

Development an Artificial Intelligence to Automate the Buying and Selling of bitcoins

Desarrollo de una Inteligencia Artificial para Automatizar la Compra y Venta de bitcoins

LEDESMA-URIBE, Norma Alejandra†*, OLVERA-MONROY, Jared Fabian, ATANACIO-MELCHOR, Jesus David and RODRIGUEZ-MIRANDA, Gregorio

Universidad Tecnológica de San Juan del Río, División de Mecatrónica, Desarrollo de Software e Ingeniería Civil

1st Author ID: Norma Alejandra, Ledesma-Uribe / ORC ID: 0000-0001-8422-2046, CVU CONACYT ID: 673202

1st Co-author ID: Jesus David, Atanacio-Melchor / ORC ID: 0000-0001-6519-9346 Researcher ID Thomson AGY-2511-2022, CVU CONACYT ID: 1198872

2nd Co-author ID: Jared Fabian, Olvera-Monroy / ORC ID: 0000-0003-4777-6642 Researcher ID Thomson AGY-2510-2022, CVU CONACYT ID: 1198873

3rd Co-author ID: Gregorio, Rodriguez-Miranda / ORC ID: 0000-0002-2512-892X, CVU CONACYT ID: 246718

DOI: 10.35429/JIT.2022.27.9.1.20

Received: March 10, 2022; Accepted June 30, 2022

Abstract

Trading consists of purchasing and selling listed assets with high market liquidity such as: stocks, currencies and futures. then this financial market is used electronically and it is regulated (the trading and generation price is freely agreed through a negotiation process between the consumer and the trader.). One of its objective is to obtain an economic benefit when the operation generates a capital income , repeating the process for a considerable number of operations, therefore it makes it possible to increase the initial capital. This article will approach the implementation of an algorithmic trading model which can help to maximize the profitability of a portfolio for cryptocurrency assets based on the application, combination and weighting of some of the most advanced mathematical techniques using the Python programming language, its libraries and some other tools in a controlled development environment and also guided by a previous research and training based on the topics planned for the completion and satisfaction of the project.

Trading, Exchange, Cryptocurrency, Bot, Genetic Algorithms

Resumen

El ‘trading’ consiste en la compraventa de activos cotizados con mucha liquidez de mercado (acciones, divisas y futuros). Y ese mercado financiero es empleado de manera electrónica y está regulado (el precio de comercialización y generación se pacta libremente mediante un proceso de negociación entre el consumidor y el comercializador.). Su objetivo es obtener un beneficio económico cuando la operación genera una plusvalía, repitiendo el proceso por un número considerable de operaciones, permitiendo así aumentar el capital inicial. En este artículo se abordará la implementación de un modelo de trading algorítmico que ayude a maximizar la rentabilidad de un portafolio de activos de criptomonedas basándose en la aplicación, combinación y ponderación de alguna de las técnicas matemáticas más avanzadas utilizando el lenguaje de programación Python, sus librerías y algunas otras herramientas en un entorno de desarrollo controlado y guiado por una investigación y capacitación previa basada en los temas previstos para la conclusión y satisfacción del proyecto.

Trading, Exchange, Criptomoneda, Bot, Algoritmos genéticos

Citation: LEDESMA-URIBE, Norma Alejandra, OLVERA-MONROY, Jared Fabian, ATANACIO-MELCHOR, Jesus David and RODRIGUEZ-MIRANDA, Gregorio. Development an Artificial Intelligence to Automate the Buying and Selling of bitcoins. Journal Information Technology. 2022. 9-27: 1-20

* Author Correspondence (e-mail: nledesma@utsjr.edu.mx)
† Researcher contributing as first author.

Introduction

Cryptocurrencies are virtual currencies. They can be exchanged and traded like any other traditional currency, however these cryptocurrencies are outside governments control as well as financial institutions. This paper discusses the Algorithmic Trading Model applied to cryptocurrencies carried out in a company in the state of Querétaro.

Cryptocurrency trading is the act of speculating on the price movements of cryptocurrencies through a CFD trading account, or buying and selling the underlying cryptocurrencies in a trading market. (Binance Academy, 2021)

Actually is intended to create an algorithm that allows us trading with cryptocurrencies in which a large number of transactions will be executed in a short time on the Binance Exchange (exchange market), where transactions must be made by acquiring coins at the lowest possible price then after selling it on the exchange platform with a higher price than what we have acquired, then lastly We can acquire a profit. In this process, you want a bot (a program that performs repetitive, predefined and automated tasks) to make multiple purchases in a short period of time (Kaspersky Lab, 2022).

One of the main needs to create a bot to be able to perform this activity comes from the problem that markets change rapidly the value of currencies varying in small prices hence complicating multiple purchases.

Currently performing a large number of operations for a human is difficult, reasoning and making a correct decision in a short period of time having emotions in between, therefore this will be the task of implementing the bot that will allow us to make a decision that at the end would give us a real gain, actually It is our main objective.

Nowadays there are several bots that can be configured to operate in an exchange, having all one thing in common, you must pay to use their tools. This development uses free software and free resources, with the main objective of using a bot, which actually it is also free.

The bot development will be developed under the visual studio code environment and programmed in the Python programming language which provides us with different libraries with which it will be necessary to work, ranging from pandas, ccxt, to the most important one called pyjuke that will make it easier to reach our final goal.

Objective

Develop an algorithmic trading model that maximizes the profitability of our portfolio of cryptocurrency assets, based on the application, combination and weighting of some of the most advanced mathematical techniques.

Problem

Currently investors, traders, speculators and anyone who wants to get into cryptocurrency trading in the so-called Exchange requires to perform a huge series of activities and calculations that a person could not perform in the time and accuracy demanded, so they are performed by a bot. Making use of computer tools, the aim is to provide a solution to market operations (trading), generating many sentences in a short time with bots, and then obtaining the highest possible profit. (Caminiti, 2021). Among bots, you can find some that may inject malicious codes, stealing their information, movements and cryptocurrencies. To prevent this attack it is recomendable to block JavaScript in browsers as well as the use of software specialized in blocking all types of processes that refer to mining in browsers, such as "MinerBlock" and "No Coin". (Malwarebytes, 2022)

Development

The bot design is based on the course "DE CERO A QUANT", Algorithmic trading for cryptocurrencies with python and machine learning. (Cajina Ramirez, 2021). The actual development was developed on a PC with a Linux operating system with the Ubuntu distribution, by using Python programming language. Before starting the development it is necessary to specify parameters to generate strategies, therefore purpose the ccxt library was used, with the objective of accessing information of different cryptocurrencies in the exchanges with which it is compatible, since the information that can be found is public.

In the event that it is required to access private information like user balance, view buy or sell orders, or make transactions, it is necessary to use public and private keys provided by the Exchange. Figure 1 shows how the library is used.

```
import ccxt

print(ccxt. exchanges)

kucoin = ccxt. kucoin({
    'apiKey': 'YOUR_APIKEY',
    'secret': 'YOUR_SECRET',
    'timeout': 3000,
    'enableRateLimit': True,
})
```

Figure 1 Data import ccxt library
Source: Own Elaboration

Figure 2 shows in which Exchange the information is searched, followed by a variable named "symbol" that will store the value of the symbol of the required cryptocurrency. The information frequency is also assigned, minutes, hours, days, etc., since the prices of digital currencies change permanently.

```
exchange = ccxt. binance()
symbol = 'BTC/USDT'
timeframe = '1h'
ohlcv = exchange. fetch_ohlcv(symbol,
timeframe)
```

Figure 2 Data import ccxt library
Source: Own Elaboration

Once information is received, it is formatted and saved in a "dataframe", with the help of the Pandas library. To use the library, it is installed in the computer and the steps shown in figure 3 are used.

```
def
ccxt_ohlcv_to_dataframe(ohlcv):
    df = pd. DataFrame(ohlcv)
    df. columns = ['time', 'open',
'high', 'low', 'close', 'volume']
    df['date'] = pd.
to_datetime(df['time']*1000000,
infer_datetime_format = True)
return df
```

Figure 3 Transforming information with pandas
Source: Own Elaboration

In order to develop a strategy that would predict the market price as well as recognizing patterns, quantitative analysis was used with indicators such as: the Simple Moving Average (SMA), the Exponential Moving Average (EMA), the Relative Strength Index (RSI) and the Relative Strength Index (RSI).

The panda_ta library allows using a sample of data, in figure 4 10 are used, which will define the strategy of the average of the prices of the sample.

```
df['sma'] = df.ta.sma(length=10)

print(df. drop(columns=['time',
'open', 'high', 'low']))
```

Figure 4 SMA calculation
Source: Own Elaboration

The simple moving average is one of the most used indicators for trading within the markets, as it allows us to measure several factors, the variation of the price of the different crypto-assets, the speed with which they change and the overbought and oversold.

The Relative Strength Index (RSI) is used to measure the volatility of prices in the market, as shown in Figure 5.

$RSI = 100 - 100 / (1 + RS)$

RS refers to the Relative Strength factor, which is calculated as shown below.

$RS = AvgU / AvgD$

AvgU refers to the average of the upward price changes in a given period.

AvgD refers to the average downward price variation over a given period.

```
df['rsi'] = df.ta.rsi(length=14)

print(df. drop(columns=['time',
'open', 'high', 'low']))
```

Figure 5 Calculate RSI
Source: Own Elaboration

This indicator is made up of three bands, the simple moving average and two more called the upper and lower standard deviation.

Calculate the upper and lower bands by standard deviation. Another indicator is the Bollinger Bands, which are the lower and upper limits at which assets will move over a period of time as shown in Figure 6.

To implement them in the code, as in the previous indicators, the functions provided by the `panda_ta` library will be used. In this function period and number of standard deviations must be indicated as shown in the previous code. The MACD stands for Moving Average Convergene Divergence.



Figure 6 Bollinger Bands

Source: (Broseta, 2016)

The MACD indicates, at each moment, the separation between the value of two moving averages with different calculation periods. (AvaTrade, 2022)

$$MACD = EMA_{12} - EMA_{26}$$

As can be seen, for this indicator, an exponential moving average with a short period and another exponential average

```
ta_bollinger_bands =
df.ta.bbands(length = 20, std = 20)

print("Bollinger Bars")
print(ta_bollinger_bands)
```

Figure 7 Calculate Bollinger Bands

Source: Own Elaboration

In this function period and number of standard deviations must be indicated as shown in the previous code. The MACD stands for Moving Average Convergene Divergence. The MACD indicates, at each moment, the separation between the value of two moving averages with different calculation periods. (AvaTrade, 2022)

$$MACD = EMA_{12} - EMA_{26} \quad (1)$$

As can be seen, for this indicator, an exponential moving average with a short period and another exponential average with a medium time period must be calculated first. The shorter the calculation period, the more sensitive the moving average is to price variation.

The formula for calculating the exponential moving average is as follows:

$$EMA = Pk + EMAA(1 - k) \quad (2)$$

EMAA refers to the exponential average of the previous day.

P refers to the price.

K equals $2/(N + 1)$

N is the number of days in the moving average. (Saenz, 2020)

Below is the line of code where the MACD is calculated with the help of the `pandas_ta` library.

```
macd_ta = df.ta.macd()

print("MACD")
print(macd_ta)
```

Figure 8 Calculate MACD

Source: Own Elaboration

To test strategy effectiveness before using it with real money, backtesting will be used, it will be analyzed how it behaves using historical data in order to test its effectiveness. This is called quantitative analysis since the predictability of the model is reviewed. Once this analysis is completed, the actions to be executed by the strategy must be defined. As a strategy, the RSI condition is reviewed and a BUY will be made when the previous candlestick crosses below the lower band and the low of the current candlestick is greater than the lower band.

A SELL is made when the high of the previous candlestick crosses the upper band and the high of the current candlestick is lower than the upper band.

For the programming of the strategy, see Figure 9, the pandas and pandas_ta libraries are used. A class called "BBStrategy" is declared, which will contain the initial parameters for the strategy:

- Length of bollinger bands: number of standard deviations.
- rsi length: number of periods.
- Limit on overbuying and over selling.

```
def __init__(self, bb_len = 20, n_std = 2.0, rsi_len = 14, rsi_overbought = 60, rsi_oversold = 40):  
  
    self.bb_len = bb_len  
    self.n_std = n_std  
    self.rsi_len = rsi_len  
    self.rsi_overbought = rsi_overbought  
    self.rsi_oversold = rsi_oversold
```

Figure 9 Class to program the strategy
Source: Own Elaboration

A method called "setUp" is used to calculate indicators. The function provided by the library was used to obtain the bollinger bands and the rsi, and send the dataframe with the data exported from the Exchange. Once calculations have been performed, they are saved in a dataframe as shown in Figure 10.

```
def setUp(self, df):  
    bb = ta.bbands(  
        close = df['close'],  
        length = self.bb_len,  
        std = self.n_std  
  
    df['lbb'] = bb.iloc[:,0]  
    df['mbb'] = bb.iloc[:,1]  
    df['ubb'] = bb.iloc[:,2] = bb.iloc[:,2].  
  
    df['rsi'] = ta.rsi(close = df['close'], length = self.rsi_len)  
  
    self.dataframe = df
```

Figure 10 Method for calculating indicators
Source: Own Elaboration

As can be seen in the second "if" of the code in Figure 11, some conditions are established that must be met to see if a command will be executed or not.

```
def checkLongSignal(self, i = None):  
    df = self.dataframe  
  
    if i == None:  
        i = len(df)  
  
    if (df['rsi'].iloc[i] < self.rsi_overbought) and \  
        (df['rsi'].iloc[i] > self.rsi_oversold) and \  
        (df['low'].iloc[i-1] < df['lbb'].iloc[i-1]) and \  
        (df['low'].iloc[i] > df['lbb'].iloc[i]) :  
  
        return True  
    return False
```

Figure 11 Method to detect signals
Source: Own Elaboration

The function will return True or False according to the following assumptions:

- The rsi must be less than our overbought value.
- Our rsi must be higher than our oversold level.

- The minimum of the previous candle is less than the band of bolliger i-1 to be the previous one.
- The low of the current candlestick is greater than the bollinger band.

Once the strategy programming is completed, methods are integrated, results are displayed and monitored to the user on the screen, see Figure 12.

```
import ccxt
from utils import ccxt_ohlcv_to_dataframe

exchange = ccxt.binance()
symbol = 'BTC/USDT'
timeframe = '1h'
ohlcv = exchange.fetch_ohlcv(symbol, timeframe)
df = ccxt_ohlcv_to_dataframe(ohlcv)
strategy = BBStrategy()

strategy.setUp(df)

for i in range(len(df)):
    print(strategy.checkLongSignal(i))
```

Figure 12 Data import and execution of methods
Source: Own Elaboration

Figure 13 shows a program code for testing a strategy in simulation mode, with real-time stock market data, before running it with real money. Once variables to work with have been defined, the "open_position" method is implemented, It will open a position. A position can be defined as a type of binding commitment to buy or sell transactions, in this case cryptocurrencies.

As it can be shown in the code in figure 14, the method requires some variables to operate, the price of cryptocurrencies and the "from_opened". Following this is the variable that counts the number of operations that are performed, which must be increased each time the code is executed. If there is an open trade we average the prices and add or subtract the profit or loss to the variable "amount". If there is nothing open we update the variable "is_long_open" to "True" which indicates that a trade has been opened, we also update the variable "long_open_price" by the new currency price (this happens every time this code is executed) and update the amount.

Finally, if a percentage loss was set and the value of the variable "from opened" will be updated.

```
class Backtester():
    def __init__(self, initial_balance = 1000, leverage = 10, trailing_stop_loss = False):
        self.initial_balance = initial_balance
        self.balance = initial_balance
        self.amount = 0
        self.leverage = leverage
        self.fee_cost = 0.02 / 100
        self.inv = self.balance * 0.01 * self.leverage
        self.profit = []
        self.drawdown = []
        self.winned = 0
        self.losed = 0
        self.num_operations = 0
        self.is_long_open = False
        self.trailing_stop_loss = trailing_stop_loss
        self.from_opened = 0
```

Figure 13 Backtester Class
Source: Own Elaboration

A method will be created that does the opposite action to open a position, i.e. close it. It will be called "close_position" and will need to receive only the price to perform the calculation. See figure 15.

```
def open_position(self, price, from_opened = 0):
    self.num_operations += 1

    if self.is_long_open:
        self.long_open_price = (self.long_open_price + price)/2
        self.amount += self.inv/price
    else:
        self.is_long_open = True
        self.long_open_price = price
        self.amount = self.inv/price

    if self.trailing_stop_loss:
        self.from_opened = from_opened
```

Figure 14 Method opening a position
Source: Own Elaboration


```
def close_position(self, price):
    self.num_operations += 1

    if self.is_long_open:
        result = self.amount *
(price - self.long_open_price)
        self.is_long_open =
False
        self.long_open_price = 0

    self.profit.append(result)
    self.balance += result

    if result > 0:
        self.winned += 1
        self.drawdown.append(0)
    else:
        self.losedd += 1
        self.drawdown.
append(result)

    self.take_profit_price = 0
    self.stop_loss_price = 0
```

Figure 15 Method of closing a position
Source: Own Elaboration

At the very begging the number of operations will be incremented, then it is requested if a long operation is open, if this is true we will start calculating the results, for this the amount is multiplied by the result of the difference between the current price of the currency and the price it had when the operation was opened. We also change the value of the variable that indicates if there is something open to "False" and the variable that stores the price returns to be 0.

As for the list called "profit", the results will be added to it and the balance will be incremented in the same way. If results are greater than zero, it means that there was a profit, therefore, the variable "winned" will be incremented and a 0 will be added to the "drawdown". On the other hand, if it is not, it means that there was a loss and the "losedd" variable will be incremented and the results will also be added to the drawdown.

Finally, the variables "take_profit_price" and "stop_loss_price" return to a value of 0.

Now the methods that will set the "profit" and "stop_loss" will be defined.

The first method to create will be called "set_take_profit" which must receive as parameter the price (from which we will calculate the profit) and the percentage of that price that we want to earn.

```
def set_take_profit(self, price,
tp_long = 1.05):

    self.take_profit_price =
price * tp_long
```

Figure 16 Method that establishes a profit percentage
Source: Own Elaboration

As shown in Figure 16, the percentage assigned in this case was 5%, concluding with the operation of the current price multiplied by the percentage, assigning this value to the variable "take_profit_price".

To calculate the "stop_loss" the "set_stop_loss" method will be created which will receive the current price of the currency and the percentage of that price willing to lose.

```
def set_stop_loss(self, price, sl_long
= 0.98):

    self.stop_loss_price = price
* sl_long
```

Figure 17 Method that establishes a percentage of loss
Source: Own Elaboration

In this case, the loss was set at 2% below the price, calculated by multiplying the price by the percentage, assigning this value to the variable "stop_loss_price".

In order to have a better management of the strategy's operation, the data must be visualized, and a method will be created that will show the results of the strategy's bakctesting. This method should receive as a parameter the symbol of the desired cryptocurrency, the start date of the period in which you want to test the end date.

```
def return_results(self, symbol,
start_date, end_date):
    profit = sum(self.profit)
    drawdown = sum(self.drawdown)
    fees = (abs(profit) * self.
fee_cost * self.num_operations)
    results = {
        'symbol': symbol,
        'start_date': start_date,
        'end_date': end_date,
        'balance': self.balance,
        'profit': profit,
        'drawdown': drawdown,
        'profit_after_fees': profit - fees,
        'num_operations': self.
num_operations,
        'num_long': self.num longs,
        'num_shorts': self.num shorts,
        'winned': self.winned,
        'losses': self.losses
    }
    if self.num_operations > 0 and
(self.winned + self.losses) > 0:
        winrate = self.winned /
(self.winned + self.losses)
        results['winrate'] = winrate
        results['fitness_function'] =
(self.num longs + self.num shorts) *
(profit - abs(drawdown)) * winrate / self.
num_operations
    else:
        results['winrate'] = 0
        results['fitness_function'] =
0
    return results
```

Figure 18 Method that returns results

Source: Own Elaboration

In the last mentioned method some variables will be used and their value will be the sum of the lists that are generated during the operations, as well as the cost that is paid for these operations. After this, values will be added to "results", among them the "winrate" and "fitness_function" that must be calculated first.

For the "winrate" it must be asked if the operations performed are greater than zero and that the sum of the won and lost operations is also greater than zero, if this is true, the variable will be assigned the value of the won operations between the sum of the won and lost operations, once calculated it is added to the results.

The fitness_function will be calculated as the sum of the number of operations multiplied by the subtraction of the profit minus the absolute value of the drawdown multiplied by the division of the winrate by the number of operations. If no operation was performed, only the value of zero is added to "results" in both fields. The return of the information is finished.

Now a function will be created which will be in charge of executing the backtesting. This function would receive the dataframe and the strategy that needs to be tested. This backtesting will be based on the highest, lowest and closing price of a candle.

First we will check an individual that is positive, we will go through the dataframe with a cycle to do this. After this we check if there is a buy signal, if this is true we open a purchase that will be equal to the closing price, that is, we call the method that opens a position by sending it the necessary arguments. This is also done in the same way with the functions for setting the profit and stop loss.

If there is nothing on the current candlestick, make sure that there is an open trade, in order to update the stop loss by comparing the previous prices with the current ones. We ask if the variable that indicates whether a stop loss was set is "True" and if a trade is open. If the above is true with the help of a variable we will save the maximum price reached, this to check if it is necessary to set a new stop loss.

To conclude a conditional is made which operates to calculate if it is a good time to close the position because it has been possible to obtain some profit or instead there was a loss and the stop loss should be activated.

Finally, we need a method that simply returns all variables to their initial value for further operations. The code for this function is as follows.


```
def reset_results(self):

    self. balance = self.
initial_balance
    self. amount = 0
    self. profit = []
    self. drawdown = []
    self. wonned = 0
    self. lossed = 0
    self. num_operations = 0
    self. num longs = 0
    self. num shorts = 0
    self. is_long_open = False
    self. is_short_open = False
    self. from_opened = 0
```

Figure 19 Method performing backtesting
Source: Own Elaboration

An algorithm is a series of organized steps that describes the process to be followed, to provide a solution to a specific problem. A genetic algorithm (or GA for short) is a programming technique inspired by the reproduction of living beings and mimics biological evolution as a strategy to solve optimization problems.

In general, genetic algorithms (GAs) are part of the so-called artificial intelligence, i.e., problem solving using computer programs that mimic the functioning of natural intelligence. (Garduño Juárez, 2018).

```
def __backtesting__(self, df, strategy):
    high = df['high']]
    close = df['close']]
    low = df['low']]
    for i in range(len(df)):
        if self. balance > 0:
            if
strategy.checkLongSignal(i):
                self.
open_position(price = close[i], side =
'long', from_opened = i)
                self.
set_take_profit(price = close[i], tp_long
= 1.03)
                self.
set_stop_loss(price = close[i], sl_long =
0.99)
            else:
                if self.
trailing_stop_loss and (self.
is_long_open):
                    new_max =
high[self. from_opened:i].max()
                    previous_stop_loss
= self. stop_loss_price
                    self.
set_stop_loss(price = new_max)
                    if
previous_stop_loss > self.
stop_loss_price:
                        self.
stop_loss_price = previous_stop_loss
                        if self. is_long_open:
                            if high[i] >=
self. take_profit_price:
                                self.
close_position(price = self.
take_profit_price)
```

Figure 20 Method returns variables to their initial values
Source: Own elaboration

In summary, a GA consists of the following steps:

- Initialization: an initial population of possible solutions to a problem, also called individuals, is randomly generated.
- Evaluation: application of the evaluation function to each individual.
- Evolution: application of genetic operators (such as selection, reproduction and mutation).

- And term: the GA should stop when the optimal solution is reached, but this is usually unknown, so several stopping criteria are used.

In this project we will make use of genetic algorithms to optimize the parameters of the trading strategies following using the above mentioned steps, being these variables the population.

To start with the implementation of these algorithms, a file called "GA" is created in which some methods will be programmed to carry out each of the necessary steps to calculate the most optimal option or also called "the best individual".

Initially, the class called individual is declared, there we will create a data structure is used to access the methods, data and properties of individuals once the population has been created. This should receive as parameter the number of genes it will have and the range of those genes, that is, the parameters used by the indicators of the strategy.

```
class Individual:
    def __init__(self, n_genes,
gene_ranges):
        self. genes = [np. random.
randint(gene_ranges[x][0],
gene_ranges[x][1]) for x in
range(n_genes)]

        self. backtester = Backtester(
            initial_balance = 1000,
            leverage = 10,
            trailing_stop_loss = True
```

Figure 21 Individual class
Source: Own Elaboration

The variable "genes" is matched to a random number generated in a range from the first interval to the last in each of the defined tuples, which at the end is stored as a list. The backtester attribute is also added to be able to access all the characteristics of each individual, as well as to analyze how each one of them behaves with the historical data of the Exchange.

To continue he defined the class "Population", and within it a constructor, since in this class the individuals will be defined. This constructor must receive some parameters, such as the number of individuals per population named "generation_size" which will be a list according to the size required, the number of genes in "n_genes", the rank, the number or percentage of best individuals found and a parameter that indicates the probability that a gene mutates.

```
class Population:
    def __init__(self,
generation_size, n_genes,
gene_ranges, n_best,
mutation_rate):

        self. population =
[Individual(n_genes, gene_ranges)
for _ in range(generation_size)].
        self. n_genes = n_genes
        self. gene_ranges =
gene_ranges
        self. n_best = n_best
        self. generation_size =
generation_size
        self. mutation_rate =
mutation_rate
```

Figure 22 Population Class
Source: Own Elaboration

In this class we will perform the whole process to calculate the best individual using genetic algorithms, so to continue we will create a method called "selection" which will return the best individuals, as shown below.

```
def selection(self):
    return sorted(
        self. population,
        key = lambda individual:
individual. backtester.
return_results(
            symbol = '-',
            start_date = '-',
            end_date = '-',

)['fitness_function'],
        reverse = True
```

Figure 23 Method for selecting individuals
Source: Own elaboration

The next method is called "crossover", as its name suggests, its function is to combine the best individuals to obtain "offspring", trying to obtain the best combinations of these. First of all, we have the variable "selected" in which the best individuals are saved, followed by "point" which will be the genetic cut-off point between parents and offspring, and finally the "father" list in which the parents will be saved. Now, with a cycle all the individuals will be cycled through to reproduce them, to continue with the selection of two parents and the elimination of these once they are selected, to conclude by saving the individual object in the list of the parents. Continuing with the process, a "crossover" point must be calculated, looking for a random point between the first element of the genes and the last one.

Once the point is calculated, we proceed to replace the first value of the list up to the calculated point by the values of the parent list in the same way, that is, from the first value to the calculated point. This is repeated, only now it is to be done as well from the calculated point to the end. By the end of this process the population will be reproduced.

The last method used with genetic algorithms is the one that will perform mutation. This function will randomly add new genes to the population.

```
def mutation(self):
    for i in range(self.
generation_size):
        point = 0
        for j in range(self.
n_genes):
            point = np.
random. randint(0, self. n_genes)
            if np. random.
random() <= self. mutation_rate:
                new_gen = np.
random. randint(self.
gene_ranges[point][0], self.
gene_ranges[point][1])
                while new_gen
== self. population[i].
genes[point]:
                    new_gen =
np. random. randint(self.
gene_ranges[point][0], self.
gene_ranges[point][1])
                    self.
population[i]. genes[point] =
new_gen
```

Figure 24 Method that generates the mutation
Source: Own Elaboration

Initially, all the elements of the population will be run through and a variable will be declared which will store a randomly calculated point. With a cycle we will mutate "n" number of times according to the number of genes we have. After this, the random point mentioned above will be calculated, and then a conditional will be made that will ask if a random number is less than or equal to the mutation point, a new gene will be created and we must ensure that it is not equal to the one that is being replaced, so a cycle will be used that will be repeated while the new gene is equal to the previous one performing the same action over and over again until they get different, once this is fulfilled, the "point" gene will be changed by the "new_gen".

Finally, we will use a tool called Pyjuque, this name comes from Py-thon Ju-ju Qu-ant E-engine, which is oriented to the crypto world, it has everything you need to start doing algorithmic trading, it helps us save time in creating databases, code functions and other things. To use it just enter the command "pip install pyjuque" in the terminal. Once this is done we can start using it.

A file must be generated in which some parameters necessary for the bot to operate must be declared, then it can be better appreciated.

```
from pyjuque. Bot import defineBot
from BBStrategy import BBStrategy
import pandas as pd
import time
bot_config = {
    'name' : 'cryptobot9',

    'exchange' : {
        'name' : 'binance',
        'params' : {
            'api_key':
            'PC6UTw1cX5BB4p0I5nGWZJgaoo1iSW2glsnr
            FMC9WcuCWSFLxAmRsXylR1UnycQ7',
            'secret':
            'B28IOjlIWCD41FdWIn3gkrf1Scu6bWGptM
            BUS5vhpPdTF2PuGjggd982fccgUWG'
        },
    },
    'symbols' : ['CLV/USDT', 'ADA/USDT',
    'VET/USDT',
    'MINA/USDT'],

    'starting_balance' : 20.00,

    'strategy': {
        'class': BBStrategy,
        'params': {
            'bb_len' : 20,
            'n_std' : 2.0,
            'rsi_len' : 14,
            'rsi_overbought': 60,
            'rsi_oversold' : 40,
        },
    },
}
```

Figure 25 Pyjuque file
Source: Own Elaboration

Initially the necessary imports are made, such as the "defineBot", the strategy, the panda library and the time. Following this, a dictionary where certain parameters are established that are required to operate in a certain Exchange, i.e. the name of the Exchange, the keys provided to access its api, one public and one private. The symbols of the currencies to be operated, an initial balance and the strategy that was developed are also declared. For the strategy you must send parameters with which you want the indicators to operate.

```
'timeframe' : '5m',
'entry_settings' : {
    'initial_entry_allocation': 75,
    'signal_distance': 0.001
},
'exit_settings' : {
    'take_profit' : 0.2,
    'stop_loss_value': 1
},
display_status : True
}
def main():
    bot_controller =
    defineBot(bot_config)
    to True:
        try:
            bot_controller.
            executeBot()
        except KeyboardInterrupt:
            return
            time.sleep(15)
```

Figure 26 Execute Bot
Source: Own Elaboration

To complement this, you must also declare some other parameters, among the most important in which. We have the frequency with the candles that are analyzed, the percentage of profit you expecting and the percentage of the loss limit that you are allowed to lose. And finally the method "main" that will execute everything mentioned above.

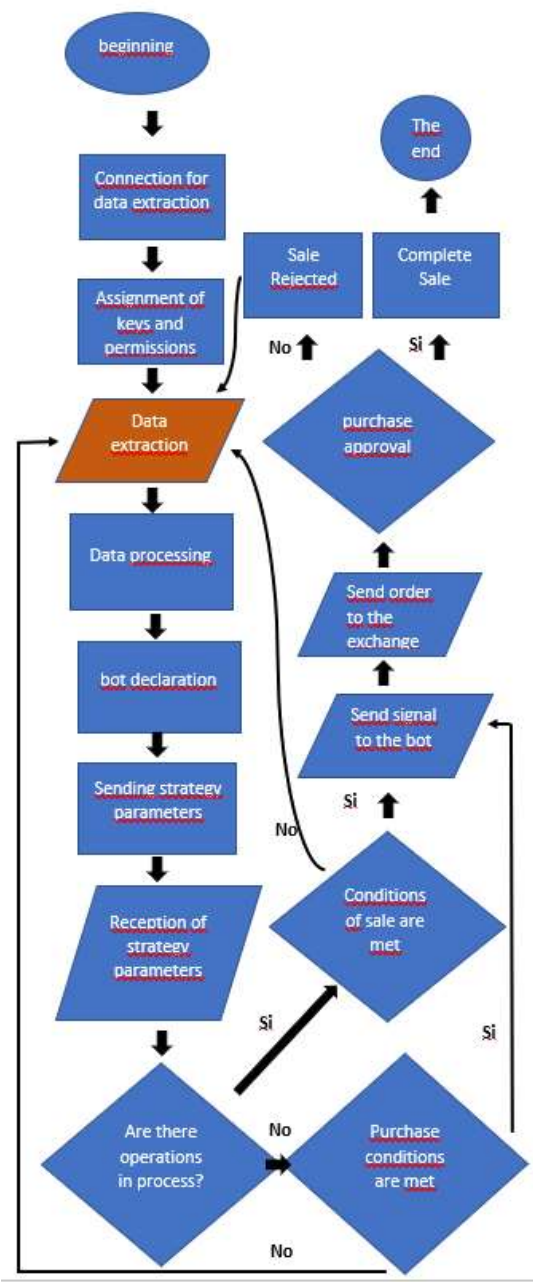


Figure 27 Flow diagram
Source: Own Elaboration

Results

It was concluded that working with a safe strategy and with enough time to try different and better options It is very likely to get beneficial incomes.

To evaluate the proper functioning of the indicators and the strategy, "backtesting" will be performed, It consists of taking historical information of the prices that a certain cryptocurrency has had and calculates the indicators of the program to compare the results obtained with those observed in that historical data. As for the strategies, they can also be executed and then you can check if they correctly execute the operations at the most appropriate time.

In order to perform the backtesting, historical data from cryptocurrency exchanges was required, which, as mentioned in the previous chapter, was obtained with the "ccxt" library through some code sentences. The result is shown below

```
PS C:\Backtesting> python 01get_data.py

['aax', 'acx', 'aofex', 'ascendex', 'bequant', 'bibox',
'bigone', 'binance', 'binanceus', 'bit2c', 'bitbank',
'bitbay', 'bitcoincom', 'bitfinex', 'bitfinex2', 'bitflyer',
'bitforex', 'bitget', 'bithumb', 'bitkk', 'bitmart',
'bitmax', 'bitmex', 'bitpanda', 'bitso', 'bitstamp',
'bitstamp1', 'bittrex', 'bitvavo', 'bitz', 'bl3p',
'bleutrade', 'braziliex', 'btcalpha', 'btcbox',
'btcmart', 'btctradeua', 'btcturk', 'buda', 'bw',
'bybit', 'bytetrade', 'cdax', 'cex', 'chilebit', 'coinbase',
'coinbaseprime', 'coinbasepro', 'coincheck',
'coinecheck', 'coinegg', 'coinex', 'coinfalcon',
'coinfoor', 'coingi', 'coinmarketcap', 'coinmate',
'coinone', 'coinspot', 'crex24', 'currencycom', 'delta',
'deribit', 'digifinex', 'equos', 'eterbase', 'exmo', 'exx',
'fcoin', 'fcoinjp', 'flowbtc', 'foxbit', 'ftx', 'gateio',
'gemini', 'gopax', 'hbtc', 'hitbtc', 'hollaex', 'huobijp',
'huobipro', 'idex', 'independentreserve', 'indodax',
'itbit', 'kraken', 'kucoin', 'kuna', 'lakebtc', 'latoken',
'lbank', 'liquid', 'luno', 'lykke', 'market', 'mixcoins',
'ndax', 'novadax', 'oceanex', 'okcoin', 'okex',
'paymium', 'phemex', 'poloniex', 'probit', 'qtrade',
'rightbtc', 'ripio', 'southxchange', 'stex', 'surbitcoin',
'therock', 'tidebit', 'tidex', 'timex', 'upbit', 'vbtc', 'vcc',
'wavesexchange', 'whitebit', 'xbtce', 'xena', 'yobit',
'zaif', 'zb']
```

Figure 28 ccxt library test
Source: Own Elaboration

As shown in the previous example, it was possible to extract from the "ccxt" library the names of the different exchanges from which information about their cryptocurrencies can be obtained. With this test we can check which library was installed and is being used correctly. A test was also performed for the Simple Moving Average indicator.

[500 rows x 5 columns]						
PS C:\Backtesting\ana> python 01_sma.py						
	close	volume	date	sma_ours	sma_target	
0	39519.93	276.27537	2022-04-22 20:38:00	NaN	NaN	
1	39623.54	331.47312	2022-04-22 20:45:00	NaN	NaN	
2	39607.32	192.40663	2022-04-22 21:00:00	NaN	NaN	
3	39663.83	263.35132	2022-04-22 21:15:00	NaN	NaN	
4	39622.90	135.49320	2022-04-22 21:30:00	NaN	NaN	
...						
495	39264.44	230.15400	2022-04-28 00:15:00	39188.188	39188.188	
496	39314.78	434.34164	2022-04-28 00:30:00	39195.629	39195.629	
497	39385.87	416.49159	2022-04-28 00:45:00	39212.505	39212.505	
498	39543.96	1040.09013	2022-04-28 01:00:00	39258.701	39258.701	
499	39519.72	43.49674	2022-04-28 01:15:00	39303.239	39303.239	

Figure 29 Simple Moving Average
Source: Own Elaboration

The calculation was made with a certain number of candles from the historical data, and this was the result. As it can be seen the code that generates this indicator all together with the table of the other information works correctly calculating the SMA only where it is indicated. This can be seen in the last column called "sma_target", and you can also check that this is correct with the other data being printed on the screen.

For the "RSI" indicator, a test was performed in the same way as the previous one, taking historical data from the Exchange.

```
PS C:\Backtesting\ema> python 03_rsi.py
```

	close	volume	date	rsi_ours	rsi_target
0	43380.94	929.11382	2022-04-07 06:00:00	NaN	NaN
1	43445.33	1246.42515	2022-04-07 07:00:00	100.000000	NaN
2	43521.92	964.22773	2022-04-07 08:00:00	100.000000	NaN
3	43457.98	1191.87092	2022-04-07 09:00:00	66.449650	NaN
4	43409.73	954.90136	2022-04-07 10:00:00	52.213586	NaN
...
495	39221.97	827.47953	2022-04-27 21:00:00	54.789268	54.789268
496	39074.34	884.48570	2022-04-27 22:00:00	51.736436	51.736436
497	39235.72	629.92794	2022-04-27 23:00:00	54.638184	54.638184
498	39385.87	1321.56279	2022-04-28 00:00:00	57.215608	57.215608
499	39612.20	948.03094	2022-04-28 01:00:00	60.828594	60.828594

Figure 30 RSI
Source: Own Elaboration

Figure 30 shows the extracted data to verify that the calculation is correct. The "RSI" can be visualized in the last two columns, the reason why we have two columns is because one was calculated manually with code statements and the other column was calculated by the pandas_ta library function. By observing both columns you can deduce that both are the same and therefore correct.

The last indicator we used was the Bollinger Bars. For this one, the calculation was made in the same way as the previous ones, i.e., with the same data. The result is shown below.

```
PS C:\Backtesting> python 04_bollbands.py
```

Target				
	BBL_20_20.0	BBM_20_20.0	BBU_20_20.0	
	BBB_20_20.0	BBB_20_20.0	BBP_20_20.0	
	BBP_20_20_20.0			
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN
495	32962.502780	38884.6720	44806.841220	
	30.460173	0.528478		
496	33333.655724	38917.0615	44500.467276	
	28.693871	0.514084		
497	33880.167303	38961.6885	44043.209697	
	26.084707	0.526964		
498	34277.228335	39010.4355	43743.642665	
	24.266364	0.539660		
499	34486.951875	39070.5765	43654.201125	
	23.463307	0.559164		
[500 rows x 5 columns] [500 rows x 5 columns]				
Ours				
	BBL	MID	BBU	
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN
495	32808.654583	38884.6720	44960.689417	
496	33188.608043	38917.0615	44645.514957	
497	33748.157756	38961.6885	44175.219244	
498	34154.267413	39010.4355	43866.603587	
499	34367.876861	39070.5765	43773.276139	

Figure 31 Bollinger Bars
Source: Own Elaboration

As we can see in the results we have two different tables showing the calculation of the Bollinger Bars, the first one is calculated by the pandas_ta library and the second one was calculated manually with code statements. In this case the first one shows more complete information of the bars, but nevertheless the calculations coincide and both are correct.

To analyze the behavior of the programmed strategy, the test was carried out with historical data of some cryptocurrencies, and thus establishing percentages of profit, loss, time periods, among other things, in order to observe if the operation is correct or if certain parameters, variables, indicators, etc. had to be adjusted.

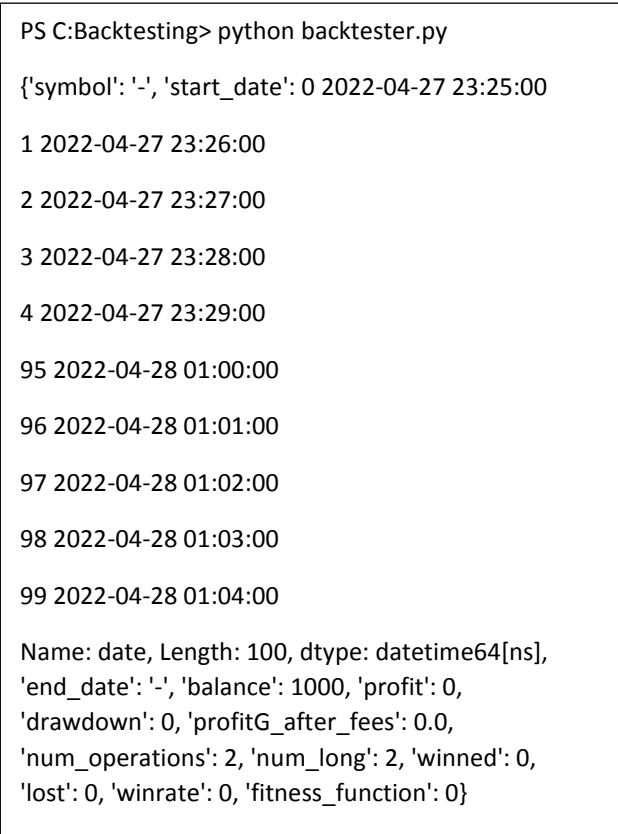


Figure 32 Strategy backtesting results
Source: Own Elaboration

The results show some data such as the dates on which the execution of the strategy was simulated, start and end dates, total number of trades, win-loss, this execution was carried out for a short time and the conditions for opening orders were not met, but the actions carried out are the most appropriate for the scenario proposed.

In the genetic algorithms, all the indicators used for the strategy were taken as parameters and the most optimal combination of these indicators was searched for 20 generations. The result is shown below.

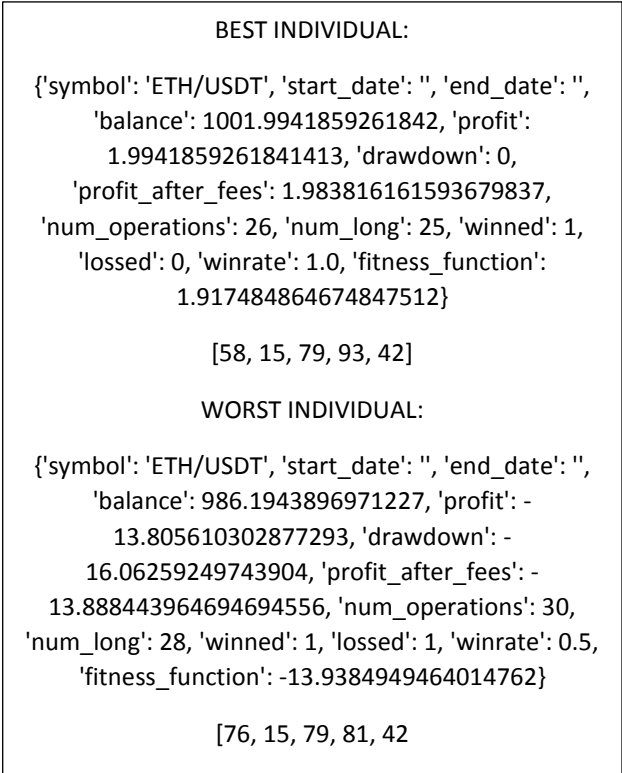


Figure 33 Results of genetic algorithms
Source: Own Elaboration

As it can be seen in Figure 33, we ran from 0 to 19 generations, followed by the best individual along with the results obtained by looking for the best and worst combination of the indicators This helps to optimize the strategy by finding the best conditions to say whether it is time to buy or sell. The test of the bot with real money was performed on Monday, April 4 of this year at 13:00 hours in the Binance Exchange with the cryptocurrencies CLV, ADA, VET, MINA with an initial balance of 20 dollars, a profit percentage of 2% and a loss percentage of 1%. The time this test lasted was approximately one hour in which the bot was trading and detecting signals in order to know when to enter or exit, finding a buy opportunity and a sell opportunity a few seconds after the cryptocurrency "VET", obtaining a profit of approximately 2%. in the chart below a trade can be seen saved in the order history in Binance.



Figure 34 Order history of the 1st execution
Source: Own Elaboration

After this execution it was possible to deduce that the bot could open and close operations correctly depending on how it was configured. So it was decided to perform a full 24-hour test starting that same day at approximately 4 pm and ending the next day at the same time. Below is a screenshot showing most of the operations performed during that period of time, in this screenshot We can see some data about the orders, among the most important we have gotten the time and date that was executed, the type of order (buy or sell), the price at which it was bought or sold and its status. The complete results of this test can be seen in Annex A.



Figure 35 Order history of the 2nd execution
Source: Own Elaboration

When analyzing the results carefully and at the same time seeing how cryptocurrency prices behaved, it is observed that the program had lost profit opportunities since the purchase was made, the price of some cryptocurrencies rose enough to make a profit, but the percentage initially indicated to the bot was somehow ambitious and therefore complicated to fulfill. therefore it was decided to modify that value for one that was not so high in order to take advantage of all possible opportunities. Once the parameter was changed, another test was performed on the same day (April 5, 2022). The following image shows some of the operations performed during that time. The complete results of this test can be seen in Annex B.

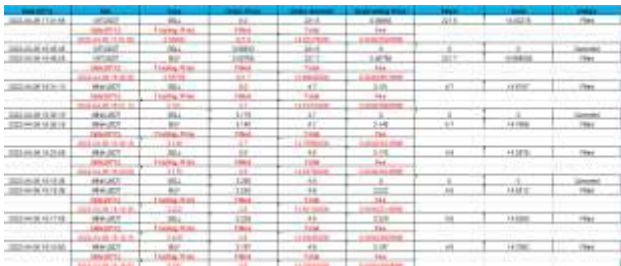


Figure 36 Order history of the 3rd execution
Source: Own Elaboration

Finally, on this test it was observed that the price of some assets was falling and when this happened the bot would have to activate the maximum percentage of loss and close the operation to protect the trader from loss, but apparently that percentage of loss was too low, so there was a considerable loss of assets. That is why it was decided to stop the test execution so as to change that parameter.

Currently (date of preparation of this report) the bot continues to operate with the above mentioned adjustments. This project will continue to be developed to improve, optimize and/or create new trading strategies to maximize investor's profitability.

Conclusion

Algorithmic trading is a tool that allows to automate processes that sometimes may be done better by a machine than by a human, however, it may not always be so. This tool is rather used to automate strategies and to study which things work in the market and which ones do not. It is something that may be very helpful, but it does not make anyone rich overnight whatsoever.

A cryptobot is very useful in many aspects compared to a human, for instance when calculating indicators quicker, opening and closing orders, does not affect the factor of emotions that in humans is very common to influence decisions made when trading, they can also be operating 24 hours a day 7 days a week without resting and then taking advantage of all the opportunities that may occur at any time.

In this bot we introduced a basic strategy that operates well in the Exchange sending the most optimal possible parameters according to the situation that arises, however, we could improve its efficiency to operate with minimal human supervision, as well as increase the ability to predict the market through mathematical models, and the correct use of some indicators to find patterns or any other advantage that can be exploited that cannot be detected at a naked eye or if is very difficult to calculate for a person.

Algorithmic trading of cryptocurrencies is an environment where you can take advantage of several factors that may help increase the profitability of a investor, this all through taking advantage of all the tools that are available today such as programming languages, libraries to calculate indicators and extracting data, among many others that can be used in the most convenient way. This project is planned to improve it and continue it. In a near future we will create new strategies and ways to increase efficiency when predicting the market.

References

- Andbank (April 15, 2014). *Andbank*. Retrieved from Andbank: <https://www.andbank.es/observatoriodelinversor/que-es-el-scalping/#:~:text=The%20scalping%2C%20ta mbi%C3%A9n%20known%20as,operations%20in%20questions%20of%20seconds>.
- AvaTrade (2022). *AvaTrade* . Retrieved from <https://www.avatrade.es/educacion/professional-trading-strategies/macd-trading-strategies>
- Binance Academy (December 02, 2021). *Binance Academy*. Retrieved from Binance Academy: <https://academy.binance.com/es/articles/what-is-arbitrage-trading>
- Bit2Me Academy (July 2021). *Bit2Me Academy*. Retrieved from Bit2Me Academy: <https://academy.bit2me.com/>
- Broseta, A. (October 04, 2016). *Rankia*. Retrieved from <https://www.rankia.cl/blog/analisis-ipsa/3346628-como-calculan-bandas-bollinger-formula-interpretacion>
- Burrueco, D. (February 2019). *InteractiveChaos*. Retrieved from InteractiveChaos: <https://interactivechaos.com/es/manual/tutorial-de-matplotlib/tutorial-de-matplotlib>
- Caminiti, G. (August 31, 2021). *Coder House*. Retrieved from Coder House: https://www.coderhouse.com.mx/blog/que-es-python?utm_term=&utm_campaign=14822878552&utm_source=google_performance_max&utm_medium=cpc
- Cardellino, F. (March 20, 2021). *Freecodecamp*. Retrieved from Freecodecamp: 2021.
- Caro Mora, C. (August 18, 2021). *Admiral Markets*. Retrieved from Admiral Markets: <https://admiralmarkets.com/es/education/articles/forex-basics/que-es-trading>
- Caro Mora, C. (January 17, 2022). *Admiral Markets*. Retrieved from Admiral Markets: <https://admiralmarkets.com/es/education/articles/automated-trading/trading-algoritmico>
- Chacón, J. L. (March 22, 2021). *Profile*. Retrieved from Profile: <https://profile.es/blog/pandas-python/>
- CMC Markets (January 2022). *CMC Markets*. Retrieved from CMC Markets: <https://www.cmcmarkets.com/es-es/aprenda-a-operar-con-criptomonedas/>
- Cryptonews. (2022). *cryptonews.com*. Retrieved from cryptonews.com: <https://es.cryptonews.com/guias/los-10-principales-exchanges-descentralizados-dex-en-2022.htm>
- Datademia. (March 02, 2020). *datademia.es*. Retrieved from datademia.es: <https://datademia.es/blog/author/sebadmin25>
- Expansion (2021). What is bitcoin and how does it work? *Expansion*.
- Garduño Juárez, R. (October 12, 2018). *Conogasi.org*. Retrieved from <https://conogasi.org/articulos/algoritmos-geneticos/>: <https://conogasi.org/articulos/algoritmos-geneticos/>
- Gonzalez, L. (September 21, 2018). *Aprendeia*. Retrieved from Aprendeia: <https://aprendeia.com/introduccion-a-numpy-python-1/>
- GuiaBitcoin (March 08, 2022). *GuiaBitcoin*. Retrieved from <https://guiabitcoin.co/robot-bitcoin/cryptohopper#Que-es-Cryptohopper>
- Gunbot.es (2022). *Gunbot.co.uk*. Retrieved from Gunbot.co.uk: <https://gunbot.es/>.

Hernández Barrera, G., Sánchez Ruanova, S., & Ordóñez Sánchez, S. G. (July 2018). *Fintech law and regulation for Bitcoin cryptocurrency*. Retrieved from facpya: http://www.web.facpya.uanl.mx/vinculategica/Vinculategica6_2/55_Ordo%C3%B1ez_Hernandez_Sanchez.pdf

IG (January 2022). *IG.com*. Retrieved from IG: <https://www.ig.com/es/ethereum-trading/que-es-ether-y-como-funciona>

I'mnovation#hub (2021). *I'mnovation#hub*. Retrieved from I'mnovation#hub: <https://www.imnovation-hub.com/es/transformacion-digital/que-es-blockchain-y-como-funciona-esta-tecnologia/>

InvertirenBolsa. (2022). *invertirenbolsa.mx*. Retrieved from <https://www.invertirenbolsa.mx/plataformas-de-trading/plataformas-criptomonedas/#:~:text=We%20can%20say%20that%20it%20is%20a,se%20will%20be%20of%20normal%20currencies>.

Konfio (January 2022). *Konfio*. Retrieved from Konfio: <https://konfio.mx/>

malwarebytes. (n.d.). *malwarebytes*. Retrieved from <https://es.malwarebytes.com/cryptojacking/>

Martín Garrido, I. (February 17, 2022). *Roams*. Retrieved from Roams: <https://finanzas.roams.es/academia/criptomonedas/bot-trading/>

Montero Castellanos, Y. (February 28, 2014). *Economipedia.com*. Retrieved from Economipedia.com: <https://economipedia.com/definiciones/arbitraje.html>

Monex (October 30, 2021). *Monex*. Retrieved from Monex: <https://blog.monex.com.mx/instrumentos-financieros/clasificacion-de-los-mercados-bursatiles-en-los-mercados-financieros>

Müller, M., Rougier, N., & Varoquau, G. (August 21, 2013). *Claudiovz*. Retrieved from Claudiovz: <https://claudiovz.github.io/scipy-lecture-notes-ES/intro/matplotlib/matplotlib.html>

My Satoshi World (July 05, 2020). *My Satoshi World*. Retrieved from My Satoshi World: <https://mysatoshiworld.com/blog/2020/07/05/bituniverse-el-portafolio-tracker-y-robot-de-trading-gratuito/>

Pachón Díaz, M. (July 30, 2020). *Business Insider*. Retrieved from Business Insider: <https://www.businessinsider.es/ethereum-cumple-6-anos-reto-finanzas-descentralizadas-678183>

Pastorino, C. (September 04, 2018). *Welivesecurity*. Retrieved from Welivesecurity: <https://www.welivesecurity.com/la-es/2018/09/04/blockchain-que-es-como-funciona-y-como-se-esta-usando-en-el-mercado/>

Plus500 (2022). *Pluss500*. Retrieved from Pluss500: <https://www.plus500.com/es/Instruments/XRPUSD/What-is-Ripple-XRP~1>

Puente, J. (January 04, 2020). *Background*. Retrieved from <http://www.fondos.com/blog/fondos-de-inversion-de-criptomonedas>

ripio launchpad. (February 02, 2022). *launchpad.ripio.com*. Retrieved from launchpad.ripio.com: <https://launchpad.ripio.com/guias-capitulos/brokers-de-criptomonedas>

Rus Arias, E. (October 05, 2021). *Economipedia.com*. Retrieved from Economipedia.com: <https://economipedia.com/definiciones/exchange-de-criptomonedas.html>

Saenz, F. (April 29, 2020). *Rankia*. Retrieved from <https://www.rankia.cl/blog/analisis-ipsa/2039072-medias-movil-simple-exponencial-ponderada-formulas-ejemplos>

Sanchez Alberca, A. (September 21, 2020). *Learn with Alf*. Retrieved from Learn with Alf: <https://aprendeconalf.es/docencia/python/manu al/>

Sánchez, J. (September 12, 2021). *Economia3*. Retrieved from Economia3: <https://economia3.com/que-es-dash-criptomonedas/>

Santander Universities (April 09, 2021). *Santander Scholarships*. Retrieved from <https://www.becas-santander.com/es/blog/python-que-es.html>

Solunion (August 26, 2021). *Solunion*. Retrieved from <https://www.solunion.cl/blog/que-es-y-para-que-sirve-la-tecnologia-blockchain/>

Staff kueski (March 03, 2020). *kueski*. Retrieved from kueski: <https://kueski.com/blog/finanzas-personales/diccionario-finanzas/ley-fintech/>

Tradepark (May 21, 2021). *Tradepark*. Retrieved from Tradepark: <https://tradespark.la/blog/articulos/python-for-at/cuales-son-las-principales-estrategias-de-trading-algoritmico/#:~:text=One%20of%20the%20languages%20of,Python%20and%20there%20is%20a%20lot%20of%20content>

Trecet, J. (March 12, 2019). *Business Insider*. Retrieved from Business Insider: <https://www.businessinsider.es/scalping-quick-trading-como-saber-ti-362323>

University of Alcalá (2021). *University of Alcalá*. Retrieved from master-data-scientist.com: <https://www.master-data-scientist.com/pandas-herramienta-data-science/>

Yunius. (December 25, 2018). *archive.yunius*. Retrieved from www.archivo.yunius.com: <http://archivo.yunius.com/la-criptomoneda-mexicana/#:~:text=Being%C3%A1%20with%20all%20safety%20in,for%20your%20presentation%20in%20society>

Annex A

Table of Results

Date(UTC)	Pair	Type	Order Price	Order Amount	AvgTrading Price	Filled	Total	status
2022-04-05 21:29:02	VEUSDT	SELL	0.07718	194.9	0.07718	194.9	15.042382	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 21:29:42	0.07718	194.9	15.04238200	0.00025178NB			
2022-04-05 20:49:44	VEUSDT	SELL	0.07847	194.9	0	0	0	Canceled
2022-04-05 20:49:42	VEUSDT	BUY	0.07893	194.9	0.07891	194.9	14.986759	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 20:49:42	0.07891	194.9	14.98675900	0.00025032NB			
2022-04-05 19:19:08	CUUSDT	SELL	0.0	33	0.448	33	14.817	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 19:19:08	0.448	33.0	14.81700000	0.00024732NB			
2022-04-05 19:21:43	CUUSDT	SELL	0.463	33	0	0	0	Canceled
2022-04-05 19:20:15	CUUSDT	BUY	0.454	33	0.454	33	14.982	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 19:21:42	0.454	33.0	14.98200000	0.00025118NB			
2022-04-05 19:12:19	ADAUUSD	SELL	0.0	12.5	1.182	12.5	14.775	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 19:12:19	1.182	12.5	14.77500000	0.00024818NB			
2022-04-05 18:50:07	ADAUUSD	SELL	1.219	12.5	0	0	0	Canceled
2022-04-05 18:50:04	ADAUUSD	BUY	1.194	12.5	1.194	12.5	14.925	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 18:50:04	1.194	12.5	14.92500000	0.00024832NB			
2022-04-05 17:52:59	ADAUUSD	SELL	1.197	12.6	1.197	12.6	15.0822	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 17:54:11	1.197	12.6	15.08220000	0.00024876NB			
2022-04-05 17:30:30	ADAUUSD	SELL	1.2	12.6	0	0	0	Canceled
2022-04-05 16:19:10	ADAUUSD	SELL	1.208	12.6	0	0	0	Canceled
2022-04-05 16:06:08	ADAUUSD	BUY	1.184	12.6	1.184	12.6	14.9194	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 16:06:08	1.184	12.6	14.91940000	0.00024876NB			
2022-04-05 13:32:29	VEUSDT	SELL	0.0	189.3	0.07829	189.3	14.820297	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 13:32:29	0.07829	189.3	14.82029700	0.00024832NB			
2022-04-05 07:02:36	VEUSDT	SELL	0.08081	189.3	0	0	0	Canceled
2022-04-05 07:02:34	VEUSDT	BUY	0.07923	189.3	0.07923	189.3	14.98029	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 07:02:34	0.07923	189.3	14.98029000	0.00024832NB			
2022-04-05 06:55:50	ADAUUSD	SELL	0.0	12.4	1.196	12.4	14.8304	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 06:55:50	1.196	12.4	14.83040000	0.00024832NB			
2022-04-05 04:55:23	ADAUUSD	SELL	1.233	12.4	0	0	0	Canceled
2022-04-05 04:54:47	ADAUUSD	BUY	1.209	12.4	1.209	12.4	14.9916	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 04:55:09	1.209	12.4	14.99160000	0.00024703NB			
2022-04-05 04:11:21	MINAUSD	SELL	3.21	4.7	3.21	4.7	15.087	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 04:54:23	3.210	4.7	15.08700000	0.00024882NB			
2022-04-05 03:52:15	MINAUSD	BUY	3.147	4.7	3.147	4.7	14.7939	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 04:11:09	3.147	4.7	14.79390000	0.00024832NB			
2022-04-04 22:36:40	CUUSDT	SELL	0.467	32.7	0.467	32.7	15.2709	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-05 01:28:36	0.467	32.7	15.27090000	0.00025578NB			
2022-04-04 22:35:09	CUUSDT	BUY	0.458	32.7	0.458	32.7	14.9766	Filled
	Date(UTC)	Trading Price	Filled	Total	Fee			
	2022-04-04 22:36:35	0.458	32.7	14.97660000	0.00025238NB			

Annex B

Table of Results 2

Date(UTC)	Pair	Type	Order Price	Order Amount	AvgTrading Price	Filled	Total	status
2022-04-06 17:01:59	VEUUSD	SELL	0.0	221.5	0.00000000	221.5	14.62079	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:45:45	VEUUSD	SELL	0.06833	221.8	0.00000000	0	0	Cancelled
2022-04-06 16:45:25	VEUUSD	BUY	0.06765	221.7	0.06765	221.7	14.99805	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:31:13	ADALUSD	SELL	0.0	4.7	0.00000000	4.7	14.8347	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:30:19	ADALUSD	SELL	3.171	14.54710000	0.00000000	0	0	Cancelled
2022-04-06 16:30:18	MNVALUSD	BUY	3.148	4.7	3.148	4.7	14.7596	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:30:16	ADALUSD	SELL	3.148	4.7	14.75960000	0.00000000	14.7596	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:30:09	ADALUSD	SELL	3.172	4.5	14.26750000	0.00000000	14.2675	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:18:38	MNVALUSD	BUY	3.225	4.5	14.26750000	0.00000000	14.2675	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:18:36	MNVALUSD	BUY	3.225	4.5	14.26750000	0.00000000	14.2675	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:18:30	ADALUSD	SELL	3.225	4.5	14.26750000	0.00000000	14.2675	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:17:09	MNVALUSD	BUY	3.229	4.5	14.26750000	0.00000000	14.2675	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:16:50	ADALUSD	SELL	3.192	4.5	14.50000000	0.00000000	14.5000	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:16:17	MNVALUSD	BUY	3.192	4.5	14.50000000	0.00000000	14.5000	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:16:17	MNVALUSD	BUY	3.192	4.5	14.50000000	0.00000000	14.5000	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:15:22	MNVALUSD	BUY	3.203	4.5	0	0	0	Cancelled
2022-04-06 16:15:04	MNVALUSD	BUY	3.231	4.5	3.231	4.5	14.8625	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:15:12	ADALUSD	SELL	3.231	0.4	0.00000000	0.00000000	0.0000	Cancelled
2022-04-06 16:15:10	ADALUSD	SELL	3.231	0.4	0.00000000	0.00000000	0.0000	Cancelled
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:35:06	ADALUSD	SELL	0.408	37.1	0.408	37.1	15.1969	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:35:04	ADALUSD	SELL	0.404	37.1	0.404	37.1	14.9984	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:35:04	ADALUSD	SELL	0.404	37.1	14.99840000	0.00000000	14.9984	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:28:55	ADALUSD	SELL	0.402	37.1	14.99840000	0.00000000	14.9984	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:25:20	ADALUSD	SELL	0.411	14.99840000	0.00000000	0	0	Cancelled
2022-04-06 16:25:16	ADALUSD	SELL	0.407	36.8	0.407	36.8	14.9776	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 16:25:16	ADALUSD	SELL	0.407	36.8	14.97760000	0.00000000	14.9776	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 14:10:26	ADALUSD	SELL	1.116	13.5	1.116	13.5	15.0525	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 14:10:23	ADALUSD	SELL	1.110	13.5	15.05250000	0.00000000	15.0525	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 14:10:23	ADALUSD	SELL	1.104	13.5	1.104	13.5	14.904	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 14:10:23	ADALUSD	SELL	1.104	13.5	14.90400000	0.00000000	14.9040	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 14:10:17	ADALUSD	SELL	1.116	13.4	1.116	13.4	14.9544	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 13:37:25	ADALUSD	SELL	1.116	13.4	14.95440000	0.00000000	14.9544	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 13:35:15	ADALUSD	BUY	1.114	13.4	1.114	13.4	14.9276	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 13:35:15	ADALUSD	BUY	1.114	13.4	14.92760000	0.00000000	14.9276	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 12:59:13	ADALUSD	BUY	1.111	13.5	1.111	13.5	14.9585	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 12:59:11	ADALUSD	BUY	1.109	13.5	14.95850000	0.00000000	14.9585	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 12:56:30	ADALUSD	BUY	1.111	13.5	14.95850000	0.00000000	14.9585	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 12:55:19	ADALUSD	BUY	1.109	13.5	1.109	13.5	14.9575	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 12:54:21	ADALUSD	BUY	1.104	13.5	14.95750000	0.00000000	14.9575	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:35:12	ADALUSD	BUY	1.119	13.4	1.119	13.4	14.9546	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:34:52	ADALUSD	BUY	1.119	13.4	14.95460000	0.00000000	14.9546	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:34:52	ADALUSD	BUY	1.117	13.4	1.117	13.4	14.9676	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:34:52	ADALUSD	BUY	1.117	13.4	14.96760000	0.00000000	14.9676	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:34:36	VEUUSD	SELL	0.07152	210	0.07152	210	15.0182	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:34:36	VEUUSD	SELL	0.07152	210	15.01820000	0.00000000	15.0182	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:33:36	VEUUSD	BUY	0.07138	210.1	0.07138	210.1	14.99838	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:33:36	VEUUSD	BUY	0.07138	210.1	14.99838000	0.00000000	14.99838	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:33:21	ADALUSD	BUY	1.119	13.4	1.119	13.4	14.9546	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:33:21	ADALUSD	BUY	1.119	13.4	14.95460000	0.00000000	14.9546	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:33:17	ADALUSD	BUY	1.117	13.4	1.117	13.4	14.9676	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:27:54	ADALUSD	BUY	1.117	13.5	14.95850000	0.00000000	14.9585	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:27:54	ADALUSD	BUY	1.111	13.5	1.111	13.5	14.7953	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:21:41	ADALUSD	BUY	1.125	13.3	0	0	0	Cancelled
2022-04-06 11:21:38	ADALUSD	BUY	1.125	13.3	1.125	13.3	14.8935	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:21:38	ADALUSD	BUY	1.123	13.3	14.89350000	0.00000000	14.8935	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:21:20	MNVALUSD	BUY	3.092	4.7	3.092	4.7	14.5324	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:21:19	MNVALUSD	BUY	3.086	4.8	14.81280000	0.00000000	14.8128	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:21:20	MNVALUSD	BUY	3.086	4.8	14.81280000	0.00000000	14.8128	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:02:43	ADALUSD	SELL	0.0	13	1.122	13	14.586	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 11:02:43	ADALUSD	SELL	1.122	13.0	14.58600000	0.00000000	14.5860	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:45:08	ADALUSD	SELL	1.106	13.2	1.106	13.2	0	Cancelled
2022-04-06 10:45:08	ADALUSD	SELL	1.134	13.2	1.134	13.2	14.968	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:45:08	ADALUSD	SELL	1.134	13.2	14.96800000	0.00000000	14.9680	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:40:29	VEUUSD	BUY	0.07207	205.9	0.07207	205.9	15.0207	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:40:27	VEUUSD	BUY	0.07205	205.9	0.07205	205.9	14.99515	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:33:51	VEUUSD	SELL	0.07337	204.8	0.07337	204.8	15.02176	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:34:25	VEUUSD	BUY	0.07301	204.8	15.02176000	0.00000000	15.02176	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:24:42	VEUUSD	BUY	0.07322	204.8	0.07322	204.8	14.95406	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:24:42	VEUUSD	BUY	0.07322	204.8	14.95406000	0.00000000	14.95406	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:20:43	VEUUSD	BUY	0.07301	205.3	0.07301	205.3	15.02756	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:24:23	VEUUSD	BUY	0.07320	205.3	15.02756000	0.00000000	15.02756	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:20:21	VEUUSD	BUY	0.07305	205.3	0.07305	205.3	14.99795	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:19:56	VEUUSD	BUY	0.07377	203.6	0.07377	203.6	15.01922	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 10:19:11	VEUUSD	BUY	0.07377	203.6	15.01922000	0.00000000	15.01922	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 09:58:13	VEUUSD	BUY	0.07360	203.7	0.07361	203.7	14.98497	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 09:58:11	VEUUSD	BUY	0.07361	203.7	14.98497000	0.00000000	14.98497	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 09:38:37	ADALUSD	BUY	1.142	13	1.142	13	14.846	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 09:38:37	ADALUSD	BUY	1.142	13.0	14.84600000	0.00000000	14.8460	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 09:35:22	ADALUSD	BUY	1.14	13.1	1.14	13.1	14.934	Filed
	Demot(UTC)	Trading Price	Filled	Total	Fee			
2022-04-06 09:35:22	ADALUSD	BUY	1.14	13.1	14.93400000			