutate the chromosome (with a certain probability) by changing bits from 1 to a 0 or vice versa and replace population with off springs. This procedure continues after some iterations are done. The best trading rule found in the validation set is evaluated in the out of sample set.

3) Data Pre-processing & Features: In this section we will discuss the data pre-processing techniques and features used in the machine learning models. These techniques only apply to the proposed machine learning models, none of the techniques discussed here apply to the baseline models. Every model was trained on the same set of features and these include the technical indicators described in the baseline models Section IV-B1. So, for features we used the following indicators: EMA short window, EMA long window, SMA short window, SMA long window, RSI, Williams%R and NATR. The reason for using these features is to check whether a machine learning model can outperform these vanilla technical indicators when trained on these values as features.

Apart from the features at time t a lagged window for all the indicators was also applied and added as features, basically this will take the n previous readings from the current data point at time t for each of the indicator described, a technique which is quite common when training a machine learning model on financial data. This technique of lagging technical indicators was also employed by [30] and applied a lagging effect to the EMA indicator and used both the EMA at time t and the lagged EMA as features to train a Neural Network to forecast stock market prices. Each indicator together with the lagged readings for that specific indicator are scaled to a value between 0 and 1. So for example if we set a lagged window of 2 the RSI would be scaled relative to its lagged reading, RSI_0, RSI_1, RSI_2, RSI_3, where RSI_0 is the reading at the current data point. The scale function is shown in Formula 12.

$$Scaled\ x_t = \frac{x_t}{\sum_{i=0}^{k} x_{ti}} \tag{12}$$

12: Scaling function used to scale indicators relative to their lagged readings. x_t is the data point which is being scaled, k is the lagged window set, and x_{ti} is the reading at datapoint i. Basically, it divides the reading at time t by the summation of the lagged reading and the reading at the current data point.

This scaling of features relative with their lagged indicators was mainly done for the Logistic Regression as it may decrease the training time when working with small values. Furthermore, since we have two indicators of the same type for the SMA and EMA (short and long window), feature reduction was applied. This was achieved by subtracting the scaled value for the long window with the scaled value of the short window for every i in the lagged window. This is better shown in Formula 13.

$$\{\text{SMA}^{i}_{long} - \text{SMA}^{i}_{Short}, ..., \text{SMA}^{i-k}_{long} - \text{SMA}^{i-k}_{Short}\}^{k}_{i=0} \tag{13}$$

13: Feature reduction for moving averages. This was applied for both the SMA and EMA. In the above formula it is shown only on the SMA. k is the lagged window set, each SMA long window reading is subtracted by the SMA short window at each i.

The reason for doing this is that we are interested in the short and long moving average with respect to each other, so by applying this method we could reduce the readings for the short and long windows to one for each reading in the lagged window including the current data point. The lagged window used for the features and the parameters used for the technical indicators will be explained in Section IV-C.

C. Configurations and Parameters

The parameters used for each technical indicator was set by using brute force as a hyperparameter tuning method on the in-sample dataset to find the most optimal configurations and then these were evaluated on the out of sample dataset. The brute force method was trying to optimise the Sharpe Ratio. The same configuration for each indicator was then used when creating features for the machine learning models.

For both the EMA and SMA, the lead and lagged windows were tuned by trying values in the range of 5 to 50 with a step size of +5 for the lead window and trying values in the range of 20 to 80 with a step size of +5 for the lagged window. A large step size was used to avoid overfitting. The parameters chosen for both the EMA and SMA for each currency pair is shown in Table II.

Model	BTC	ETH	XRP	LTC	EOS	XLM
EMA	(45,75)	(35,75)	(15,35)	(5,50)	(30,35)	(20,35)
SMA	(35,75)	(10,75)	(15,55)	(10,55)	(25,75)	(15,40)

TABLE II: The chosen (lead, lag) windows for both moving average indicators for each currency. Was optimised using the in sample dataset using brute force and optimising on the Sharpe Ratio.

For the momentum indicators (RSI and Williams%R) 3 different variables were optimised using the same method as

the EMA and SMA. The variables which were tuned were the window period, the overbought threshold and the oversold threshold. For the RSI the window period was tested from the range of 2 to 20 with a step size of +4, the overbought threshold was tested from the range of 50 to 90 with a step size of +5, and the oversold threshold was tested between the values of 5 and 55 with a step size of +5. For the Williams%R the same range for the window period was tested, however the overbought threshold was tested from the range of -5 to -50 with a step size of +5, and the oversold threshold was tested from the range of -50 to -95 with a step size of +5. A large step size was selected to avoid overfitting. Again, these were tuned using the in-sample dataset and finally evaluated using the out of sample set. The final chosen parameters are shown in Table III.

Model	BTC	ETH	XRP	LTC	EOS	XLM
RSI	(2,75,15)	(6,55,20)	(14,85,5)	(14,75,25)	(14,90,10)	(18,80,5)
Wills%R	(2,-15,-85)	(2,-5,-90)	(2,-5,-85)	(2,-10,-90)	(2,-5,-90)	(2,-10,-90)

TABLE III: The chosen (window, overbought, oversold) parameters for both momentum indicators for each currency. Was optimised using the in sample dataset using brute force and optimising on the Sharpe Ratio.

In the combined indicators method, we used the best configurations found in the previous two tuning procedures for all the indicators. In this phase will also tuned the NATR indicator using the in-sample dataset and once the best parameter was found it was evaluated on the out of sample set like the previous procedures. For the NATR two variables where tuned which are the window period and the volatility threshold to fire up a signal which is better explained in Section IV-E. The parameters were tuned for BTC, ETH, XRP, LTC, EOS and XLM and the best configurations were (window, threshold): (10, 0.3), (14, 0.8), (14, 0.7), (14, 0.8), (18, 0.8) and (6, 0.2), respectively. Again, these were tuned to optimise the Sharpe Ratio. We also tuned some parameters when training our models which are the number of lagged features to use, maximum tree depth for the RF and XGBoost and the confidence/probability level of the predicted value (take a position if above this confidence level). Obviously for the Logistic Regression there was no need to tune the maximum tree depth. To tune this parameter each model was trained on the train set and validate these tuned parameters on the validation set. Once the best configuration which maximised the Sharpe Ratio when applying brute force was found, the trained models were evaluated on the out of sample set. For all the models lagged features were tested with 2 and 4 as lagged window for the features and this was limited to these two values, so we wont end up with high dimensions. For the maximum depth for the trees we tested 4, 6 and 8, to avoid overfitting. As for the confidence level we tested values between 0.55 and 0.80 with a step size of +0.5, and the reason for limiting it to 0.80 is that we dont want it to be too high a lose opportunities. For all pairs in the XGBoost and RF the best parameters were 4 as the lagged window for the features, 8 for the maximum tree depth and 0.55 for the probability level. For the Logistic Regression model the best parameter are as follow (Coin, Lagged Features, Confidence Level): (BTC, 4, 0.55), (ETH, 4, 0.55), (XRP, 4, 0.65), (LTC, 4, 0.65), (EOS, 2, 0.55) and (XLM, 2, 0.55). The Logsitic Regression also had a maximum iteration size of 1000 for the training phase. For other hyperparameters for the XGBoost we set as the default values which can be viewed from https://xgboost.readthedocs.io/en/latest/python/python_api.html, the default hyperparameters for the Random Forest Classifier can be viewed from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html and for the Logisitc Regression they can be viewed from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

For the Genetic Algorithm we used the following hyperparameters: population size of 30, a mutation rate of 0.08, a crossover rate of 0.75 and a termination condition to terminate after 200 epochs. The reason for setting a low population size and epochs is that this task was computationally heavy to run on our machine.

D. Model Evaluation

1) Performance Metrics: We evaluated both the baseline models and the proposed models using the following metrics which can be grouped by two different types, measures of performance and measures of robustness.

Measures of performance:

- Number of Trades
- Average Trades per Day
- Profit & Loss (cumulative log returns)
- Maximum Gain (maximum cumulative log returns)
- Equity Curve Plot

Measures of robustness (risk-adjusted performance):

- Annualised Sharpe Ratio
- Maximum Drawdown

The annualised Sharpe Ratio is computed using Formula 14. Apart from these metrics the machine learning models were also evaluated using accuracy and F1 score.

$$S_A = \sqrt{96 * 365}\frac{R_m - R_{rf}}{R_\sigma} \tag{14}$$

14: Sharpe Ratio: Annualised using $\sqrt{96 * 365}$, since the market is open 24/7 and we are using 15 minute data. R_m is the mean for the log returns obtained by the model, R_σ is the standard deviation of the log returns obtained by the model and R_{rf} is the risk free rate which we set to 0.

2) Datasets: All the experiments conducted in this study made use of the datasets shown in Table IV. The data was collected from Binance [31] which is a well-known cryptocurrency exchange which provides historical data for various cryptocurrencies traded on its platform. Using the REST Web API provided by this exchange [32], we collected historical prices for the following pairs: *BTC/USDT, ETH/USDT, XRP/USDT, LTC/USDT, EOS/USDT* and *XLM/USDT*. The data was saved as *'.csv'* format prior to the model evaluation phase. All the datasets contain 15-minute data for the following attributes: OHLC, volume, open time and close time. The USDT [33] cryptocurrency was used as a base currency

since we wanted to simulate our trading strategy as if it was running on Binance exchange. Since Binance do not have fiat base currencies for trading, we opted to select a stable coin backed by the US dollar. Furthermore, this coin has the most volume when compared with other stable coins according to CoinMarketCap [13] as from 1 June 2019.

To evaluate both the baseline models and the proposed model we split the data into three parts: training, validation and testing. First the dataset was split into two, the in-sample set (training/validation) and the out of sample set (testing) using an 80:20 split. Then the in-sample set was split to get the training dataset and validation dataset using an 80:20 split. All the models will be evaluated on the test dataset, trained using the training dataset and if any tuning is required, it would be done using the validation dataset.

Furthermore, prior to the experiments we collected descriptive statistics and investigated the underlying distribution for each dataset shown in Table IV. The four moments of distribution (mean, standard deviation, skew, kurtosis), the minimum/maximum, the lower/upper/50% percentiles for the log returns for each dataset are shown in Table V. Moreover, Figure 1 shows the log returns density plots for all the datasets. These plots visualises the distribution of the log returns for each cryptocurrency pair.

E. Trading Simulation

In our trading algorithm/simulation we took into consideration both long and short positions. A hold/sit position is also a possibility, meaning you are currently not in any position and no trade is made. The long position is denoted as 1, the short position is denoted as -1 and the hold/sit position (no position) is denoted as 0. These different types of positions can be better viewed in the Formula 15.

$$position = \begin{cases} 1 & \text{Long Position} \\ 0 & \text{No Position} \\ -1 & \text{Short Position} \end{cases} \tag{15}$$

15

A transaction cost of 0.1% on every trade is also considered. We chosen that specific transaction cost since it is the fee issued on Binance exchange on every trade as of 1 June 2019 [35]. The trading simulation works the same for all the trading strategies tested and it works as follows:

1) Generate features for a given strategy.
2) Take positions based on signals from the generated features for a given strategy.
3) Evaluate strategy and apply transaction fees.

Lets break down the trading simulation, first a trading strategy need to generate its features, for example the SMA will generate both the long and short moving averages in this phase. After the features for every data point is generated the positions are taken based on those features. So, keeping with the SMA example, a long position signal is generated if the short moving average is greater than the short moving average denoted as 1 and vice-versa for a short position signal -1. The final position is taken by adding the short and long position, this part was added as some strategies may have more than one signal and when adding the long signal (1) with the short signal (-1) you would end up with a 0 meaning take no position.

Moreover, at this phase we also check positions where a fee must be applied (entered/exited long/short positions), and this is done by checking if the previous position is different from the current position. This means that if you entered/exited a different position and a trade was made. A special case where a position is entered and exited on the same data point was also taken into consideration and for that case the fee was applied two times, as you will be entering and exiting a position in a 15 minute time interval (one data point). Furthermore, in the evaluation method the log returns are computed for each data point and the profit and loss for each position is calculated by multiplying the log return at time t with the position at $t-1$, also a subtraction from the profit and loss is computed if a fee was issued. Finally, the profit and loss are cumulatively summed up to get the total profit and loss at time t. The last data point evaluated hold the final profit and loss which is the cumulative profit and loss in that data point. Log returns are computed using Formula 16.

$$R_t = \ln(\frac{C_t}{C_{t-1}}) \tag{16}$$

16: Log returns are computed by taking the difference between the natural log for the closing price at time t and subtracting it by the natural log for the closing price at time $t-1$. This will output the log return at time t (R_t).

Each strategy tested uses its own features and takes positions based on those features, in the following list we will describe in detail the features and rules used for each strategy tested:

- **Buy and Hold Strategy [Baseline]**
  As discussed in Section IV-B1, this strategy will take a long position at t_0 and hold the asset without taking no further trading positions.
- **Moving Average Crossover [Baseline]**
  This trading strategy was tested using both the EMA and SMA indicators/models. For both indicators a short and long window was generated using the parameters described in Section IV-C, and this strategy takes a long position if the lead/short moving average is greater than the lag/long moving average and vice-versa for a short position. This moving average crossover strategy is better shown in Formula 17.

$$position = \begin{cases} Long(1), & \text{if } W_{short} > W_{long} \\ Short(-1), & \text{otherwise} \end{cases} \tag{17}$$

17: Moving Average Crossover strategy, p refers to the position, the conditions refer to the signals. The W_{short} refers to the short/lead window in the moving average, on the other hand, the W_{long} refers to the long/lag window in the moving average. Using this strategy you are always in a long or short position (conditions are exhaustive).

- **Momentum Overbought/Oversold [Baseline]**

Dataset	Training		Validation		Testing		All	
	Dates	*Total*	*Dates*	*Total*	*Dates*	*No. Samples*	*Dates*	*Total*
BTC/USDT	17/08/2017 06:00:00 to 9/10/2018 06:59:59	39,881	9/10/2018 07:00:00 to 21/01/2019 13:14:59	9,971	21/01/2019 13:15:00 to 01/06/2019 02:14:59	12,464	17/08/2017 06:00:00 to 01/06/2019 02:14:59	62,316
ETH/USDT	17/08/2017 06:00:00 to 09/10/2018 06:59:59	39,881	09/10/2018 07:00:00 to 21/01/2019 13:14:59	9,971	21/01/2019 13:15:00 to 01/06/2019 02:14:59	12,464	17/08/2017 06:00:00 to 01/06/2019 02:14:59	62,316
XRP/USDT	04/05/2018 10:00:00 to 10/01/2019 16:44:59	24,008	10/01/2019 16:45:00 to 14/03/2019 11:29:59	6,003	14/03/2019 11:30:00 to 01/06/2019 02:14:59	7,503	04/05/2018 10:00:00 to 01/06/2019 02:14:59	37,514
LTC/USDT	13/12/2017 04:30:00 to 21/11/2018 01:29:59	32,650	21/11/2018 01:30:00 to 14/02/2019 02:14:59	8,163	14/02/2019 02:15:00 to 01/06/2019 02:14:59	10,204	13/12/2017 04:30:00 to 01/06/2019 02:14:59	51,017
EOS/USDT	28/05/2018 07:00:00 to 19/01/2019 06:59:59	22,541	19/01/2019 07:00:00 to 19/03/2019 05:59:59	5,636	19/03/2019 06:00:00 to 01/06/2019 12:14:59	7,045	28/05/2018 07:00:00 to 01/06/2019 02:14:59	35,222
XLM/USDT	31/05/2018 11:30:00 to 20/01/2019 10:29:59	22,345	20/01/2019 10:30:00 to 19/03/2019 21:14:59	5,587	19/03/2019 21:15:00 to 01/06/2019 02:14:59	6,984	31/05/2018 11:30:00 to 01/06/2019 12:14:59	34,916

TABLE IV: A detailed description for all the datasets. Each dataset was split into training, validation and testing. The dataset was first split in sample (training/validation) and out sample (testing) using a 80:20 split and then the in sample dataset was split 80:20 ratio to get the training and validation datasets. The range for all the datasets is shown in this table, with the following date format *'dd/mm/yyyy HH:MM:SS'*. The total number of samples is also shown in the *'Total'* column. The last column *'All'* is the description for all the datasets combined. All the datasets contain historical prices (OHLC, volume, open time and close time) and the time interval used is 15 minutes.

Dataset	Mean	Std.	Skew	Kurtosis	Min.	Max.	25%	50%	75%
BTC/USDT	0.000011	0.005573	0.045194	21.390842	-0.078635	0.090803	-0.001634	0.000022	0.001743
ETH/USDT	-0.000002	0.006652	0.009216	25.606661	-0.126609	0.138168	-0.002244	0.000000	0.002315
XRP/USDT	-0.000020	0.005486	0.278156	31.207233	-0.106577	0.097354	-0.001931	-0.000031	0.001889
LTC/USDT	-0.000018	0.006991	0.109748	30.928835	-0.152350	0.136735	-0.002466	0.000000	0.002401
EOS/USDT	-0.000011	0.005942	-0.623118	32.572065	-0.147268	0.080855	-0.002175	-0.000014	0.002147
XLM/USDT	-0.000023	0.005855	0.125108	16.640248	-0.077377	0.094950	-0.002338	0.000000	0.002278

TABLE V: Descriptive statistics for the log returns for each currency pair.

This trading strategy was tested using both the RSI and Williams%R indicators/models. For both indicators an upper/overbought value is specified, and a lower/oversold value is also specified. When the reading in the indicator is above the overbought value, we take a short position, on the other hand if the value is below the oversold a long position is taken (the configurations for the overbought/oversold parameters were described in Section IV-C). The difference in this strategy in context of the momentum indicators is that the RSI reading is between 0 and 100 and the Williams%R is between -100 and 0, however the same conditions apply. This overbought/oversold strategy is better shown in Formula 18.

$$position = \begin{cases} Long(1), & \text{if } X < \text{Oversold} \\ Short(-1), & \text{if } X > \text{Overbought} \\ Sit(0), & \text{otherwise} \end{cases} \tag{18}$$

18: Overbought/oversold strategy, p refers to the position, the conditions refer to the signals. X refers to the reading of the momentum indicator and the $overbought$ and $oversold$ thresholds refer to the parameters set to indicate an overbought or oversold signal, respectively. This strategy is not exhaustive meaning there are times when you hold instead (no position at all 0).

- **Combined Indicators + NATR (Volatility Measure) [Baseline]**
  In this strategy we combined all indicators together and added the NATR indicator as a volatility measure. In this strategy we take a long position if both the EMA and SMA signals a long position or the RSI and Williams%R signals a long position and vice-versa for a short position. However, both conditions also consider that the NATR is above a specific value to take both the short and long positions. This strategy is better shown in Formula 19. We created this strategy to base our positions if indicators of the same type for example EMA and SMA signals the same position, this was done as it furthers confirm the signal, and same goes for the momentum indicators. An *'or'* condition between the trend and momentum indicators was used as we are also interested to make several trades in the period tested and an *'and'* condition would have limited the number of trades (less signals). The NATR also gives us a gist of the current volatility and using a threshold we would only take a long or short position if the spread is above this threshold. All the parameters used in these indicators are described in Section IV-C.

$$position = \begin{cases} Long(1), & \text{if } ((\text{EMA } \& \text{ SMA} == \text{Long}) \\ & or\ (\text{RSI } \& \text{ Will\%R} == \text{Long})) \\ & and\ \text{NATR} > \text{threshold} \\ Short(-1), & \text{if } ((\text{EMA } \& \text{ SMA} == \text{Short}) \\ & or\ (\text{RSI } \& \text{ Will\%R} == \text{Short})) \\ & and\ \text{NATR} > \text{threshold} \\ Sit(0), & \text{otherwise} \end{cases} \tag{19}$$

19: Combined technical indicators strategy, p refers to the position, the conditions refer to the signals. This strategy is not exhaustive meaning there are times when you hold instead (no position at all 0). We also added a volatility measure using the NATR indicator.

- **AI/ML Classifiers Strategy [Proposed]**
  This trading strategy was tested on the AI/ML classifiers

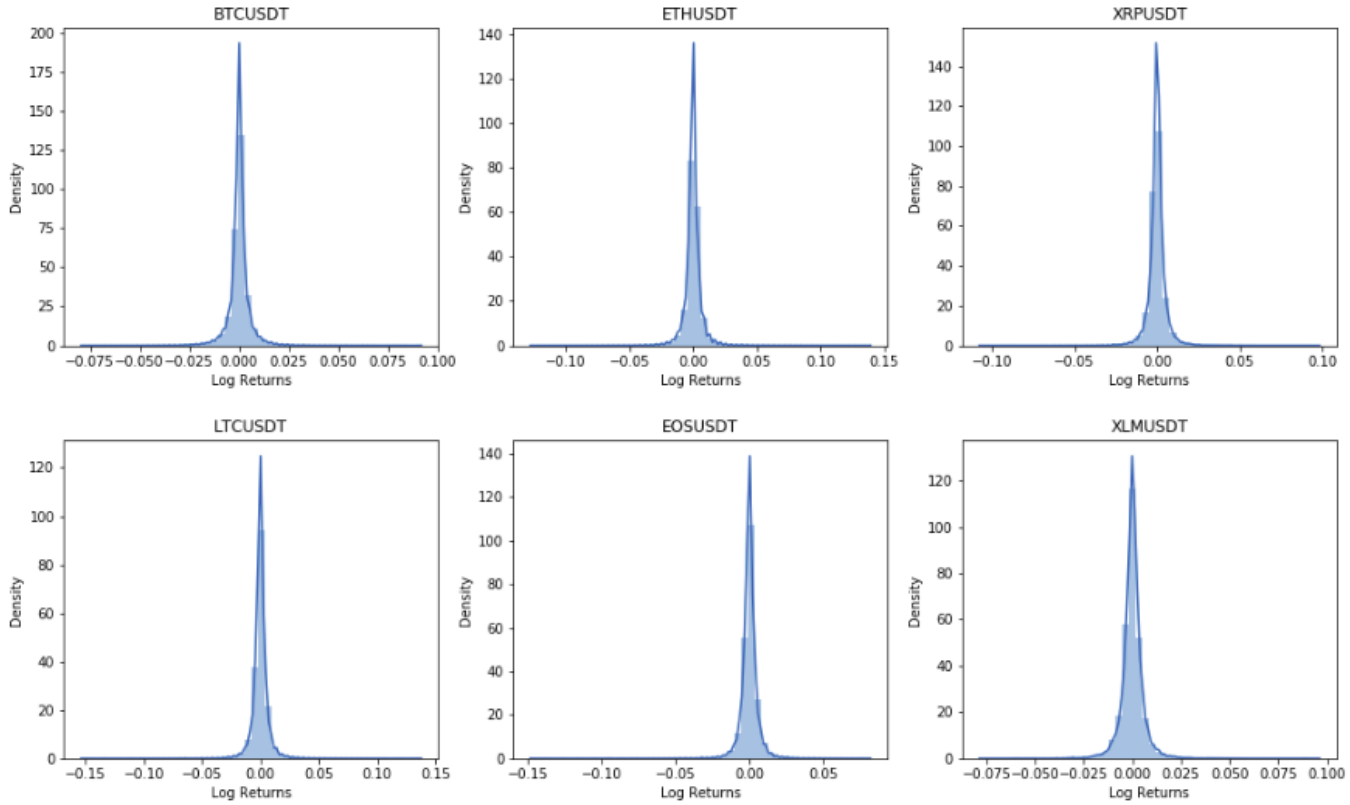

Fig. 1: The distribution plots for the log returns for all the datasets. Top row plots show the probability distribution for the following pairs *BTC/USDT, ETH/USDT* and *XRP/USDT*. The bottom row shows the plots for *LTC/USDT, ETH/USDT* and *XLM/USDT*. All the plots were generated using seaborn python library [34].

which are as follows: XGBoost, Logistic Regression and Random Forest. Basically, this strategy takes a long position if the trained model classifies that the price at $t+1$ of the current data point is classified as an up move (1) or takes a short position if the classification is a down move (0). For both positions an *'and'* condition on the probability of the classification is also taken into consideration, which means both the short and long positions are taken if the confidence probability of the classification is above a certain threshold. This strategy is better shown in Formula 20. The parameters used for each model in this strategy is described in Section IV-C.

$$position = \begin{cases} Long(1), & \text{if } y == 1 \ \& \ y_{prob} > \text{ confidence} \\ Short(-1), & \text{if } y == 0 \ \& \ y_{prob} > \text{ confidence} \\ Sit(0), & \text{otherwise} \end{cases} \tag{20}$$

20: AI/ML strategy, p refers to the position, the conditions refer to the signals. This strategy is not exhaustive meaning there are times when you hold instead (no position at all 0). The y variable refers to the classification made by a specific model, while the y_{prob} refers to the confidence probability of that classification.

- **GA Rules [Proposed]**
  The strategies shown here were found to be the most optimal when training the GA to optimise the Sharpe Ratio given a total of n iterations. These rules are shown in Table VI.

	Encoding	Rule
BTC/USDT	0 1 0 0 0 0 1	XGB xor RF and not(LogReg)
ETH/USDT	0 1 0 0 0 0 1	XGB xor RF and not(LogReg)
XRP/USDT	0 1 1 0 1 1 0	XGB or RF or LogReg
LTC/USDT	0 1 1 0 1 1 0	XGB or RF or LogReg
EOS/USDT	0 1 0 0 1 0 0	XGB xor RF xor LogReg
XLM/USDT	0 1 1 0 1 1 0	XGB or RF or LogReg

TABLE VI: Optimal rules found for each currency when using GA and optimising on Sharpe Ratio.

In total we tested 10 different trading strategies, 6 of which are baseline strategies and the other 4 are our proposed trading strategies. In the next Section V we will show and discuss the results obtained when running this simulation on the strategies described above.

V. Results and Evaluation

The results obtained when testing the baseline models on the out of sample dataset are shown in Table VII. In terms of profit/loss the best performing strategy in the baseline model was the Buy and Hold strategy as it obtained the highest percentages when tested against all currencies except for XRP/USDT where the best performing model was the SMA and for XLM/USDT the best performing model was EMA. From the results obtained the Williams%R seem to be the winner when evaluating in terms of average trades per

day as it obtained the highest score in all the currency pairs except of BTC/USDT where RSI was the top result in terms of average trades per day. As for maximum gain the Buy and Hold strategy obtained the best results in all the currency pairs except of XRP/USDT where the SMA obtained the highest result. When evaluating on the risk-adjusted performance, Sharpe Ratio, the best performing model was the Williams%R as it gained the top Sharpe Ratio in all tests except when testing EOS/USDT and XLM/USDT where the Buy and Hold strategy obtained the highest score. From these results we can deduce that the best performing model in terms of risk-adjusted performance was the Williams%R with the highest Sharpe Ratio of 6.681 when tested on XRP/USDT however the best performing model in terms of Profit and Loss was the Buy and Hold strategy with a maximum result of 100.15% when tested on LTC/USDT.

BTC/USDT						
Model	**Trades**	**ATD**	**P/L (%)**	**MG(%)**	**MDD(%)**	**ASR**
Buy and Hold	1	N/A	88.04*	92.7*	-4.99	4.388
EMA	296	2.26	11.39	18.44	-16.05	0.568
SMA	396	3.02	24.04	40.01	-18.86	1.199
RSI	5494*	41.94**	33.19	37.42	0**	2.202
Williams %R	5402	41.24	73.32	74.32	-1.34	6.681**
Combined TA	1668	12.73	34.1	40.62	-4.86	3.309
ETH/USDT						
Model	**Trades**	**ATD**	**P/L (%)**	**MG(%)**	**MDD(%)**	**ASR**
Buy and Hold	1	N/A	82.6*	89.69*	-13.48	2.841
EMA	308	2.35	46.55	56.77	-10.14	1.602
SMA	596	4.55	-1.76	32.74	-8.04	-0.061
RSI	2732	20.85	-42.65	6.34	-53.08	-2.05
Williams %R	2854*	21.79*	41.73	41.75	-2.14*	3.766*
Combined TA	232	1.77	0.68	5.7	-9.46	0.085
XRP/USDT						
Model	**Trades**	**ATD**	**P/L (%)**	**MG(%)**	**MDD(%)**	**ASR**
Buy and Hold	1	N/A	33.57	41.98	-7.38	1.714
EMA	448	5.67	15.39	32.82	-10.75	0.787
SMA	356	4.51	39.86*	49.01*	-4.01	2.038
RSI	14	0.18	-6.69	0.43	-8.63	-2.074
Williams %R	2424*	30.68*	20.34	26.96	-1.51*	2.304*
Combined TA	76	0.96	6.94	11.4	-3.44	1.611
LTC/USDT						
Model	**Trades**	**ATD**	**P/L (%)**	**MG(%)**	**MDD(%)**	**ASR**
Buy and Hold	1	N/A	100.15**	104.09**	-2.99	3.148
EMA	788	7.36	-44.16	12.2	-69.18	-1.389
SMA	624	5.83	31.65	40.36	-11.38	0.996
RSI	242	2.26	3.72	9.73	-14.52	0.331
Williams %R	3188*	29.79*	69.31	75.57	-5.22	5.333*
Combined TA	218	2.04	33.14	35.24	-0.43*	4.356
EOS/USDT						
Model	**Trades**	**ATD**	**P/L (%)**	**MG(%)**	**MDD(%)**	**ASR**
Buy and Hold	1	N/A	83.38*	85.21*	-4.0	4.095*
EMA	310	4.19	26.43	35.57	-7.64	1.298
SMA	258	3.49	54.44	58.93	-10.37	2.675
RSI	10	0.14	5.81	5.81	0**	4.084
Williams %R	1740*	23.51*	30.33	43.69	0**	3.571
Combined TA	60	0.81	6.99	8.78	-1.57	1.847
XLM/USDT						
Model	**Trades**	**ATD**	**P/L (%)**	**MG(%)**	**MDD(%)**	**ASR**
Buy and Hold	1	N/A	16.14	35.97	-25.78	0.788*
EMA	280	3.84	33.95*	48.32*	-1.27*	1.66
SMA	428	5.86	4.92	23.29	-18.1	0.240
RSI	22	0.3	0.43	1.76	-9.49	0.088
Williams %R	2348*	32.16*	-12.87	14.82	-19.01	-1.191
Combined TA	338	4.63	-21.83	0	-24.4	-2.548

TABLE VII: Results obtained when testing the baseline models on the test dataset. The column *Trades* is the number of trades done, *ATD* is the average trades per day, *P/L(%)* profit and loss for log returns, *MG (%)* maximum gain for log returns, *MDD(%)* maximum drawdown for log returns and *ASR* is the annualised Sharpe Ratio. All values were rounded to two decimal places apart from the *ASR* column where 3 decimal places were used. The values denoted by * are the best performing values and values denoted by ** are the best performing values across all currencies (except for trades).

Now we will discuss results obtained by our proposed models, the accuracy and F1 Scores obtained by each ML model when tested on the test set are shown in Table VIII. In terms of accuracy the best performing model was the Random Forest model where it obtained the highest results when tested against all currency pairs except when tested on ETH/USDT where the XGBoost seem to be the highest. In terms of F1 score the best results were obtained by the XGBoost as it had the highest scores in all tests except when tested against BTC/USDT. The maximum accuracy score (0.546) and F1 score (0.584) were obtained by the RF model when tested against BTC/USDT. It is also worth noting that the accuracy levels achieved for both the RF and Logistic Regression are very similar to [16].

	XGBoost		**RF**		**LogReg**	
Model	**ACC**	**F1**	**ACC**	**F1**	**ACC**	**F1**
BTC/USDT	0.544	0.569	0.546**	0.584**	0.529	0.555
ETH/USDT	0.525	0.521*	0.531*	0.494	0.519	0.514
XRP/USDT	0.515	0.48*	0.535*	0.431	0.528	0.474
LTC/USDT	0.532	0.469*	0.539*	0.398	0.525	0.388
EOS/USDT	0.520	0.501*	05.32*	0.494	0.52	0.458
XLM/USDT	0.509	0.496*	0.516*	0.484	0.512	0.475

TABLE VIII: Accuracy and F1 score results obtained when testing XGBoost, Random Forest and Logistic Regression trained classifiers on the test dataset. The values denoted by * are the best performing values and values denoted by ** are the best performing values across all currencies (except for trades).

As for the trading strategy the results for the proposed methods are shown in Table IX. In terms of profit and loss the combined trading rules found by the Genetic Algorithm seem to obtain quite good results, with the top results for LTC/USDT, EOS/USDT and XLM/USDT for the other 3 pairs XGBoost obtained the best results in terms of P/L. Although the combined signals performed well in terms of P/L it did not obtain the highest results in terms of annualised sharpe ratio. For BTC/USDT and EOS/USDT the RF model obtained the highest results in terms of Sharpe ratio, for the ETH/USDT, XRP/USDT and LTC/USDT the XGBoost obtained the highest result and the Logistic Regression achieved the highest Sharpe Ratio for XLM/USDT. From the results obtained its hard to say which model performed best as each obtained high results in different currency pairs however, the XGBoost model had the highest Sharpe Ratio in three different currency pairs. The highest P/L result in all currency pairs was achieved by the combined trading rules for the LTC/USDT pair with a total log return of 152.96% and in terms of Sharpe Ratio the RF obtained the highest score when tested on BTC/USDT pair with a score of 8.424.

When comparing both the baseline models and the proposed ML models it is apparent that our proposed models outperform the baseline models. There are instances when comparing against the risk adjusted metric, the less performing proposed models fall behind for example in BTC/USDT the lowest Sharpe Ratio was obtained by the combined trading rules (ML models) with a score of 0.318 and in the baseline model the worst result was obtained by EMA indicator with a score of 0.568. When comparing the highest scores obtained for both P/L and Sharpe Ratio for each currency pair from both the baseline and proposed models, the ML models always

BTC/USDT						
Model	**Trades**	**ATD**	**P/L(%)**	**MG(%)**	**MDD(%)**	**ASR**
XGBoost	7872*	60.55*	124.27*	126.41*	-0.68	8.06
RF	5296	40.74	99.09	99.51	-0.66*	8.424**
LogReg	3304	25.42	66.1	67.39	-1.95	8.418
Combined	7866	60.51	6.37	49.82	-11.55	0.318
ETH/USDT						
Model	**Trades**	**ATD**	**P/L(%)**	**MG(%)**	**MDD(%)**	**ASR**
XGBoost	7008*	53.91*	123.2*	130*	-1.58	6.04*
RF	1862	14.32	73.24	79.11	-0.71*	5.665
LogReg	322	2.48	19.11	22.74	-2.21	3.452
Combined	6498	49.98	49.5	59.69	-4.7	1.709
XRP/USDT						
Model	**Trades**	**ATD**	**P/L(%)**	**MG(%)**	**MDD(%)**	**ASR**
XGBoost	5104*	65.44**	81.96*	95.92*	-4.35	5.357*
RF	1888	24.21	30.98	31.04	-7.0	2.686
LogReg	4	0.05	2.42	2.42	0**	3.148
Combined	4494	57.62	41.38	63.08	-10.82	2.125
LTC/USD						
Model	**Trades**	**ATD**	**P/L(%)**	**MG(%)**	**MDD(%)**	**ASR**
XGBoost	6414*	60.51*	144.36	147.95	-3.99	6.311*
RF	2494	23.53	78.7	94.36	-14.32	5.546
LogReg	2	0.02	-1.77	0	-1.77*	-1.858
Combined	4852	45.77	152.69**	158.96**	-8.64	4.820
EOS/USDT						
Model	**Trades**	**ATD**	**P/L(%)**	**MG(%)**	**MDD(%)**	**ASR**
XGBoost	4708*	64.49*	78.18	79.53	-6.62	5.205
RF	2462	33.73	84.6	87.66	-3.54	7.124*
LogReg	680	9.32	5.97	8.6	-2.7*	0.913
Combined	4358	59.7	111.91*	113.68*	-6.07	5.534
XLM/USDT						
Model	**Trades**	**ATD**	**P/L(%)**	**MG(%)**	**MDD(%)**	**ASR**
XGBoost	4554*	62.38*	0.31	32.5	-6.14	0.02
RF	1654	22.66	7.54	24.72	-0.95*	0.650
LogReg	416	5.7	12.15	18.19	-1.47	2.867*
Combined	4512	61.81	14.44*	52.39*	-1.33	0.708

TABLE IX: Results obtained when testing the proposed models on the test dataset. The column *Trades* is the number of trades done, *ATD* is the average trades per day, *P/L(%)* profit and loss for log returns, *MG (%)* maximum gain for log returns, *MDD(%)* maximum drawdown for log returns and *ASR* is the annualised Sharpe Ratio. All values were rounded to two decimal places apart from the *ASR* column where 3 decimal places were used. The values denoted by * are the best performing values and values denoted by ** are the best performing values across all currencies (except for trades).

obtained the highest scores. It seems that AI/ML models provide more efficient trading strategies when trained using technical indicators as features rather than using standard technical indicator strategies. It is also worth noting that although we anticipated that the combined ML signals will outperform individual classifiers in the risk-adjusted score (which was not the case), it seems to have potential as it ranked top for three different currency pairs in terms of P/L and as stated the highest P/L obtained was achieved by this strategy.

In view of our contributions we can say that overall the AI/ML models outperform the standard technical indicators strategies in terms of P/L and Annualised Sharpe Ratio. We also managed to combine the signals generated from individual AI/ML models using a Genetic algorithm. However, we anticipated that the combined optimal rules will obtain better scores in terms of Annualized Sharpe Ratio (since it was optimised on Sharpe Ratio), although it seems to have potential as it outperformed the other strategies in three different tests when evaluating on P/L. We would like to add that there are many ways to try to improve the Genetic Algorithm itself for example adding the concept of elitism where you keep the best performing chromosomes in the next population, using a different selection method which gives more opportunity to select weak individuals and applying a different crossover method. Also, finding the right hyperparameters for the GA can also be challenging and should be investigated. It is also worth to investigate the use of GA on the standard technical indicators only and compare the results with the ones obtained by the combined AI/ML models.

VI. Conclusion

XGBoost, Random Forest and Logistic Regression is evidently effective when used as trading strategies, with the tree-based ensemble models performing better than the Logistic Regression. As for combining the models using a GA, also seem to be effective in terms of P/L and as stated we anticipated better results when using this method in terms of risk-adjusted returns. Using technical indicators as features to train these classifiers seem to show that they outperform the standard technical indicators strategies. We also tried to combine the standard technical indicators as one trading strategy using our own intuition however that method seem to obtain poor results and should not be compared against the GA optimal rules method.

To properly investigate the performance of the combined technical indicator signals as one strategy against the combined AI/ML strategy, the same GA technique should have been applied. This is worth investigating as future work. There are many ways to try to improve our strategy given the number of hyperparameters used, however the following list is the most important work (in our opinion) that can be done for future work

- Apply the same GA method to combine the individual technical indicators together and compare with our proposed model
- Apply the concept of stop loss in the trading strategy as this may improve the results drastically
- For the machine learning models tweak the hyperparameters using some tuning algorithm such as Grid search as our hyperparameters for these models were not changed except for the max tree depth for the tree based models

All the code used in this study was written in Python, and the code/data used to evaluate our approach is publicly available on this project's GitHub repository https://github.com/achmand/ari5123_assignment. The equity curve plots and other graphs are also supplied in this repository and can be viewed from a Jupyter Notebook found in the repository.

References

[1] T. Oberlechner, "Importance of technical and fundamental analysis in the european foreign exchange market," *International Journal of Finance & Economics*, vol. 6, no. 1, pp. 81–93, 2001.

[2] D. Olson, "Have trading rule profits in the currency markets declined over time?" *Journal of banking & Finance*, vol. 28, no. 1, pp. 85–105, 2004.

[3] W.-K. Wong, M. Manzur, and B.-K. Chew, "How rewarding is technical analysis? evidence from singapore stock market," *Applied Financial Economics*, vol. 13, no. 7, pp. 543–551, 2003.

[4] M. A. Dempster and V. Leemans, "An automated fx trading system using adaptive reinforcement learning," *Expert Systems with Applications*, vol. 30, no. 3, pp. 543–552, 2006.

[5] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock market prediction based on generative adversarial network," *Procedia computer science*, vol. 147, pp. 400–406, 2019.

[6] A. ElBahrawy, L. Alessandretti, A. Kandler, R. Pastor-Satorras, and A. Baronchelli, "Evolutionary dynamics of the cryptocurrency market," *Royal Society open science*, vol. 4, no. 11, p. 170623, 2017.

[7] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[8] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[9] B. Chase and E. MacBrough, "Analysis of the xrp ledger consensus protocol," *arXiv preprint arXiv:1802.07242*, 2018.

[10] Litecoin, "Litecoin github page," *Litecoin (accessed 1 June 2019) https://github.com/litecoin-project/litecoin*, 2011.

[11] E. IO, "Eos. io technical white paper," *EOS. IO (accessed 1 June 2019) https://github.com/EOSIO/Documentation*, 2017.

[12] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," *Stellar Development Foundation*, 2015.

[13] "CoinMarketCap: Top 100 cryptocurrencies by market capitalization," 2013–, [Online; accessed June 2019]. [Online]. Available: https://coinmarketcap.com/

[14] X. Sun, M. Liu, and Z. Sima, "A novel cryptocurrency price trend forecasting model based on lightgbm," *Finance Research Letters*, 2018.

[15] L. Alessandretti, A. ElBahrawy, L. M. Aiello, and A. Baronchelli, "Anticipating cryptocurrency prices using machine learning," *Complexity*, vol. 2018, 2018.

[16] I. Madan, S. Saluja, and A. Zhao, "Automated bitcoin trading via machine learning algorithms," *URL: http://cs229. stanford. edu/proj2014/Isaac% 20Madan*, vol. 20, 2015.

[17] B. J. de Almeida, R. F. Neves, and N. Horta, "Combining support vector machine with genetic algorithms to optimize investments in forex markets with high leverage," *Applied Soft Computing*, vol. 64, pp. 596–613, 2018.

[18] T. M. Tupinambás, R. A. L. Cadence, and A. P. Lemos, "Cryptocurrencies transactions advisor using a genetic mamdani-type fuzzy rules based system," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–7.

[19] E. Bouri, R. Gupta, A. K. Tiwari, and D. Roubaud, "Does bitcoin hedge global uncertainty? evidence from wavelet-based quantile-in-quantile regressions," *Finance Research Letters*, vol. 23, pp. 87–95, 2017.

[20] J.-Z. Huang, W. Huang, and J. Ni, "Predicting bitcoin returns using high-dimensional technical indicators," *The Journal of Finance and Data Science*, 2018.

[21] K. Michalak, P. Filipiak, and P. Lipinski, "Usage patterns of trading rules in stock market trading strategies optimized with evolutionary methods," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2013, pp. 234–243.

[22] T.-C. Fu, C.-P. Chung, and F.-L. Chung, "Adopting genetic algorithms for technical analysis and portfolio management," *Computers & Mathematics with Applications*, vol. 66, no. 10, pp. 1743–1757, 2013.

[23] S. Ha and B.-R. Moon, "Finding attractive technical patterns in cryptocurrency markets," *Memetic Computing*, vol. 10, no. 3, pp. 301–306, 2018.

[24] W. Brock, J. Lakonishok, and B. LeBaron, "Simple technical trading rules and the stochastic properties of stock returns," *The Journal of finance*, vol. 47, no. 5, pp. 1731–1764, 1992.

[25] T. Kilgallen, "Testing the simple moving average across commodities, global stock indices, and currencies," *The Journal of Wealth Management*, vol. 15, no. 1, p. 82, 2012.

[26] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[27] C. Gini, "Italian: Variabilità e mutabilità (variability and mutability)," *Cuppini, Bologna*, 1912.

[28] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[29] R. D. McKelvey and W. Zavoina, "A statistical model for the analysis of ordinal level dependent variables," *Journal of mathematical sociology*, vol. 4, no. 1, pp. 103–120, 1975.

[30] M.-C. Chan, C.-C. Wong, and C.-C. Lam, "Financial time series forecasting by neural network using conjugate gradient learning algorithm and multiple linear regression weight initialization," in *Computing in Economics and Finance*, vol. 61, 2000, pp. 326–342.

[31] "Binance: a global cryptocurrency exchange that provides a platform for trading cryptocurrencies." 2017–, [Online; accessed June 2019]. [Online]. Available: https://www.binance.com/en

[32] "Binance-WebAPI: official documentation for the binance apis and streams." 2017–, [Online; accessed June 2019]. [Online]. Available: https://github.com/binance-exchange/binance-official-api-docs

[33] U. W. Chohan, "Tethering cryptocurrencies to fiat currencies without transparency: A case study," *Available at SSRN 3129978*, 2018.

[34] M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, C. Fonnesbeck, A. Lee, and A. Qalieh, "mwaskom/seaborn: v0.8.1 (september 2017)," Sep. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.883859

[35] "Binance: current fees on the exchange." 2017–, [Online; accessed June 2019]. [Online]. Available: https://www.binance.com/en/fee/schedule