Artificial Intelligence and Machine Learning in Financial Environments

-

AI powered Trading Robots

Prof. Paulo André L. de Castro

pauloac@ita.br

www.comp.ita.br/~pauloac

Sala 110, IEC-ITA

Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos-SP

Package mt5b3

- Package mt5b3provides access to the B3 market to python programs through Metatrader and some Brazilian brokers (XP, clear corretora, ...). URL

- It allows access to price data (open, close, high, low) and book data (bid, ask) and order placement! More information: https://github.com/paulo-al-castro/mt5b3/

- In the first chapter, we presented some simple trading robots based on algorithms, but that were not really based on Artificial Intelligence

- We are going to present some AI powered trading robots, based on Machine learning, Probabilistic Reasoning and search algorithms

AI powered Trading Robots

- This chapter shows how to create AI powered trading robots using mt5b3 and well known AI framework like: TensorFlow, Pytorch, SciKit-Learn and others
- You will be able to create Trading Robots using Neural networks, Random Forests, Support Vector Machines, Genetic Algorithms, Bayesian Networks, Reinforcement Learning, Deep Learningand other techniques
- Note that we assume here, that you are **already** familiar with mt5b3 and Articificial Intelligence algorithms, specially Machine learning
- We are going to present the differences between Financial Environment and more traditional environments

Summary

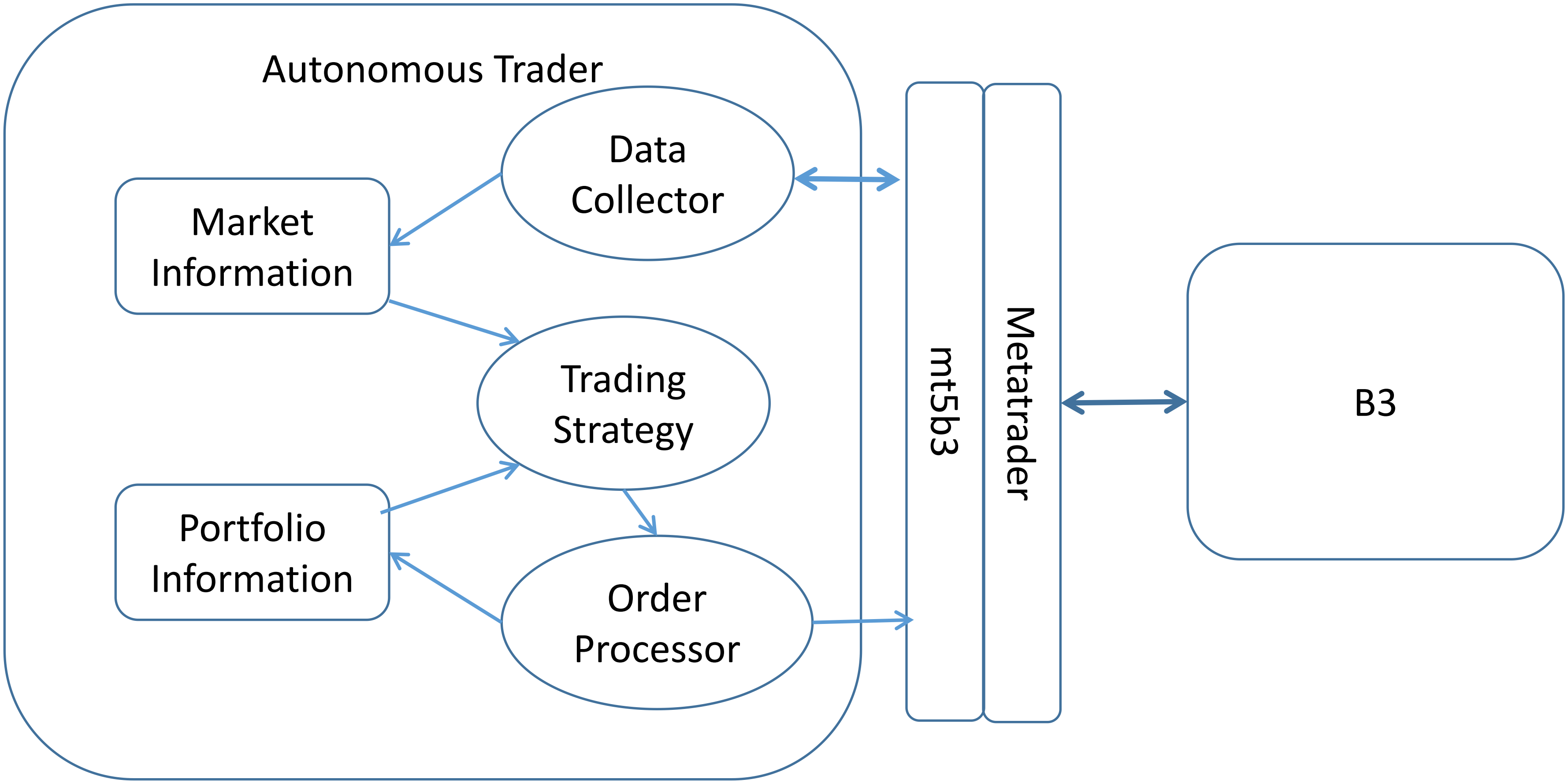
- **Modelling trading robots**
- A Simple AI Trading Robot
  - Preparing datasets
  - Training
  - [Back]Testing
  - Evaluation

Four fundamental types of data

Fundamental Data	Market Data	Analytics	Alternative Data
• Assets • Liabilities • Sales • Costs/earnings • Macro variables • . . .	• Price/yield/implied volatility • Volume • Dividend/coupons • Open interest • Quotes/cancellations • Aggressor side • . . .	• Analyst recommendations • Credit ratings • Earnings expectations • News sentiment • . . .	• Satellite/CCTV images • Google searches • Twitter/chats • Metadata • . . .

- Source: Prado, Marcos Lopez de. Advances in Financial Machine Learning.Wiley, New York, 2018

Autonomos Trader Architecture



Trader agent's Components

- **Data Collector**: According to theStrategy, there are setof data points that needs to be collected and properly stored in theMarket informationin a continuous way. That is the role of the Data collector component.

- **Market information**: it keeps record of all relevantinformation regarding the target assets (technical, funda-mentalist and others). Which are the relevant informationis defined by the set of assets and the robot's strategy

- **Portfolio Information**: it keeps tracks about the sharesand capital owned by the trading strategy

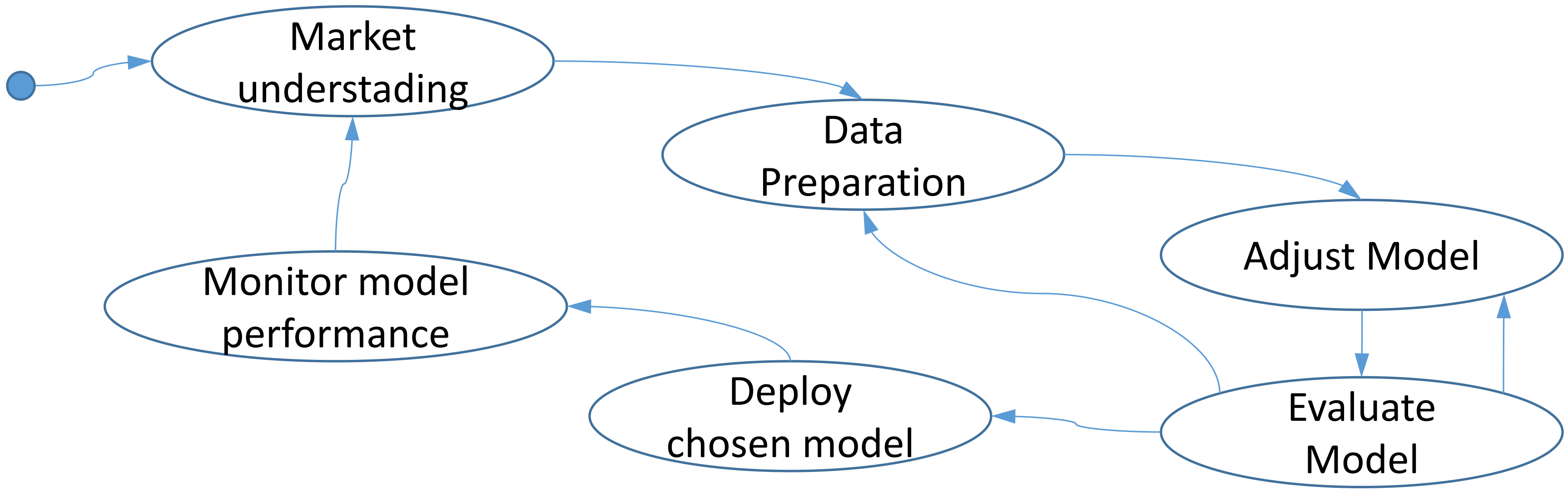
- **Trading Strategy**: it is the implementation of the function thatdecides the Robot's actions according to current portfolio information and Market information

- **Order Processor**: Once the Strategy decides which orders to submit according to the observed Market information and current portfolio. These orders are dispatched to some Market (real orsimulated) by the Order processor component to be completely or partially executed or even not executed according to the market conditions. The order processor also updates the portfolio information according to the result of the submitted orders

AI based Trading Robots

- You may create AI based Trading Robots using AI algorithms to define the Trading strategy, the others components remain basically unchanged

- The development process is basically the same for data mining processes. We can separate it in six steps:



AI based Trading Robots – development Process

- Market understading:
- Data preparation:
- Adjust [train] model:
- Evaluate model:
- Deploy chosen model:
- Monitor model performance

Summary

- Modelling trading robots
- **A Simple AI Trading Robot**
  - Preparing datasets
  - Training
  - [Back]Testing
  - Evaluation

Simple AI Trading Robot – Market Understanding

- We are going to show a simple AI trading robots with the following features:
  - it deals with just one asset [PETR4]
  - it uses historical prices (open, close, high, low), collected from B3 (with 1 minute interval)
  - it is based on decision trees

Simple AI Trading Robot – Data preparation

```
import mt5b3 as b3
from datetime import datetime
b3.connect()
bars=b3.getBars('PETR4',datetime(2020,1,1),
datetime(2020,2,1),b3.INTRADAY)
```

From bars to Dataset (X and Y features) . timeFrame (tF)=2, horizon=h

x1	x2	x3	x4	y
x1[0]	x2[0]	x3[0]	x4[0]	y[0]
x1[1]	x2[1]	x3[1]	x4[1]	y[1]
x1[2]	x2[2]	x3[2]	x4[2]	y[2]
x1[3]	x2[3]	x3[3]	x4[3]	y[3]
..	.	..	…	…
x1[n]	x2[n]	x3[n]	x4[n]	y[n]

older to newer…

Table n rows, X+Y columns

x1[t-1]	x2[t-1]	x3[t-1]	x4[t-1]	x1[t]	x2[t]	x3[t]	x4[t]	y[t+h]
x1[0]	x2[0]	x3[0]	x4[0]	x1[1]	x2[1]	x3[1]	x4[1]	y[h+tF]
x1[1]	x2[1]	x3[1]	x4[1]	x1[2]	x2[2]	x3[2]	x4[2]	y[h+tF+1]
x1[2]	x2[2]	x3[2]	x4[2]	x1[3]	x2[3]	x3[3]	x4[3]	y[h+tF+1]
x1[3]	x2[3]	x3[3]	x4[3]	x1[4]	x2[4]	x3[4]	x4[4]	y[h+tF+1]
		…					…	…
x1 [n-h-1]	x2 [n-h-1]	x3 [n-h-1]	x4 [n-h-1]	x1 [n-h]	x2 [n-h]	x3 [n-h]	x4 [n-h]	y[n]

Table (n–h–timeFrame) rows

and

(X*timeFrame+Y) columns

From Bars to Dataset

- X features

```
bars=b3.getBars('PETR4',100)
# You may also obtain bars by getIntradayBars, readBarsFile....

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
# remove the columns that should not be used in ML
# remove all that are not float or categorical !!!
del bars['time']
# convert from bars to dataset
ds=b3.ai_utils.bars2Dataset(bars,target,timeFrame,horizon)
ds.describe() # see the data
```

Discretizing Target (or other columns)

- Some machine learning algorithms (like CART for decision trees) require a discrete target (dependent variable, Y) or even independent variables (X variables)
- There are many methods to bring continuos variables to the discrete domain. The main methods are:
  - Equal-width or uniform [Width= (maximum value - minimum value)/ N]
  - Equal-frequency or quantile (All bins with the same number of points)
  - K-means (based on K-means clustering algorithm, each bin has the closest centroid)

```
# Python
from sklearn.preprocessing import KBinsDiscretizer

# set the number of bins and encode. Strategy may be 'quantile', 'uniform' or 'kmeans'
discretizer = KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='uniform')

ds[target]=b3.ai_utils.discTarget(discretizer,ds[target])
```

Discretizing np arrays

```
from sklearn.preprocessing import KBinsDiscretizer

discretizer = KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='uniform')

x=np.array(ds.target)
x.reshape(-1,1) # it turns into a column array

dx=discretizer.transform(x)  # you may also use fit_transform(x)

dx=discretizer.fit_transform(x)  # you may also use fit_transform(x)
```

Decision Tree algorithms in scikit

- ID3 algorithm creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets

- C4.5 is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals
  - C5.0 is a new version released under a proprietary license. It uses less memory and builds smaller rulesets than C4.5 while being more accurate.

- CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

- scikit-learn uses an optimised version of the CART algorithm;

Creating an decision tree for Trading

```
from sklearn import tree
ds # dataset with X features and Y feature
X = b3.ai_utils.fromDs2NpArray(ds,['open1','high1','close1','low1'])
# or
#X=b3.ai_utils.fromDs2NpArrayAllBut(ds,['target'])
discretizer = KBinsDiscretizer(n_bins=3, encode='ordinal',
strategy='uniform')

ds[target]=b3.ai_utils.discTarget(discretizer,ds[target])

Y=b3.ai_utils.fromDs2NpArray(ds,['target'])

clf = tree.DecisionTreeClassifier()

clf = clf.fit(X, Y)
```

Getting decisions

```
# get new information (bars), transform it in X
#remove irrelevant info
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=clf.predict([X[-1]])
if p==2:
        #buy it
elif p==0:
        #sell it
else:
        # else do nothing
```

Putting it all together – Simple AI trading robot

```
from sklearn import tree
import mt5b3 as b3

class SimpleAITrader(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]]
        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')

        ds[target]=b3.ai_utils.discTarget(discretizer,ds[target])
        Y=b3.ai_utils.fromDs2NpArray(ds,['target'])

        clf = tree.DecisionTreeClassifier()

        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(asset)
        money=b3.backtest.getBalance()/len(assets) # divide o saldo
        free_shares=b3.backtest.getAfforShares(asset,money,dbars)
        # get new information (bars), transform it in X
        bars=dbars[asset]
        #remove irrelevant info
        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
```

Backtesting the Simple AI Trading

```
trader=SimpleAITrader()
# sets Backtest options
prestart=b3.date(2018,12,10)
start=b3.date(2019,1,10)
end=b3.date(2019,2,27)
capital=100000
results_file='data_equity_file.csv'
verbose=False
assets=['PETR4']
# Use True if you want debug information for your Trader
#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)
if b3.backtest.checkBTS(bts): # check if the backtest setup is ok!
    print('Backtest Setup is Ok!')
else:
    print('Backtest Setup is NOT Ok!')
# Running the backtest
df= b3.backtest.run(trader,bts)
# run calls the Trader. setup and trade (one for bar)
# evaluates the backtest results
b3.backtest.evaluate(df)
```

Evaluating Backtesting results

- The method backtest.run creates a data file with the name given in the backtest setup (bts)
- This will give you a report about the trader performance
- We need ot 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 operations.
- More about that in mt5b3 backtest evaluation chapter.
- For a deeper discussion, we suggest:
  - Is it a great Autonomous Trading Strategy or you are just fooling yourself Bernardini,M. and Castro, P.A.L
- In order to analyze the trader's backtest, you may use :

```
b3.backtest.evaluateFile(fileName)  #fileName is the name of file generated by the backtest
or
b3.bactest.evaluate(df)  # df is the dataframe returned by b3.backtest.run
```

Example of Backtest Report

------------------------ Backtest Report --------------------------------

Total Return (%)=-3.21 in 33 bars

Average Bar Return (%)=-0.099

Std Deviation of returns (%) =1.17

----------------------- End of Report ---------------------------------

- As you may see in the numbers, it is an example of Backtest Trader report showing a **bad** performance

Evaluating Trading Robots

- The method backtest.run creates a data file with the name given in the backtest setup (bts)

- In order to analyze the trader's backtest, you may use stse package :

```
import stse
stse.evaluateFile(fileName)
```

- This will give you a report about the trader performance

- We need ot 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 operations. For a deeper discussion, we suggest:
  - Is it a great Autonomous Trading Strategy or you are just fooling yourself Bernardini,M. and Castro, P.A.L

Conclusions and Next steps

- Building autonomous trader is a very complex task!
- The financial environment could be classified in a classic way as:
  - partially observable,
  - sequential,
  - stochastic,
  - dynamic,
  - continuous
  - and multiagent,
- Which is the most complex environment class
- However, it does not really represents the whole complexity of the problem.
  - More than stochastic, such environment is also a non-stationary process (Probability distributions do change along the time) and it is also strategic in the sense that two active investors compete for a more accurate valuation of assets and their actions may change other agents’ behavior.
- Let's talk about the Financial Environments and The present and Future of Autonomous Investments

Financial Environment and its distinct features

- “The rate of failure in quantitative finance is high, particularly so in financial ML. The few who succeed amass a large amount of assets and deliver consistently exceptional performance to their investors. However, that is a rare outcome…”[1]
- “..Just because a theorem is true in a logical sense does not mean it is true in a physical sense…”

- On the other hand, many initiatives in Financial ML misuse mathematical tools to describe actual observations. Their models are overfit and fail when implemented.

- “Academic investigation and publication are divorced from practical application to financial markets, and many applications in the trading/investment world are not grounded in proper science”

The Present and The Future…[??]

- Present? "…Investors are lured to gamble their wealth on wild hunches originated by charlatans and encouraged by mass media…" [1]
  - Do you agree?
- "*One day in the near future, ML will dominate finance, science will curtail guessing, and investing will not mean gambling*" Marcos Lopez de Prado
  - Do you believe that revolution is going to happen? If yes, would you like to be part of it?

Can Machines invest [better than us]?

- This question reminds the one raised by Alan Turing in the paper "Can machines think?" [1]. Turing discuss many objections pointed out to reinforce the idea that machines will never be able to really think.
- Some of these objections could be raised against the idea of machines that can analyze investment…but not all of them. For instance, the theological objection. It seems unlikely that someone would argue that analyzing investments is a function of man's immortal soul.
- Let's briefly discuss what seems to us the most relevant objections to machine that can invest…

This section comes from the paper: CASTRO, P. A. L.; ANNONI JUNIOR, R. SICHMAN, Jaime Simão Análise Autônoma de Investimento: Uma Abordagem Probabilística Discreta .Revista de Informática Teórica e Aplicada (RITA). vol. 25. num. 1. (Jan,2018).pp. 23-38. 2018.

Objections to Machine that can invest

- **The Mathematical objection**: Investment analysis or management is more than logic, it is kind of art, so it is beyond the limits of computability.
  - We use here Turing's short answer "...although it is established that there are limitations to the powers of any particular machine, it has only been stated without any sort of proof, that no such limitations apply to the human intellect..."

- **The Heads in the Sand objection**: The consequences of machines controlling investment would be too dreadful or : Machines controlling investment would steel jobs from real people or even creating catastrophic crises in global markets.
  - Turing answers such objection, stating that this argument is not sufficiently substantial to require refutation and consolation would be more appropriate. We would add that if AIA can be really efficient, perhaps it would be more likely that financial crises would become more rare

- **Other disabilities**: Machines could do significant part of the job, but no machine will ever be able to do X in investment analysis. Numerous features X can be pointed out, for instance: be intuitive, have common sense, be innovative, think something really new.
  - In fact, some of this features can be very hard to achieve, However, there is no hard evidence it is impossible. Furthermore, one may argue that would be possible the existence of an efficient AIA even without common sense, provided it has access to all relevant information

Why not just give the money to any professional manager ?

- Choosing among professional managers is pretty much the same problem of choosing among investments, but there is another issue:

- **Conflict of interests:** there are possible conflicts of interest among analysts and investors, when analysts have investments in the 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 (please, read the paper and citations to more info about that)

- **Interest of Machines:** Due to the fact that machine can have controlled or at least formally verifiable interests, possible conflict of interests can be avoided or at very least controlled in a more efficient way.

What happens if autonomous investment analysts or managers become ubiquitous?

- Would everybody become rich or at least present very high average returns on their investments?
- The short answer is **no**.
- We believe the scenario described by Fama [6] in his Efficient Market Hypothesis (EMH) 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 for simple chance

Next: Advanced Trading Robots

multiasset, multistrategy, investor profile awareness, risk control:

towards autonomous portfolio management