mt5b3: A Framework for Building Autonomous Traders

Paulo André Lima de Castro
Autonomous Computational Systems Lab
Technicological Institute of Aeronautics (ITA - Instituto Tecnológico de Aeronáutica)
São José dos Campos-SP, Brazil
pauloac@ita.br

***Abstract*—Autonomous trading robots have been studied in artificial intelligence area for quite some time. Many AI techniques have been tested in finance field including recent approaches like convolutional neural networks and deep reinforcement learning. There are many reported cases, where the developers are successful in creating robots with great performance when executing with historical price series, so called backtesting. However, when these robots are used in real markets or data not used in their training or evaluation frequently they present very poor performance in terms of risks and return. In this paper, we discussed some fundamental aspects of modelling autonomous traders and the complex environment that is the financial world. Furthermore, we presented a framework that helps the development and testing of autonomous traders. It may also be used in real or simulated operation in financial markets. Finally, we discussed some open problems in the area and pointed out some interesting technologies that may contribute to advance in such task. We believe that mt5b3 may also contribute to development of new autonomous traders.**

I. Introduction

An autonomous trader must be able of selecting, buying and selling financial assets in order to fulfill it's investor requirements about risk and return within the investor's horizon of investment. This period may range from very some seconds or fractions of seconds (high frequency trading [1]), to longer periods as several days or even years. It is an autonomous agent that perceives and acts in a very particular environment: financial markets, which brings several complex features that may make it very challenging for autonomous agents. It is important to note that there are also ethical and legal implications that should be taken in consideration in the process of building automated trading robots. We recommend Wellman and Rajan's paper [2] about ethical issues in Autonomous Trading Agents.

In this paper, we propose a framework for building and testing trading robots called mt5b3. This framework allow the development of autonomous traders on python language and access markets through MetaTrader 5 platform [3]. You may perform historical simulations (often called backtest), with you traders in order to evaluate theirs performance. Furthermore, it is possible the use of same trader with minor changes in real operations in any market accessible through Metatrader5 in netting accounting system [3] (check https://www.metatrader5.com/en/stocks-ecns, for a list accessible stock exchanges). It is possible to create Trading Robots using Neural networks, Random Forests, Support Vector Machines, Genetic Algorithms, Bayesian Networks, Reinforcement Learning, Deep Learning and other techniques using the wide range of available libraries in python.

We assume the reader is already familiar with Finance Theory, including Modern Portfolio Theory [4], Efficient Market Hypothesis [5], Capital Asset Pricing Model (CAPM) and Market micro-structure. If that is not the case, a very short introduction to these concepts are available in section 2 of [6]. The remain of this paper is organized as follows: section II discuss the main aspects of modelling autonomous traders, including some distinct characteristics of financial markets that make them an unique challenge for machine learning and artificial intelligence approaches. We propose and present our framework in section III. We provide several examples of autonomous traders and their source. In section IV, we point and discuss some open problems in the development high performance autonomous traders and we also present some AI related technologies that may be help in this process in the near future. Finally, we discuss some possible extensions and future work in section V.

II. Modelling Autonomous Traders

The spectrum of used AI techniques in finance field is wide and it includes since reinforcement Learning [7], [8], multiagent systems [9], [10] complex networks [11], decision trees [12], genetic algorithms [13], random forests [14] to more recent approaches like convolutional neural networks [15] and deep reinforcement learning [16]. Regardless of the picked AI technology, there are some aspects that are always present. We discuss these aspects in in this section: the environment where autonomous trader perceive and act, two possible conceptual architectures, four different types of data that autonomous trader should be able to handle and a suitable development process for AT.

A. Financial environment for AI agents

The problem of creating models with great performance in some previous known data set, but with bad performance in new data is well known in Machine Learning field and it is usually called **overfitting**. Nevertheless, this problem seems to be unknown or at least despised by many practitioners in autonomous trading. However, the dangers of failing to avoid overfitting are even more severe in the financial field for two reasons. Overfitting is more likely to happen in finance than in traditional machine learning problems, like face recognition. Second, overfitting in finance leads to unrealistic expectation about the performance and therefore it can be used to mislead investors to bet in overfitted autonomous trading systems, which could be built by naive or even malicious developers [17].

In financial environments, it is hard to separate signal (real information) from noise, because they are almost at same level and if the signal is clear, it will not be for long. Arbitrage (exploiting price differences of similar financial instruments on different markets) will decrease the signal-to-noise ratio, making easier to confuse signal and noise. This confusion may also lead to **overfitted** models.

In fact, it is not rare to see very high promises of return by trading strategy vendors in digital media. As pointed out by Prado [17] without control for backtest overfitting, good backtest performance may be an indicator for negative future results.

Unfortunately, it is not easy to prove that a given model is not overfitted and it is always important to note that past returns do not guarantee future returns. In fact, it is fundamental to have a deep understanding about the environment faced by trading strategies in order to avoid problems. By that we mean the reasons that make overfitting more likely to happen in financial robots and some common mistakes about testing trading robots, or the so-called trading backtesting. We tackle these issues in section II-A.

The environment (Financial Environment) where live the trading robots could be classified as: partially observable, sequential, stochastic, dynamic, continuous and multi-agent, according to Russell and Norvig taxonomy [18], which it is the most complex environment class pointed by them. It is also strategic in the sense that two active investors compete for a more accurate valuation of assets and their acts may change other economic agent's behavior. However, it does not really represents the whole complexity of the problem. More than stochastic, such environment is also **non-stationary process** (the probability distribution do change along the time).

Due this non-stationary feature, a specific robot may present great performance in a given period of time, but a terrible performance in the next period [9]. In fact, it can be checked by searching for structural breaks in financial time series. There are abundant empirical evidence which shows that there are structural breaks in financial markets [19]. Strong empirical evidence shows that various economic events can lead to structural changes. These events can be, for instance, the financial liberalization of emerging markets and integration of world equity markets, changes in exchange rate regimes such as the collapse of exchange rate systems and the introduction of a single currency in Europe [19]

Moreover, **different assets may require different information and models**. For instance, oil companies may be very sensitive to changing of petrol prices, probably it is not the same for banks. In another terms, a specific asset price can be very correlated to some time series, but that time series can have no relevant correlation with another asset price.

Investor may have different preferences regarding risk, return and time of the investment . It is common to observe different requirements according to the period of time the investor intends to keep his resources invested. This **investment horizon** may range from several years to few milliseconds. It may lead to strategy that can be very effective in very short horizons, but present poor performance in the long run [20]. Another aspect that should be addressed by any trading strategy, but it often forgotten is that people do not have the same **level of acceptable risk**. Some investors may present a much stronger risk aversion than others, for example. A trading robot must be aware of its investor's preferences, in order to trade appropriately.

Building trading strategies is a complex task and after building them, it is hard to tell which one is the most likely to have good performance in the near future. Developers picks their trading strategy usually through historical simulations (also called backtesting). The idea is simple: put the financial robot (with its strategy already defined) to run in a virtual environment created that feeds the robot with market information respecting the time barrier. It could be compared to send your robot in a time travel back to the beginning of that market behavior and then account how would be the robot performance, if it was available at that time and their existence would not change anything in the market behavior.

B. Autonomous Traders

In figure 1, we present a simple autonomous trader architecture the main conceptual components of any Autonomous Trader and its interactions with the environment from it gets information and submits its buy or sell orders. These components include required information for the agent to do a proper job (Market information, investor preferences and portfolio information), an algorithm (here called strategy) and two accessory services that act as sensors and actuators, data collector and order dispatcher, respectively. Those components are described next

- **Investor information**: it defines the goals that should be pursued by the trading robot. It should describe the investor preferences over maximum acceptable risk, besides the investor horizon of investment. The current investor's portfolio is also relevant, because it may change operations that are available or not to a given investor. Unfortunately many trading robot developers assume simply pursues the highest possible return regardless the risks. It is unrealistic for almost all investor, but unfortunately it is also a common implicit assumption in robot trading development. **Investor's Portfolio Information** registers the shares and capital to be managed by the trading strategy
- **Asset Data**: it keeps record of all relevant information regarding the target assets (technical, fundamentalist and others). Which are the relevant information is defined by the set of assets and the trading strategy. Observe that as

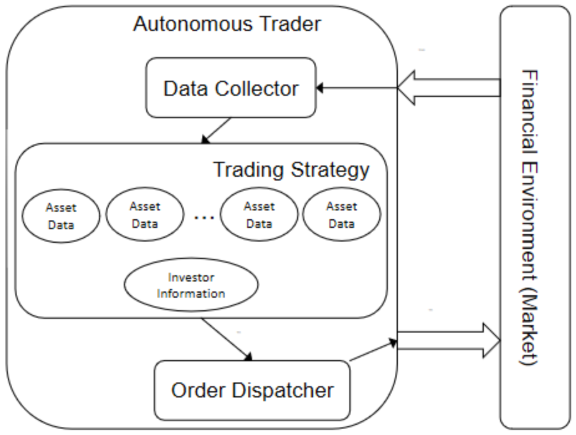


Fig. 1. Simple Autonomous Trader Architecture

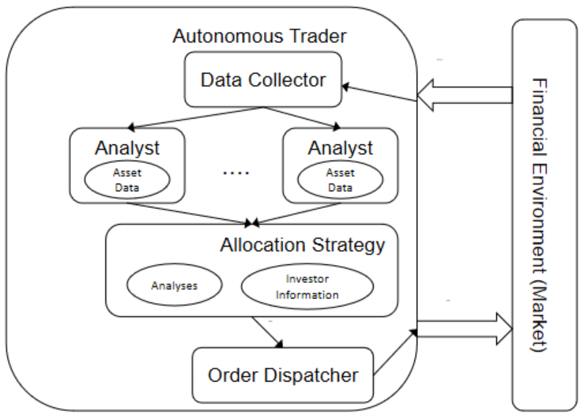
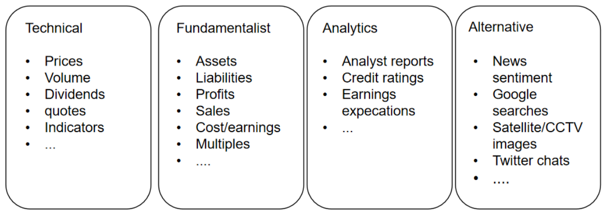


Fig. 2. Alternative Autonomous Trader Architecture



Technical	Fundamentalist	Analytics	Alternative
• Prices • Volume • Dividends • quotes • Indicators • ...	• Assets • Liabilities • Profits • Sales • Cost/earnings • Multiples •	• Analyst reports • Credit ratings • Earnings expecations • ...	• News sentiment • Google searches • Satellite/CCTV images • Twitter chats • ....

Fig. 3. Types of Financial Data

the number of target assets grows, the volume of asset data may become huge really fast, because each asset may require a significant amount of data.

- **Trading Strategy**: it is the implementation of the function that decides the Robot's actions according to Investor preferences, current Investor information and Asset Data
- **Data Collector**: According to the Strategy, there are set of data points that needs to be collected and properly stored in the Asset Data in a continuous way. That is the role of the Data collector component.
- **Order Dispatcher**: Once the Strategy decides which orders to submit according to the observed Asset Data and, Investor information. These orders need to be dispatched to some Market (real or simulated) to be completely or partially executed or even not executed according to the market conditions. That is the role of the Order Dispatcher, that should also updated the portfolio information according to the result of the submitted orders.

C. *Alternative Architecture for Autonomous Traders*

Analyzing figure 1, it is not hard to realize that each asset will demand different data streams in order to be analyzed When analyzing many assets, it may be required to process a significant amount of data volume. One should also note that investor information is also part of data needs to be considered to define the trading strategy. That may bring the so called **curse of dimensionality**. As the number of dimensions increases, the volume of the data space increases so fast that the available data may become sparse. This sparsity is problematic because the amount of data needed to support result (in a significant way) will likely grow exponentially with dimensionality. In another hand, it is well known that diversification (a large number of target assets) may provide a better risk-reward relation in portfolios, by reducing the risks.

Furthermore, it is important to observe that autonomous trading requires trust from investors and the ability to explain its decisions would be highly desirable or even required from an autonomous trader, in order to be build such trust. These problems (curse of dimensionality and diversification) and the fact that analyses are not dependent on investor preferences suggest another possible architecture, that were inspired by [9]. This architecture is presented on figure 2, it counts with some additional components in relation to the first architecture. Such new components are described next.

- **Analyst**: It provides prediction about the asset return. Ideally, a return distribution. Naturally, it is dependent of the asset, but it is independent of investor's preferences, capital and horizon of investment. Ensemble several models to create an analyst is a good idea, since it can provide better predictions but also help to understand the reason behind its predictions
- **Allocation Strategy**: It uses the analyses provided by the analysts and according with investor profile it searches for the best allocation (or re-allocation) of resources among the target assets. It does not need to know more about the assets itself, since it works with the analyses, therefore given the analyses, it is asset independent.

D. *Types of Financial Data*

The data that can and should be used by autonomous traders is far more diverse and complex than historical prices. This data can be splitted in four different types: technical, fundamentalist, analytical and alternative data [21]. Figure 3 presents some examples for each type.

Fundamental data encompasses information that can be found in regulatory filings. It is mostly accounting data, reported quarterly or yearly. A particular aspect of this data is that it is reported with some temporal lapse. You must confirm exactly when each data point was released, so that your analysis uses that information only after it was publicly available. A common error is to assume that this data was published at the end of the reporting period, which is almost never the case. Another problem is the use of backfilled or reinstated data. Backfilling means that missing data is assigned a value, even if those values were unknown at that time. A "reinstated value" is a corrected value that amends an incorrect initial release. The problem is, the corrected values were not known on that first release date. The use of such data gives a unrealistic advantage to models, that will not be present in real operation. In another words, we could say it also leads to overfitting. Nevertheless, it is common to find studies that use the final released value and assign it to the time of the first release, or even to the last day in the reporting period.

Technical or Market data includes all trading activity that takes place in an exchange, including open, high, low and closes prices and their historical series. It is common to use the term bar to refer to open, high, low and close prices in a given time frame (1 minute, 1 hour or 1 day). Your Market data provider may also allow access to the order books or even to semi-structured information, like FIX messages. In this case, you may try to recognize patterns that identify some traders. For instance, human GUI traders often trade in round lots, and you can use this fact to estimate what percentage of the volume is coming from them at a given point in time.

Analytics data is usually provided by human experts. It is a kind of derivative data, based on an original source, which could be fundamental, market, alternative, or even a collection of other analytics. It is not available from an original source, and it was processed in a particular way (hopefully not biased). Independent analysts, investment banks and research firms sell valuable information that results from deep analyses of companies' business models, activities, competition and so on.

Alternative data may refer to satellite image or video feeds include monitoring of tankers, tunnel traffic activity, or parking lot occupancy. Before an Oil company reports increased earnings, before its market price shot up, before analysts wrote their commentary of their latest filings, before all of that, there were movements of tankers and drillers and pipeline traffic. They happened months before those activities were reflected in the other data types. What truly characterizes alternative data is that it is primary information, by that we mean the information has not made it to the other sources.

There are some problematic aspects of alternative data. It may bring high cost and privacy concerns, in fact it could even bring legal concerns. Capture those data may be very expensive, and the observed company may object, not to mention bystanders. Furthermore, it is usually hard to process. However, Alternative data offers the opportunity to work with unique datasets. Data that is hard to store, manipulate, and operate may be very rewarding, because perhaps your competitors did not try to use it for logistic reasons, gave up midway, or processed it incorrectly. Therefore, you may have a truly unexploited field to mining.

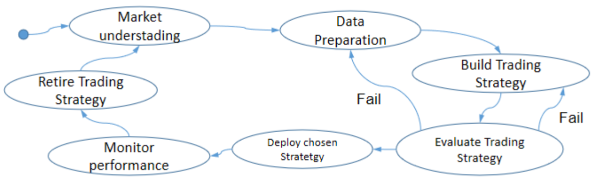


Fig. 4. Development process of trading strategies based on AI

E. Development Process of Autonomous Trader

The process of building Autonomous may be different according with the adopted architecture. For the simple architecture (figure 1), The development process is basically the same for data mining processes and can be separated in six steps, presented in figure 4. These steps are detailed next.

- **Market understanding**: In this phase, you should define your objectives and requirements and determining what kind of data can be collected to build a deployable model
- **Data preparation**: An initial dataset is established and studied to see whether it is suitable for processing. If the data quality is poor, it may be necessary to collect new data. Preparation also involves pre-processing the raw data so that machine learning algorithms can produce a model
- **Build trading strategy**: Model building and Data preparation usually go hand in hand. It is almost always necessary to iterate: results obtained during modeling provide new insights that affect the choice of pre-processing techniques and model's parameters definition
- **Evaluate trading strategy**: The main question in this phase is The built model is really good enough to be deployed? It is critical to success of any development of Trading robot and its importance cannot be overstated! It is not uncommon that it would be necessary to go back to data preparation, after achieving models with poor performance
- **Deploy chosen model**: This phase is really connected to the problem of deploying secure and reliable software
- **Monitor model performance**: It does not matter, how good performs a given model, after some time its performance can degrade along the time. (remember the financial environment is not i.i.d).

The strategy construction is absolutely critical to the performance and it can be done using expert knowledge or simply based on AI techniques. As mentioned before high dimensionality, diversification, and overfitting avoidance are significant challenges. However, if one uses the Alternative Architecture, some of these problems may be mitigated. A development process for the Alternative Architecture can be seen in figure 5. It includes some new steps that are detailed next.

- **Build Analyst** : Rather than building a trading strategy, the idea is creating a model (or an ensemble of models)

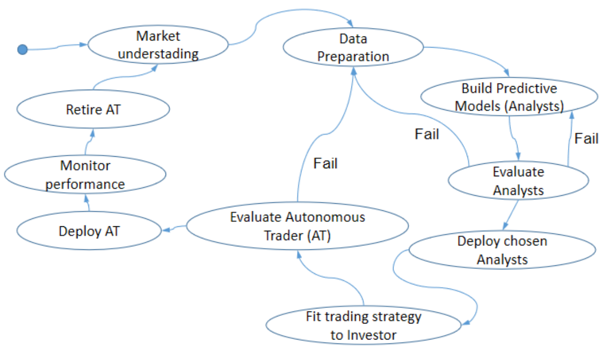


Fig. 5. Development process of autonomous analyses and trading strategies based on AI

that provides good accuracy and such model building is independent among assets

- **Evaluate Analyst**: The evaluation here is basically the same faced by predictive machine learning models with the remark that it should take into consideration the cost of errors. For a deeper discussion about cost of errors in autonomous trading see [22]
- **Fit Trading Strategy**; Adjust the trading strategy to the investor preferences, however the complexity of Asset's data is reduced because they are represented just by asset's return distributions
- **Evaluate Autonomous Trader**: Such evaluation is basically the same used for evaluate Trading Strategy, despite the fact that here the trading strategy is splitted in an allocation strategy, and several analysts.

F. *Trading Strategy Performance Measure*

Most of the work related to measure performance for autonomous trading strategies or portfolio management by human rely on Sharpe ratio to guide their processes. Sharpe ratio is well known and often used in evaluating funds performance. However, there are significant criticism about it. One of the critics is about that it does not distinguish between upside and downside volatility. According to Rollinger and Hoffman [23], high outlier returns could increase the value of the Sharpe ratio's denominator (standard deviation) more than the value of the numerator (return in excess), thereby lowering the value of the ratio and for positively skewed return distributions (observed some strategies), the Sharpe ratio could be increased by removing the largest positive returns. It would make sense because investors welcome large positive returns in general [23]. Another point is that the assumption of normality in return distributions are been relaxed and Sharpe ratio become questionable tools for selecting optimal portfolios [24]. In fact, some indexes have been proposed as alternative to Sharpe ration, for instance: Sortino ratio [25] and Modigliani index [26]. Despite the quality of these proposals, Sharpe ratio "...definitely is the most widely used.." measure of risk-adjusted performance [23](pg.42). Furthermore, as stated by Bailey and others [27](pg. 15) Sharpe ratio can provide evidence of investment skill since it is observed a proper track record length, despite its known deficiencies.

III. A FRAMEWORK FOR BUILDING AUTONOMOUS TRADERS: MT5B3

In this section, we present a framework to build autonomous traders, called mt5b3. It allows access to price data (open, close, high, low) and book data (bid, ask) and order placement in simulated or real financial markets. It was designed to operate in any financial market accessible through Metatrader platform, that uses the so called netting accounting system. However, it was tested in simulated and real operation connected just to the Brazilian stock exchange (B3).

The framework mt5b3 may be used to create simple trading robots based on algorithms, but it also supports the creation of traders based on Artificial Intelligence techniques, such as decision trees [12], neural networks [28], Bayesian dynamic networks [29], reinforcement learning [30] and so on.

In listing 1, we present a **very simple** autonomous trader built using mt5b3. It is able to trade with any number of target assets, but it does that randomly. It select aleatory a number of shares between and 1 and 1000, and then also aleatory picks buying or selling that number of shares at market price for each asset, at each moment.

```
import numpy.random as rand
import mt5b3 as b3
class RandomTrader(b3.Trader):
    def trade(self,dbars):
        orders=[]
        shares=rand.randint(1,1000)
        for asset in assets:
            if rand.randint(2)==1:
                order=b3.buyOrder(asset,shares)
            else:
              order=b3.sellOrder(asset,shares)
            orders.append(order)
        return orders
```

Listing 1. Random Autonomous Trader Example

A. *Installation and Further information about mt5b3*

The framework mt5b3 may be download from github or PyPI platforms. In its repository [31], there are also available tutorials presenting mt5b3 API and how to create simple autonomous Trader, and also Traders based on AI techniques. Some jupyter notebooks with several examples are also available. In order to install mt5b3 in a Python, you may just use one of the ways presented in listing 2

```
# this package is required by mt5b3
pip install MetaTrader5
# installing mt5b3 package
pip install mt5b3

# within a jupyter notebook, you may use:
#import sys
#!{sys.executable} -m pip install Metatrader5
#!{sys.executable} -m pip install mt5b3
```

Listing 2. Installing mt5b3

B. Building Autonomous Traders with mt5b3

Built an autonomous trader, one needs to verify if it is suitable for operation. The basic form of testing an autonomous trader is often called backtest. It is a kind of evaluation for trading robots. It is basically a trading robot executing with historical price series , and its performance is computed. In backtesting, time is discretized according with bars and mt5b3 controls the information access to the Trader according with the simulated time. As simulation time advances, the function 'trade' is called and receives the new bar info and decides which orders to send. In order to backtest one strategy, one just need to create a Trader, establish the test parameters and execute it. These parameters define trading guidelines (target assets, available capital and horizon of investment) and some system parameters that define operation and log register. In listing 3, we present an example of backtest definition and execution.

```
    # trading data options
capital=100000
results_file='data_equity_file.csv'
assets=['PETR4','VALE3','ITUB4']

#backtest options
prestart=b3.date(2019,12,10)
start=b3.date(2019,1,10)
end=b3.date(2019,2,27)
# Use True if you want debug information for your
     Trader
verbose=False
#sets the backtest setup
period=b3.DAILY
 # it may be b3.INTRADAY (one minute interval)

bts=b3.backtest.set(assets,prestart,start,end,period
     ,capital,results_file,verbose)
#create trader instance
trader=RandomTrader()
# Running the backtest
df= b3.backtest.run(trader,bts)
```

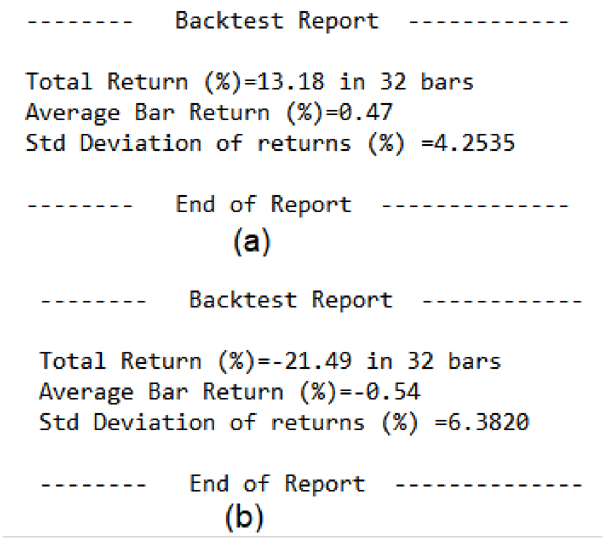
Listing 3. Backtesting an Autonomous Trader

If we execute the listing 3, we backtest the trader presented in listing 1. As we said before, the random trader does make intelligent decisions, just picks randomly numbers of shares and sells or buys them. Naturally, its performance may vary widely. In order to evaluate the trader's performance, you may use the function b3.backtest.evaluate that generates a report, see listing 4. In figure 6, we present two very different results for the Random Trader using the same assets and trading period. In execution (a), the performance is very good, where trader achieves profit (13.18% return) in less than two months (January 10 to February 27, 2019), however another execution (b) using the same random trader and period achieved a negative return of 21.49%.

```

#Run the trader according setup and get the results
df=b3.operations.run(trader,ops)
#evaluate the results
b3.backtest.evaluate(df)
#Alternatively, you can evaluate using the generated
      file
#b3.backtest.evaluateFile(fileName)
#fileName is the name of file generated by the
     backtest
```

Listing 4. Evaluating performance an Autonomous Trader


```
--------   Backtest Report  ------------

Total Return (%)=13.18 in 32 bars
Average Bar Return (%)=0.47
Std Deviation of returns (%) =4.2535

--------   End of Report  --------------
```
(a)
```
--------   Backtest Report  ------------

Total Return (%)=-21.49 in 32 bars
Average Bar Return (%)=-0.54
Std Deviation of returns (%) =6.3820

--------   End of Report  --------------
```
(b)

Fig. 6. Two different results for the same Trader and setup

We need to note that it is hard to perform meaningful evaluations using backtest. There are many pitfalls to avoid and it may be easier to get trading robots with great performance in backtest, but that perform really badly in real operation. For a deeper discussion about trading strategies evaluation, we suggest [6], [21] and [27].

After achieving good performance in backtesting properly conceived and executed, the next phase would be to operate in real market with limited portfolio. As mentioned before you can use mt5b3 traders in real environments, but you may have to perform some minor changes in a trader so it can operate in real mode and you need to define the operational parameters, which are a little different from backtest parameters. In listing 6, we show how to establish the parameters for trader's operation, create a trader instance from a class called MultiAssetTrader and run it according with its setup. Some brokers provide dedicated servers and accounts for simulated operations, often called demo accounts. For instance, XP Inc is one of those brokers in B3 stock exchange that provide demo accounts. Despite being simulated account, under the point of view of an autonomous trader they are just like real accounts and can use exactly the same code, as presented in listings 6 and 5.

```
class MultiAssetTrader(b3.Trader):
    def trade(self,bts,dbars):
        assets=dbars.keys()
        orders=[]
        for asset in assets:
            bars=dbars[asset]
            curr_shares=b3.backtest.getShares(bts,
    asset)
            money=b3.backtest.getBalance(bts)/len(
    assets)  # divide o saldo em dinheiro igualmente
    entre os ativos
```

```
            # number of shares that you can buy of asset
            free_shares=b3.backtest.getAfforShares(bts,dbars,asset,money)
            rsi=b3.tech.rsi(bars)
            if rsi>=70 and free_shares>0:
                order=b3.buyOrder(asset,free_shares)
            elif  rsi<70 and curr_shares>0:
                order=b3.sellOrder(asset,curr_shares)
            else:
                order=None
            if order!=None:
                orders.append(order)
        return orders
```

Listing 5. Autonomous Trader suitable for real operation

In listing 5, we present the complete code of an autonomous trader called MultiAssetTrader, ready for real operations. It is based on a simple interpretation of the Relative Strength Index (RSI), and splits equally the available capital among assets.

```
      #trading data
# target assets (three of the main assets in B3)
assets=['PETR4','VALE3','ITUB4']
# available capital
capital=100000
# it will run by the end of session!
endTime=b3.operations.sessionEnd()
#if market not open, keep waiting
waitForOpen=True
#get information and decise every minute
timeFrame=b3.INTRADAY

     # System information
# gives information during executing
verbose=True
# operations register file
data_file='data_equity_file.csv'
# seconds to wait between trade calls
delay=1
# number of bars to take in each decision
mem=10
# setup operation (ops)
ops=b3.operations.set(assets,capital,\
endTime, mem,timeFrame,data_file,\
verbose,delay,waitForOpen)

# Create an instance of the trader
trader=MultiAssetTrader()

#Connect to B3 using default account in MT5
b3.connect()

#Run the trader according setup
 b3.operations.run(trader,ops)
```

Listing 6. Setup and Running an Autonomous Trader for real operation

C. Autonomous traders based on AI

In this section, we are going to present some AI powered trading robots, based on Machine learning and Artificial Intelligence algorithms. As discussed in section II-B, one can create autonomous trader building trading strategies that encompass the whole decision process, taking into account data about target assets and investor information. Another option would be split those concerns in different models or agents, as discussed in II-C. We present an autonomous trader based on the first alternative, using Random Forests [32] and deals with investor preferences in a very simplistic way, by dividing the capital equally among the assets and using the same model in listing 7

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import KBinsDiscretizer

class RandomForestTrader(b3.Trader):

    def setup(self,dbars):
        assets=list(dbars.keys())
        if len(assets)!=1:
            print('Error, this trader is supposed to deal with just one asset')
            return None
        bars=dbars[assets[0]]
        # remove irrelevant info
        if 'time' in bars:
            del bars['time']
        timeFrame=10 # it takes into account the last 10 bars
        horizon=1 # it project the closing price for next bar
        target='close' # name of the target column
        ds=b3.ai_utils.bars2Dataset(bars,target,timeFrame,horizon)

        X=b3.ai_utils.fromDs2NpArrayAllBut(ds,['target'])
        discretizer = KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='uniform')
        # creates the discrete target
        ds['target']=b3.ai_utils.discTarget(discretizer,ds['target'])
        Y=b3.ai_utils.fromDs2NpArray(ds,['target'])

        #clf = tree.DecisionTreeClassifier()
        clf = RandomForestClassifier(n_estimators=10)
        clf = clf.fit(X, Y)
        self.clf=clf

    def trade(self,bts,dbars):
            assets=dbars.keys()
            orders=[]
            timeFrame=10 # it takes into account the last 10 bars
            horizon=1 # it project the closing price for next bar
            target='close' # name of the target column
            for asset in assets:
                curr_shares=b3.backtest.getShares(bts,asset)
                money=b3.backtest.getBalance(bts)/len(assets) # divide o saldo em dinheiro igualmente entre os ativos
                free_shares=b3.backtest.getAfforShares(bts,dbars,asset)
                # get new information (bars), transform it in X
                bars=dbars[asset]
                #remove irrelevant info
                if 'time' in bars:
                    del bars['time']
                # convert from bars to dataset
                ds=b3.ai_utils.bars2Dataset(bars,target,timeFrame,horizon)
                # Get X fields
                X=b3.ai_utils.fromDs2NpArrayAllBut(ds,['target'])

                # predict the result, using the latest info
            p=self.clf.predict([X[-1]])
            if p==2:
                #buy it
                order=b3.buyOrder(asset,
free_shares)
            elif p==0:
                #sell it
                order=b3.sellOrder(asset,
curr_shares)
            else:
                order=None
            if order!=None:
                orders.append(order)
        return orders
```

Listing 7. Example of Autonomous Trader based on AI (Random Forest)

IV. Open Problems and Promising Technologies

There is a long road ahead in the path to build autonomous traders that can beat the best human experts in a consistent way. There are many initiatives that deal with Autonomous trading or some closely related concept. Despite all the work done, there is no system that could be pointed as the "Deep blue" of autonomous trading. In fact, we believe there is a long way towards an autonomous trading systems that can beat a "world champion" but they are close to beat average human trader or may be already there.

One aspect that we believe may be a fundamental advantage for autonomous traders is accountability. Especially regarding eliciting possible conflict of interests. It is well known that there are possible conflicts of interest among analysts, managers and investors. One common conflict of interest may happen among managers and stockholders [26] (pg.12). The conflicts of interest among analysts and investors, may take place when analysts have investments on target assets themselves or are contracted by securities emitters. In fact, SEC (U.S. Securities and Exchange Commission) has a long history of examining potential conflicts of interests among such roles, for more information see [33] and [34]. Due to the fact that machine can have controlled or at least formally verifiable interests through software verification and validation, possible conflict of interests can be avoided or at very least controlled in a more efficient way. On the other hand, the use of autonomous traders require that trust can be established in its development, deployment, and operation. It is a challenge faced by AI in many scenarios and it is not different in autonomous traders. The concept of trustworthy artificial intelligence has five foundational principles according to [35]: (1) beneficence, (2) non-maleficence, (3) autonomy, (4) justice, and (5) explicability. One may argue that accountability is strongly related to the first two principles. Autonomy refers in large extent on the promotion of human oversight (e.g., Guidelines), others also consider the restriction of AI-based systems' autonomy, where humans retaining the right to decide when to decide at any given time. The justice principle is not to be understood judicially, as in adhering to laws and regulations, but instead in an ethical way. For instance, the utilization of AI should to amend past inequities like discrimination of any kind. The last principle, explicability is without any doubt, critical and challenging for autonomous traders, because given the environment complexity mistakes will happen and the ability to explain and justify past decisions are crucial, in order to build and keep trust in the system.

When reasoning about autonomous investments one may ask What would happen if autonomous investment analysts or managers become ubiquitous. We believe the scenario described by Fama in his Efficient Market Hypothesis (EMH) [5] would take place. The EMH states that financial markets are efficient in pricing assets. Asset prices would reflect all information publicly available and the collective beliefs of all investors over the foreseeable future. Thus, it would not be possible to overcome the performance of the market, using information that is known by the market, except by simple chance.

We have discussed the financial environment complexity in section II-A. In fact, it is a challenging environment not just for autonomous agents, but also for human experts. Some argue that it would be beyond the limits of algorithms. As stated by Marks, "Valid approaches work some of the time but not all. And investing can't be reduced to an algorithm and turned over to a computer. Even the best investors don't get it right every time." [36]. In other words, Marks point to the fact that the environment is not stationary and circumstances rarely repeat exactly. Furthermore, Marks recognizes that psychology plays a major role in financial markets. We believe that true autonomous trader should some how try to model and reason about market psychological aspects. There are some initiatives aiming to walk towards software and hardware that can mimic processes that exist within the human brain, such as intuition and emotional memory concepts [37] and [38]. However, there are more questions then answer about how to model and emotions in autonomous agents and certainly it is a topic that requires further research.

A. Related Work

In table I, we present a comparative analysis of some selected systems with similar propose of mt5b3. Such analysis is based on some features that facilitate the development and test of autonomous trading strategies. We do not intend to judge the overall quality of the cited systems, but just identify differences (positive and negative) with the mt5b3 framework proposed here. It is relevant to note that mt5b3 relies on MT5 [3] to connect to stock markets. The MT4 [39] is not just an earlier version of MetaTrader 5, but it is only focused on Forex markets. While, MetaTrader 5 allows trading in Forex or stock markets, however it comes with the cost of higher complexity and two accounting systems: netting and hedging systems. Therefore, MT4 is still wildey used for Forex operations. The other system, called AgEx [40], is an open source financial market simulation tool that uses FIPA ACL language and a market ontology created specifically to be used for trader agents.

agents.

Feature	MT4	AgEx	MT5	mt5b3
Open source	No	Yes	No	Yes
Python support	No	No	No	Yes
Object Oriented	No	Yes	Yes	Yes
Backtest support	Yes	Yes	Yes	Yes
Real Operation support	Yes	No	Yes	Yes
Forex operation support	Yes	No	Yes	No
Stock operation support	No	Yes	Yes	Yes

TABLE I
COMPARISON AMONG SELECTED SYSTEMS

B. Promising Technologies

Recent work point to some promising technologies, specially regarding the use of convolutional neural networks and Gated Recurrent Units (GRU) [41]; reinforcement learning [30], [42], specially when used deep learning architectures [43], [44] and ensemble methods [12], [45], [46]. For a good review of recent work in the field, we suggest [47].

V. CONCLUSIONS AND FUTURE WORK

In this paper, we discussed some fundamental aspects of modelling autonomous traders, their complex environment under the point of view of an autonomous agent based on Artificial Intelligence. We proposed two conceptual architectures for autonomous traders and discussed the main steps in the developments of such entities. Furthermore, we presented a framework that helps the development and testing of autonomous traders. This framework, called mt5b3, is freely available [31] and it may also be used in real or simulated operation in financial market accessible through platform MetaTrader 5 [3]. Finally, we discussed some open problems in the area such as accountability and trustworthiness in autonomous systems and recognized there is still a long road ahead in the path to build autonomous traders that can beat the best human experts consistently. We also pointed out some interesting technologies that may contribute to advance in such task. The proposed framework mt5b3 may also contribute to development of new autonomous traders.

REFERENCES

[1] Olympia Hadjiliadis Michael Carlisle and Ioannis Stamos. Trends and trades. In *Handbook of High-Frequency Trading and Modeling in Finance, First Edition.*, New York, 2016. John Wiley Sons, Inc.

[2] Michael P. Wellman and Uday Rajan. Ethical issues for autonomous trading agents. *Minds and Machines*, 27(4):609–624, Dec 2017.

[3] Andrew R. Young. *Expert Advisor programming for Metatrader 5*. Edgehill Pushishing, Nashville, TN, USA, 2018.

[4] Harry M. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[5] Eugene Fama. Efficient capital markets:a review of theory and empirical work. *Journal of Finance*, 25:383–417, 1970.

[6] Paulo André Lima de Castro. Is it a great autonomous fx trading strategy or you are just fooling yourself? *arxiv.org/3558639*, 2020.

[7] Renato Oliveira and Adriano Pereira. Agente de negociação de ações utilizando aprendizado por reforço. In *Proceeding of the Workshop of Artificial Intelligence Applied to Finance (WAIAF 2019)*, Sao Jose dos Campos, Brazil, 2019. WAIAF.

[8] Alexander Sherstov and Peter Stone. Three automated stock-trading agents: A comparative study. In *Proceedings of the Agent Mediated Electronic Commerce (AMEC) Workshop - AAMAS 2004*, New York, 2004.

[9] Paulo Andre Lima Castro and Jaime Simao Sichman. Automated asset management based on partially cooperative agents for a world of risks. *Applied Intelligence*, 38:210–225, 2013.

[10] J. Paulin, A. Calinescu, and M. Wooldridge. Agent-based modeling for complex financial systems. *IEEE Intelligent Systems*, 33(2):74–82, Mar 2018.

[11] Leandro Anghinoni and Liang Zhao. Time series trend detection and forecasting using complex network topology analysis. In *Proceeding of the Workshop of Artificial Intelligence Applied to Finance (WAIAF 2018)*, Sao Jose dos Campos, Brazil, 2018. WAIAF.

[12] Rafael Silva Wagner and André Alves Portela Dos Santos. Forecasting the direction of high-frequency returns: An ensemble-trees application. In *Proceeding of the Workshop of Artificial Intelligence Applied to Finance (WAIAF 2018)*, Sao Jose dos Campos, Brazil, 2018. WAIAF.

[13] Harish Subramanian, Subramanian Ramamoorthy, Peter Stone, and Benjamin J. Kuipers. Designing safe, profitable automated stock trading agents using evolutionary algorithms. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1777–1784, New York, NY, USA, 2006. ACM Press.

[14] Flavio Abdenur Elias Cavalcante-Filho and Rodrigo De-Losso. Machine learning applied to accounting variables yields the risk-return metrics of private company portfolios. In *Proceeding of the Workshop of Artificial Intelligence Applied to Finance (WAIAF 2019)*, Sao Jose dos Campos, Brazil, 2019. WAIAF.

[15] Felipe Dias Paiva and Carolina Magda da Silva Roma. Métodos de deep learning aplicados a candlestick como estratégia de investimento. In *Proceeding of the Workshop of Artificial Intelligence Applied to Finance (WAIAF 2019)*, Sao Jose dos Campos, Brazil, 2019. WAIAF.

[16] Y. Hu and S. Lin. Deep reinforcement learning for optimizing finance portfolio management. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 14–20, Feb 2019.

[17] David H. Bailey, Jonathan Borwein, Marcos Lopez de Prado, and Qiji Jim Zhu. Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. *Notices of the American Mathematical Society*, 61:458–471, May 2014.

[18] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach Third Edition*. Prentice Hall, Englewood Cliffs - NJ, 2013.

[19] Elena Andreou and Eric Ghysels. *Structural Breaks in Financial Time Series*, pages 839–870. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[20] Paulo Andre Lima de Castro and Ronnald Annoni Jr. Towards autonomous investment analysts - helping people to make good investment decisions. In *Proceeding of the Future Technologies Conference 2016*, san Francisco, CA, December 2016. FTC 2016.

[21] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley, New York, 2018.

[22] Paulo Andre Lima de Castro and Marcel Ribeiro. Online learning applied to autonomous valuation of financial assets. In *Proceeding of the Web Intelligence Conference 2018*, Santiago, Chile, December 2018. WI 2018.

[23] Tom Rollinger and Scott Hoffman. Sortino ratio: A better measure of risk. *Futures magazine*, 1(1):40–42, 2 2013.

[24] Simone Farinelli, Manuel Ferreira, Damiano Rossello, Markus Thoeny, and Luisa Tibiletti. Beyond sharpe ratio: Optimal asset allocation using different performance ratios. *Journal of Banking and Finance*, 32(10):2057–2063, 10 2008.

[25] F. A.Sortino and H. J.Forsey. On the use and misuse of downside risk. *Journal of Portfolio Management*, 22(1):35–42, 1996.

[26] Aswath Damodaran. *Applied Corporate Finance*. Wiley, New York, NY, 2010.

[27] David H. Bailey and Lopez de Prado. The sharpe ratio efficient frontier. *Journal of Risk*, 15:3–44, Feb 2012.

[28] A. Tan, C. Quek, and K. Yow. Maximizing winning trades using a novel rspop fuzzy neural network intelligent stock trading system. *Applied Intelligence*, 29:116–128, 2008. 10.1007/s10489-007-0055-1.

[29] Gordon Ritter and Petter Kolm. Multiperiod portfolio selection and bayesian dynamic models. *SSRN Electronic Journal*, 09 2014.

[30] Yi Feng Yuriy Nevmyvaka and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on Machine Learning- ICML 2006*, Pittsburgh, Pennsylvania, 6 2006.

[31] Paulo André Lima de Castro. mt5b3: A framework for building autonomous trader for the brazilian stock exchange - b3.

https://github.com/paulo-al-castro/mt5b3, 2020. A Python Framework for Autnomous Traders.

[32] Mark A. Hall Eibe Frank and Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, New York, 2016.

[33] U.S. Securities and Exchange Commission. Analysts conflicts of interest: Taking steps to remove bias, 2016.

[34] Ingo Walter. Conflicts of interest and market discipline among financial services firms. In *Proc. of the Federal Reserve of Chicago conference on Market Discipline: Evidence Across Countries and Industries*, Chicago, USA, 11 2003. Fed-Chicago.

[35] Sebastian Lins Scott Thiebes and Ali Sunyaev. Trustworthy artificial intelligence. *Electronic Markets*, 10 2020.

[36] Howard Marks. *The most important thing: uncommon sense for the thoughtful investor*. Columbia Press, New York, 2011.

[37] James A. Crowder and Shelli Friess. Artificial psychology: The psychology of ai. *SYSTEMICS, CYBERNETICS AND INFORMATICS*, 11:64–68, May 2013.

[38] Luis-Felipe Rodríguez and Félix Ramos. Development of computational models of emotions for autonomous agents: A review. *Cognitive Computation*, 6:351–375, 09 2014.

[39] MetaQuotes Ltd. Metatrader 4 - forex platform, 2020.

[40] Paulo Andre Castro and Jaime S. Sichman. Agex: A financial market simulation tool for software agents. In Will Aalst, John Mylopoulos, Norman M. Sadeh, Michael J. Shaw, Clemens Szyperski, Joaquim Filipe, and Jos? Cordeiro, editors, *LNBIP*, volume 24, pages 704–715. Springer, Berlin, 2009.

[41] ; Carmine Ventre Zheng Gon and John O'Hara. Classifying high-frequency fx rate movements with technical indicators and inception model. In *Proceedings of the ACM International Conference on AI in Finance 2020*, New York, 2020.

[42] Renato Oliveira and Adriano Pereira. A tabular sarsa-based stock market agent. In *Proceeding of the ACM International Conference on AI in Finance - 2020*, New York, 2020. ICAIF 2020.

[43] Hongda Shen and Eren Kursun. Deep q-network based adaptive alert threshold selection policy for payment fraud systems in retail banking. In *Proceeding of the ACM International Conference on AI in Finance - 2020*, New York, 2020. ICAIF 2020.

[44] L. Conegundes and A. C. M. Pereira. Beating the stock market with a deep reinforcement learning day trading system. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[45] Bashar Alhnaity and Maysam Abbod. A new hybrid financial time series prediction model. *Engineering Applications of Artificial Intelligence*, 95:103873, 2020.

[46] Hongyang Yang; Xiao-Yang Liu; Shan Zhong and Anwar Walid. Deep ensemble reinforcement learning for automated stock trading. In *Proceeding of the ACM International Conference on AI in Finance - 2020*, New York, 2020. ICAIF 2020.

[47] Watthanasak Jeamwatthanachai Mehtabhorn Obthong, Nongnuch Tantisantiwong and Gary Wills. A survey on machine learning for stock price prediction: algorithms and techniques. In *Proceeding of the 2nd International Conference on Finance, Economics, Management and IT Business*, pages 63–71, Prague, Czech Republic, 2020. Vienna House Diplomat.