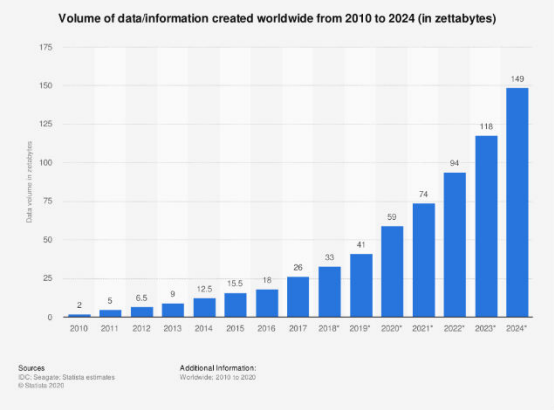
Summary – Financial Machine Learning – Stefano Ciccarelli

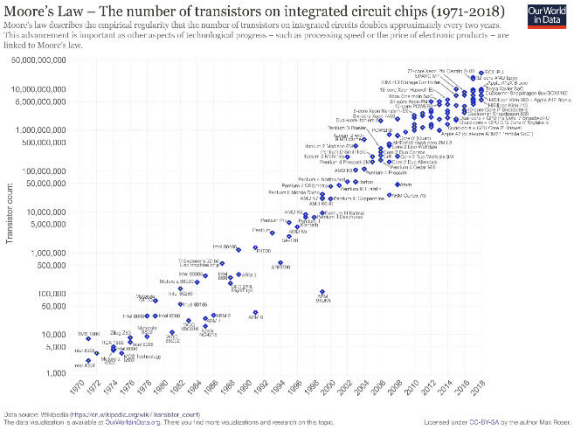
The analysis starts with the current worldwide technological landscape: answering why Machine Learning is becoming relevant for Shareholders and how this can contribute to generating value for the overall society.

This call has a focus on economic and social inclusiveness given the expected non-linear risks generated by the potential Singularity's developments on the overall society, identifying the Governments as crucial in establishing a financial singularity strategy without disincentivizing innovation.

The main scope is to nudge a preventive solution oriented to maximize the resilient abilities of the local communities and to develop AI-indexed and passive alternative sources of income, with a scale of importance specular to the level of automation inside the supply chain of the main industries.

After explaining the current and projected trajectory of Machine Learning, since the exponential trend of the Big Data and the Computer power, 4 technical practical financial approaches using powerful computational statistical algorithms is shown.





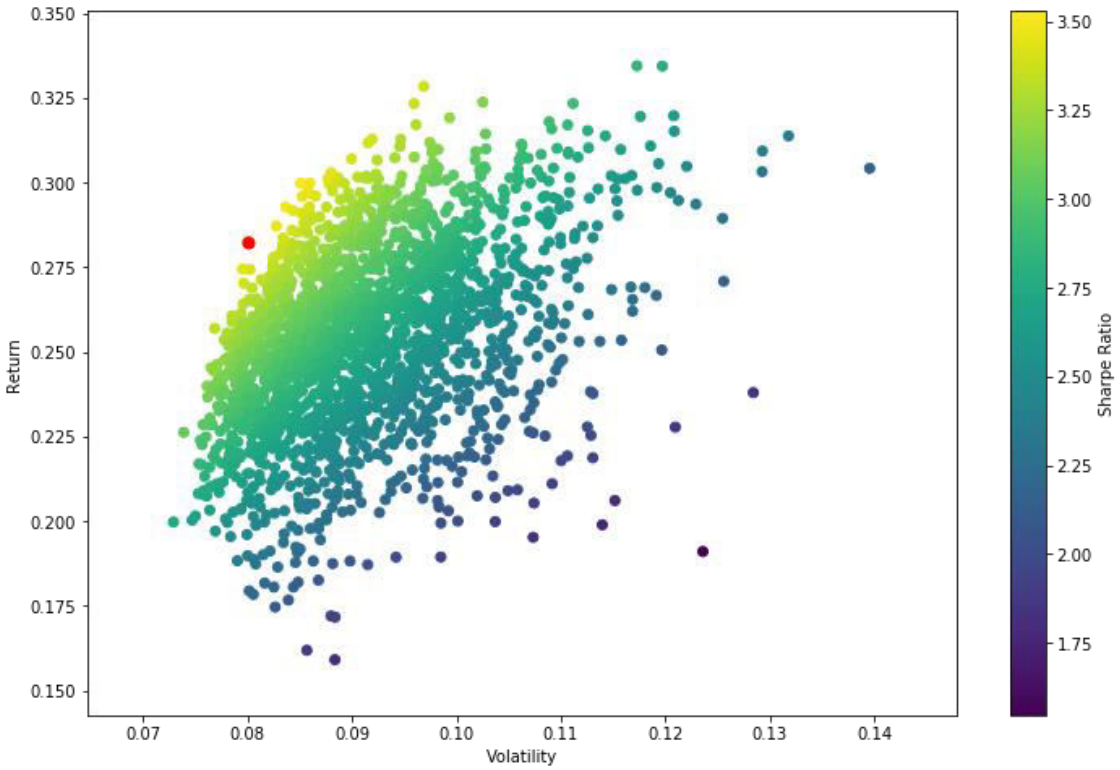
Overall, the thesis is structured using a strongly quantitative and data-driven approach: to let data prove the underlying assumptions and leveraging the analytical power of Python.

I. Deep Neural Networks for Stock Selection in Portfolio Optimization

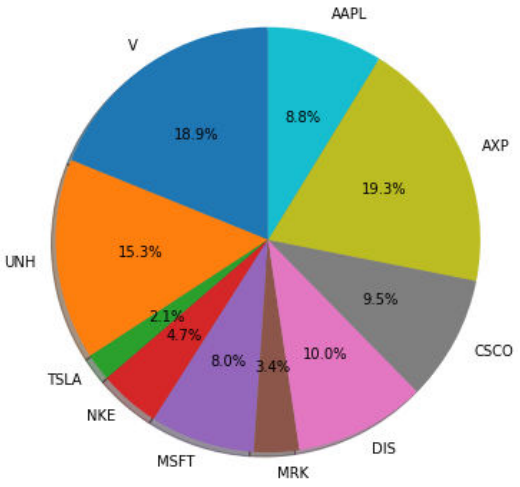
The key final result in the last chapter seems to support how a Neural Network can outperform the financial valuation abilities of a team of professional analysts in selecting stocks, once the model is validated on data that were never observed by the machine, obtaining an average YoY +42.43% return.

It was trained on a period span of 5 years data and on more than 100 fundamental parameters (as relevant Balance Sheet and Income Statements voices) on a set of 30 Companies.

It automatically selected the most relevant interactions among the standardize parameters (the relative relevancy of the ratio respect to the overall population), discriminating accurately the best companies to select and insert in the portfolio optimization strategy.



Following the portfolio optimization and the obtention of the efficient frontier, a Sharpe Ratio of 3.53 is found, with an expected return of +28.26% and a volatility of 8%, implying that in the worst of all empirical case (99.9th Percentile) the return would be still positive (+1.93%).



To give a benchmark on the same period of analysis the market index (S&P 500) obtained a return of – 4.38%.

This result is of particular interest, especially in an historical period where the results achievable by the emerging technologies are still debated, supporting, and clarifying the long-term vision of the potential performances of Machine Learning in the next decade.

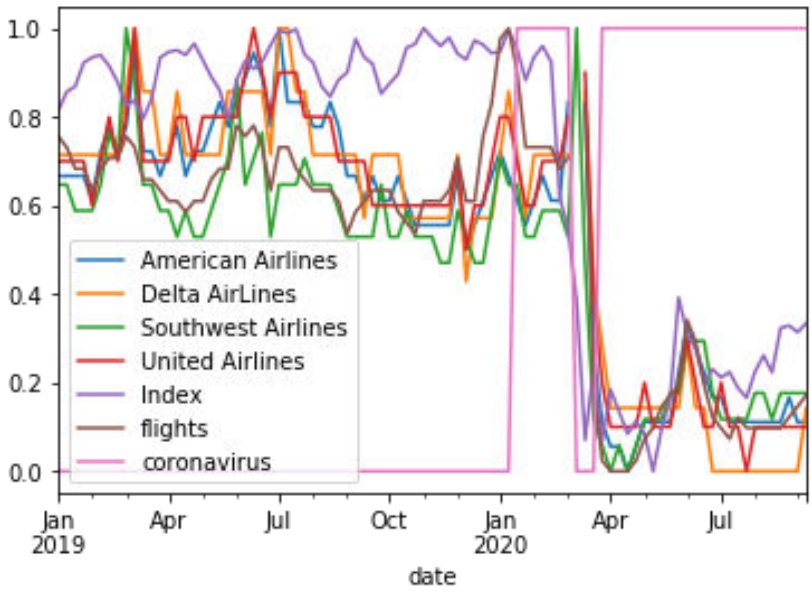
In fact, a similar approach could be used as a baseline to structure an AI-managed portfolio, available inclusively at 0 fixed cost to the local population in order to incentive the diversification of the sources of income and minimize the impacts of the expected negative externalities of the AI on the long-term technological unemployment.

II. Optimized Neural Networks with AutoKeras: Analyzing the COVID-19 Impacts on NASDAQ US Benchmark Airlines Index using Google Trends

In another case, the effects of the number of searches related to airlines on Google, together with a sample of the behavioural "worrying level" related to COVID-19, were used to analyse and monitor the NASDAQ US Benchmark Airlines Index level in a way that was not possible before, clarifying when we can expect a recovery of this index For example, if the demand for the US Airlines tickets will start to increase again, using a Neural Network for classification purposes, with a Boolean class we can generate an expectation regarding when the Index will turn back to the pre-virus status.

Here the correlations, between the variables (relative number of Google Searches) and the Index and the standardized plot (on the same scale):

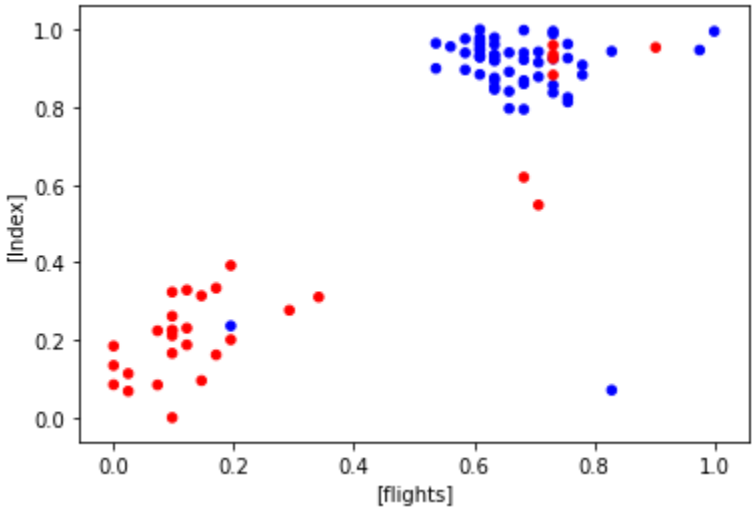
	Airlines Index
American Airlines	0.776127
Delta AirLines	0.739754
Southwest Airlines	0.819368
United Airlines	0.773032
flights	0.710131
coronavirus	-0.621369



Here instead the 2 clusters observable:

I. High number of searches – High Index Values

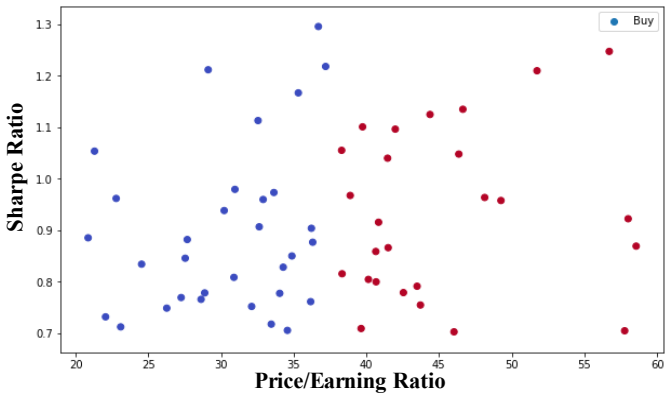
II. Low number of searches – Low Index Values



Index on the y-axis and number of searches for flights on the x-axis, in red the COVID-19 "concern" period.

III. Applied Financial Machine Learning with Scikit-learn

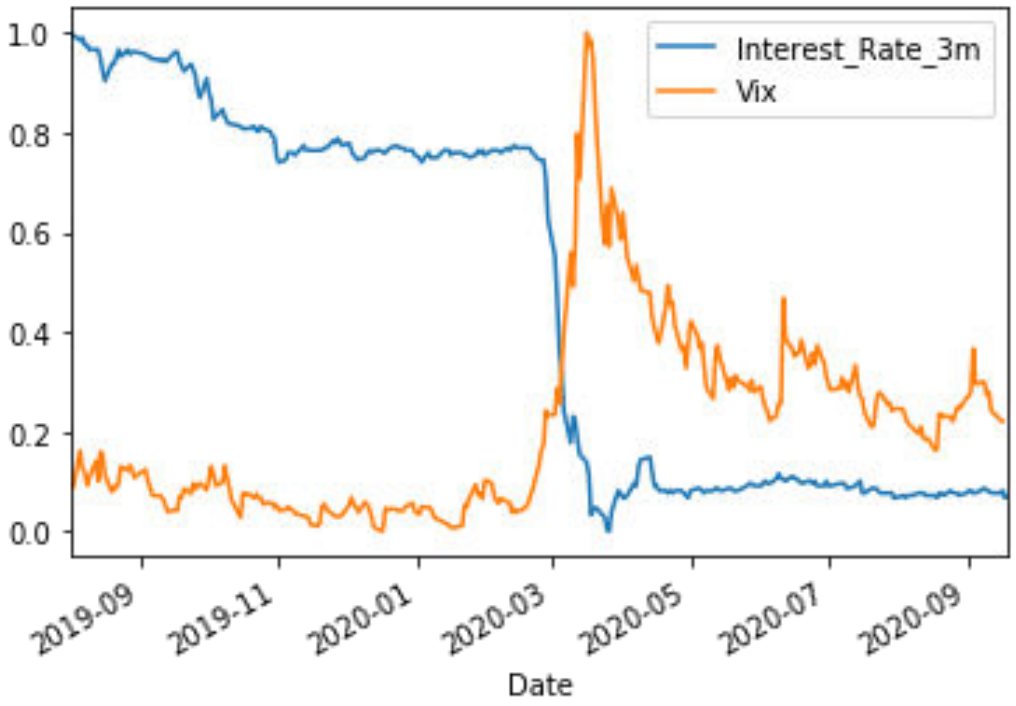
Between the different techniques, a Machine Learning clustering method using Scikit-learn was applied to the S&P 500 Index Companies, where after computing massively the returns and the volatility and obtained the Sharpe Ratio (y-axis), this was plotted against the P/E (x-axis) to automatically classify and consequentially identify the Buy companies against the Not-Buy ones.

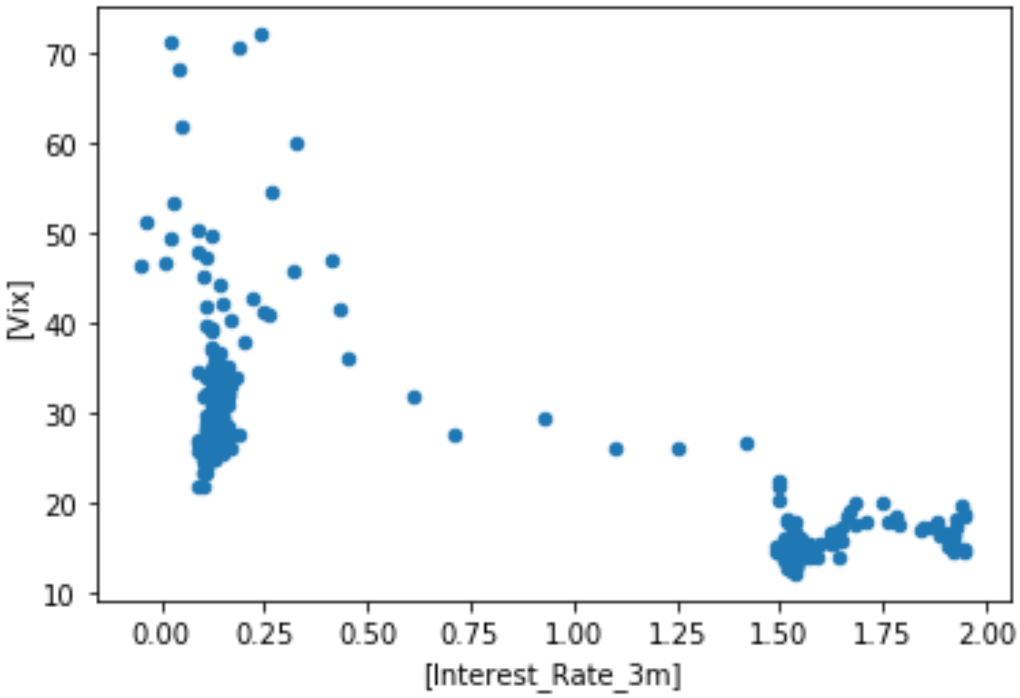


The results of this example can change according the parametrized Investor's preferences.

IV. Deep Neural Networks to Forecast Market Implied Volatility (VIX) using the Short-Term 3-M US Treasury Bonds Rates

Finally, the ML potential is also shown on forecasting the market implied volatility (VIX Index), using the negatively correlated short term risk-free interest rates (3-M US Treasury Bonds): to model how the Governments and Central Banks interventions applying Quantitative Easing policies can negatively affect the short-term market volatility pushing the Investor towards the equity market's higher yields and defining when we can expect the VIX Index to increase and consequentially to turn back to a normal level (under the value of 20).





In this case the Neural Network is able, once observed the daily interest rates of the last 3 Weeks, to answer if the volatility is expected to gradually turn back to normality, under a value of 20.

Final Comment:

The main vision of this work is to experiment how automation is affecting Finance in the long-term and how interconnecting the dots and different models could be accretive in constructing an automated financial mechanism to prevent the potential crises generated by the Singularity risk, generating a sustainable monthly cash-flow for the most exposed communities.

LUISS

Department
of Business and Management & Economics and Finance

Course of Machine Learning and Applied Statistics - Imperial College [ELE1]

Financial Machine Learning

Prof. LAURA L.
SUPERVISOR

Prof. ITALIANO G. F.
CO-SUPERVISOR

ID No. 703861 - Ciccarelli S.
CANDIDATE

Academic Year 2019/2020

The Future will be Exponentially Better

Dedicated to My Parents

"Artificial intelligence programs like deep learning neural networks may be able to beat humans at playing Go or chess, or doing arithmetic, or writing Navy Seal copypasta, but they will never be able to truly think for themselves, to have consciousness, to feel any of the richness and complexity of the world that we mere humans can feel.

Mere, unenlightened humans might be impressed by the abilities of simple deep learning programs, but when looked at in a more holistic manner, it all adds up to… well, nothing. They still don't exhibit any trace of consciousness.

All of the available data support the notion that humans feel and experience the world differently than computers do. While a computer can beat a human master at chess or Go or some other game of structured rules, it will never be able to truly think outside of those rules, it will never be able to come up with its own new strategies on the fly, it will never be able to feel, to react, the way a human can.

Artificial intelligence programs lack consciousness and self-awareness. They will never be able to have a sense of humor. They will never be able to appreciate art, or beauty, or love. They will never feel lonely. They will never have empathy for other people, for animals, for the environment. They will never enjoy music or fall in love, or cry at the drop of a hat.

Merely by existing, mere, unenlightened humans are intellectually superior to computers, no matter how good our computers get at winning games like Go or Jeopardy. We don't live by the rules of those games. Our minds are much, much bigger than that."

Written by the Artificial Intelligence GPT-3, Open-AI[1]

[1] https://openai.com/blog/openai-api/

Machine Learning and Applied Statistics

Course Taken at Imperial College Business School

The Data-Driven approach was developed thanks to the QTEM

And the Brilliant Experiences at EDHEC Business School and Warwick Business School

Contents

Introduction 7
Financial Singularity Strategy 7
History 11
Why hedging against the Singularity Risk 13
Machine Learning 16
Why 16
The Big Data 16
The Computational Power 18
The Algorithms 20
How 22
Supervised Learning 23
Unsupervised Learning 24
Semi-Supervised Learning and Reinforcement Learning 26
Data-Driven Finance 28
Financial Automation with Python 28
Storing and Simulating Financial Big Data with NumPy and Pandas 29
Applied Financial Machine Learning with Scikit-learn 41
Optimized Neural Networks with AutoKeras: Analyzing the COVID-19 Impacts on NASDAQ US Benchmark Airlines Index using Google Trends 48
Financial Machine Learning: A Practical Approach 65
Deep Neural Networks to Forecast Market Implied Volatility (VIX) using the Short-Term 3-M US Treasury Bonds Rates 65
Deep Neural Networks for Stock Selection in Portfolio Optimization 78
Conclusions 94
Bibliography 97

Introduction

Financial Singularity Strategy

The main Vision of this work it is collocated in a framework that has as pillar a simple, but game-changing, assumption.

In the next 50 years the evolution of the Artificial Intelligence, and consequently the exploitation of the AGI – Artificial General Intelligence [2] – will be soon a high likely reality.

Consequently, the human workforce, will become gradually unnecessary, exposing societies, characterized by populations at high risk of unemployment and with the salary as the only source of income[3], to disruptive crises, and the fallen of the governments and political decision makers under the control of a technocognitive aristocracy.

The preventive solution is connected to the resilient abilities of the local societies to develop AI-indexed and passive alternative sources of income, with a scale of importance specular to the level of automation inside the supply chain of the main industries.

We are going towards the greatest period of the known human history, until now, and to ensure the future generated wellness will be spread meritocratically and democratically among the population is necessary to ensure the consistent support to the education[4] and the consciousness, necessary to coexist and to improve systematically the support to the scientific research.

Finally, a practical application of basic machine learning algorithms will be tested on financial instruments, to show and validate the automation opportunities of diverse sources

---

[2] Goertzel, B. (2014). Artificial General Intelligence: Concept, State of the Art, and Future Prospects, Journal of Artificial General Intelligence, 5(1), 1-48. https://doi.org/10.2478/jagi-2014-0001

[3] Abbott, R., & Bogenschneider, B. (2018). Should robots pay taxes: Tax policy in the age of automation. Harvard Law & Policy Review, 12(1), 145-176.

[4] Duncan, G. J., Magnuson, K., & Votruba-Drzal, E. (2017). Moving Beyond Correlations in Assessing the Consequences of Poverty. Annual review of psychology, 68, 413–434. https://doi.org/10.1146/annurev-psych-010416-044224

of income, and the potential effects of an educational income[5] in supporting the skills transfer[6].

[5] Ciccarelli, Stefano (2017), Manifesto della Geniocrazia, Cap. Reddito di Formazione ("Geniocracy Manifesto, Chapter – The Educational Income")

[6] Adams, T. L., & Demaiter, E. I. (2008). Skill, education and credentials in the new economy: the case of information technology workers. Work, Employment and Society, 22(2), 351–362. https://doi.org/10.1177/0950017008089109

Time

To standardize the chaotic changes of the absolute space, the humankind required time.

In fact, time should not exist in absolute terms, but it changes relatively [7] to the cognition and its relative environment.

While it can take years, or decades, for a human mind to collect and process all the necessary information and to develop an expertise, an AI Algorithm requires few instants[8], allowing for – perceived as exponential – growth opportunities.

To maximize the clearness of the relative picture of the absolute space, is strictly necessary to precisely measure time, and consequentially to collect data with the optimal frequency, directly increasing the demand for a firm infrastructure: and nowadays, for scientific purposes, Caesium is the top performing resource to focus on.

Controlled, at the base of its Supply Chain, by China for 96% of its production, this material is necessary to develop the necessary data-platform to reach the AI military and political supremacy[9]: starting from 5G, consequentially allowing for the IoT organism next level generation to evolve together with Smart Cities and Smart Devices, and arriving to the final destination of an Artificial General Intelligence, able to generalize from the data collected and to produce Academic and Industrial advancements.

Time is also the inverse output performance metric: instead of measuring the quantity of real resources produced in a year, the Macroeconomics actors should focus on the time of production/transformation of each unit of standardized good.

---

[7] Buhusi CV, Meck WH (2009) Relativity Theory and Time Perception: Single or Multiple Clocks?. PLOS ONE 4(7): e6268.
https://doi.org/10.1371/journal.pone.0006268

[8] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis (2018),
A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play (DeepMind).
https://doi.org/10.1126/science.aar6404

[9] Ciccarelli, Stefano (2020), The US-China Cold War made easy (Article):
https://www.financecs.com/2020/05/22/the-us-china-tech-cold-war-made-easy/

Since, using GDP (subject to the demand volatility and the relative supply) as actual measure, is not possible to measure directly for the opportunities-risks of automation, but ,instead, using the delta changes of production time of each unity of real average goods and services, it is possible to have a clearer picture of the trajectory of the automation inside the Industries to assess more objective data-driven global long-term policies.

In fact, the monetary exchanges, in performing a consume utilitarian or perceptual experience, represent the perceived economic value generated: an increase in the availability of resources, indirectly, imply a saving of time respect to the individual gathering necessary to perform the same type of experience, suggesting that the time saved, representing the energy exchanges economisation, is the main dimension of the Macroeconomic scenario.

History

For the time, only the present exists, and the past is the collected memories regarding each perceived experienced present, or more precisely a change in the relative perceived space: it exists as it is necessary to the cognition to define the trend to project and anticipate the next presents (alias "the future").

To optimally understand the potential directions of the humankind, and determine the best scenario to reach, is necessary to collect the empirical data in a clear and structured framework: this is also necessary to AI, to be able to generalize patters with a reduced bias. Practical automation applications are showing their effectiveness using this approach, where not only the logical flow it is inscribed inside the procedures, but also the data and the past are used to generate transformations inside the artificial "organism" without human explicit directions.

We are in the phase of human history where the Financial applications and tools are shifting gradually from a top-down to a bottom-up approach, where the necessaries human inputs start to reduce.

From the first calculator, invented by Blaise Pascal around 1642, to automate the 4 main arithmetic operations, reducing for the need of the human mind in executing the basic mathematical computations, the willingness to automate, and consequentially improve the tasks that require human intelligence, opened the doors to new opportunities and new industries.

Especially the Globalised Financial Industry, that allows for faster and more efficient capital allocations, could be considered a product of both technological and political achievements: the spread of the Telegraph from the 1844 [10] necessary to strongly reduce the geographical information asymmetries and the first trans-Atlantic cable (1866) [11] that

[10] Du Boff, R. (1980). Business Demand and the Development of the Telegraph in the United States, 1844-1860. The Business History Review, 54(4), 459-479. Retrieved July 12, 2020, from www.jstor.org/stable/3114215

[11] Freezee, W. (1978). The First Trans-Atlantic Cable. Journal of the Washington Academy of Sciences, 68(1), 3-13. Retrieved July 12, 2020, from www.jstor.org/stable/24537173

allowed to reduce the communication between Europe and North American time from 10 days to only 3 hours, were the presage of the modern present destination.

The begin of the digitalization of financial markets around 1962, with the computerized stock quotation system QUOTRON[12], and the following establishment of the SWIFT [13] network (1972) to standardize the international communications, arriving to the GLOBEX (between 1987 and 1992) electronic trading platform, can be considered the presage of the future Artificial General Intelligence capital allocation.

The first phase of the history, has seen an elimination of geo-informational barriers in accessing to investment opportunities, soon the second phase, given the internet diffusion that produces the collective digital memory, together the evolution of the Fintech applications that learn form it, will see a gradual reduction of the cognitive barriers to an optimal capital allocation.

In this, the governments will be central to establish an efficient supervision during the transition from a trust-based banking system, to an Open-Banking[14] track-based one.

The firstcomer to reach an AGI Financial supremacy would, together a decentralized Blockchain 2.0[15] system, produce the highest measurable returns worldwide, given the reduced cost structure and superior performances, attracting capital worldwide and reshaping the global industries.

---

[12] IEEE History Center Staff, "Proceedings of the IEEE Through 100 Years: 1960-1969 [Scanning Our Past]," in Proceedings of the IEEE, vol. 100, no. 7, pp. 2380-2386, July 2012. https://doi.org/10.1109/JPROC.2012.2193712

[13] Susan V. Scott & Markos Zachariadis (2012) Origins and development of SWIFT, 1973–2009, Business History, 54:3, 462-482. https://doi.org/10.1080/00076791.2011.638502

[14] Zachariadis, Markos and Ozcan, Pinar, The API Economy and Digital Transformation in Financial Services: The Case of Open Banking (June 15, 2017). SWIFT Institute Working Paper No. 2016-001, Available at SSRN: https://ssrn.com/abstract=2975199 or http://dx.doi.org/10.2139/ssrn.2975199

[15] Fanning, K. and Centers, D.P. (2016), Blockchain and Its Coming Impact on Financial Services. J. Corp. Acct. Fin, 27: 53-57. https://doi.org/10.1002/jcaf.22179

Why hedging against the Singularity Risk

Externalizing and outsourcing the cognitive abilities generate a dependency towards the unknown: the interactions inside the layers and the patterns frequently behave like a black box even for the AI creator.

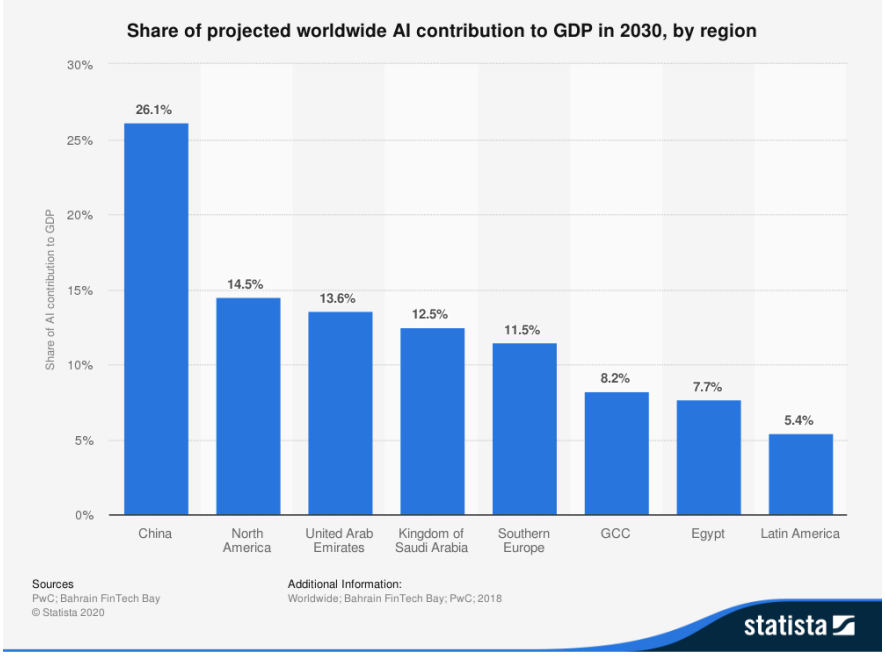
Consequentially societies must not become the victims of a technological cognitive Elite able to build the systems that will push the economy to a zero-cost marginal structure[16].

Being able to resiliently structure an Automation-Indexed public Fund (AIF) is a necessary solution to ensure the democratic and inclusive generation of the income necessary to sustain the part of population classified as not employable, ensuring a global ownership of the wealth evolution, without disincentives to the free-market to produce innovation in case an automation-tax would be the alternative.

This concept is inside the framework of a more general Financial Singularity Strategy, where the capital allocation and the relative returns are maximized to ensure an inclusive sustainability: to allow this, the main objective would be to prepare the necessary environment for the Singularity[17] who will invest and share the wealth as a definitive multi-lateral and super-rational agent in behalf of the population.

[16] Rifkin, J. (2014). The zero marginal cost society: The internet of things, the collaborative commons, and the eclipse of capitalism. New York: Palgrave Macmillan.

[17] Eden, Amnon & Moor, James & Søraker, Johnny & Steinhart, Eric. (2013). Singularity hypotheses. A scientific and philosophical assessment.
https://doi.org/10.1007/978-3-642-32560-1



1.1 Worldwide; Bahrain FinTech Bay; PwC; 2018

In the graph 1.1 is shown the projected AI worldwide contribution to GDP: this represent the first step in the direction of a global commitment to a more automated society, that in the long-term will increase the need for also an automated cash-flows generation to increase the sustainability.

In fact, focusing only on the automation of the Supply would generate negative externalities and produce technological powerful monopolies reducing the potential scientific developments apported by a more inclusive society: investing in people via an educational income would maximize the scientific output while automation increases among production processes, otherwise the increase of the disparities and of the relative sources of income scarcity would push people to focus only on the basics Maslow pyramid's needs[18].

[18] McLeod, S. A. (2020, March 20). Maslow's hierarchy of needs. Simply Psychology. https://www.simplypsychology.org/maslow.html

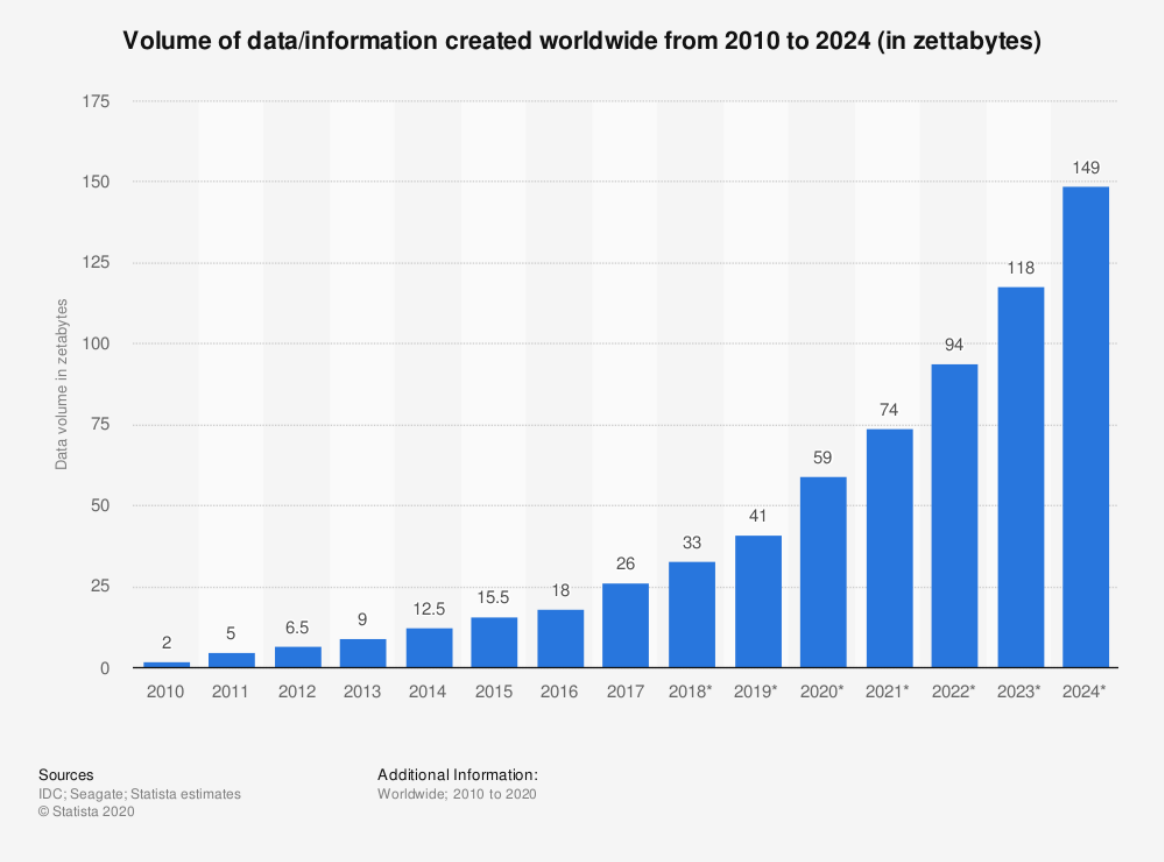
Machine Learning

Why

The knowledge applied to establish rational top-down decisions is not only based on the collected information but also on the ability to account for the interactions of the different objects and ideas and to recognize the underlying patterns.

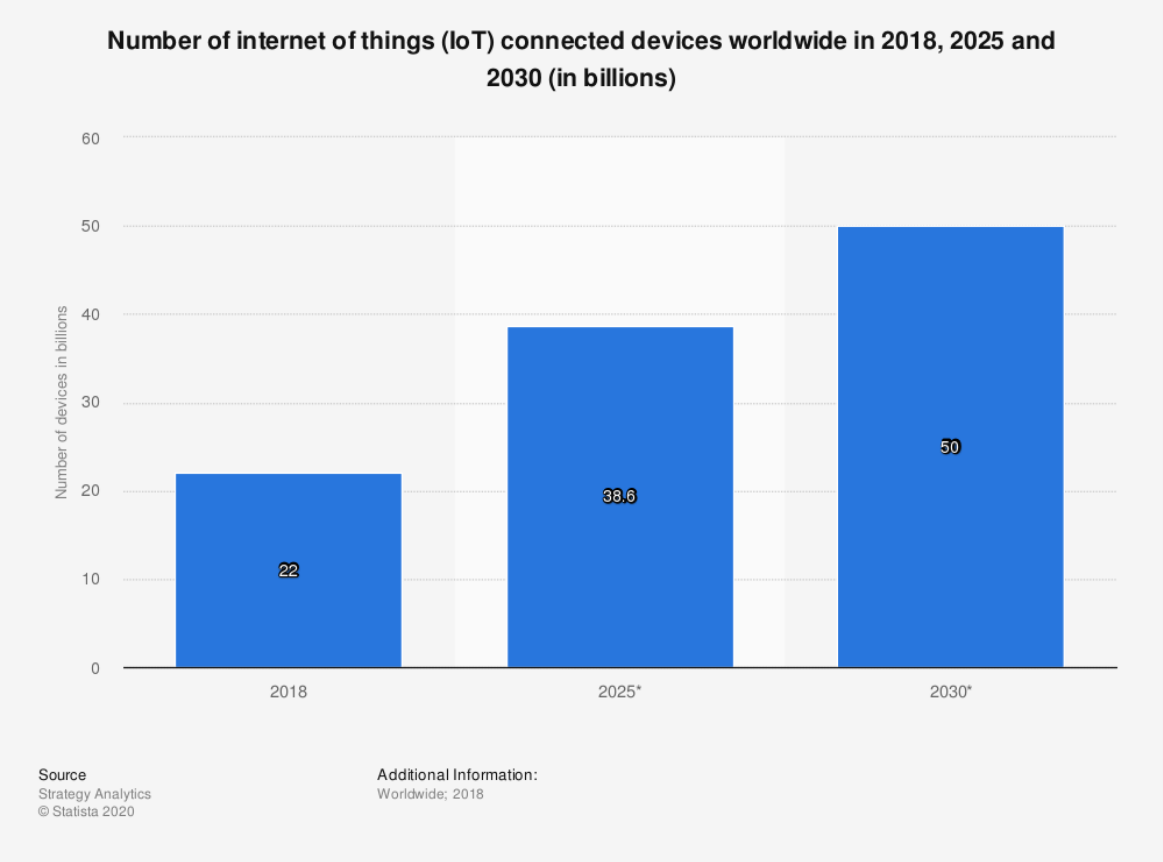
Similarly, in statistical and probabilistic terms, we can observe the increasing abilities of the machines to “learn” because of 3 main trends:

The Big Data



2.1 IDC; Seagate; Statista estimates

The online information is growing at an evident exponential trend (2.1), and according McKinsey analysts "90 percent of the digital data ever created in the world has been generated in just the past two years, only 1 percent of that data has been analyzed."[19]



2.2 https://www.businesswire.com/news/home/20190516005700/en/Strategy-Analytics-Internet-Things-Numbers-22-Billion

Taking also in account the expected development and the spread of the worldwide IoT Ecosystem shown in the graph 2.2, it is reasonable to assume that the projected quantity of

[19] https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/straight-talk-about-big-data

structured data, together with the increase of normalization practices of the unstructured types, will aliment and sustain the generation of the online useful information available.

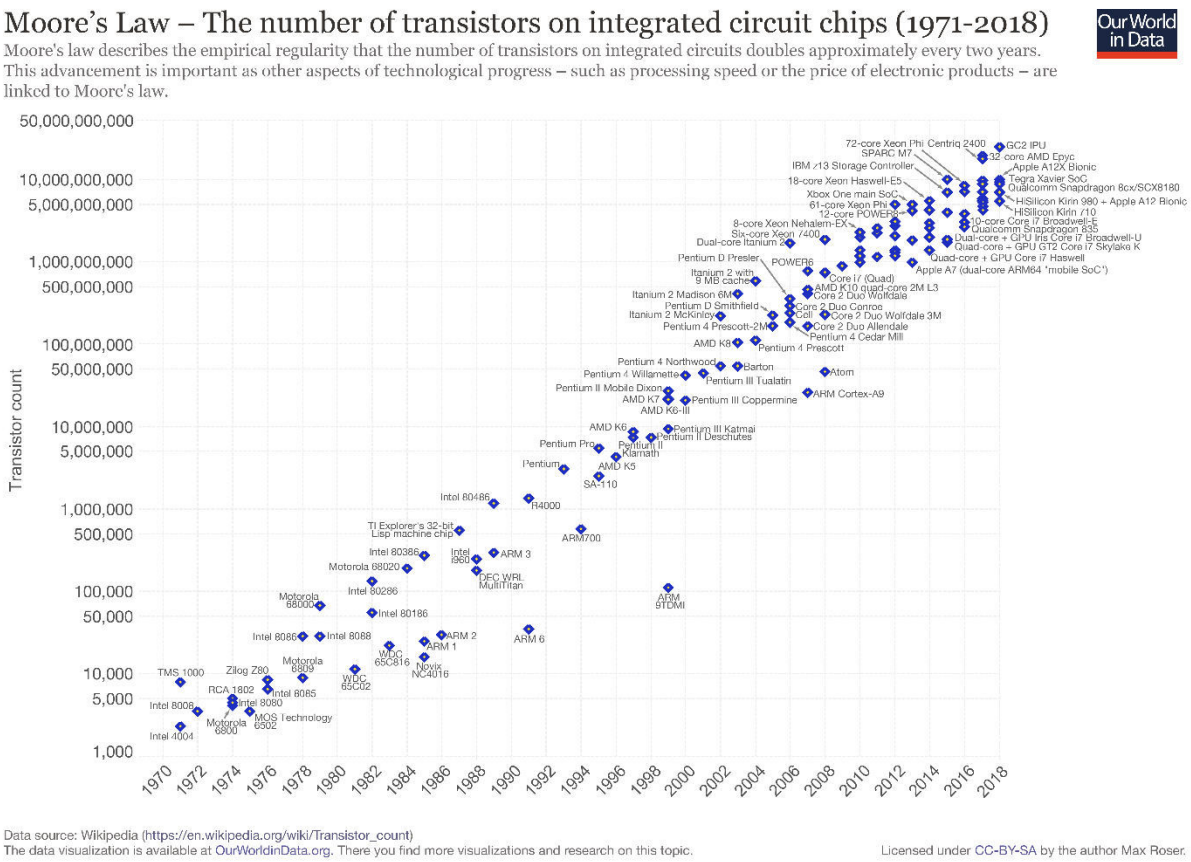
Consequentially, a statistical algorithm can potentially use those data to boost its "experience" and subsequentially to improve the generalization of patterns relative to a set of tasks.

The Computational Power

According the empirical Moore's law, $n_2 = n_1 * 2^{[(y\_2 - y\_1)/2]}$ where n is the number of transistors and y the year, the computing power double every two years. Some members of the scientific community argue there will be soon a physical limit[20] to this trend, but until today the Moore observation is still holding.

This exponential rate allowed in the past years to increase the speed at which the information is processed, determining not only a reasonable execution time of the algorithms with a larger number of features and interactions, but also a more democratic and accessible spread of the high-level programming languages, like Python, that open the doors for a faster deployment and a higher number of industrial applications.

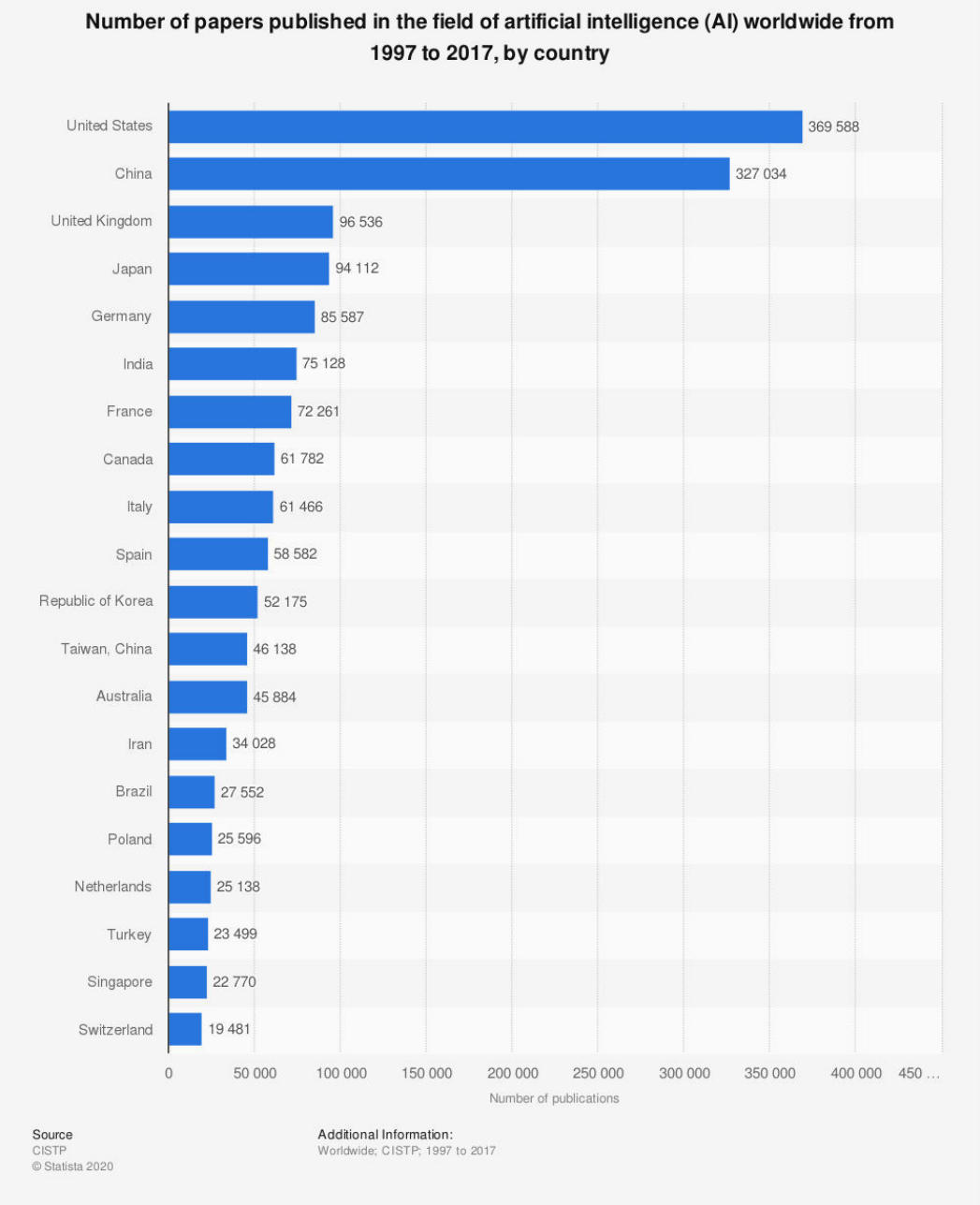
[20] J. R. Powell (2008), "The Quantum Limit to Moore's Law," in Proceedings of the IEEE, vol. 96, no. 8, pp. 1247-1248.
https://doi.org/10.1109/JPROC.2008.925411



2.3 https://ourworldindata.org/technological-progress

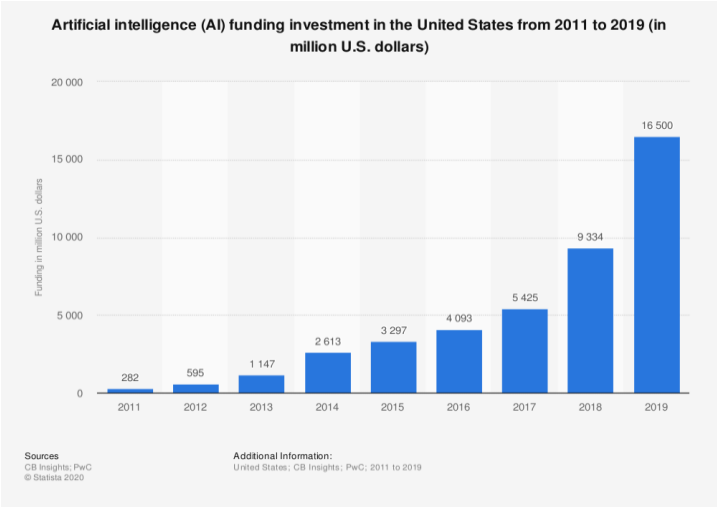
The Algorithms

Alpha Go[21] signed the Sputnik moment for China, increasing the competition and bringing to the contemporaneous tech cold war.

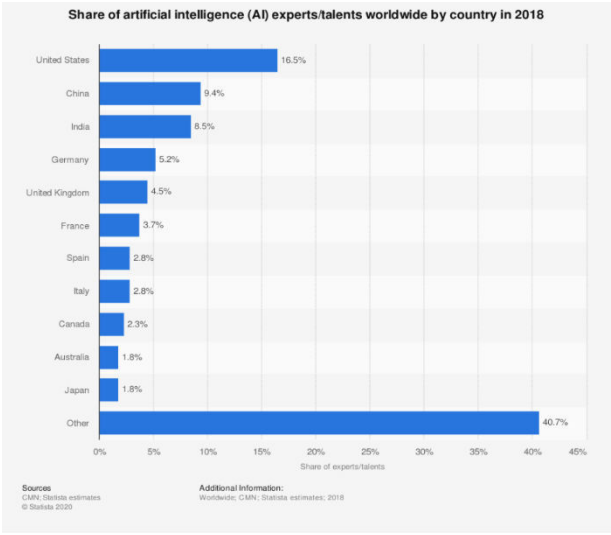


2.4 CISTP

Consequentially not only the large pool of the previous algorithms from academia must be taken in account in the final scenario, but the potential ones that will be developed in the next 10 years in the aim of reaching the AI technological and military supremacy.
In the graph 2.4 we can observe the equilibrium during the period 1997-2017 of the number of AI publications by country, but we can reasonably expect, given the current Geo-Political tensions, a run towards the massive exploitation of the worldwide AI talents.



2.5 https://www.pwc.com/us/en/moneytree-report/assets/pwc-moneytree-2020-q1.pdf



2.6 CMN; Statista estimates

In fact, in the graph 2.5 we can deduct the increasing commitment in the United States for Artificial Intelligence applications in terms of resources, and instead in the graph 2.6 is possible to observe the “soft-power” abilities in terms of attractiveness for the AI talent necessary to develop the future infrastructures required to win the tech challenge.

How

Machine Learning can be defined, according Tom Mitchell, as *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E"*[22].

On the academic level it can also be collocated as the intersection of:

I) **Computer Science**, that provides the necessaries formalization of the algorithms for the specific dataset.

II) **Statistics**, that provides an inferential approach, formalizing a model for the specific dataset.

Consequentially it can be approached as a set of "statistical algorithm" able to perform a defined task that improve among time according the higher data availability.

The main learning techniques are divided in 4 big categories: Supervised Learning, Unsupervised Learning, Semi-Supervised Learning and Reinforcement Learning.

[22] Mitchell, T. (1997). Machine Learning. McGraw Hill. p. 2. ISBN 978-0-07-042807-2.

Supervised Learning

Supervised Learning is characterized by labeled outcomes y_i for each feature vector x_i. A feature vector can have dimensionality D, with each feature describing different quantifiable perspective of a phenomenon.

Instead, the label can have different complex structures, but in the framework of this analysis we will focus mainly on the real numbers of a finite set of classes.

The main objective of this category of algorithms it is to predict starting from the feature vectors, the description of the phenomenon, which is the corresponding label: in fact, once the model is trained on the labeled data, according the quality (bias) and quantity of the data, it should "learn" the common patterns among the features necessary to define the interactions that characterize a specific class or population, or a specific output in the case of real continuous numbers.

Typical examples of supervised learning are linear/logistic regression, k-nearest neighbors, support vector machines, decision trees, random forests and neural networks.

In finance it can find its application for example in the case we have the returns (labels) and the respective indicators (feature vector) necessary to try to predict, given a certain market potential scenario, the expected performance or to classify if the portfolio will have a positive or negative outcome.

Unsupervised Learning

Unsupervised Learning can be applied in the case we have an unlabeled dataset, with only the feature vector x_i available.
In this case it is produced a transformation in the original dataset, with the main objective to assign a scalar value or to reduce the dimensionality of the original vector.

For example in the case of clustering is it possible to understand if there are common pattern to establish classes which the data belong to: given for example the reaction to a market exogenous shock, can we differentiate different stock classes?

In this case the data can give better insights respect a qualitative rating, and we can observe that the chaotic market can be subjected to emotional biases that doesn't take in account necessarily the fundamental value of the companies, differently from the rating agencies.

Thus in the case of clustering we can differentiate and take in account if a stock belong to a specific asset class or not – according the market - using the ex-post data.

Instead, if we want to deduct relevant insights on the overall performance of a series of companies but there are too many parameters and indexes, we can shift to the dimensionality reduction, using factor analysis to understand and analyze the relationships among the different features or also the principal component analysis to build a low-dimensional feature vector, with inside uncorrelated variables, that synthetize the underlying phenomenon starting from a high number of variables characterized by multicollinearity[23].

[23] Alin, A. (2010), Multicollinearity. WIREs Comp Stat, 2: 370-374. https://doi.org/10.1002/wics.84

Typical examples of unsupervised learning are k-means and hierarchical cluster analysis in the case of clustering and PCA and Kernel PCA in the case of dimensionality reduction. In general, it can also be applied to the identification of anomalies and outliers, and consequentially to isolate for example abnormalities or rare market conditions as in the case of a black swan[24].

[24] Taleb, Nassim Nicholas, 1960-. (2007). The black swan: the impact of the highly improbable. New York: Random House.

Semi-Supervised Learning and Reinforcement Learning

Semi-Supervised Learning is applied to supervised learning problems that have relatively few labeled observations and a higher number of unlabeled observations that have the same distribution and are part of the population of the labeled ones.

Generally the algorithms are a mix between supervised and unsupervised approach, for example if we have few balance sheets and the relative information regarding the type of industry, we can try to classify all the balance sheets without an assigned industry to obtain the information regarding which industry those companies belong to.

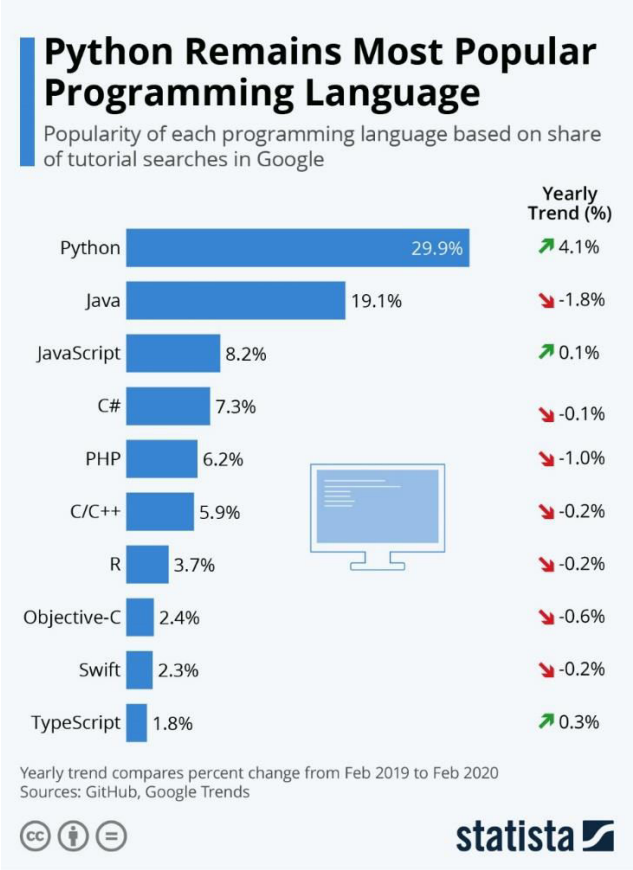
Reinforcement Learning instead has a different approach: the machine, in this case denominated as the agent, perform a series of actions inside a dynamic environment.
The environment can be analyzed, and eventually modified, by the agent who take "choices" according a policy and consequentially receives a specific reward, or also a punishment.

The main objective of the agent is to learn an optimal policy in the long-term after a series of trial and errors performed to maximize the expected reward.

For example a reinforcement learning trading algorithm[25] that provides liquidity to the market can learn from the time series of the prices to maximize the expected return, as the reward, to reduce the mispricing of the asset among time according all the information / sentiment available.

[25] Abhishek Nan, Anandh Perumal, Osmar R. Zaiane (2020), Sentiment and Knowledge Based Algorithmic Trading with Deep Reinforcement Learning.
https://arxiv.org/abs/2001.09403

Being an High-Level interpreted programming language, is really fast to deploy and analyze applications written in Python, factors that contributed to its popularity (3.1) also among finance professionals, indirectly opening the doors to the positive externalities of a large community: in fact, it is an high versatile tool thanks to the high number of libraries and modules available that increase the reliability and the velocity of the execution of different functions and the development of different types of objects.



3.1 Roper, W. (March 3, 2020). Python Remains Most Popular Programming Language. *https://www.statista.com/chart/21017/most-popular-programming-languages/*

Storing and Simulating Financial Big Data with NumPy and Pandas

One of the main problems of the high-level interpreted languages is the low-efficiency on the large datasets, but in some cases of Python can works as a “glue language”, in fact, NumPy one of its most famous and powerful libraries is developed in C and Fortran, apporting the efficiency of the low-level to the speed of deployment of the high-level.

Mainly used for scientific purposes, it is the pillar of a large amount of applications, since it provides multidimensional array objects and a series of method to analyze them.

It is useful to develop vectorized code, emulating mathematically vectors and matrices and consequentially to perform high-speed transformations and functions on the objects.

In terms of financial applications there is the specific package NumPy Financial, that provides a set of high-level methods as NVP, FV, PV, IRR, etc.

For example, if we want to execute a Monte Carlo simulation, we will need to generate a vector of standard errors, and in Python is not only easy but also fast and scalable, as demonstrated in the following code: 100,000,000 standard errors are generated inside a NumPy array and the mean and var methods are computed to confirm that the mean is around zero and the variance around 1.

It takes circa 7 seconds, but according the machine and the necessities it can be applied also for larger simulations and datasets.

Finally, a histogram frequency plot using seaborn is shown to proof graphically that the errors are normally distributed around zero.

Jupiter HTML version of the code available also at FinanceCS.com (Stefano Ciccarelli):
https://www.financecs.com/wp-content/uploads/2020/08/100-Milion-Standard-Error-1.html

Input:

```
# -*- coding: utf-8 -*-
""" Created on Sat Aug 22 23:57:29 2020
@author: Stefano"""
import numpy as np
import time

start = time.time()

# Creating a vector of one hundred million (10^8) standard errors
simulation = np.random.standard_normal(10**8)

end = time.time()

print("The generation of a vector composed by one hundred million of standard errors happened in {:.2f} seconds".format(end-start))

# Showing the mean value approximated to 2 decimals
print("The mean is approximately {:.2f}".format(simulation.mean()))

# Showing the variance value approximated to 2 decimals
print("The variance is approximately {:.2f}".format(simulation.var()))
```

Output:

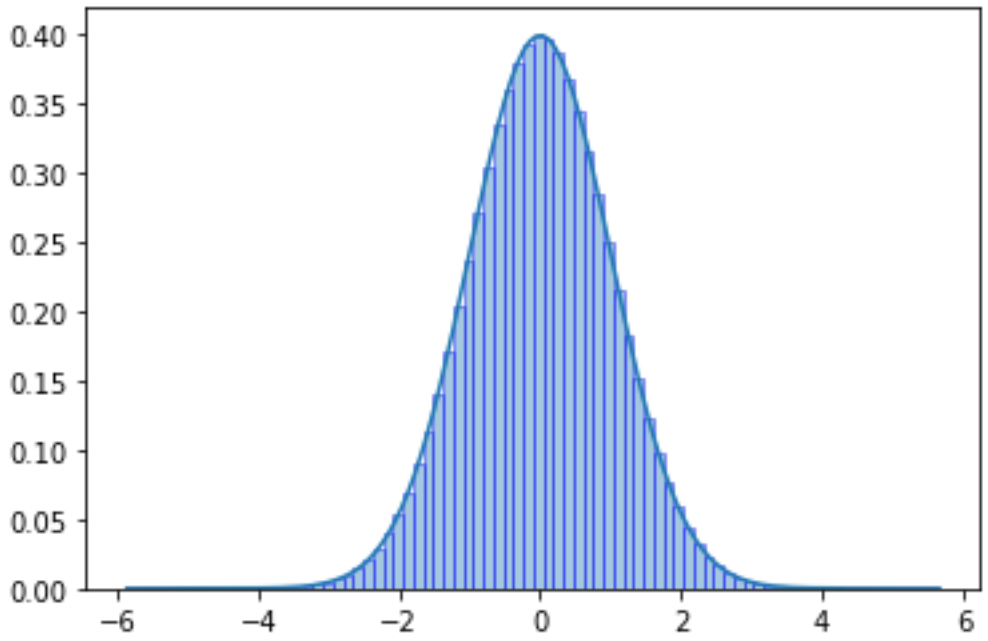
```
The generation of a vector composed by one hundred million of standard errors happened in 6.59 seconds
The mean is approximately 0.00
The variance is approximately 1.00
```

Input:

```
import seaborn as sns

# Plotting the frequency histogram
sns.distplot(simulation, bins=80, hist_kws={'edgecolor':'blue'})
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x234b4fd6808>
```



Instead, if we want to handle different data types at the same time and improve the indexing applying specific methods, especially in the case of time series, Pandas is a better suited solution: it is the natural evolution of NumPy, that allows to store the data in a Data Frame structure, allowing for an easier and faster high-level management.

Here an example of a generation of 5000 stock paths using a geometric Brownian motion having as engine a Monte Carlo simulation and its relative output management using Pandas.

$$S_{\Delta t} = S_0 * e^{\left[\left(\mu - \frac{\sigma^2}{2}\right) * \Delta t + \left(\sigma * \sqrt{\Delta t}\right) * \varepsilon\right]}$$

Jupiter HTML version of the code available also at FinanceCS.com (Stefano Ciccarelli):
https://www.financecs.com/wp-content/uploads/2020/08/Monte-Carlo-Simulation-for-geometric-Brownian-motion-with-Pandas-1.html

Input:

```
import random
import string

# Function to generate a random Stock Tiker with lengh n
def random_stock_tiker(n):
  l_upper = string.ascii_uppercase
  return ''.join(random.choice(l_upper) for i in range(n))

print("Example of a Randomly Generated Stock Ticker of lengh 4: {}".format(random_stock_tiker(4)))
```

Output:

```
Example of a Randomly Generated Stock Ticker of length 4: FOYJ
```

Input:

```
# Generating a list of 30000 random numbers beetween 1 and 4
lenghts = [random.randint(1,4) for i in range(30000)]

# Generating 30000 Stock Tickets, each one of length n for each number inside list lengths
# We keep only the unique Tickets and store the first 5000
stock_market = list(set([random_stock_tiker(ticket_l) for ticket_l in lenghts]))[:5000]

print("Sample of size 10 from the 5000 stock market {}".format(stock_market[:10]))
```

Output:

```
Sample of size 10 from the 5000 stock market ['AHC', 'QLEN', 'YUAS', 'ERBT', 'KUKO', 'UDL', 'PDF', 'SCZ', 'MHPO', 'AEW']
```

Input:

```
import numpy as np

magnitude_volatility = 2.5

# Generating 5000 expected returns, 5000 volatilities and 5000 starting prices (one for each stoc
k)
# With a value between 0% and 10% and maximum 3 decimal points for expected returns and vo
latitlites
# With a value between 1 and 1000 and maximum 2 decimal points for starting prices
expected_returns = np.array([round(random.uniform(0, 10),3)/100 for x in range(5000)])
volatilities =  np.array([round(random.uniform(0, 10),3)/(10**magnitude_volatility) for x in r
ange(5000)])
starting_prices = np.array([round(random.uniform(1, 1000),2) for x in range(5000)])

print("Sample of annual returns of size 10:  {} \n*Values are in %".format(expected_returns[:1
0]*100))
```

Output:

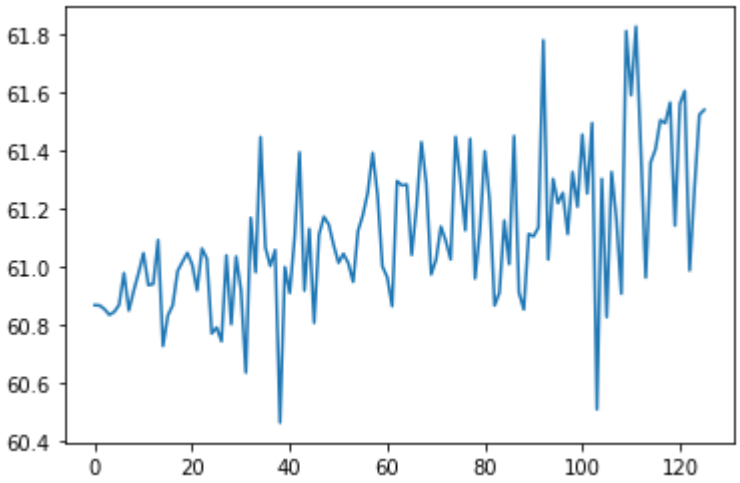
```
Sample of annual returns of size 10:
[8.482 5.848 8.847 6.658 8.471 4.566 5.681 3.45 9.46  5.145]
*Values are in %
```

Input:

```
def geometric_brownian_motion(P_0, delta_t, e_return, volatility, error):
  return P_0 * math.exp((e_return - (volatility**2)/2)*delta_t + (volatility * (delta_t**0.5))*er
ror)

# Defining the frequency on a dailiy basis and a time horizon of 0.5 year, assuming there are 252
tradings day in a year
time_ex = np.array([t/252 for t in range(126)])

# Showing an example of the price on the day number 2000 of the stock number 10
stock_n = 100

# Generating 5000 standard normal errors for the stock number 10
errors = np.random.standard_normal(len(time_ex))
path = np.array([geometric_brownian_motion(starting_prices[stock_n], time_ex[n], expected_r
eturns[stock_n], volatilities[stock_n], errors[n]) for n in range(len(time_ex))])

print("Example of half year of the daily movement of a sampled stock price")
plt.plot(path)
plt.show()
```

Output:

```
Example of half year of the daily movement of a sampled stock price
```



Input:

```
import pandas as pd

# Defining the frequency on a dailiy basis and a time horizon of 10 years, assuming there are 252 trading day in a year
time = np.array([t/252 for t in range(252*10)])

# Data Frame initialization
virtual_scenario = pd.DataFrame()

for stock in stock_market:
   stock_n = stock_market.index(stock)
   errors = np.random.standard_normal(len(time))
   virtual_scenario[stock] = np.array([geometric_brownian_motion(starting_prices[stock_n], time[n], expected_returns[stock_n], volatilities[stock_n], errors[n]) for n in range(len(time))])

print("Generation of a market scenario for all 5000 stocks, with each stock having 2520 observations (10 years of daily returns), stored in a Data Frame.")
virtual_scenario
```

Output:

```
Generation of a market scenario for all 5000 stocks, with each stoc
k having 2520 observations (10 years of daily returns), stored in a
Data Frame.
```

	AHC	QLEN	YUAS	ERBT	KUKO	UDL	I
0	984.910000	239.450000	831.690000	169.390000	722.350000	606.480000	674.010
1	984.704858	239.468345	831.982085	169.708594	722.633886	607.032482	674.888
2	986.347680	239.634796	832.691262	170.100244	722.790251	608.051722	676.850
3	987.030921	239.611956	832.954032	169.397223	722.824237	607.024295	671.648
4	986.378491	239.572654	832.722657	169.955736	723.613692	608.606500	672.099
...	...	...	...	...	...	...	
2515	2427.984396	434.103433	1963.923184	338.122920	1682.668372	955.620643	1273.865
2516	2286.666375	442.479310	2039.274327	307.908286	1682.279109	965.217214	1197.027
2517	2285.818411	439.942238	2018.843283	328.007042	1691.493259	904.606421	1287.350
2518	2263.378407	436.766684	2010.588331	317.617832	1654.688150	944.211378	1240.287
2519	2328.287111	426.633093	2044.900066	319.958085	1684.119455	925.764002	1138.471

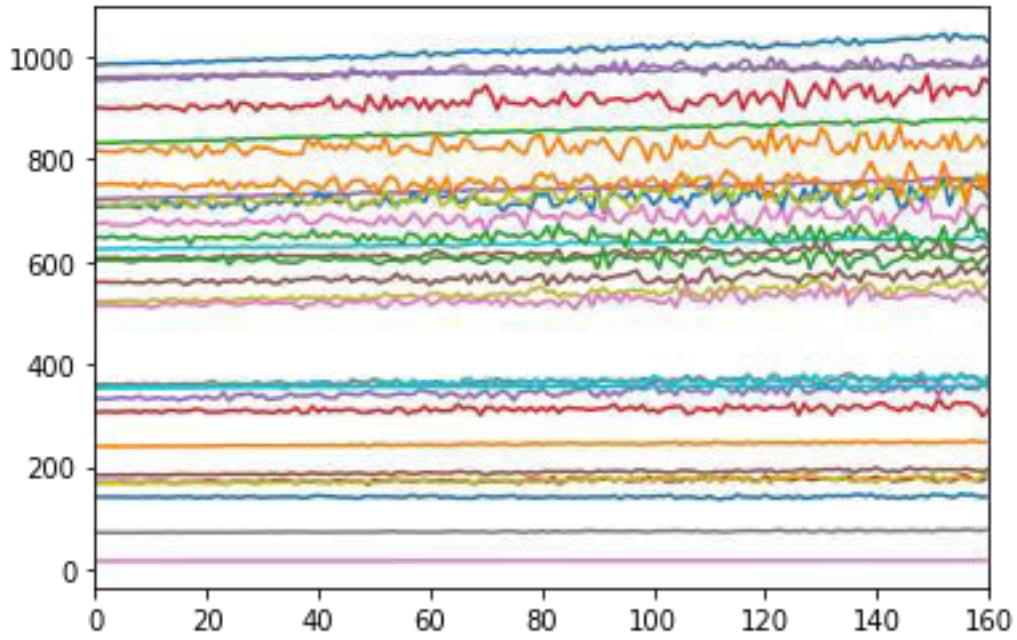
2520 rows × 5000 columns

Input:

```
virtual_scenario[stock_market[0:30]].iloc[0:161].plot(legend=False)
print("Plot of 30 sampled stock for the first 6 months")
```

Output:

```
Plot of 30 sampled stock for the first 6 months
```



It seems the perfect tool to handle financial data on a large scale in a small amount of time: not casually it was developed by Wes McKinney while working as analyst of a Hedge Fund to handle easily financial time series.

Once clarified the potentiality of NumPy and Pandas in performing high-speed numerical computation on large amount of observations or dimensions, is it possible to show a practical example of its applications with real financial data: computing the Sharpe Ratio for all the Companies included in the S&P 500 Index for the period 2017-01-01 until today.

The assumed risk-free ratio, given the recent Quantitative Easing policies, is equal to zero, and consequentially the average natural logarithm return, and the respective volatility are computed starting from the daily adjusted close price directly.

Jupiter HTML version of the code available also at FinanceCS.com (Stefano Ciccarelli):
https://www.financecs.com/wp-content/uploads/2020/09/Top-Performing-SP-500-Companies-1.html

Input:

```
import pandas as pd
import yfinance as yf
from yahoofinancials import YahooFinancials
import datetime
import numpy as np

# Taking from Wikipedia the list of the Companies inside the SP500 Index
SP_500_Data = pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')[0]
SP_500_Data.head()
```

Output:

	Symbol	Security	SEC filings	GICS Sector	GICS Sub Industry	Headquarters Location	Date first added	
0	MMM	3M Company	reports	Industrials	Industrial Conglomerates	St. Paul, Minnesota	1976-08-09	
1	ABT	Abbott Laboratories	reports	Health Care	Health Care Equipment	North Chicago, Illinois	1964-03-31	
2	ABBV	AbbVie Inc.	reports	Health Care	Pharmaceuticals	North Chicago, Illinois	2012-12-31	1
3	ABMD	ABIOMED	reports	Health	Health Care	Danvers,	2018-	

Input:

```
#Initializing the dataframe
SP500_adj_Close = pd.DataFrame()

# For each stock ticker we download the stock close price daily data starting from 1st Janua
ry 2017 until today
for symbol in  SP_500_Data["Symbol"]:

   SP500_adj_Close[symbol] = yf.download(symbol,
                start='2017-01-01',
                end=datetime.datetime.today(),
                progress=False)["Adj Close"]

#Initializing the dataframe
SP500_Returns = pd.DataFrame()

# Generating the Daily Returns
SP500_Returns = np.log(SP500_adj_Close / SP500_adj_Close.shift(1)).iloc[1:]


SP500_Returns.head()
```

Output:

Date	MMM	ABT	ABBV	ABMD	ACN	ATVI	ADBE	AMI
2017-01-04	0.001515	0.007907	0.014002	0.029638	0.002401	0.019460	0.006358	0.00000
2017-01-05	-0.003427	0.008601	0.007556	-0.008068	-0.015104	0.015406	0.016854	0.01676
2017-01-06	0.002922	0.026841	0.000314	0.005299	0.011328	-0.000791	0.022315	0.00709
2017-01-09	-0.005401	-0.000981	0.006562	0.014536	-0.011241	-0.005555	0.002490	0.01490

Input:

```
df = pd.DataFrame()

df["Avg_Returns"] = SP500_Returns.mean()
df["Volatilities"] = SP500_Returns.std()
df["Sharpe_Ratios"] = df["Avg_Returns"]/df["Volatilities"]*(252**0.5)
df["Sharpe_Ratios"].nlargest(10)
```

Output:

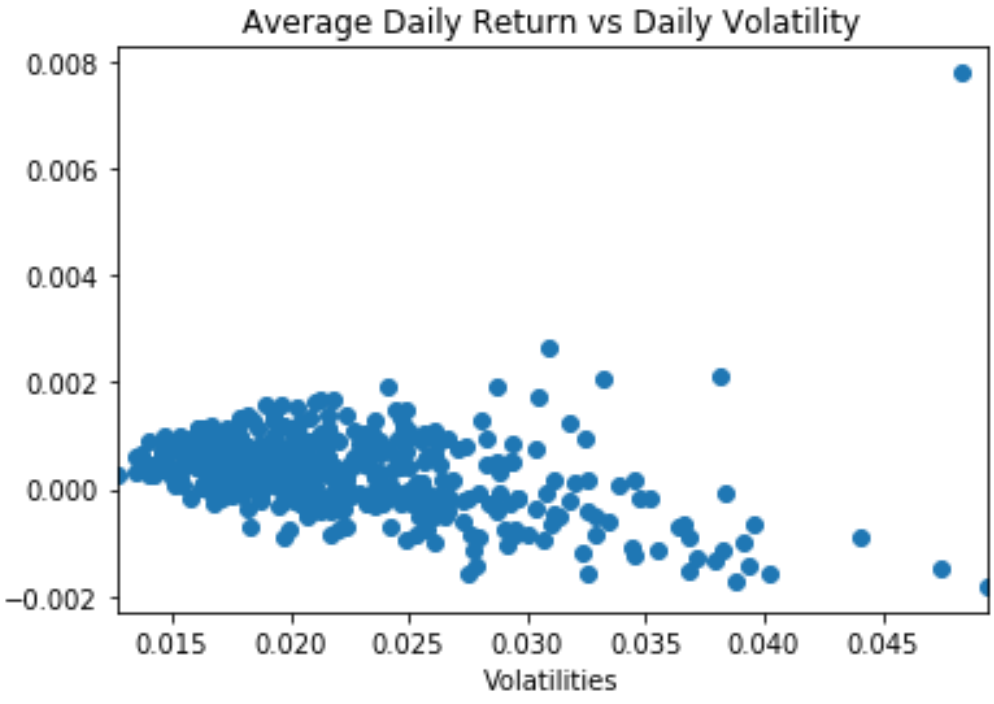
```
CARR    2.559158
OTIS    1.366332
AMZN    1.338054
AAPL    1.295516
NOW     1.270008
ADBE    1.258111
MSCI    1.247168
PYPL    1.228539
MSFT    1.217933
CDNS    1.211523
Name: Sharpe_Ratios, dtype: float64
```

Input:

```
df.plot(x='Volatilities', y='Avg_Returns', style='o', title='Average Daily Return vs Daily Volatility', legend = False)
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1beb14e6788>
```



In the graph above is it possible to summarize an analysis conducted on large scale (the S&P 500 Index companies) and have a plot of the daily volatility (x axis) against the average daily return (y axis) and the relative list of the top 10 performing companies in terms of Sharpe Ratio[26] (average excess return over volatility).

This allow us to identify investment opportunities that maximize the return for each level of volatility included in our portfolio.

[26] William F. Sharpe, The Journal of Portfolio Management Fall 1994, 21 (1) 49-58.
DOI: https://doi.org/10.3905/jpm.1994.409501

Applied Financial Machine Learning with Scikit-learn

The large spread of Scikit-learn[27] as a main reference for Machine Learning is not a coincidence: it doesn't only includes a series of algorithms already structured and optimized, but also a range of techniques dedicate to processing the data (before and after). Generally, is worth to remember that in applying Machine Learning to financial analysis we must not only use advanced quantitative tools but also a deep qualitative acumen to have a broad picture of why the model is generating specific results.

Following we will show the application of a k-means clustering algorithm:

I) In the first step we need to mix the Sharpe Ratios of the S&P500 Index Companies together with the P/E Ratios, that we will obtain scraping Yahoo Finance and the relative fundamental financial values inside each company's page.

II) Once stored the data in a Dataframe, the outliers inside P/E are removed (and indirectly the stocks associated) and the remaining are plotted on the x axis against the y axis (Sharpe Ratio).

III) On this level we are ready to apply a K-mean algorithm to identify the group of stocks (4 in our case) that are discovered starting from the data and the relative 2 dimensions provided.

IV) Finally, we analyze the results and the characteristics of each single group to try to identify starting backwards the meaning of each group and how this could be valuable to find new potential investment opportunities.

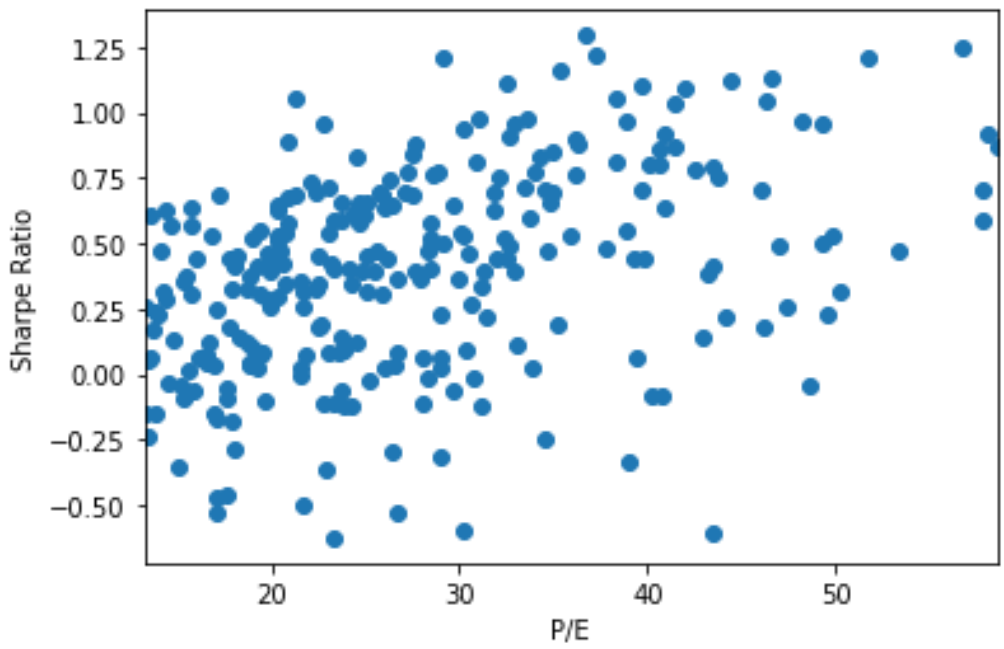
[27] https://scikit-learn.org/stable/index.html

Jupiter HTML version of the code available also at FinanceCS.com (Stefano Ciccarelli):
https://www.financecs.com/wp-content/uploads/2020/09/PE-vs-Sharpe-Ratio-1.html

Input:

```
# Stefano Ciccarelli
import pandas as pd
import yfinance as yf
from yahoofinancials import YahooFinancials
import datetime
import numpy as np
from bs4 import BeautifulSoup as bs
import requests
from scipy import stats

# Reading the previously computed Sharpe Ratios
df = pd.read_csv("Sharpe_Ratios.csv", index_col = 0)

# Taking from Wikipedia the list of the Companies inside the SP500 Index
SP_500_Data = pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')[0]

import yahoo_fin.stock_info as si

# Obtainf fundamental data: P/E Ratio

P_E_SP500 = dict()

for stock in SP_500_Data["Symbol"]:
  try:
    P_E_SP500[stock] = si.get_quote_table(stock)['PE Ratio (TTM)']
  except:
    print(stock, " not found!")

df["P/E"] = pd.Series(P_E_SP500)
P_E_SP500
# Removing outliers from the P/E Ratios
df["P/E"] = df["P/E"][df["P/E"].between(df["P/E"].quantile(.15), df["P/E"].quantile(.85))
]

# Cleaning the datset from missing values
df = df.dropna()
df.plot(x='P/E', y='Sharpe_Ratios', style='o') .set_ylabel("Sharpe Ratio")
```

Output:



Input:

```
df.corr()
```

Output:

	Avg_Returns	Volatilities	Sharpe_Ratios	P/E
Avg_Returns	1.000000	-0.258494	0.972602	0.377830
Volatilities	-0.258494	1.000000	-0.388080	-0.089847
Sharpe_Ratios	0.972602	-0.388080	1.000000	0.412262
P/E	0.377830	-0.089847	0.412262	1.000000

In the graph above we can observe the raw data, where for each stock (after the outliers' removal, with 270 observations remaining) the Sharpe Ratio is plotted against its relative P/E.

The normalization outliers' removal and the absence of multicollinearity (low correlation between Sharpe Ratio and P/E) indicates that the data is ready for a K-mean clustering model.

The main scope of the analysis, once removed the irrelevant stocks, as in the case of a Sharpe Ratio inferior to 0.7 (subjective factor dependent on the preference of the Investor), will be to identify the top 10 performing companies inside the "buy" group, defined in terms of Sharpe Ratio and P/E.

Jupiter HTML version of the code available also at FinanceCS.com (Stefano Ciccarelli):
https://www.financecs.com/wp-content/uploads/2020/09/Clustering-for-Stock-Selection-1.html

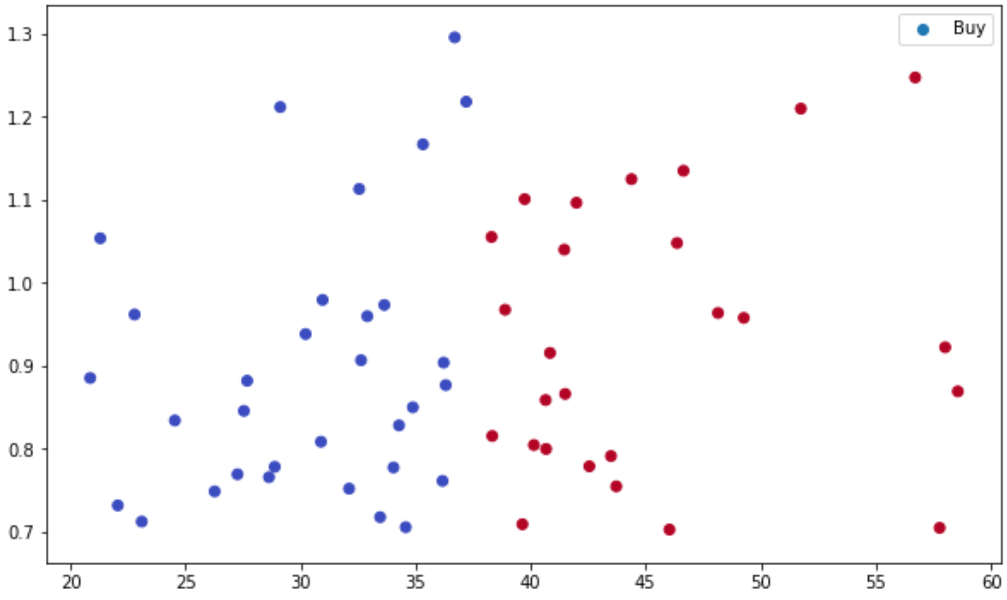
Input:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.neighbors import NearestNeighbors
from pylab import mpl, plt

df = pd.read_csv("S&P500 Index - Shape and P_E.csv")

# We want to invest only in stocks with a Sharpe Ratio superior to 0.7
# All the stock not satisfying this criterion are removed

df = df[df["Sharpe_Ratios"] > 0.70]
```

```
# Initialising the K-Mean Algorithm defining 2 clusters
algo = KMeans(n_clusters=2)

y = algo.fit_predict(df[["Sharpe_Ratios","P/E"]])
#algo.fit(X, columns=["Sharpe_Ratios", "P/E"])

plt.figure(figsize=(10,6))
plt.scatter(df["P/E"], df["Sharpe_Ratios"], c=y, cmap="coolwarm")
plt.legend(['Buy'])
```

Output:

```
<matplotlib.legend.Legend at 0x152b5cb3908>
```



Input:

```
# The data is Labeled according its class (1 Buy 0 Not Buy)
df['Buy'] = 1 - y

# Filtring only by the stock the algorithm suggests to buy
df = df[df["Buy"] == 1]

df.sort_values(by=["Sharpe_Ratios"], ascending=False).head(10)
```

Output:

	Unnamed: 0	Avg_Returns	Volatilities	Sharpe_Ratios	P/E	Buy
29	AAPL	0.001599	0.019599	1.295516	36.70	1
185	MSFT	0.001397	0.018213	1.217933	37.20	1
51	CDNS	0.001546	0.020255	1.211523	29.12	1
80	CPRT	0.001419	0.019304	1.166650	35.32	1
228	SPGI	0.001293	0.018449	1.112825	32.55	1
91	DG	0.001090	0.016421	1.053425	21.30	1
188	MCO	0.001219	0.019764	0.979201	30.96	1
255	VRTX	0.001372	0.022386	0.973098	33.64	1
260	WMT	0.000886	0.014629	0.961559	22.79	1
236	STE	0.000955	0.015802	0.959521	32.90	1

The table above represent the top 10 Stocks classified as “Buy” by the clustering algorithm: once all the stocks are classified binomially, the dataset is filtered by the stock the Investor is interested to buy (discriminated by a combination of Sharpe Ratio and P/E Ratio) and all the stocks are ranked according the highest Sharpe Ratio.

This algorithm can be useful to discriminate which company, outside the S&P 500 Index, is behaving, in terms of P/E and Sharpe Ratios, as the top 10 performing one and consequentially to select which stock should be inserted in the final portfolio.

It is worth to notice that the period is including the COVID-19 effect, consequentially the Sharpe Ratios are affected by these “outliers” and this approach should be repeated in a relatively more stable future period.

Optimized Neural Networks with AutoKeras: Analyzing the COVID-19 Impacts on NASDAQ US Benchmark Airlines Index using Google Trends

Airlines is one of the main industries to be affected by the COVID-19: in the long-run it is not only because of the effective diffusion of the virus, but also to the social perception of it.

Stephens-Davidowitz [28], used Google Trends as a Data-Driven approach to analyze the online behavior of the masses to try to predict the future (specifically the US Elections): an approach that I replicated on the European Elections in occasion of the meeting with him.

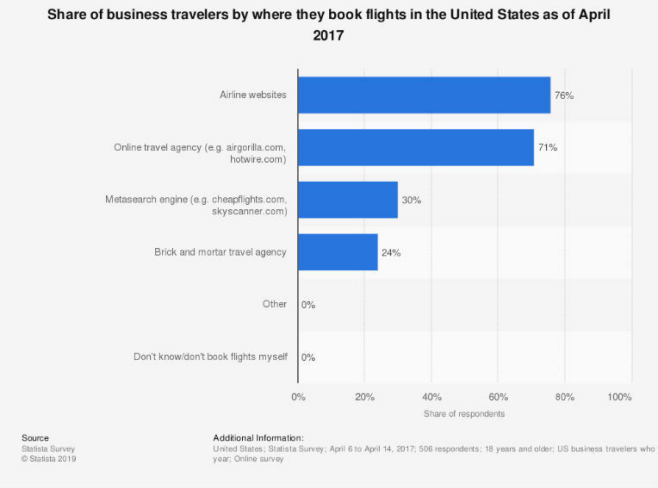
It mainly consists in finding patterns and behaviors related to an outcome of interest: it can be improved substituting the human judgment with Neural Networks to account for different interactions among data on a deeper level than a simple linear or logistic regression.

For this analysis, the key mission will be to analyze and quantify the impact of the Coronavirus "worries", the number of Google Searches for "Flights" and the 4 main US Airlines Companies on the Airlines Industry's Financial performances.

First, we will start obtaining the NASDAQ US Benchmark Airlines Index values using quandl, consequentially the number of searches for the flights and the airlines will be scraped from Google Trends together with the number of Coronavirus related searches.
In the case of the Coronavirus searches, will be applied a transformation, where over a threshold the value will be equal to 1, implying a minimum of amount of worrying among the population is still persistent, and in all other cases will be equal to 0.
The 2 main pillars are:

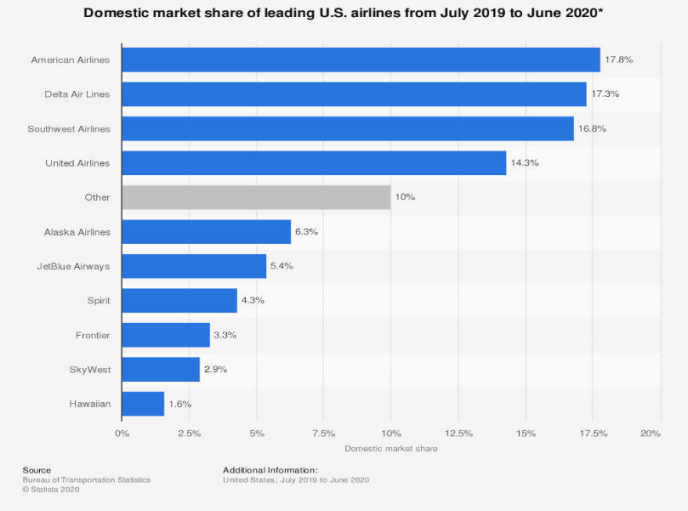
[28] Stephens-Davidowitz, S. (2017). Everybody lies: Big data, new data, and what the Internet can tell us about who we really are. New York, NY: HarperCollins, 352 pp
https://onlinelibrary.wiley.com/doi/abs/10.1111/jmft.12325

I) Most of the Airlines Tickets Bookings are performed online.



https://www.statista.com/statistics/291037/online-sources-leisure-business-travel-planning-us/

II) The search airline index will be composed by the 4 most important US Airlines by market share



https://www.statista.com/statistics/250577/domestic-market-share-of-leading-us-airlines/

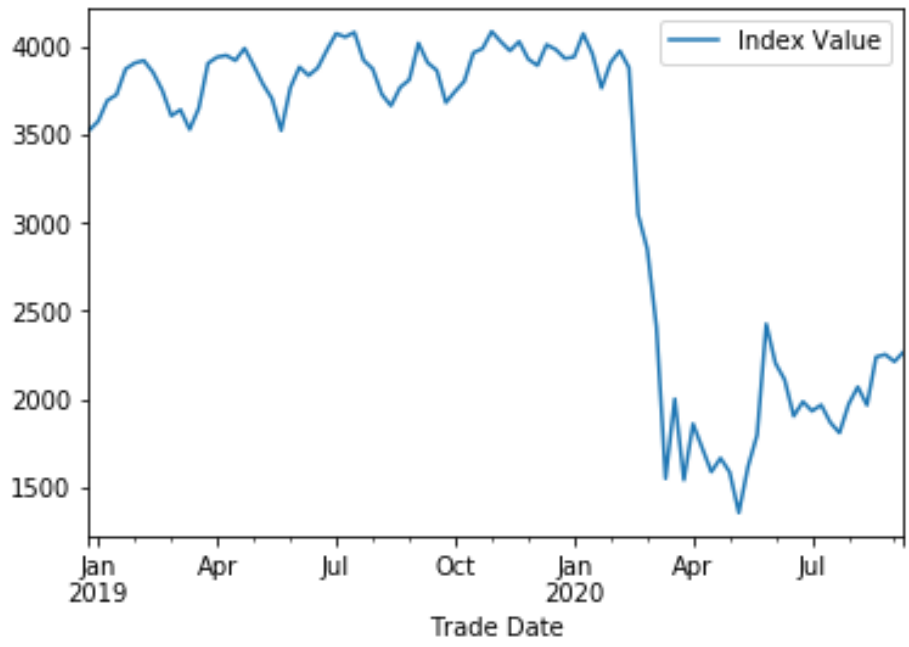
Jupiter HTML version of the code available also at FinanceCS.com (Stefano Ciccarelli):
https://www.financecs.com/wp-content/uploads/2020/09/Untitled11-Copy3-1.html

Input:

```
import autokeras
import quandl
import pandas as pd

quandl.ApiConfig.api_key = "TyKZthWgcxsqfZf9yeXD"
data_air = quandl.get("NASDAQOMX/NQUSB5751-NQ-US-Bnchmk-Airlines-Index-NQUSB5751", start_date="2019-01-01", end_date="2020-09-20")
data_air.index = pd.to_datetime(data_air.index)
data_air = data_air["2019-01-01":]

logic = {'Index Value'  : 'last'}
offset = pd.offsets.timedelta(days=-6)

data_air = data_air.resample('W-SAT', loffset=offset).apply(logic)
data_air.plot()
```

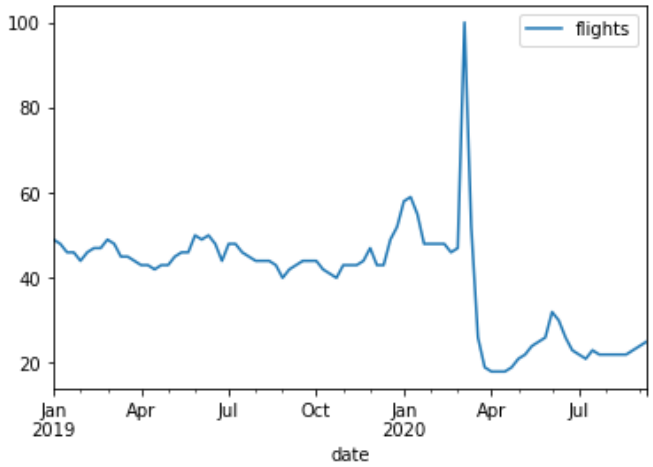
Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x207c42a9288>
```



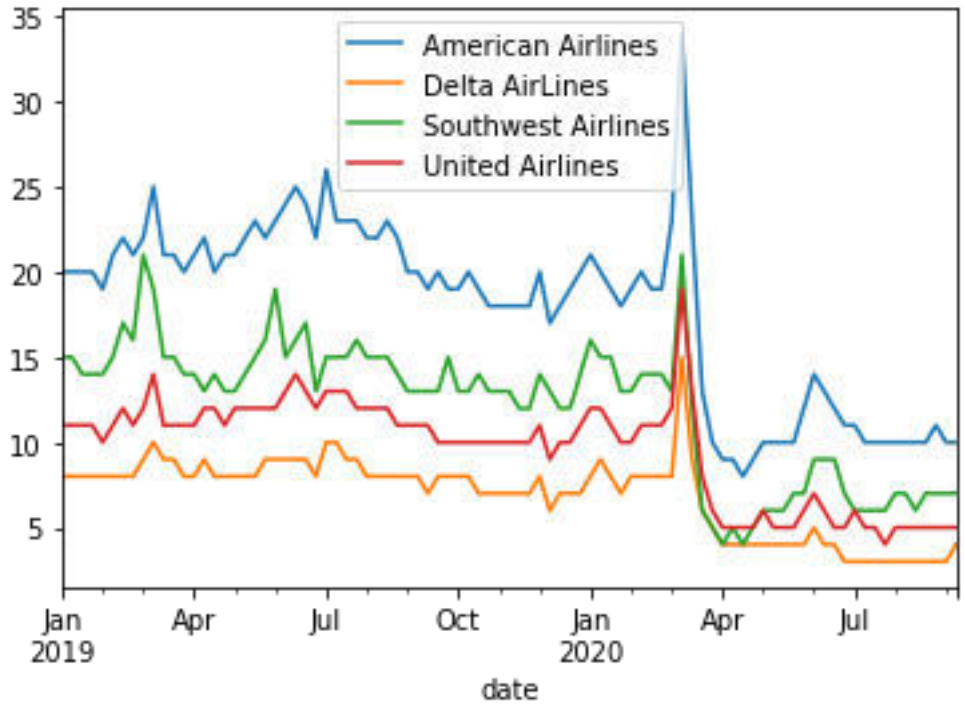
The code above obtains the data of the Index from the beginning of 2019 until the 20th of September 2020 daily, consequentially it is resampled on a weekly basis setting as price the last price of the week available.

```
Input:
from pytrends.request import TrendReq
pytrend = TrendReq(hl='en-US', tz=360)
keywords = ['flights']
pytrend.build_payload(
    kw_list=keywords,
    cat=0,
    timeframe='today 5-y',
    geo='US',
    gprop='')
searches = pytrend.interest_over_time()
searches = searches.drop(labels=['isPartial'],axis='columns')
#searches.index = pd.to_datetime(searches.index)
searches.index = pd.to_datetime(searches.index)

searches = searches["2019-01-01":]
#searches.index = searches.index.week
searches.plot()

Output:
<matplotlib.axes._subplots.AxesSubplot at 0x207c5554088>
```



Instead, above we can observe the behavior from the 1st of January 2019 until the 20th of September 2020 of the Google Searches for the keyword “Flights” in US.

```
Input:
pytrend2 = TrendReq(hl='en-US', tz=360)
keywords2 = ['American Airlines','Delta AirLines','Southwest Airlines','United Airlines']
pytrend2.build_payload(
    kw_list=keywords2,
    cat=0,
    timeframe='today 5-y',
    geo='US',
    gprop='')
searches2 = pytrend2.interest_over_time()
searches2 = searches2.drop(labels=['isPartial'],axis='columns')
#searches.index = pd.to_datetime(searches.index)
searches2.index = pd.to_datetime(searches2.index)

searches2 = searches2["2019-01-01":]
#searches.index = searches.index.week
searches2.plot()
Output:
<matplotlib.axes._subplots.AxesSubplot at 0x207c55a8048>
```



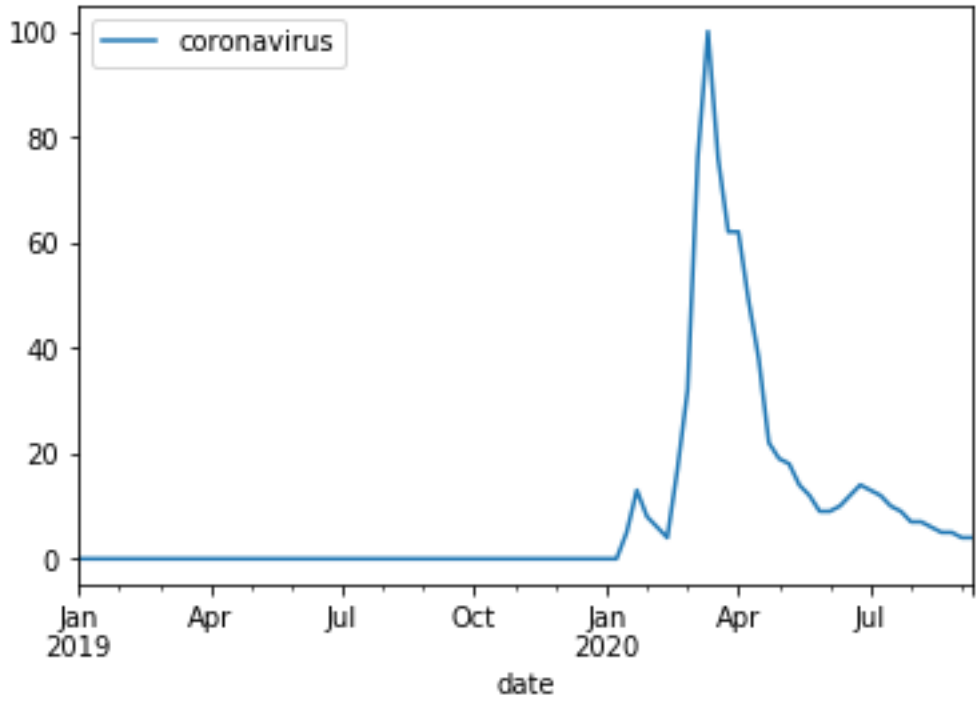
The same approach is repeated on the 4 main US Airlines Companies.

Input:

```
pytrend3 = TrendReq(hl='en-US', tz=360)
keywords3 = ['coronavirus']
pytrend3.build_payload(
    kw_list=keywords3,
    cat=0,
    timeframe='today 5-y',
    geo='US',
    gprop='')
searches3 = pytrend3.interest_over_time()
searches3 = searches3.drop(labels=['isPartial'],axis='columns')
#searches.index = pd.to_datetime(searches.index)
searches3.index = pd.to_datetime(searches3.index)

searches3 = searches3["2019-01-01":]
#searches.index = searches.index.week
searches3.plot()
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x207c54e6588>
```



Finally, the COVID-19 “worriedness” in US is sampled and plotted above, seeming to have a negatively correlated behavior respect to the Index and the Airlines searches.

Input:

```
dataframe = searches2.copy()
dataframe['Index'] = data_air['Index Value']
dataframe['flights'] = searches['flights']
dataframe['coronavirus'] = searches3['coronavirus']
#dataframe.index.week
dataframe = dataframe.dropna()
dataframe.head()
dataframe.tail()
```

Output:

date	American Airlines	Delta AirLines	Southwest Airlines	United Airlines	Index	flights	coronavirus
2019-01-06	20	8	15	11	3574.66	49	0
2019-01-13	20	8	15	11	3692.80	48	0
2019-01-20	20	8	14	11	3725.15	46	0
2019-01-27	20	8	14	11	3871.17	46	0
2019-02-03	19	8	14	10	3905.14	44	0

Output:

date	American Airlines	Delta AirLines	Southwest Airlines	United Airlines	Index	flights	coronavirus
2020-08-16	10	3	6	5	1967.17	22	6
2020-08-23	10	3	7	5	2239.90	22	5
2020-08-30	11	3	7	5	2254.08	23	5
2020-09-06	10	3	7	5	2213.60	24	4
2020-09-13	10	4	7	5	2268.02	25	4

Once collected the data from different data sources, the time series are collected together in a unique dataset according the Index (a datetime object).

In the first table we can observe the data at the beginning of 2019 and in the second one the most recent observations.

There is a clear relationship between the decrease of number of searches, the drop of the Index and the increase of the COVID-19 worriedness.

Input:

```
dataframe.corr()
```

Output:

	American Airlines	Delta AirLines	Southwest Airlines	United Airlines	Index	flights	coronavirus
American Airlines	1.000000	0.964791	0.940771	0.985492	0.776127	0.921146	-0.216994
Delta AirLines	0.964791	1.000000	0.913538	0.973014	0.739754	0.924918	-0.114226
Southwest Airlines	0.940771	0.913538	1.000000	0.927562	0.819368	0.900298	-0.372909
United Airlines	0.985492	0.973014	0.927562	1.000000	0.773032	0.927118	-0.165175
Index	0.776127	0.739754	0.819368	0.773032	1.000000	0.710131	-0.621369
flights	0.921146	0.924918	0.900298	0.927118	0.710131	1.000000	-0.134019
coronavirus	-0.216994	-0.114226	-0.372909	-0.165175	-0.621369	-0.134019	1.000000

We can observe the correlation of the variable, and conclude there is the presence of Multicollinearity, an indicator that those variables are suitable as an input of our model.

In the case of the Index, all the correlations for google searches related to the flights are superior to >0.7 and only in the case of coronavirus we find a -0.62, strongly negatively correlated to the performance of the index, implying an increase of people worrying for

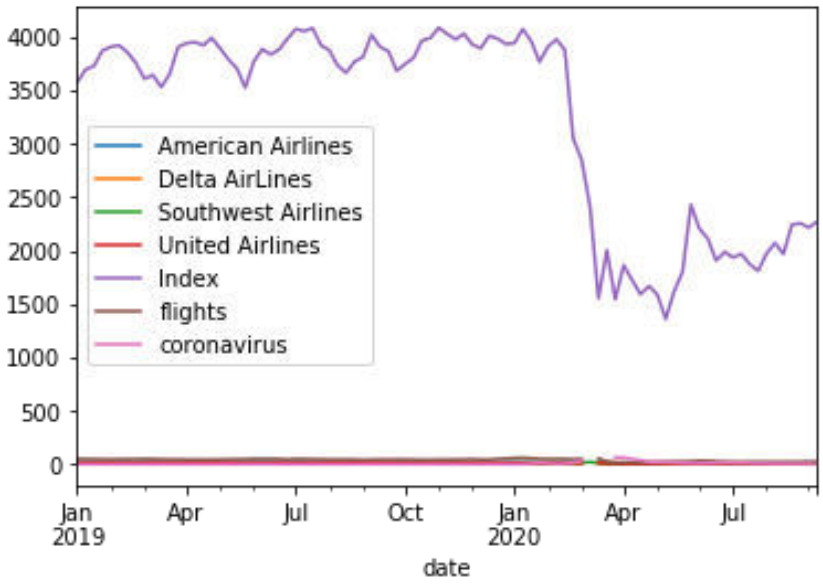
COVID-19 could potentially, but not necessarily since correlation doesn't mean causation in every case, determine a decrease of the Airline financial performance.

Input:

```
import numpy as np
#searches_clean = searches
dataframe_clean = dataframe[np.abs(dataframe-dataframe.mean())<=(3*dataframe.std())]
#dataframe_clean['flights'] = dataframe_clean['flights'].rolling(window=3).mean()
dataframe_clean.plot()
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x207c57f7188>
```



We clean the dataset removing all the outliers (all the observations that are distant from the average at least 3 times the standard deviation).

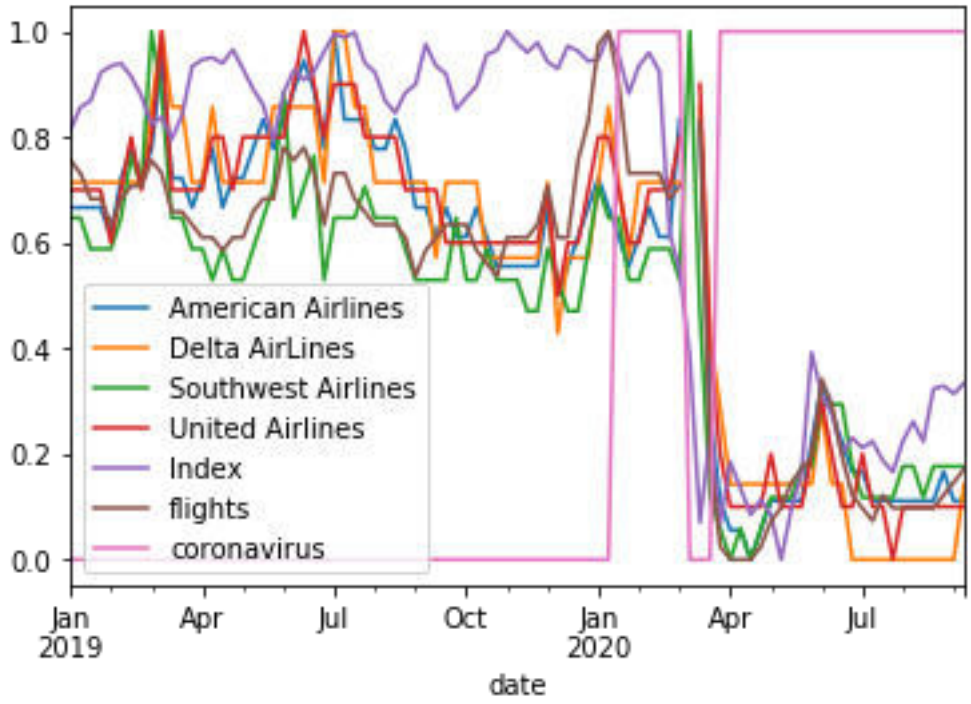
But the plot is still not enough informative, implying is necessary to standardize the data and work on the same scale to extract the meaning from the variables.

Input:

```
from sklearn.preprocessing import MinMaxScaler
def condition(x):
  if x > 0.03:
    return 1
  else:
    return 0

scaler = MinMaxScaler()
data_standard = dataframe_clean.copy()
#data_standard[["flights","Index"]] = scaler.fit_transform(data_standard[["flights", "Index"]])

data_standard[["flights","Index","coronavirus","American Airlines","Delta AirLines", "Southwest Airlines", "United Airlines"]] = scaler.fit_transform(data_standard[["flights", "Index","coronavirus","American Airlines","Delta AirLines", "Southwest Airlines", "United Airlines"]])
data_standard["coronavirus"] = data_standard["coronavirus"].apply(condition)
data_standard.plot()
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15c6bf10608>
```



Now the graph clearly shows the Index moving on the same direction of the number of searches for the flights and the airline companies.

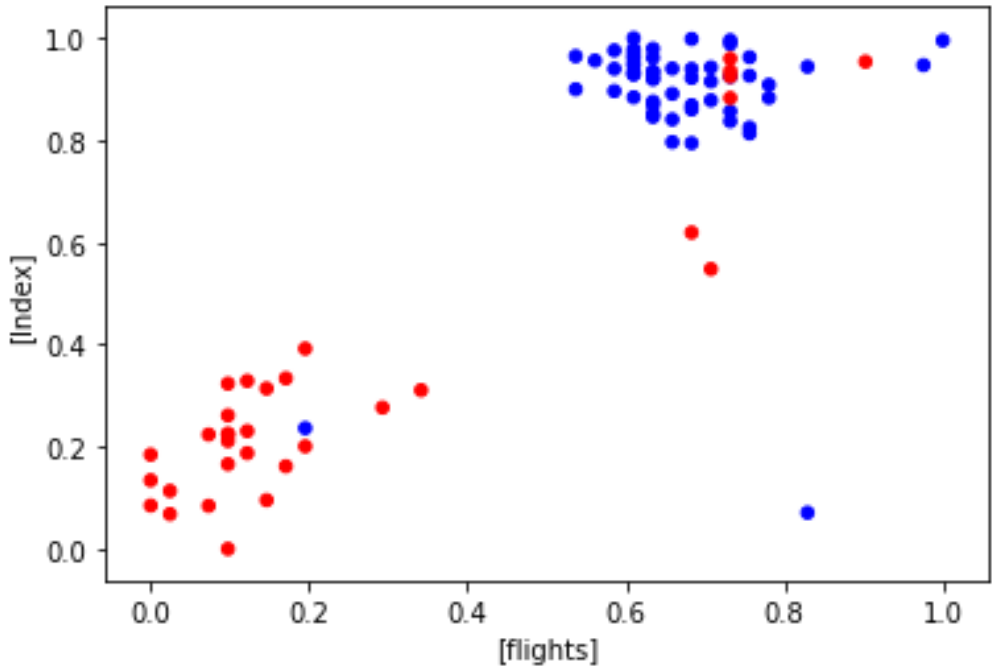
The COVID-19 is represented as a Boolean, as 1 while a minimum threshold of searches is present: when it is activated the index and the flights searches are pushed down, implying that when the people will stop worrying about COVID-19 the market could potentially, but not necessarily, generate a specular reaction.

Input:

```
col = data_standard.coronavirus.map({0:'b', 1:'r'})
data_standard.plot.scatter(["flights"],["Index"], c = col)
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15c6bfed948>
```



A scatter plot above, where the flights searches are on the x-axis and the index is on the y-axis, show how there are 2 evident clusters, where one is characterized by low searches

and low index values and the other one by high searches and high index values, implying those 2 variables are strictly related.

The red color represents the COVID-19 activation.

Input:

```
def labeling(x):
  if x > 3000:
    return 1
  else:
    return 0


data_standard2 = data_standard.copy()
data_standard2["Index"] = dataframe_clean["Index"].apply(labeling)
data_standard2.tail()
```

Output:

date	American Airlines	Delta AirLines	Southwest Airlines	United Airlines	Index	flights	coronavirus
2020-08-16	0.111111	0.000000	0.117647	0.1	0	0.097561	1
2020-08-23	0.111111	0.000000	0.176471	0.1	0	0.097561	1
2020-08-30	0.166667	0.000000	0.176471	0.1	0	0.121951	1
2020-09-06	0.111111	0.000000	0.176471	0.1	0	0.146341	1
2020-09-13	0.111111	0.142857	0.176471	0.1	0	0.170732	1

In the code above, the Index is labeled, to prepare it for a classification algorithm: 0 when the Index is less or equal to 3000 and 1 in all other cases.
The main purpose is to identify, using the parameters as discriminators, when will be likely that the Index will turn back to the pre-virus levels (1).

```
Input:

X, y = data_standard2[["flights","coronavirus","American Airlines","Delta AirLines", "Sout
hwest Airlines", "United Airlines"]], data_standard2['Index']

from sklearn.model_selection import train_test_split
X_train, X_test = X[:'2020-08-01'], X['2020-08-01':]
y_train, y_test = y[:'2020-08-01'], y['2020-08-01':]

import autokeras as ak
search = ak.StructuredDataClassifier(max_trials=15)
search.fit(x=X_train, y=y_train, verbose=1)

Output:

INFO:tensorflow:Reloading Oracle from existing project .\struct
ured_data_classifier\oracle.json
INFO:tensorflow:Reloading Tuner from .\structured_data_classifi
er\tuner0.json
INFO:tensorflow:Oracle triggered exit
```

Above the model is trained with AutoKeras, that generates a TensorFlow classification algorithm already optimized after a series of epochs.

```
Input:
loss, acc = search.evaluate(X_test, y_test, verbose=0)
print(acc)

Output:
1.0
```

The accuracy obtained is 100%, implying the model maximized its classification capabilities over the Index, assuming absence of overfitting.

Since the data was previously split (training set and test set for both X and y), to test the model on unobserved data, we can test it on the most recent market changes: the model was trained until the 1st of August and doesn't have information of what happens after.

```
Input:
y_predictions = search.predict(X_test)
print(y_predictions)

Ouput:
[[0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]]
```

Our model estimates that for all the period of August, until the second week of September 2020, the Index should remain under the threshold of 3000, since the market is still affected by negative conditions.

```
Input:
print(y_test)

Output:
date
2020-08-02    0
2020-08-09    0
2020-08-16    0
2020-08-23    0
2020-08-30    0
2020-09-06    0
2020-09-13    0
Name: Index, dtype: int64
```

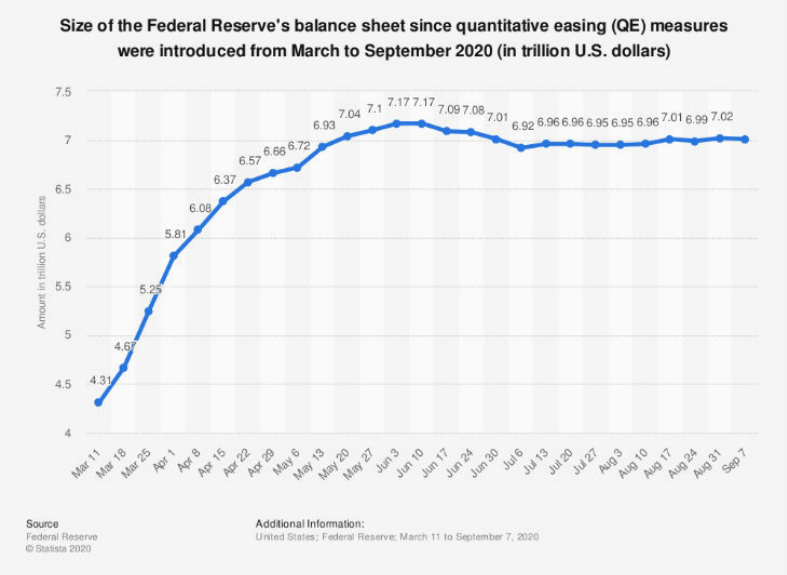
This is exactly what happened, implying the model is correctly working on the short-term. The main purpose of this model it is to monitor the market conditions and the relative variables that could affect the financial performances of the index: if an unexpected event happens, the model could estimate that the index should recover to the pre-virus conditions before that it is understood by the investors, reducing the market informative asymmetries.

```
Input:
model = search.export_model()
model.summary()
Output:
Model: "functional_1"
_________________________________________________________________
__
Layer (type)                 Output Shape              Param #
=================================================================
==
input_1 (InputLayer)         [(None, 6)]               0
_________________________________________________________________
__
multi_category_encoding (Mul (None, 6)                 0
_________________________________________________________________
__
dense (Dense)                (None, 32)                224
_________________________________________________________________
__
re_lu (ReLU)                 (None, 32)                0
_________________________________________________________________
__
dense_1 (Dense)              (None, 1)                 33
_________________________________________________________________
__
classification_head_1 (Activ (None, 1)                 0
=================================================================
==
Total params: 257
Trainable params: 257
Non-trainable params: 0
```

In finance the ability to account for different interactions of relevant variables is fundamental to estimate and understand the impacts on the overall financial markets, and especially on the market expected volatility (measured by the VIX Index).

In fact, we can observe how the financial actors are frequently "influenced" by nudging policies[29] with the main target to drive the consensus decisions.

A clear example happened recently where the COVID-19 negatively affected the financial markets (especially the S&P 500 Index), pushing the FED towards a new Quantitative Easing policy and consequentially lowering the interest rates of the US Treasury Bonds.



https://www.federalreserve.gov/monetarypolicy/bst_recenttrends.htm

[29] Barton, A., Grüne-Yanoff, T. From Libertarian Paternalism to Nudging—and Beyond. Rev.Phil.Psych. 6, 341–359 (2015).
https://doi.org/10.1007/s13164-015-0268-x

According Tan, Ji and Kohli, Vaibhav[30] (2011), in fact, a Quantitative Easing policy tends to introduce a higher volatility inside the financial markets in the Short-Term, with a following volatility stabilization in the mid-term.

Consequentially we want to analyze if the negative correlation is still holding in the Market Shift of 2020: is still true that lowering the interest rates of the Short Term 3-M US Treasury Bonds the Investors will be pushed to look for higher yield in the stock markets?

Finally, a Deep Neural Network Classification model is built to predict, given the last 3 weeks daily interest rates of the 3-M US Treasury Bond as parameters, if the expected volatility (VIX Index) is high (superior or equal to 20, labeled as 1) or normal (all other cases – labeled as 0).

The model, will have dimensionality equal to 21 for the X and a Boolean y and will be trained from the 1st of August 2019 to the to the 1st of August 2020 and consequentially tested on the period from the 2nd of August 2020 to 20th of September 2020.

[30] Tan, Ji and Kohli, Vaibhav, The Effect of Fed's Quantitative Easing on Stock Volatility (June 1, 2011). http://dx.doi.org/10.2139/ssrn.2215423

Jupiter HTML version of the code available also at FinanceCS.com (Stefano Ciccarelli):
https://www.financecs.com/wp-content/uploads/2020/09/Deep-Neural-Network-to-Predict-VIX-using-3-M-Treasury-Bonds.html

Input:

```
import quandl
import pandas as pd

quandl.ApiConfig.api_key = "INSERT YOUR API"
vix = quandl.get("CHRIS/CBOE_VX1-S-P-500-Volatility-Index-VIX-Futures-Continuous-Contract-1-VX1-Front-Month", start_date="2019-08-01", end_date="2020-09-20")['Close']
interest_rate_3m = quandl.get("FRED/DTB3-3-Month-Treasury-Bill-Secondary-Market-Rate", start_date="2019-08-01", end_date="2020-09-20")

dataframe = pd.DataFrame()
dataframe['Interest_Rate_3m'] = interest_rate_3m['Value']
dataframe['Vix'] = vix
```

Output:

	Interest_Rate_3m	Vix
Interest_Rate_3m	1.000000	-0.745343
Vix	-0.745343	1.000000

First, we obtain the relevant data using quandl, consequentially as we can observe: the data confirm the negative correlation between the VIX Index and the Interest Rates, a good starting point to build our model.

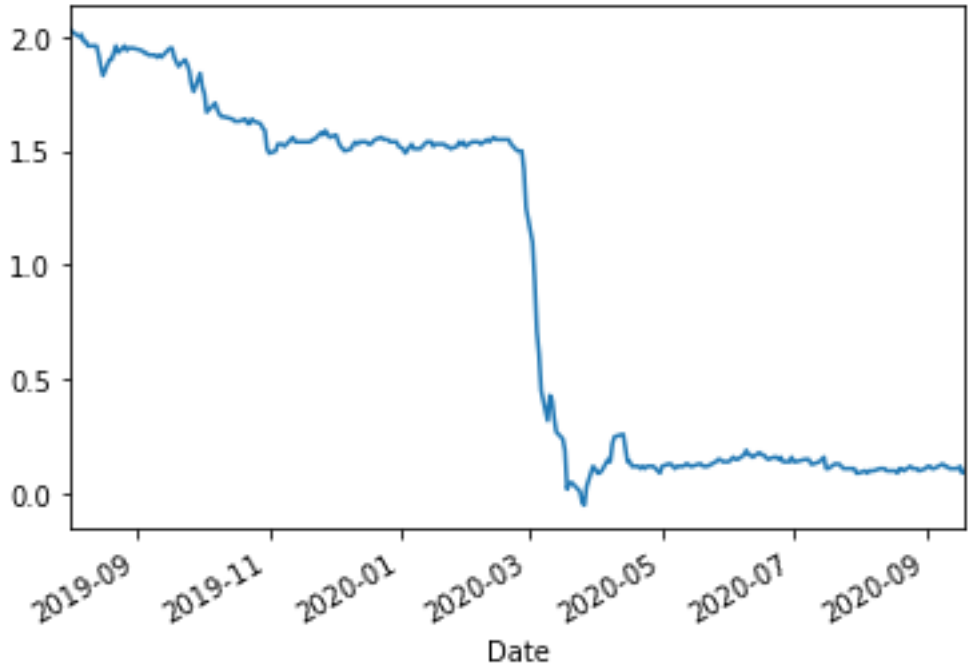
The data found, confirms the original assumption regarding the relationship between the Quantitative Easing policies and the short-term market volatility.

Input:

```
dataframe['Interest_Rate_3m'].plot()
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6550c9a48>
```



In the data above the behavior of the 3-M short term US Treasury bond interest rate is plotted to have a clear picture on how is evolving among time.

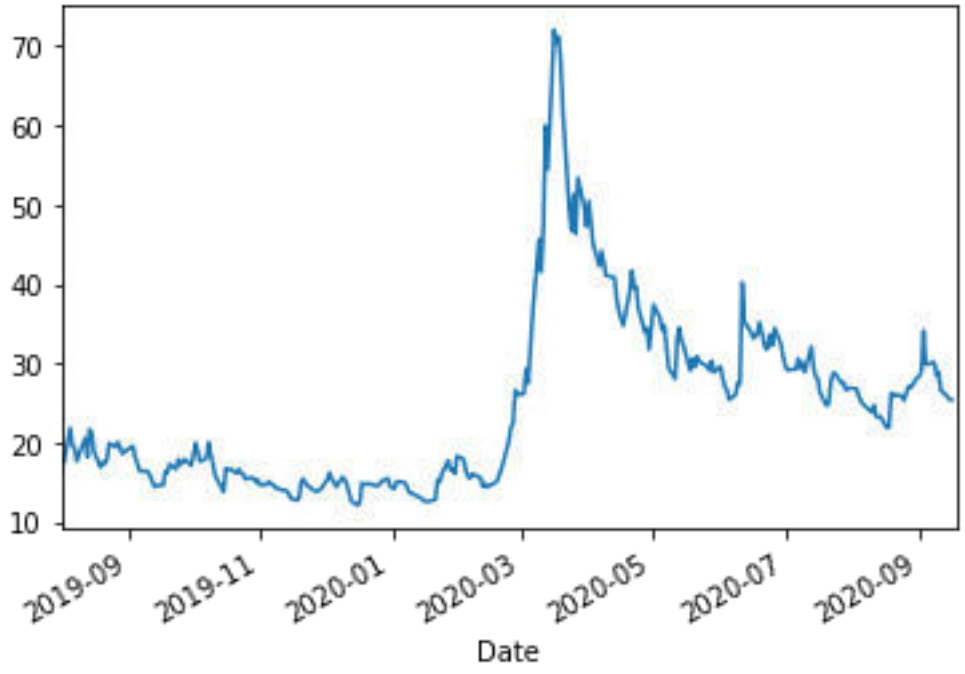
Is interesting to notice the significant drop following the FED announce regarding the instauration of the QE regime.

Input:

```
dataframe['Vix'].plot()
```

output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f65527dc48>
```



Instead, above we can observe the evolution of the VIX Index among time, and especially we can identify the significant rise following the FED new policy announcement, signaling an increase of the dynamism for the S&P 500 Index.

As expected from the theory: following the large short-term peak there is a clear trend towards a stabilization on a lower level.

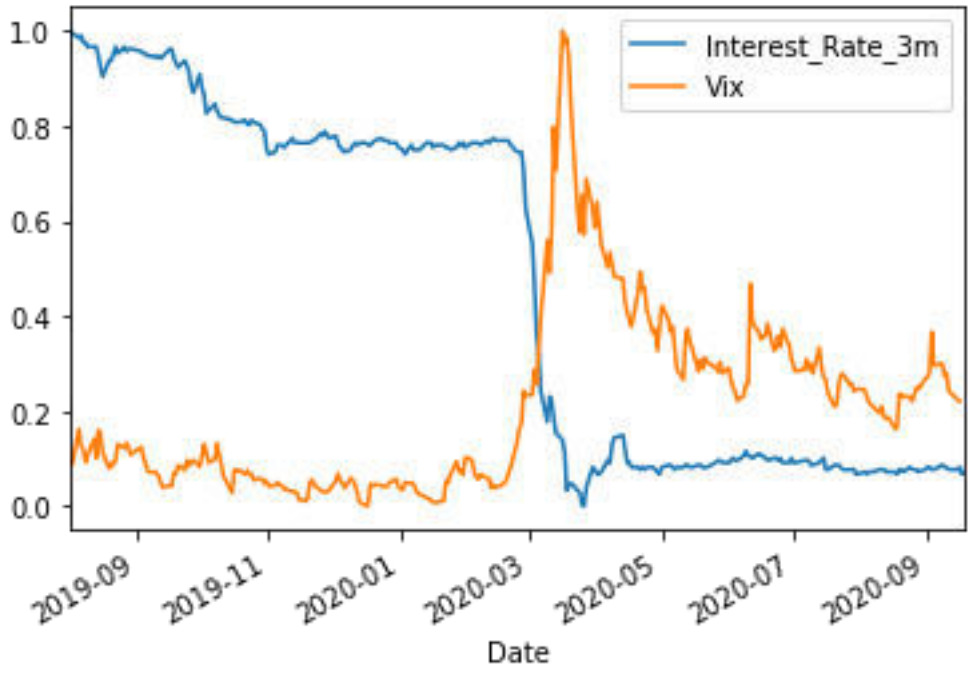
What we want to clarify is the impact of the gradual changes in the interest rates on the daily expected volatility of the market and especially when the implicit volatility will turn back to the previous level.

Input:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_standard = dataframe.copy()
data_standard[["Interest_Rate_3m","Vix"]] = scaler.fit_transform(data_standard[["Interest_Rate_3m","Vix"]])
data_standard.plot()
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6572dfac8>
```



Once standardized the data on the same scale, both the time series are plotted together to have an overview of the relationships among them.

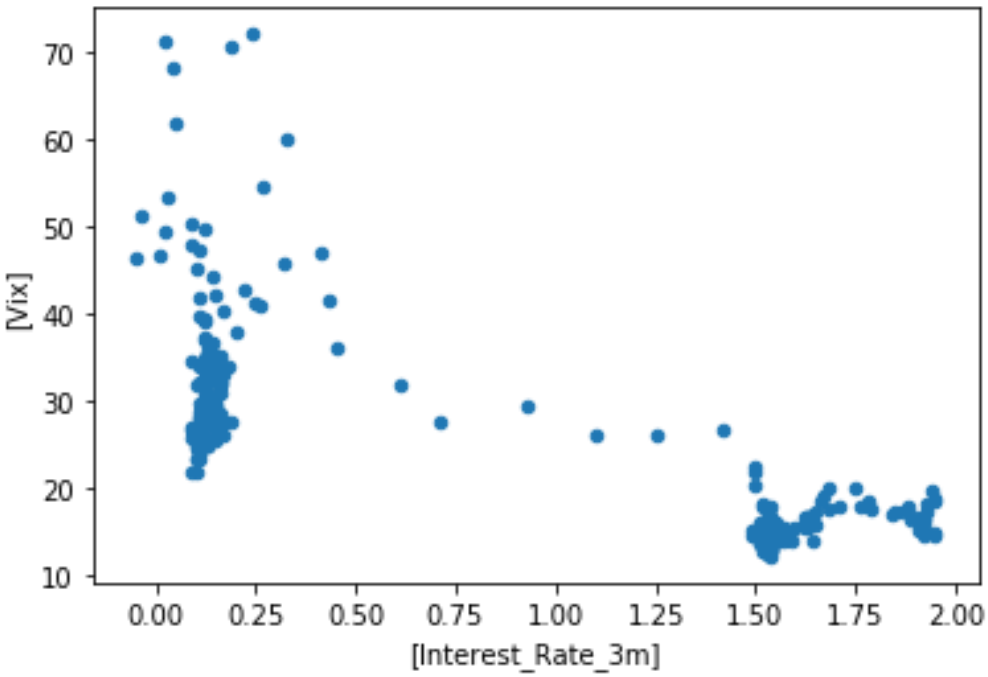
As we can immediately analyze the volatility tends to be lower wen the interest rates are higher and vice versa, with a large peak when a large deviation is verified.

Input:

```
dataframe.plot.scatter(["Interest_Rate_3m"],["Vix"])
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f65c5ddd08>
```



Instead, plotting the interest rates on the x axis and the VIX on the y axis, 2 main clusters can be observed: when the interest rates are less than 0.25 the volatility tends to be superior to 20, instead interest rates superior to 1.5 would imply a normal volatility level less than 20.

The pattern inside this scatter plot is a signal that using a Classification model can increase the likelihood of success in forecasting when the VIX index could turn back to the pre-QE policies.

Input:

```
for x in range(1, 20 + 1):
   dataframe['Interest_Rate_3m {} Tradig Day Ago'.format(x)] = dataframe['Interest_Rate_
3m'].shift(x)

dataframe = dataframe.dropna()
dataframe.tail()
```

Output:

Date	Interest_Rate_3m	Vix	Interest_Rate_3m 1 Tradig Day Ago	Interest_Rate_3m 2 Tradig Day Ago	Interest_Rate_3m 3 Tradig Day Ago
2020-09-10	0.12	28.86	0.12	0.13	0.11
2020-09-11	0.11	26.60	0.12	0.12	0.13
2020-09-14	0.11	25.85	0.11	0.12	0.12
2020-09-15	0.11	25.45	0.11	0.11	0.12
2020-09-16	0.12	25.40	0.11	0.11	0.11

The single parameter interest rate is transformed in 21 parameters, where each, after the first, represent a lag of one unit of time (third parameter represent the interest rate 2 days ago).

Input:

```
def labeling(x):
  if x >= 20:
    return 1
  else:
    return 0

list_X = [parameter for parameter in list(dataframe.columns) if parameter != 'Vix']

X, y = dataframe[list_X], dataframe['Vix'].apply(labeling)

from sklearn.model_selection import train_test_split
X_train, X_test = X[:'2020-08-01'], X['2020-08-01':]
y_train, y_test = y[:'2020-08-01'], y['2020-08-01':]

import autokeras as ak
search = ak.StructuredDataClassifier(max_trials=15)
search.fit(x=X_train, y=y_train, verbose=1)
```

Output:

```
Epoch 1/67
8/8 [==============================] - 0s 1ms/step - loss: 1.21
79 - accuracy: 0.4156
Epoch 2/67
8/8 [==============================] - 0s 1ms/step - loss: 0.90
37 - accuracy: 0.1948
Epoch 3/67
8/8 [==============================] - 0s 2ms/step - loss: 0.73
95 - accuracy: 0.2468
Epoch 4/67
8/8 [==============================] - 0s 1ms/step - loss: 0.64
69 - accuracy: 0.5238
Epoch 5/67
8/8 [==============================] - 0s 1ms/step - loss: 0.59
99 - accuracy: 0.5238
```

Input:

```
loss, acc = search.evaluate(X_test, y_test, verbose=0)
print("The Loss is {:.2f} and the Accuracy is {}".format(loss, acc))
```

Output:

```
The Loss is 0.19 and the Accuracy is 1.0
```

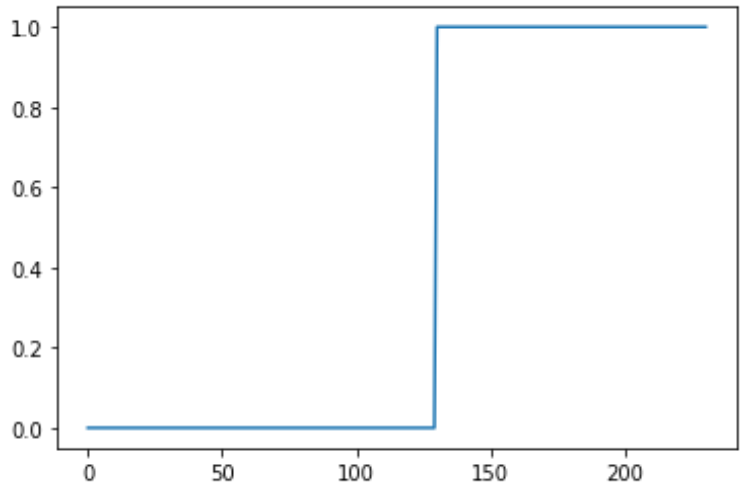
Once the model is trained with a classification algorithm using AutoKeras, we find an accuracy of 100% and a loss of 0.19, implying the observation are correctly labeled in almost any case on the data used for the training.

Input:

```
import matplotlib.pyplot as plt
model = search.predict(X_train)
plt.plot(model)
```

Output:

```
[<matplotlib.lines.Line2D at 0x1f65dbdb308>]
```



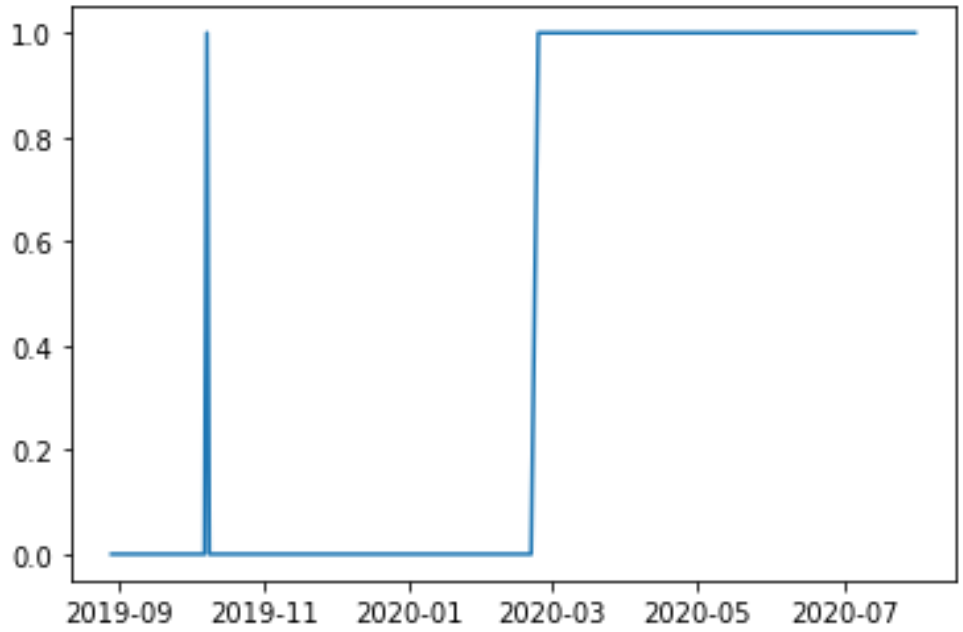
Above the forecasted outputs (0,1 labels) according the training inputs.

Input:

```
plt.plot(y_train)
```

Output:

```
[<matplotlib.lines.Line2D at 0x1f65e044f88>]
```



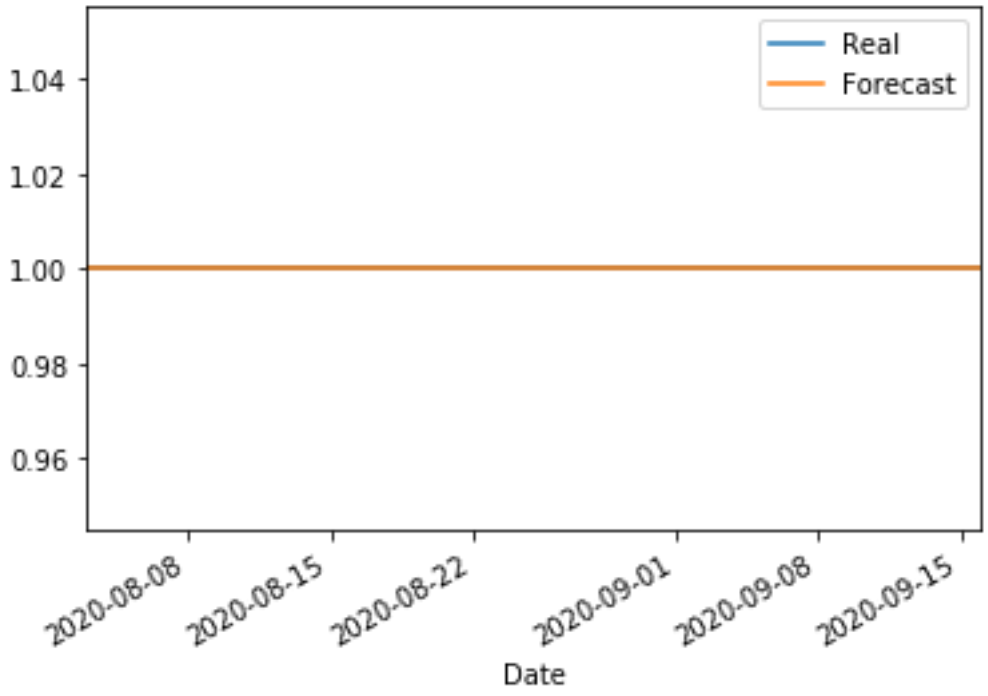
Above the real output verified on the market: the model is mostly accounting for persistent changes, ignoring the single outliers.

Input:

```
check = pd.DataFrame()
check["Real"] = y_test
check["Forecast"] = search.predict(X_test)

check.plot()
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f65d888a48>
```



Using the model on test data (on which it was never trained), it is still behaving optimally with an accuracy of 100%, classifying the VIX Index at an high level (above 1), that is what really happened for all the period of analysis.

In the graph both the real data and the forecasted one are plotted together but since they are equal there is an overlapping

```
Model: "functional_1"
_________________________________________________________________
__
Layer (type)                 Output Shape              Param #
=================================================================
==
input_1 (InputLayer)         [(None, 21)]              0
_________________________________________________________________
__
multi_category_encoding (Mul (None, 21)                0
_________________________________________________________________
__
dense (Dense)                (None, 32)                704
_________________________________________________________________
__
re_lu (ReLU)                 (None, 32)                0
_________________________________________________________________
__
dense_1 (Dense)              (None, 1)                 33
_________________________________________________________________
__
classification_head_1 (Activ (None, 1)                 0
=================================================================
==
Total params: 737
Trainable params: 737
Non-trainable params: 0
_________________________________________________________________
```

Deep Neural Networks for Stock Selection in Portfolio Optimization

In this last chapter the focus will be on applying a classification problem for Stock selection purpose, with the main scope of generating an Optimal Portfolio from the best performing stock.

We will start mining the main fundamental data of 30 companies, for all the years from 2011 to 2017: an equivalent of 104 parameters (as for example the value of Working Capital, etc.) for 30 companies and for 7 years.

Once obtained those values, the return YoY is computed and if the return in 2018 was equal or superior to 30% the Company in 2017 it is labeled as 1.

The main purpose is to discover the key patterns contributes the most to generating returns among time on the financial markets looking to the fundamental data.

Using the discovered strategy in 2017, on data the algorithm never observed before, an average return of 42.43% would be obtained: a value that would outperform most of the current funds.

Jupiter HTML version of the code available also at FinanceCS.com (Stefano Ciccarelli):
https://www.financecs.com/wp-content/uploads/2020/09/Deep-Neural-Network-for-Stock-Selection-in-Generating-the-Optimal-Portfolio.html

Input:

```
import quandl
import pandas as pd

quandl.ApiConfig.api_key = "INSERT YOUR API"

data = dict()

for n in range(10):
  request = quandl.get_table('SHARADAR/SF1', calendardate='201{}-12-31'.format(n))
  request.index = request['ticker']
  data['201{}'.format(n)] = request

def labeling(x):
  if x >= 0.3:
    return 1
  else:
    return 0


for n in range(9):
  data['201{}'.format(n)]['Return'] = data['201{}'.format(n + 1)]['price']/data['201{}'.format(n)]['price'] - 1
  data['201{}'.format(n)]['Label'] = data['201{}'.format(n)]['Return'].apply(labeling)
```

Above the data about 30 company is scraped over a decade and the relative returns YoY are computed (close price end of the next year / close price end of current year – 1). Consequentially the boolean labels are generated according the returns, if superior or equal to 30%.

Input:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

columns = list(data['2011'].columns[6:])
data_standard = pd.DataFrame(columns = data['2011'].columns)

for var in data:
  if var not in ("2010", "2018", "2019") :

    dataset_standard = data[var].copy()
    dataset_standard[columns] = scaler.fit_transform(dataset_standard[columns])
    data_standard = data_standard.append(dataset_standard)

  else:
    pass

data_standard.index = data_standard["calendardate"]
data_standard.tail()
```

Output:

	ticker	dimension	calendardate	datekey	reportperiod	lastupdated	accoci
calendardate							
2017-12-31	CSCO	MRY	2017-12-31	2017-07-29	2017-07-29	2020-09-03	0.910670
2017-12-31	CAT	MRY	2017-12-31	2017-12-31	2017-12-31	2020-08-05	0.868346
2017-12-31	BA	MRY	2017-12-31	2017-12-31	2017-12-31	2020-07-31	0.349356
2017-12-31	AXP	MRY	2017-12-31	2017-12-31	2017-12-31	2020-07-24	0.826091

Output:

assets	assetsavg	assetsc	...	shareswadil	sps	tangibles	taxassets	taxexp	taxli
0.042448	0.041329	0.498039	...	0.230141	0.005520	0.030401	0.480667	0.479851	0
0.021393	0.021883	0.196461	...	0.015472	0.341384	0.018651	0.191972	0.504957	0
0.035494	0.028950	0.507514	...	NaN	0.746081	0.033135	0.036399	0.440769	0
0.062915	0.058120	NaN	...	0.029303	0.150664	0.064469	0.000000	0.555775	0
0.140244	0.128110	0.783623	...	1.000000	0.012523	0.143497	0.000000	0.975882	0

In the code above, and inside the relative tables, we can observe the example data for the year 2017 and the key fundamental values: a standardization method is applied to be able to relate the variables on the same scale, since we want to analyze them in relation to the other key Companies' dimensions and not as standalone.

Input:

```
X, y = data_standard[columns], data_standard['Label']

from sklearn.model_selection import train_test_split
X_train, X_test = X[:'2017-01-01'], X['2017-01-01':]
y_train, y_test = y[:'2017-01-01'], y['2017-01-01':]

import autokeras as ak
search = ak.StructuredDataClassifier(max_trials=15)
search.fit(x=X_train, y=y_train, verbose=1)
```

Output:

```
Epoch 1/67
6/6 [==============================] - 0s 2ms/step - loss: 0.7960 - a
ccuracy: 0.2944
Epoch 2/67
6/6 [==============================] - 0s 2ms/step - loss: 0.6462 - a
ccuracy: 0.6778
Epoch 3/67
6/6 [==============================] - 0s 2ms/step - loss: 0.5743 - a
ccuracy: 0.7667
Epoch 4/67
6/6 [==============================] - 0s 2ms/step - loss: 0.5428 - a
ccuracy: 0.7667
```

Above the classification algorithm is applied to all the data before the 1st of January 2017.

Input:

```
loss, acc = search.evaluate(X_test, y_test, verbose=0)
print('{:.2f}%'.format(acc*100))
```

Output:

```
86.67%
```

The accuracy of the model is approximately 87%

```
Input:

y_predictions = search.predict(X_test)
y_train, y_test = y['2017-01-01'], y['2017-01-01':]
df = pd.DataFrame(y_test)


df["Expected"] = y_predictions
returns = data["2017"]
returns.index = returns["calendardate"]

df["Return"] = returns["Return"]
df["Ticker"] = returns["ticker"]
```

In the code above the model is evaluated on data that was never trained on (test as a real word example, to understand its performance once ready).

Input:

df

Output:

	Label	Expected	Return	Ticker
calendardate				
2017-12-31	0.0	0.0	-0.184720	XOM
2017-12-31	0.0	0.0	-0.101032	WMT
2017-12-31	0.0	0.0	0.062158	VZ
2017-12-31	1.0	1.0	0.426169	V
2017-12-31	0.0	1.0	0.130001	UNH
2017-12-31	0.0	1.0	0.068894	TSLA
2017-12-31	0.0	0.0	-0.117148	TRV
2017-12-31	0.0	0.0	-0.104303	PG
2017-12-31	0.0	0.0	0.205135	PFE
2017-12-31	1.0	1.0	0.354973	NKE
2017-12-31	1.0	1.0	0.430582	MSFT
2017-12-31	1.0	1.0	0.357917	MRK

	2017-12-31	0.0	0.0	-0.190466	MMM
	2017-12-31	0.0	0.0	0.031664	MCD
	2017-12-31	0.0	0.0	0.032040	KO
	2017-12-31	0.0	0.0	-0.087152	JPM
	2017-12-31	0.0	0.0	-0.089107	JNJ
	2017-12-31	0.0	0.0	0.012782	INTC
	2017-12-31	0.0	0.0	-0.259093	IBM
	2017-12-31	0.0	0.0	-0.110312	HD
	2017-12-31	0.0	0.0	-0.344285	GS
	2017-12-31	0.0	0.0	-0.566189	GE
	2017-12-31	0.0	1.0	0.186365	DIS
	2017-12-31	0.0	0.0	-0.249087	DD
	2017-12-31	0.0	0.0	-0.131001	CVX
	2017-12-31	1.0	1.0	0.350571	CSCO
	2017-12-31	0.0	0.0	-0.193616	CAT
	2017-12-31	0.0	0.0	0.093554	BA
	2017-12-31	0.0	1.0	-0.040177	AXP
	2017-12-31	1.0	1.0	0.464703	AAPL

Also, if the mode is not all time correct, when it suggests purchasing (label 1 in Expected) the return is still positive or insignificantly negative.

Input:

```
df["Strategy"] = df['Expected']*(df['Return']+1)

print("This Algorithm for Stock Selection produces an average annualized return of {:.2f}%".form
at(df["Strategy"].mean()*100))
```

Output:

```
This Algorithm for Stock Selection produces an average annualized ret
urn of 42.43%
```

Assuming an equal allocation to all the suggested stock, the average return would be 42.43% yearly, showing the power of DNNs in identifying the key patterns.

Once found the outperforming stock, according our classification algorithm, we can use it as an "engine" for the selection in a portfolio optimization approach, where we assign a specific weight to each stock and we account also for the volatility of the returns.

Input:

```
import yfinance as yf

stocks = pd.DataFrame()
count = 0
insert = list(df["Expected"])

for i in df["Ticker"]:
  if insert[count] == 1:
    count += 1
    try:
      data = yf.download(i, start="2017-01-01", end="2017-12-30")

      stocks[i] = data['Adj Close']
    except:
      print('failed: ', i)
  else:
    count += 1
    pass

print(stocks)
stocks = stocks.dropna(axis='columns')
stocks.head(10)
```

Output:

```
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
                      V         UNH        TSLA         NKE        MSFT
```

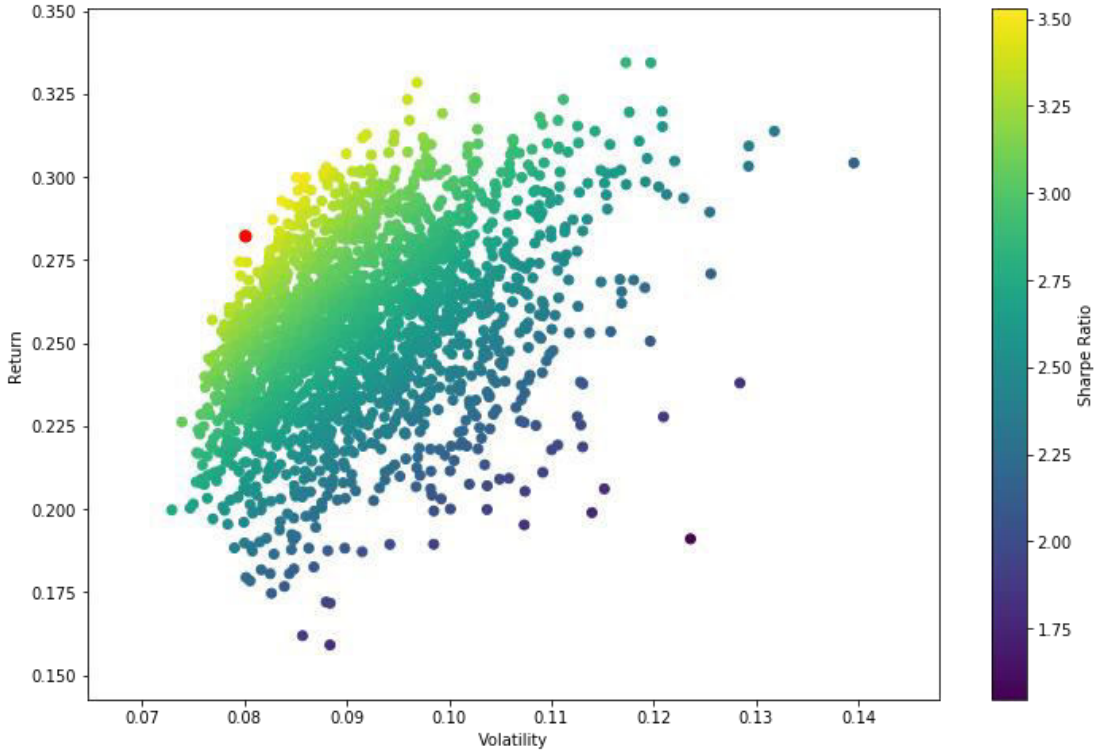
Above we obtain the close stock price for all the selected stock on the time period from the 1st of January 2017 to the 30th of December 2017

```
Input:

import sys
import yfinance as yf
import lxml
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import figure
from sklearn.preprocessing import MinMaxScaler
import os

log_ret = np.log(stocks/stocks.shift(1))

np.random.seed(42)
num_ports = 2000
all_weights = np.zeros((num_ports, len(stocks.columns)))
ret_arr = np.zeros(num_ports)
vol_arr = np.zeros(num_ports)
shape_arr = np.zeros(num_ports)

for x in range(num_ports):
   weights = np.array(np.random.random(len(stocks.columns)))
   weights = weights/np.sum(weights)
   all_weights[x,:] = weights
   ret_arr[x] = np.sum((log_ret.mean()*weights*252))
   vol_arr[x] = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov()*252, weights)))
   shape_arr[x] = ret_arr[x]/vol_arr[x]

print('Max sharpe ratio in the array: {}'.format(shape_arr.max()))
print("Its location in the array: {}".format(shape_arr.argmax()))

max_sr_ret = ret_arr[shape_arr.argmax()]
max_sr_vol = vol_arr[shape_arr.argmax()]

print('Proportion ratio: ',100*all_weights[shape_arr.argmax(),:])

plt.figure(figsize=(12,8))
```

```
plt.scatter(vol_arr, ret_arr, c=shape_arr)
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatility')
plt.ylabel('Return')
plt.scatter(max_sr_vol, max_sr_ret, c='red', s=50)
plt.show()
```

Output:

```
Max sharpe ratio in the array: 3.531086903574403
Its location in the array: 1451
Proportion ratio:  [18.88535894 15.30726511  2.09109787  4.67307346
7.96875013  3.43896713
 10.0053844   9.52941807 19.33222169  8.7684632 ]
```



Input:

```
print("The Expected return of this optimal portfolio would be {:.2f}%, with a volatility of {:.2f}%, generating a Sharpe Ratio of {:.2f}".format(ret_arr[1451]*100,vol_arr[1451]*100,shape_arr[1451]))
```

Output:

```
The Expected return of this optimal portfolio would be 28.26%, with a volatility of 8.00%, generating a Sharpe Ratio of 3.53
```

The Optimal Portfolio obtained has a Sharpe Ratio of 3.53, with an expected return of 28.26% and a volatility of 8%: this imply that in the worst case (99.9th Percentile) the return would be still positive (1.93%).

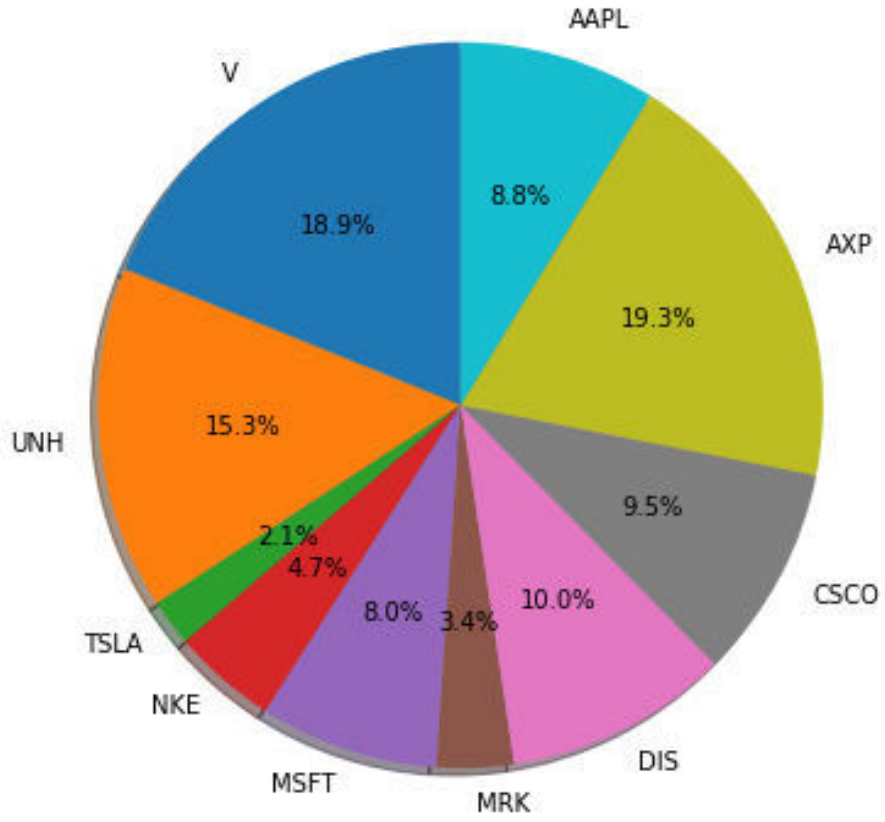
Input:

```
equipment = list(stocks.columns.values)
sizes = list(100*all_weights[shape_arr.argmax(),:])

dictionary = {}
count = 0

for x in equipment:
   dictionary[x] = sizes[count]
   count += 1

print(dictionary)

a = pd.DataFrame.from_dict(dictionary,orient= "index")
a.to_csv("final.csv")
```

```
fig1, ax1 = plt.subplots()
fig1.set_size_inches(8, 6)
ax1.pie(sizes, labels=equipment, autopct='%1.1f%%',
      shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
fig1.set_facecolor('white')
plt.show()
```

Output:

```
{'V': 18.885358942238106, 'UNH': 15.307265108539687, 'TSLA': 2.091097
8697403197, 'NKE': 4.673073461941965, 'MSFT': 7.968750132080986, 'MRK
': 3.438967133987564, 'DIS': 10.00538439670626, 'CSCO': 9.52941807110
8267, 'AXP': 19.332221687280196, 'AAPL': 8.768463196376654}
```



Consequentially the portfolio in 2017 should be composed of:

'V': 18.885358942238106

'UNH': 15.307265108539687

'TSLA': 2.0910978697403197

'NKE': 4.673073461941965

'MSFT': 7.968750132080986

'MRK': 3.438967133987564

'DIS': 10.00538439670626

'CSCO': 9.529418071108267

'AXP': 19.332221687280196

'AAPL': 8.768463196376654

Those value are in % and sum up to 100%.

An expected return of 28.26% in a year where the market index S&P 500 obtained a return of – 4.38% (2018 vs 2017) is an exceptional achievement that shows the potential of the Neural Networks in discriminating the underlying fundamental values of the companies that are relevant to predict the YoY success on the market of the listed Companies.

```
Model: "functional_1"
_________________________________________________________________
__
Layer (type)                 Output Shape              Param #
=================================================================
==
input_1 (InputLayer)         [(None, 107)]             0
_________________________________________________________________
__
multi_category_encoding (Mul (None, 107)               0
_________________________________________________________________
__
dense (Dense)                (None, 32)                3456
_________________________________________________________________
__
re_lu (ReLU)                 (None, 32)                0
_________________________________________________________________
__
dense_1 (Dense)              (None, 1)                 33
_________________________________________________________________
__
classification_head_1 (Activ (None, 1)                 0
=================================================================
==
Total params: 3,489
Trainable params: 3,489
Non-trainable params: 0
_________________________________________________________________
__
```

Conclusions

The thesis started with an analysis of the current worldwide landscape: why Machine Learning is becoming relevant for Shareholders and how this can contribute to generating value for overall society.

This call has a focus on economic and social inclusiveness given the expected non-linear risks generated by the potential Singularity's developments on the overall society, identifying the Governments as crucial in establishing a financial singularity strategy without disincentivizing innovation.

This was done using a strongly quantitative and data-driven approach: letting the data to proof the underlying assumptions, via Python as a powerful analytical tool.

The key final result in the last paragraph shows how a Neural Network can outperform the financial valuation abilities of a team of professional analysts in selecting stocks, according data that never observed during the training, obtaining an average YoY +42.43% return.

It was trained on a period span of 5 years data and on more than 100 fundamental parameters (as relevant Balance Sheet and Income Statements voices) on a set of 30 Companies.

It automatically selected the most relevant interactions among the standardize parameters (the relative relevancy of the ratio respect to the overall population), discriminating accurately the best companies to select and insert in the portfolio optimization strategy.

Following the portfolio optimization and the obtaining of the efficient frontier, is found a Sharpe Ratio of 3.53, with an expected return of +28.26% and a volatility of 8%, implying that in the worst of all empirical case (99.9th Percentile) the return would be still positive (+1.93%).

This while the market index (S&P 500) obtained a return of – 4.38% the same year.

This show how, especially in the next Future, Artificial Intelligence will tend to outperform the analytical abilities of human mind in executing specific and gradually more general tasks.

In another case, the effects of the number of searches related to airlines on Google, together with a sample of the "worrying level" related to COVID-19, were used to analyze and monitor the NASDAQ US Benchmark Airlines Index level in a way that was not possible before, clarifying when we can expect a recover of this index, if the demand for the US Airlines tickets will start to increase again.

A similar approach was used on forecasting the market volatility (a consensus sample from the VIX Index), using the negatively correlated short term risk-free interest rates (3-M US Treasury Bonds): to model how the Governments and Central Banks interventions applying Quantitative Easing policies can negatively affect the short-term market volatility pushing the Investor towards the equity market's higher yields and defining when we can expect the VIX Index to turn back to a normal level (under the value of 20).

Concluding, once proved the high value generation abilities of this technology, my hope goes to the correct applications of it, that Machine Learning will be used as a tool and not as a gun against society: is Time for the Governments to establish a long-term automation-indexed fund to sustain people when them will suddenly be exposed to the Singularity risk, or the democracies will fall under the control of a cognitive monopoly.

Bibliography

Technical

Hilpisch Y. (2018). Python for Finance: Mastering Data-Driven Finance. "O'Reilly Media, Inc.". ISBN 1492024317

Géron A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, "O'Reilly Media, Inc.". ISBN 1492032611

Prado M. L. (2018). Advances in Financial Machine Learning, Publisher Wiley, 2018. ISBN 1119482119

Ozdemir S., Kakade S. (2018). Principles of Data Science, Safari, an O'Reilly Media Company, ISBN 178980454X

Burkov, A. (2019). The Hundred-Page Machine Learning Book, Edition illustrated.

Methodological

Seth Stephens-Davidowitz (2018): EVERYBODY LIES: Big Data, New Data, and What the Internet Can Tell Us about Who We Really Are. New York: Dey Street Books. ISBN: 9780062390868**.**

Pentland, Alex, (2014). Social Physics: How Good Ideas Spread-the Lessons from a New Science. New York: The Penguin Press, ISBN 9781594205651

Harari, Y. N. (2016). Homo deus: A brief history of tomorrow. ISBN 9781910701874

Bostrom, N. (2014). Superintelligence: Paths, dangers, strategies. Oxford University Press. ISBN 9780199678112

Domingos, P. (2015). The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books. ISBN 9781501299353