

ie

Course

Day 1: Sentiment Analysis - 27 of September from 10:30am to 12:30pm - Online

Day 2: Market Trend Identification - 11th of October from 10:30am to 12:30pm – Online

Material can be download from:

Manoel Gadi

- more than 16 years in Banking and Financial Institutions like Citibank and Santander
- most in Analytics functions for Risk Management in Brazil (Sao Paulo), the UK (Milton Keynes) and Spain (Madrid)
- Teaching in IE University since 2013 and since 2019 he is fully dedicated to teaching in IE University in courses ranging from Programming, Statistics, Machine Learning, Banking, and Fintech.



identifying emerging
trends, correlations,
and predictive
patterns.

IE University – Prof. Manoel Gadi

Index

1. GATHERING AND VISUALIZING FINANCIAL DATA

Reviewing google colab environment or Jupyter/Anaconda

1. Trending Analysis key concepts
2. Gathering financial data
3. Traditional Forecast Methods (ARIMA)
4. Neural Network Models Applied to Forecasting (LSTM)
5. Trend Analysis of News Sentiment
6. LLMs applied to forecasting
7. Discussion and conclusion
8. Bibliography
9. Annex

1. TRENDING ANALYSIS KEY CONCEPTS



Trending Analysis key concepts

- Trending analysis is the process of identifying and tracking emerging patterns, topics, or ideas that are gaining popularity or significance over time. It involves identifying trends as they are happening and understanding their potential impact.



The Evolution of Trending Analysis Techniques

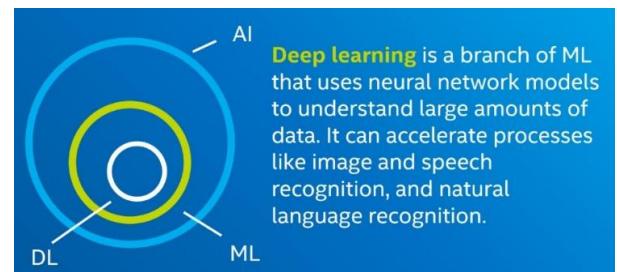
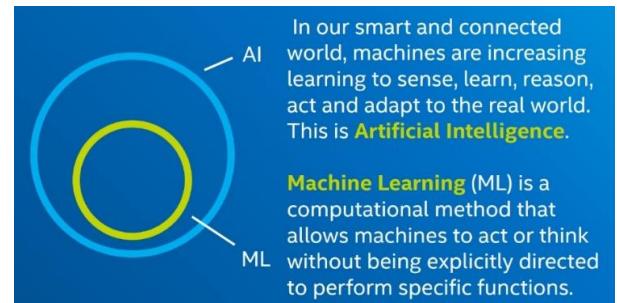
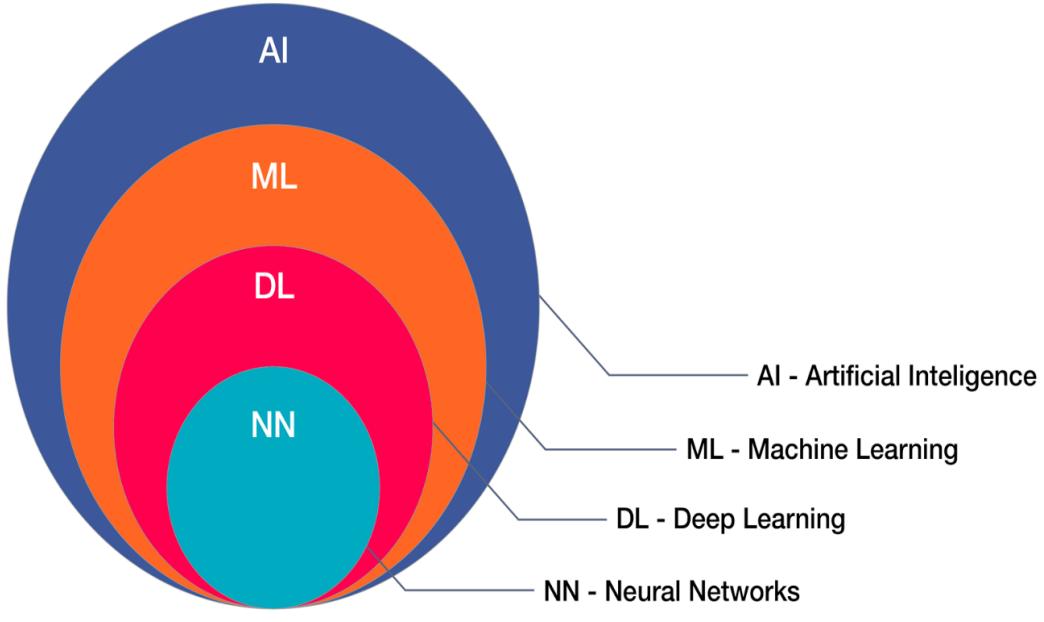
- The methods used for trending analysis have evolved significantly over time. With the advent of computers and the internet, automated techniques, such as ARIMA, Facebook Prophet has taken the part of trend analysis called forecasting to another level.



The Role of AI in Modern Trending Analysis

- Artificial intelligence (AI) has revolutionized the field of trending analysis.
- First with the advent of advanced Neural Network model, from which we can highlight LSTM (Long-Short Term Memory). Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that is designed to address the vanishing gradient problem that occurs in traditional RNNs. LSTMs are particularly well-suited for tasks that involve sequential data, such as natural language processing, **time series analysis**, and speech recognition.
- This paved the way to more advanced Large Language Models (LLM). So, nowadays AI-powered tools can process vast amounts of data from various sources, including text, images, and videos, to identify emerging trends and patterns. This has enabled analysts to gain deeper insights and make more accurate predictions.

AI taxonomy



AI types based on capabilities:

- Artificial narrow intelligence (ANI)- Narrow range of abilities.
- Artificial general intelligence (AGI)-On par with human capabilities.
- Artificial superintelligence (ASI)-More capable than a human.

How did we get here?

- Can LLM models make reasonable forecasting and trend analysis?
- It all started with traditional ARIMA, then LSTMs, then LLMs... Let's see it.

2. GATHERING FINANCIAL DATA

Source number 1: Yahoo Finance

- It contains historical & current stock prices + calculated metrics such as the beta
- Nowadays, after some changes, Yahoo Finance API provides data from sources such as FRED, Fama/French Data Library or World Bank
- To connect with the Yahoo Finance API is enough with the installation of the library called *yfinance*
- Basic example: Download Apple's stock prices from the years 2000 – 2010

```
!pip install pandas_datareader yfinance

import pandas as pd
import yfinance as yf

df_yahoo = yf.download('AAPL',
                      start='2000-01-01',
                      end='2010-12-31',
                      progress=False)
df_yahoo
```

```
import pandas_datareader.data as web

import datetime

start = datetime.datetime(2010, 1, 1)

end = datetime.datetime(2013, 1, 27)

df_gdp = web.DataReader('GDP', 'fred', start, end)
df_gdp
```

```
from pandas_datareader import wb
df_gdp = wb.download(indicator='NY.GDP.PCAP.KD', country=['US', 'CA', 'MX'], start=2005, end=2008)
df_gdp
```

Questions: What is *df_yahoo*, an array or a data-frame? How many variables and data does it contain? Solve this questions with

https://github.com/manoelgadi/ESMA/blob/main/Day2_TrendAnalysis/IE_ESMA_Day2_TrendAnalysis_nosolution.ipynb

Exercise 1: Analyzing Economic Indicators using Yahoo Finance API

Using the Yahoo Finance API, retrieve historical data for the following economic indicators from 2010-01-01 to 2023-12-31:

- **US 10-Year Treasury Constant Maturity Rate:** ^TNX
- **US 3-Month Treasury Bill Constant Maturity Rate:** ^IRX
- **US Consumer Price Index (CPI):** CPI

Exercise 1: Analyzing Economic Indicators using Yahoo Finance API

Using the Yahoo Finance API, retrieve historical data for the following economic indicators from 2010-01-01 to 2023-12-31:

- **US 10-Year Treasury Constant Maturity Rate:** ^TNX
- **US 3-Month Treasury Bill Constant Maturity Rate:** ^IRX
- **US Consumer Price Index (CPI):** CPI

```
import pandas as pd
import yfinance as yf
tickers = ['^TNX', '^IRX', 'CPI']
start_date = '2010-01-01'
end_date = '2023-12-31'
df_economic_indicators = yf.download(tickers, start=start_date, end=end_date)
df_economic_indicators
```

Exercise 2: Analyzing Economic Indicators using Yahoo Finance API

Using the pandas_datareader, retrieve historical data for the following economic indicators:

- Federal Funds Rate - the interest rate that banks charge each other for overnight loans: FEDFUND
- GDP per capita for Spain (ES), France (FR) and Germany (DE): NY.GDP.PCAP.KD

Exercise 2: Analyzing Economic Indicators using Yahoo Finance API

Using the pandas_datareader, retrieve historical data for the following economic indicators:

- Federal Funds Rate - the interest rate that banks charge each other for overnight loans: FEDFUND
- GDP per capita for Spain (ES), France (FR) and Germany (DE): NY.GDP.PCAP.KD

```
import pandas as pd
import pandas_datareader.data as web

# Retrieve data from FRED
federal_funds_rate = web.DataReader('FEDFUND', 'fred', start='2020-01-01', end='2023-12-31')
print(federal_funds_rate.tail())

from pandas_datareader import wb
df_gdp = wb.download(indicator='NY.GDP.PCAP.KD', country = ['ES', 'FR', 'DE'], start=2005, end=2008)
df_gdp
```

Source number 2: Quandl

- It was a strong provider of alternative data products for investment professionals, with an easy way to download data via a Python library
- It contains stock prices, dividends, and splits for 3000 US publicly traded companies
- It is good for getting historical data or learning how to access the databases
- However, its main drawback is that as of April 2018, it is no longer supported (there is no recent data)

```
In [6]: import pandas as pd  
import quandl
```

2. Authenticate using the personal API key:

```
In [7]: QUANDL_KEY = '3xvyiQcm6Qeyxzxo7nRZ' # replace {key} with your own API key  
quandl.ApiConfig.api_key = QUANDL_KEY
```

3. Download the data:

```
df_quandl = quandl.get(dataset='WIKI/AAPL',  
                      start_date='2000-01-01',  
                      end_date='2010-12-31')
```

Source number 3: Intrinio

- It offers Access to its free (with limits) database
- It is a good source which allows to download already calculated technical indicator such as the Moving Average Convergence Divergence (MACD)

```
In [10]: import intrinio_sdk  
import pandas as pd
```

2. Authenticate using the personal API key and select the API:

```
In [11]: intrinio_sdk.ApiClient().configuration.api_key['api_key']='OjFiYTliNmFizDzmMGU5Yjk5NjB1NzM5MjkxNmViNTzk  
security_api = intrinio_sdk.SecurityApi()
```

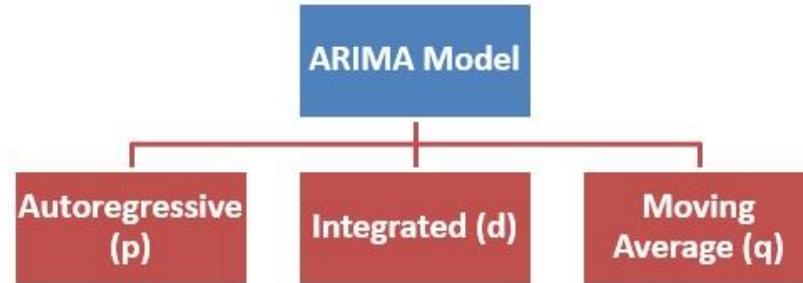
```
r = security_api.get_security_stock_prices(identifier='AAPL',  
                                         start_date='2000-01-01',  
                                         end_date='2010-12-31',  
                                         frequency='daily',  
                                         page_size=10000)
```

- Other data sources can be accessed by the following libraries: *iexfinance* , *tiingo* and *alpha_vantage*

3. TRADITIONAL FORECAST METHODS

Traditional Methods

- ARIMA model
- GARCH
- Facebook Prophet



ARIMA

- **Auto Regression (AR) part -**
Long Memory Models
- AR is a stationary model, it forecast the past values of the series based solely on the past values of the series (lags).
- In a Auto Regression 1, AR(1), the forecast is calculated based on the previous day target (also known as lagged target).
- In a Auto Regression 1, AR(p), the forecast is calculated based on the p days ago target.
- <https://www.youtube.com/watch?v=Mc6sBAUdDP4>



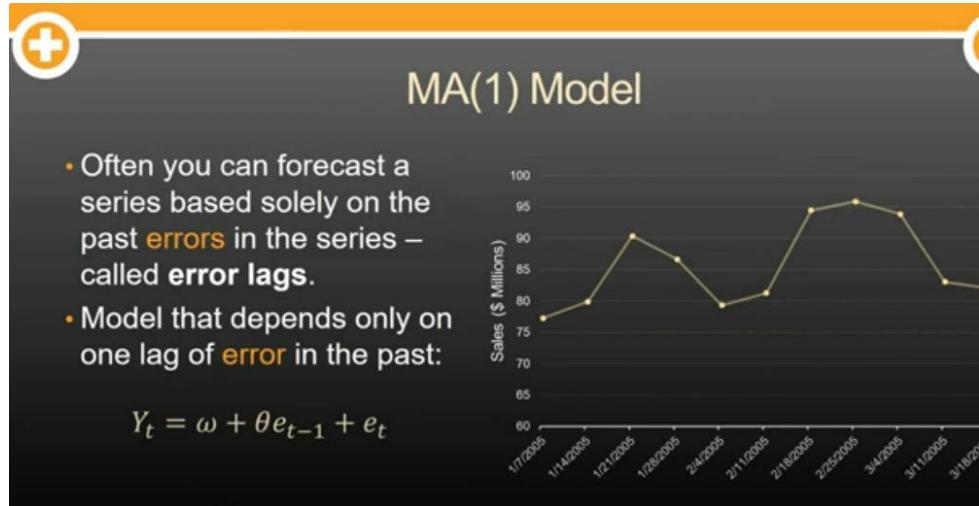
Long Memory? I Never Forget!

- This recursion in time goes back until the beginning of the series so these are called **long memory models**.

$$\begin{aligned} Y_t &= \omega + \phi Y_{t-1} + e_t \\ Y_{t-1} &= \omega + \phi Y_{t-2} + e_{t-1} \\ Y_t &= \omega + \phi(\omega + \phi Y_{t-2} + e_{t-1}) + e_t \\ Y_t &= \omega^* + \phi^2 Y_{t-2} + \phi e_{t-1} + e_t \end{aligned}$$

ARIMA

- **Moving Average (MA) part**
 - Short Memory Models
- In a Moving Average 1, MA(1), the forecast is calculated based on the error of the previous day.
- In a Moving Average q, MA(q), the forecast is calculated based on the moving average errors of the **q** previous day.
- https://www.youtube.com/watch?v=zNLG8tsA_Go



Let's Practice in Python

- https://github.com/manoelgadi/ESMA/blob/main/Day2_TrendAnalysis/IE_ESMA_Day2_TrendAnalysis_nosolution.ipynb

3.1 A NEW PARADIGM: THE PROPHET MODEL

Introduction to the model

- Prophet was a model introduced in 2017 to be freely used by Facebook in 2 versions: for R and Python <https://peerj.com/preprints/3190.pdf#section.1>

Forecasting at Scale

Sean J. Taylor^{*†}

Facebook, Menlo Park, California, United States
sjt@fb.com

and

Benjamin Letham[†]

Facebook, Menlo Park, California, United States
bletham@fb.com

Abstract

Forecasting is a common data science task that helps organizations with capacity planning, goal setting, and anomaly detection. Despite its importance, there are serious challenges associated with producing reliable and high quality forecasts – especially when there are a variety of time series and analysts with expertise in time series modeling are relatively rare. To address these challenges, we describe a practical approach to forecasting “at scale” that combines configurable models with analyst-in-the-loop performance analysis. We propose a modular regression model with interpretable parameters that can be intuitively adjusted by analysts with domain knowledge about the time series. We describe performance analyses to compare and evaluate forecasting procedures, and automatically flag forecasts for manual review and adjustment. Tools that help analysts to use their expertise most effectively enable reliable, practical forecasting of business time series.

Keywords: Time Series, Statistical Practice, Nonlinear Regression

The GAM formulation has the advantage that it decomposes easily and accommodates new components as necessary, for instance when a new source of seasonality is identified. GAMs also fit very quickly, either using backfitting or L-BFGS (Byrd et al. 1995) (we prefer the latter) so that the user can interactively change the model parameters.

- Flexibility: We can easily accommodate seasonality with multiple periods and let the analyst make different assumptions about trends.
- Unlike with ARIMA models, the measurements do not need to be regularly spaced, and we do not need to interpolate missing values *e.g.* from removing outliers.
- Fitting is very fast, allowing the analyst to interactively explore many model specifications, for example in a Shiny application (Chang et al. 2015).
- The forecasting model has easily interpretable parameters that can be changed by the analyst to impose assumptions on the forecast. Moreover, analysts typically do have experience with regression and are easily able to extend the model to include new components.

The modelo structure

- The real *prophet* formulation is not published by Facebook, but some details are left in order to understand how the model forecasts

$$y(t) = g(t) + s(t) + h(t) + e(t)$$

g(t)

- **trend** models *non-periodic* changes; linear or logistic

s(t)

- **seasonality** represents *periodic* changes; i.e. weekly, monthly, yearly

h(t)

- ties in effects of **holidays**; on potentially irregular schedules ≥ 1 day(s)

- This model has 3 deterministic and 1 random components

Components of the prophet model: the trend component

- Analyzing its components, it is possible to have an idea about of its way of forecasting

$$g(t) = \frac{C(t)}{1 + \exp(-(k + \mathbf{a}(t)^\top \boldsymbol{\delta})(t - (m + \mathbf{a}(t)^\top \boldsymbol{\gamma})))}$$

- The trend component has a logistic behavior, but on this component is used the concept of *point of change*
- In the model training process, *prophet* deduces a number of *point of change* and it adapts the *trend* component interpolating logistic curves between the *points of change*
- Doing above, *prophet* can detect jumps or important deviation in the time series

Components of the prophet model: the seasonal component

- *Prophet* makes use of the *Fourier* series to modelise the seasonal component

We rely on Fourier series to provide a flexible model of periodic effects (Harvey & Shephard 1993). Let P be the regular period we expect the time series to have (e.g. $P = 365.25$ for yearly data or $P = 7$ for weekly data, when we scale our time variable in days). We can approximate arbitrary smooth seasonal effects with

$$s(t) = \sum_{n=1}^N \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$

- Once the *trend* is modelized, a seasonal structure is added. Depend on the period of the time series, more or less *cos* and *sin* functions will be used

Other components and properties of the *prophet*

- Under this model structure, is mentioned the component called *holidays*

$$Z(t) = [\mathbf{1}(t \in D_1), \dots, \mathbf{1}(t \in D_L)]$$

and taking

$$h(t) = Z(t)\kappa.$$

As with seasonality, we use a prior $\kappa \sim \text{Normal}(0, \nu^2)$.

- This mean that calendar variable can be added as explanatory variables, but *prophet* is not restricted to this kind of variables, it is possible to add extra information in a similar way to the above sesión
- Finally some interesting properties of this model are the following ones:
 - (1) Prophet is robust to intense seasonal effects
 - (2) Prophet manages the missing and you can use series with missing data without interpolation
 - (3) Prophet is an adaptative model
 - (4) Prophet detects and manages the outliers in the time series

A Python example with prophet: Part I Data cleaning

```
import pandas as pd
import numpy as np
from datetime import datetime

#Reading of data
df = pd.read_csv('data/multivariate.csv', sep = ';')

#Handling time data
fch = []

for i in range(0, len(df[ 'Date'])):
    fch.append(datetime.strptime(df[ 'Date'][i], "%d/%m/%Y").strftime( '%Y-%m-%d'))

df[ 'Date'] = fch

#Prophet needs a variable called ds and y in the data structure
df.columns = [ 'ds', 'y', 'EURUSD', 'SP', 'FUTGAS',]

#Separating in training - test
df_train = df[df[ 'ds'] <= '2021-12-31']
df_test = df[df[ 'ds'] > '2021-12-31']

#A basic prophet model with only the time serie of interest is going to
#be built
df_train = df_train[['ds', 'y']]
```

To make forecasting with *prophet* is need two variables: *ds* and *y* Using this variables *prophet* ‘knows’ where is the date and where are the values

A Python example with prophet: Part II Modeling

```
#Modelling

import fbprophet as fb

model = fb.Prophet(yearly_seasonality=True,
                     daily_seasonality=True)

# fit the model
model.fit(df_train)
```

Prophet has many hyperparameter, to be used. Some of them are *yearly_sasonality* and *daily_seasonality* it is recommendable its activation. Doing this *prophet* ‘knows’ how the data are related

A Python example with prophet: Part III Forecasting to one day

```
#Forecasting and model evaluation
future = model.make_future_dataframe(periods=1, freq = 'D', include_history=False)
forecast = model.predict(future)[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
forecast['ds'] = '2022-01-03'

eam = np.mean(np.abs(df_test[df_test['ds'] == '2022-01-03']['y'].values-\n                    forecast['yhat'].values))
eam
```

ds	yhat	yhat_lower	yhat_upper
2022-01-03	80.8896	73.2433	88.2082

eam
4.8096411261917495

```
#Separating in training - test
df_train = df[df['ds'] <= '2022-01-03']
df_test = df[df['ds'] > '2022-01-03']

#A basic prophet model with only the time serie of interest is going to be built
df_train = df_train[['ds', 'y']]

#Modelling

import fbprophet as fb

model = fb Prophet(weekly_seasonality=True,
                    daily_seasonality=True)

# fit the model
model.fit(df_train)

#Forecasting and model evaluation
future = model.make_future_dataframe(periods=1, freq = 'D', include_history=False)
forecast = model.predict(future)[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
forecast['ds'] = '2022-01-04'

eam = np.mean(np.abs(df_test[df_test['ds'] == '2022-01-04']['y'].values-\n                    forecast['yhat'].values))
eam
```

ds	yhat	yhat_lower	yhat_upper
2022-01-04	82.5029	75.3832	89.6156

eam
5.512943123345991

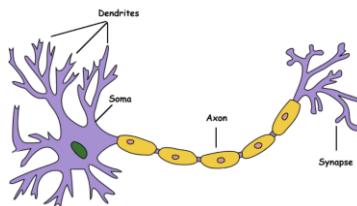
ds	y	EURUSD	SP	FUTGAS
2022-01-14	83.82	1.1325	4662.85	4.262
2022-01-13	82.12	1.1408	4659.03	4.27
2022-01-12	82.64	1.1411	4726.35	4.857
2022-01-11	81.22	1.1455	4713.07	4.249
2022-01-10	78.23	1.1442	4670.29	4.079
2022-01-07	78.9	1.1367	4677.03	3.916
2022-01-06	79.46	1.1326	4696.05	3.812
2022-01-05	77.85	1.136	4700.58	3.882
2022-01-04	76.99	1.1297	4793.54	3.717
2022-01-03	76.08	1.1314	4796.56	3.815

ds	y	EURUSD	SP	FUTGAS
2022-01-18	85.43	1.1343	4577.11	4.283
2022-01-14	83.82	1.1325	4662.85	4.262
2022-01-13	82.12	1.1408	4659.03	4.27
2022-01-12	82.64	1.1411	4726.35	4.857
2022-01-11	81.22	1.1455	4713.07	4.249
2022-01-10	78.23	1.1442	4670.29	4.079
2022-01-07	78.9	1.1367	4677.03	3.916
2022-01-06	79.46	1.1326	4696.05	3.812
2022-01-05	77.85	1.136	4700.58	3.882
2022-01-04	76.99	1.1297	4793.54	3.717

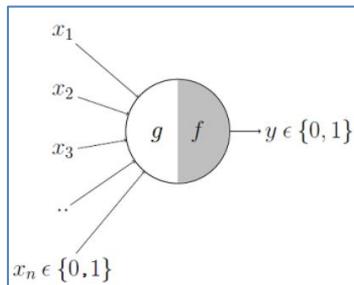
4. NEURAL NET MODELS APPLIED TO FORECASTING

A short history of the Neural: The beginings

- A neural net model is one which emulates the neurons of our brains



- The modern history of the neural net models begins around 1943 with Warren Strugis Mculloch (1898 -1969) and Walter Harry Pitts (1923 – 1969) who published the paper: *A Logical Calculus of the Ideas Immanent in Nervous Activity*



$$g(x_1, x_2, x_3, \dots, x_n) = g(\mathbf{x}) = \sum_{i=1}^n x_i$$
$$y = f(g(\mathbf{x})) = 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ = 0 \quad if \quad g(\mathbf{x}) < \theta$$

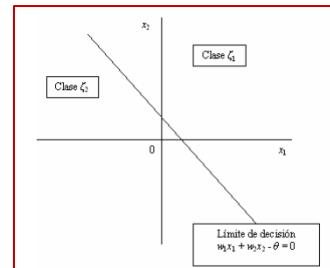
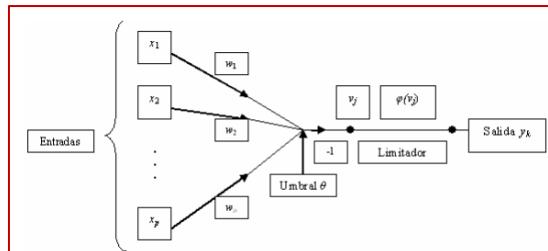
A short history of the Neural net models Part II: The first golden age



- The research about A. I. were intense between 1950 up to 1970
- In 1950 Alan Turing ask if a machine could get the ability to think and he designs the *Turing test*. (He creates the first program which can play chess)
- At 1956 is celebrated the Dortmund Congres were appeared the term *Artificial Intelligence*



- One of the most valuable result was the *Fundamental Theorem of the Perceptron* where Rossemblat shows that *Perceptron* can learn

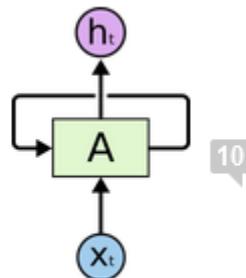


A short history of the Neural net models & AI

- The relationship between AI and time series analysis is longstanding. Over the past 30 years, algorithms using AI for time series forecasting have been developed, either independently or in combination with classical statistical methods. Among these, the most well-known algorithm is the Long Short-Term Memory (LSTM) network, which is detailed in this review.
- LSTM networks can be used for various time series forecasting tasks. In this tutorial, you will learn how to develop several LSTM models for different types of time series forecasting problems.
- But let's start with Recurrent Neural Networks (RNNs).

Recurrent Neural Networks (RNN)

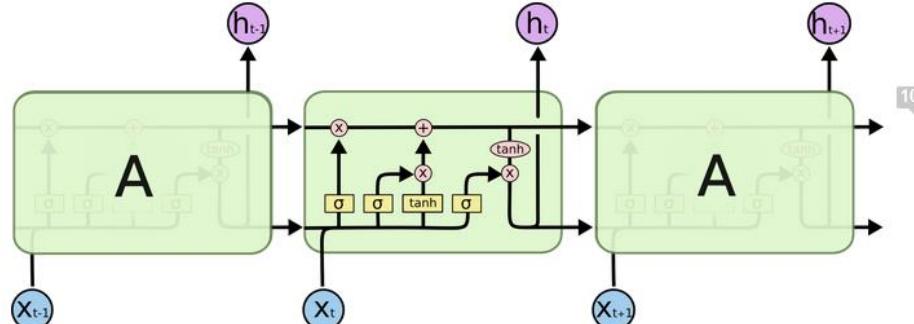
- Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.
- Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.
- RNNs address this issue by introducing loops, allowing information to persist.



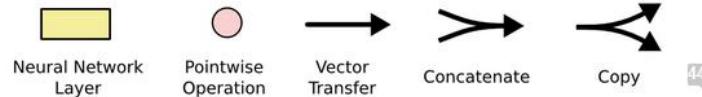
Recurrent Neural Networks have loops.

Recurrent Neural Networks (RNN)

- However, RNN suffer from the long term dependency problem. Overtime, with more information, RNN become less effective of learning new things.
- LSTM provides solution to long term dependency problem, by introducing a cell state. Image
- The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.



The repeating module in an LSTM contains four interacting layers.



<https://www.youtube.com/watch?v=b61DPVFX03I>

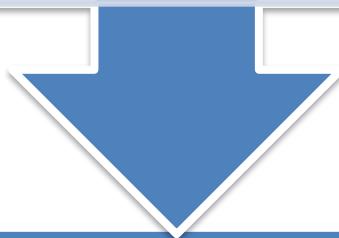
Let's Practice in Python

- https://github.com/manoelgadi/ESMA/blob/main/Day2_TrendAnalysis/IE_ESMA_Day2_TrendAnalysis_nosolution.ipynb

5. TREND ANALYSIS OF NEWS SENTIMENTS

GDELT

Extraction of news using GDELT



TextBlob Sentiment score

Polarity

Subjectivity

Let's Practice in Python

- https://github.com/manoelgadi/ESMA/blob/main/Day2_TrendAnalysis/IE_ESMA_Day2_TrendAnalysis_nosolution.ipynb

6. LLMs APPLIED TO FORECASTING

Technical story, how we got here: ChatGPT

2014: Dzmitry Bahdanau is an **Adjunct Professor at McGill University** and a **research scientist at ServiceNow Element AI**.

A screenshot of the arXiv preprint page for "Neural Machine Translation by Jointly Learning to Align and Translate" by Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. The page has a red header with the arXiv logo and navigation links. The main content area shows the title, authors, abstract, and a large block of text. At the bottom, there's a comments section and a DOI link.

Neural Machine Translation by Jointly Learning to Align and Translate

Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder-decoders and consists of an encoder that encodes a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of the basic encoder-decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

Comments: Accepted at ICLR 2015 as oral presentation
Subjects: Computation and Language (cs.CL); Machine Learning (cs.LG); Neural and Evolutionary Computing (cs.NE); Machine Learning (stat.ML)
Cite as: arXiv:1409.0473 [cs.CL]
(or arXiv:1409.0473v7 [cs.CL] for this version)
<https://doi.org/10.48550/arXiv.1409.0473>

<https://arxiv.org/abs/1409.0473>



2017: Google Brain and Google Research Teams

A screenshot of the arXiv preprint page for "Attention Is All You Need" by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. The page has a red header with the arXiv logo and navigation links. The main content area shows the title, authors, abstract, and a large block of text. At the bottom, there's a comments section and a DOI link.

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Comments: 15 pages, 5 figures
Subjects: Computation and Language (cs.CL); Machine Learning (cs.LG)
Cite as: arXiv:1706.03762 [cs.CL]
(or arXiv:1706.03762v2 [cs.CL] for this version)
<https://doi.org/10.48550/arXiv.1706.03762>

<https://arxiv.org/abs/1706.03762>

2017: Google Brain and Google Research Teams

Cornell University

arXiv > cs > arXiv:1706.03762

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5)]

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrent or convolutional operations entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly fewer parameters than previous models. On the WMT’14 English-to-German translation task, our model achieves 28.4 BLEU on eight GPUs, a small fraction of the training cost of the best prior model, by parsing both with large and limited training sets. We show that

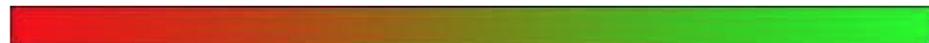
achieves 28.4 BLEU

Comments: 15 pages, 5 figures
Subjects: Computation and Language (cs.CL); Machine Learning (cs.LG)
Cite as: arXiv:1706.03762 [cs.CL]
(or arXiv:1706.03762v5 [cs.CL] for this version)
<https://doi.org/10.48550/arXiv.1706.03762>



BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

The following color gradient can be used as a general scale [interpretation of the BLEU score](#):



0 10 20 30 40 50 60 70 >80

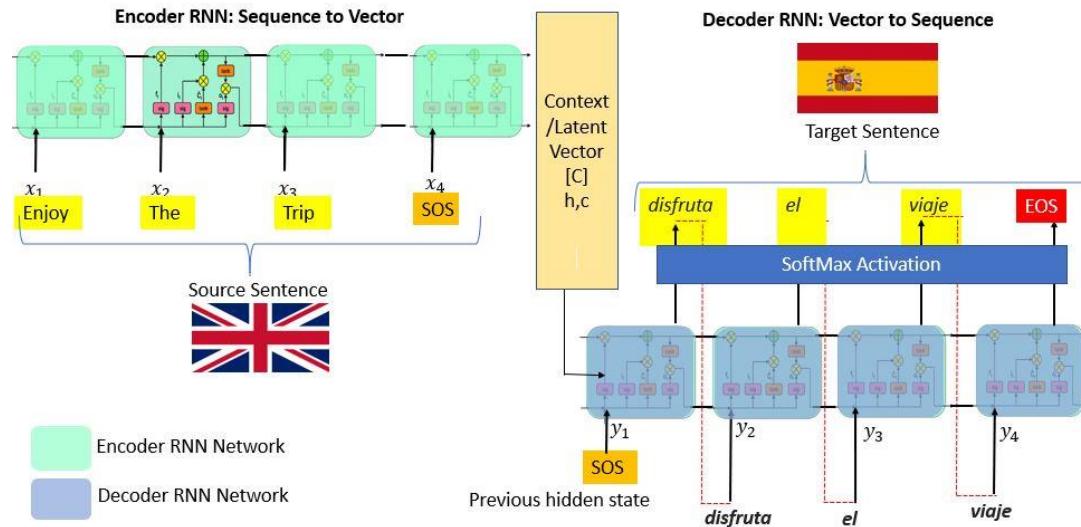
BLEU:

<https://cloud.google.com/translate/automl/docs/evaluate>

After this paper, publications stopped! Revolution was about to break through with BLEU higher than 30

What was prior to the attention model?

Encoder-decoder RNNs/LSTMs – Ok for short sentences, but bad for long sentences.



But this is not what a Realtime translator does



Me gustaria entender más sobre ChatGPT, sus aplicaciones y limitaciones.

In order to translate “gustaria” into English the translator needs the previous and the next word and remember the context of what has already been translated – the next word becomes the target in a supervised machine learning:

I would like to understand

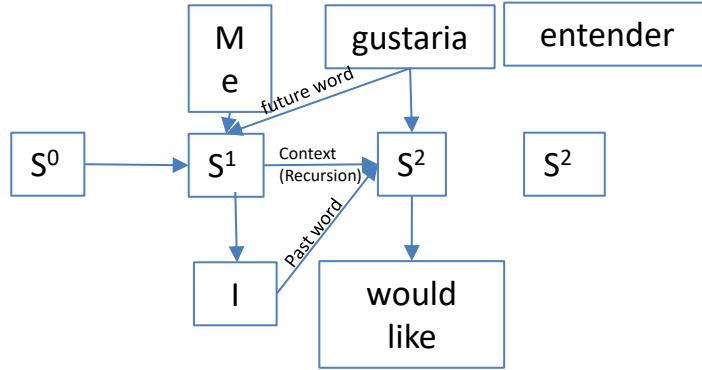
But this is not what a Realtime translator does

Me **gustarian** entender más sobre ChatGPT, sus aplicaciones y limitaciones.

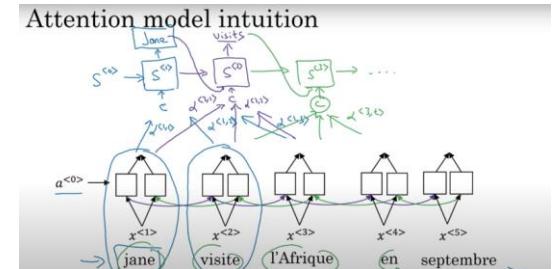
In order to translate “gustaria” into English the translator needs the previous and the next word and remember the context of what has already been translated:

I **would like** to understand

Attention model simplified, technically it is predicting the “next word” – target = next word:



Real attention model is more complex:



This can be used not only for translation,
But for generating any text!

What has OpenAI done differently?

1. **Attention Model (supervised)**, not LSTM/RNN as Transformers with Billions of parameters achieve much better results
2. **Reinforcement learning with human feedback**: Human Feedback generates policy (not supervised) – 4 answers verified by OpenAI employees.
3. **Self-Supervised Learning** (SSL) is a Machine Learning paradigm where a model, when fed with unstructured data as input, generates data labels automatically, which are further used in subsequent iterations as ground truths. For example, removing parts of the text and trying to reconstruct the text. Learn own its own!

Current Limitations:

- 1) No experience, only read/see* things.
- Like a male gynaecologist explaining the pain of giving birth → Agents are coming soon to solve this....



Current Limitations:

- 2) Out-of-date → Retrieval Augmented Generation is an option to solve this....
- Prompt: create a table with top 10 countries by gdp with 2 columns: country name, gdp, population, and name of the president.

Country	GDP (Trillion USD)	Population (Millions)	President
United States	25.43	335.9	Joe Biden
China	14.72	1.428	Xi Jinping
Japan	4.25	125.8	Fumio Kishida
Germany	3.85	83.3	Olaf Scholz
India	3.41	1.417	Narendra Modi
United Kingdom	2.67	67.2	Rishi Sunak
France	2.63	67.1	Emmanuel Macron
Russia	2.24	144.5	Vladimir Putin
Canada	2.16	38.6	Justin Trudeau
Italy	2.04	60.4	Giorgia Meloni



The Hill
Keir Starmer, UK's new prime minister ...

Current Limitations:

- 3) Hallucination (No source) / answering what was not asked → RAG & fine-tunning models with internal information is an option to solve this....
- Prompt: create a table with top 10 countries by gdp with 2 columns: country name, gdp, population, and name of the president.

Table

Country	GDP (in trillion USD)	Population (in millions)	President/Head of State
United States	25.5	331	Joe Biden
China	17.7	1441	Xi Jinping
Japan	5.1	126	Fumio Kishida (Prime Minister)
Germany	4.2	83	Frank-Walter Steinmeier
India	3.5	1393	Droupadi Murmu
United Kingdom	3.2	67	Charles III (King)
France	3.1	67	Emmanuel Macron
Italy	2.1	60	Sergio Mattarella
Canada	2.0	38	Mary Simon
Brazil	1.9	213	Luiz Inácio Lula da Silva



Practice using IA - Electricity Trend

- Create a table with electricity consumption of USA, China, India and European Union by year from 1991 to 2023. Show all the rows.

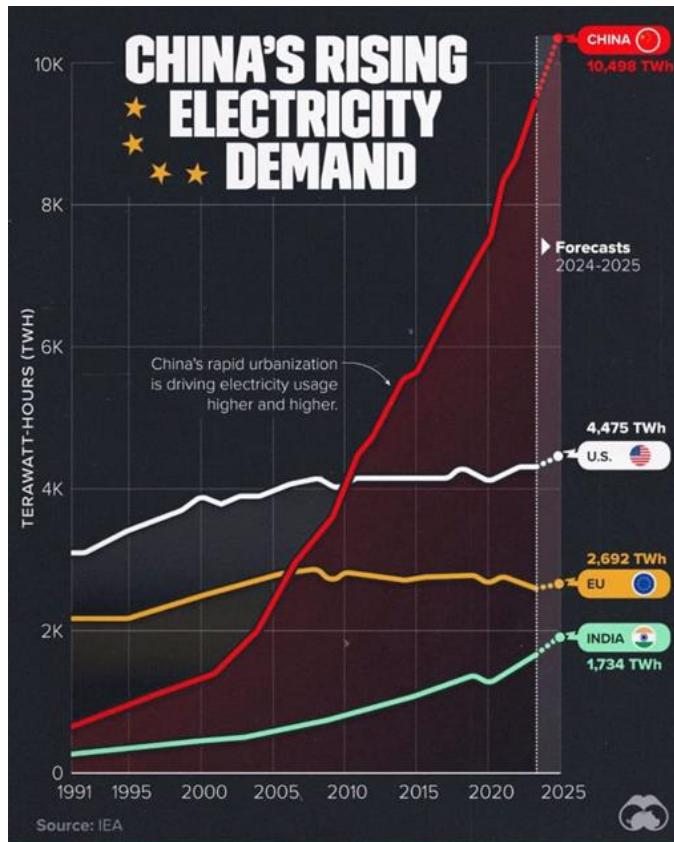
Year	USA (TWh)	China (TWh)	India (TWh)	European Union (TWh)
1991	3,195	710	225	1,582
1992	3,295	760	240	1,615
1993	3,395	815	255	1,650
1994	3,495	870	270	1,685
2021	6,195	2,730	675	2,630
2022	6,295	2,800	690	2,665
2023	6,395	2,870	705	2,700



Export to Sheets
<https://gemini.google.com/>

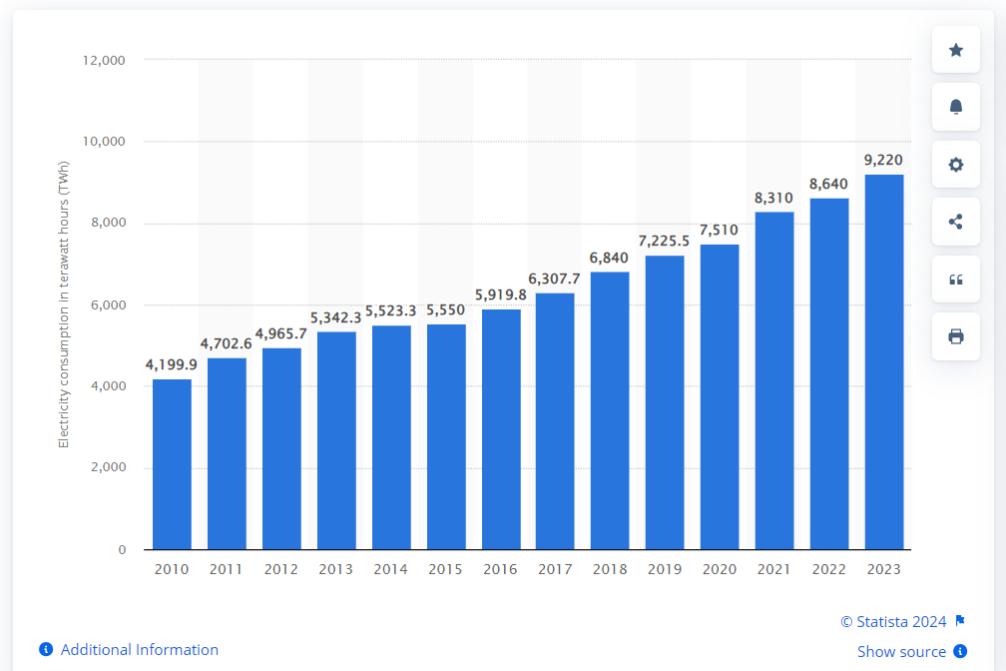
Electricity Trend

- REAL DATA:



Electricity consumption in China from 2010 to 2023

(in terawatt hours)

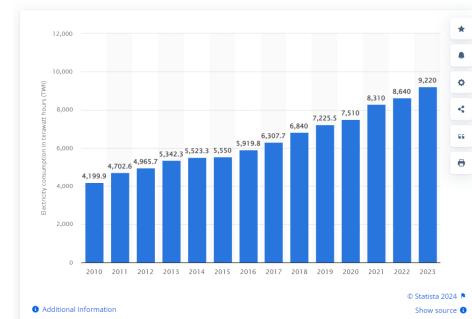


<https://www.statista.com/statistics/302203/china-electricity-consumption/>

Correct China data using the picture provided:

Electricity consumption in China from 2010 to 2023

(in terawatt hours)



2017	5,795	6,307.7	615	2,490
2018	5,895	6,840	630	2,525
2019	5,995	7,225.5	645	2,560
2020	6,095	7,510	660	2,595
2021	6,195	8,310	675	2,630
2022	6,295	8,640	690	2,665
2023	6,395	9,220	705	2,700

Export to Sheets



Expand the trend between 2024 to 2030



Projected Electricity Consumption Trends (2024-2030)

Note: These projections are based on current trends, government policies, and economic forecasts. Actual consumption may vary due to unforeseen factors.

Year	USA (TWh)	China (TWh)	India (TWh)	European Union (TWh)
2024	4,150	9,220 (based on image)	1,550	2,025
2025	4,200	9,500	1,600	2,050
2026	4,250	9,800	1,650	2,075
2027	4,300	10,100	1,700	2,100
2028	4,350	10,400	1,750	2,125
2029	4,400	10,700	1,800	2,150
2030	4,450	11,000	1,850	2,175



Export to Sheets

Exercise

- **Edit the shared Excel** naming a topic/industry/sector to select data to define the trend.
https://docs.google.com/spreadsheets/d/1_EMR9BBLJr-g_B1PnLjSZcSzvuuyHpgg/edit?usp=sharing&ouid=100949985522117931296&rtpof=true&sd=true
- Adapt the prompt for the topic/industry/sector of your choice. **Share your new prompt in the shared Excel.**
- Search the web to validate the data. Provide the chatbot with image(s) / or tables to fix the data.
- Ask the chatbot to include forecast between 2024 and 2030. Make a **last post** with the data/excel created.

**Training AI with internal data will
be the next big thing after the AI
hype of billions of parameters...**



Conclusion & Discussion

- Summary: Recap of key points.
- Future Outlook: What is the potential of AI in market trend analysis?

AI for Trends

One of the current limitations of AI for trend analysis is that AI not being able to see the most recent data. Techniques to Overcome Limitations:

Prompt Engineering

RAG

Real-time Data Integration

Incremental Learning

Hybrid Approaches

Social Media Monitoring

Time Series Analysis

Social Media Monitoring

Real-time Tracking:
Monitoring social
media for trends and
sentiment shifts.

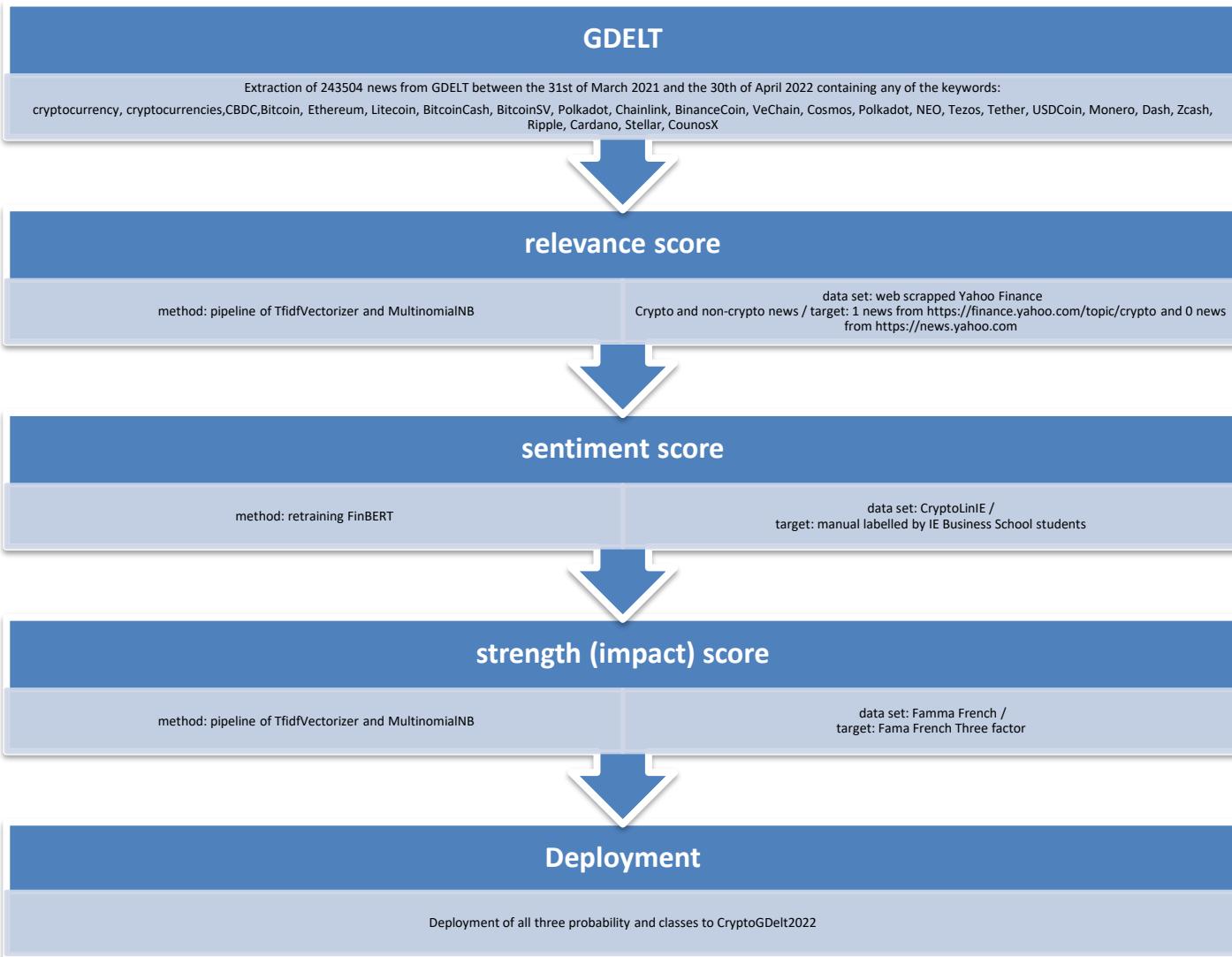
Influencer Analysis:
Identifying influential
individuals shaping
market trends.

Relevance, Sentiment, Impact (RSI) Framework

Relevance: Assessing the relevance of data points.

Sentiment: Analyzing sentiment in the data.

Impact: Evaluating the potential impact on the market.



<https://github.com/manoelgadi/CryptoGDelt2022>

BIBLIOGRAPHY

Bibliography

- The majority of the examples has been extracted from:

Matilla García M.; Pérez Pascual P.; Sanz Carnero B. (2013) *Econometría y Predicción* Mac Graw Hill.
ISBN: 978-84-481-8310-3)

Hyndman R. J; Athanasopoulos G (2018) *Forecasting: Principles and Practice* O Reilly

- Other sources:

Cowpertwait P. S. P. Metcalfe A. V. (2009) *Introductory Time Series with R* Springer ISBN 978-0-387-88698-5

Bibliography

- The majority of the examples has been extracted from:

Lewinson E. (2020) *Python for finance*. Cookbook Packt Publishing ISBN 978-78961-851-1
(Some notebook codes had to be corrected)

- Other sources:

Cowpertwait P. S. P. Metcalfe A. V. (2009) *Introductory Time Series with R* Springer ISBN 978-0-387-88698-5

Thank you very much



mfalonso@faculty.ie.edu



<https://github.com/manoelgadi>



<https://www.linkedin.com/in/phd-manoel-gadi-97821213/>

ANNEX

The Python language

- Take into account that both, Jupyter and Spyder use the same python language
- The usual way of work is: Firstly, the code is developed in spyder by statistician or data scientist or ... Secondly if a director wants to understand how the results are generated then it is a good idea to pass it to a Jupyter environment where we will be able to give more visual and understandable explanations with a technical touch
- Finally, when we make an analysis under Python code it is important to follow an order and to write it in the most readable way. Maybe other users could revise or use it. Consequently, it is mandatory to follow some general rules like the following ones:
<https://www.python.org/dev/peps/pep-0008/>

```
# Correct:
```

```
# Aligned with opening delimiter.
```

```
foo = long_function_name(var_one, var_two,  
                         var_three, var_four)
```

```
# Wrong:
```

```
# Arguments on first line forbidden when not using vertical alignment.  
foo = long_function_name(var_one, var_two,  
                         var_three, var_four)
```

```
my_list = [  
    1, 2, 3,  
    4, 5, 6,  
]  
result = some_function_that_takes_arguments(  
    'a', 'b', 'c',  
    'd', 'e', 'f',  
)
```

```
with open('/path/to/some/file/you/want/to/read') as file_1, \  
     open('/path/to/some/file/being/written', 'w') as file_2:  
    file_2.write(file_1.read())
```

```
income = (gross_wages  
          + taxable_interest  
          + (dividends - qualified_dividends)  
          - ira_deduction  
          - student_loan_interest)
```

TIME SERIES KEY CONCEPTS

What is a Time Series?

- A time series is a sequence of data points that occur in successive order over some period of time. It can be represented with the following expression:

$$\{Y_t\}_{1 \leq t \leq T}$$

- Exists 2 principal kinds of time series:
 - Time series with data equally spaced: annually, monthly, daily, ...
 - Time series with data not equally spaced: this kind of series often appears in data generated at level of minutes or seconds, where many data will be missing, and it is not possible to make missing – imputations

Here will be analyzed the time series of class 1

Traditional sources of time series

- By traditional sources of time series is referred to governmental time series:

Here in Spain:

Theme-based classification

A link to a ZIP file is provided for each theme, containing all of its time series plus an additional CSV file with the series catalogue. Users can also download a ZIP file with the time series of each subtheme. Please click on the theme to display its download links:

- Financial accounts ↗
- Financial corporations ↗
- General Government ↗
- Non-financial corporations ↗
- External statistics ↗
- Financial markets ↗
- General economic statistics ↗
- Interest rates ↗
- Exchange rates ↗

https://www.bde.es/webbde/en/estadis/infoest/descarga_series_temporales.html

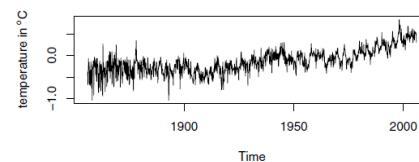
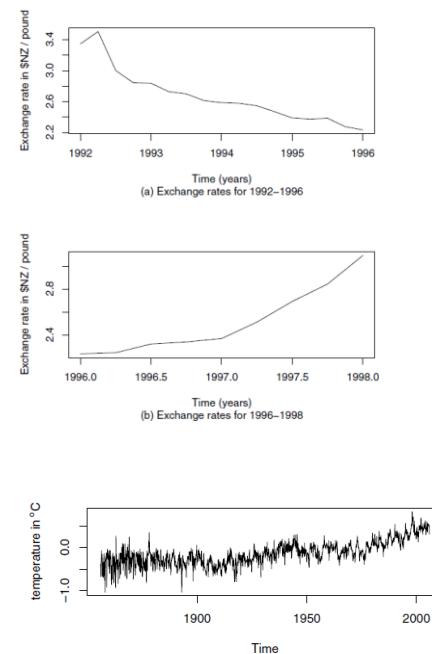
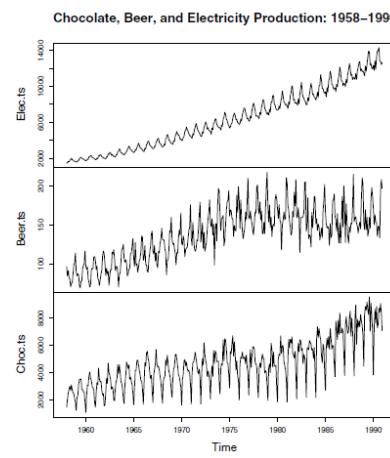
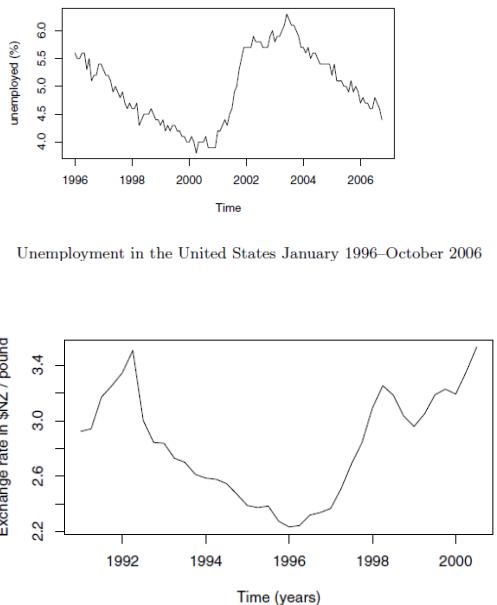
- Furthermore, time series can be found under international institutions:

<https://databank.worldbank.org/home.aspx>

The screenshot shows the DataBank website homepage. The top navigation bar includes links for 'DataBank Home', 'Databases', 'Create Report', 'Saved Reports', 'Saved Datasets', and 'Metadata Glossary'. The main content area features a heading 'Explore. Create. Share: Development Data' and a brief description of the tool's purpose. Below this are links for 'FAQs' and 'Feedback'. At the bottom, there are sections for 'Explore databases' and 'WHAT'S POPULAR'.

Examples of time series

- Time series can show very different patterns
- It is impossible to find general statistics rules for modelling time series
- Depend on the type of time series a theory or algorithm will be more or less appropriate



Stock and Flow series

- This classification is based on the dynamicity of the time series
- A stock series is a measure of certain attributes at a point in time and can be thought of as “stocktakes”. One example would be a *Monthly Labour Force Survey*. This time series is a stock measure because it takes stock of whether a person was employed in the reference week
- Flow series are ones which are a measure of activity over a given period
- The main difference between a stock and a Flow series is that Flow series can contain effects related to calendar (trading day effects). Both types of series can still be seasonally adjusted using the same seasonal adjustment process
- Many time series are seasonal adjusted. Seasonal adjustment is a process of estimating and then removing from a time series influences that are systematic and calendar related such as Easter. With this process avoid conceal both:
 - The true underlying movement
 - Certain non-seasonal characteristics which may be of interest to analysts

Main elements of a Time Series: Error, Trend and Seasonality

- The error component: it is unexplainable White noise
- The trend component: it is a sum of two sub-components called *level* or average value for a specific period of time and *growth* or average variation of the value over a period of time
- The seasonal component: is a pattern that repeats itself with a fixed periodicity

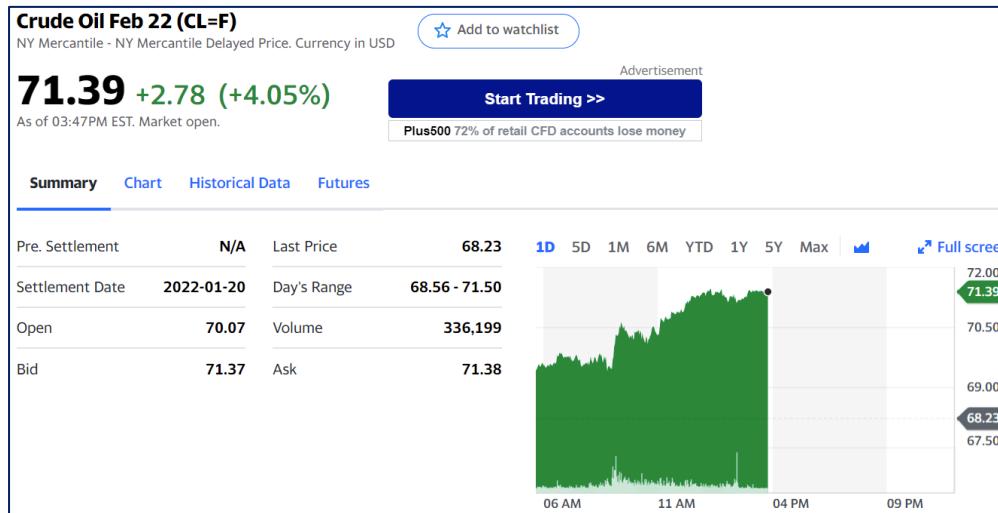
These 3 elements are the building blocks of the simplest time series model called as the ETS model, where time series are decomposed in its components, which are modelled and projected ahead in an additive or multiplicative way.

In Python these models can be built through the library *statsmodels* which offers in Python similar output to R

Our first time series “toy-model”: the additive ETS model

It is going to be associated a ETS time series model to data coming from yahoo finance data:

<https://finance.yahoo.com/quote/CL%3DF?p=CL%3DF>



DATA PRE-PROCESSING

Converting prices to returns

- Asset prices are usually non-stationary, mean and variance change over time
- By transforming the prices into returns, we attempt to make the time series stationary, which is the desired property in statistical modeling
- There are 2 types of returns:
 - **Simple returns:** They aggregate over assets; the simple return of a portfolio is the weighted sum of the returns of the individual assets in the portfolio. Simple returns are defined as:
$$R_t = (P_t - P_{t-1})/P_{t-1} = P_t/P_{t-1} - 1$$
 - **Log returns:** They aggregate over time; it is easier to understand with the help of an example—the log return for a given month is the sum of the log returns of the days within that month. Log returns are defined as:
$$r_t = \log(P_t/P_{t-1}) = \log(P_t) - \log(P_{t-1})$$
- There is not much difference between simple and log returns for daily and intraday data, the general rule is that log returns are smaller than simple ones

Converting prices to returns in practice

- From a time series perspective, the conversion may be done with a very simple python instruction for both cases:

```
df = yf.download('AAPL',
                 start='2000-01-01',
                 end='2010-12-31',
                 progress=False)

df = df.loc[:, ['Adj Close']]
df.rename(columns={'Adj Close':'adj_close'}, inplace=True)
```

A normalized name of variable is used

This pandas function allows generate simple returns directly

```
df['simple rtn'] = df.adj_close.pct_change()
df['log rtn'] = np.log(df.adj_close / df.adj_close.shift(1))
```

With return generation, a row in the data is lost. In this case, the first one

In [18]:	df.head()
Out[18]:	
	adj_close simple rtn log rtn
	Date
	1999-12-31 0.787035 → NaN NaN
	2000-01-03 0.856887 0.088754 0.085033
	2000-01-04 0.784643 -0.084310 -0.088078
	2000-01-05 0.796124 0.014633 0.014527
	2000-01-06 0.727229 -0.086538 -0.090514

As can be seen, a normalized name of variable allows that Python methods can be applied to columns

Inflation corrections

- Inflation effects are not always constant over time, and it must be eliminated in time series analysis
- In the other hand, returns transformation not only convert a series in stationary, it shows information about price dynamic that has to be conserved
- In order to manage above issues, the following transformation is recommended using an adequate price index

$$R_t^r = \frac{1 + R_t}{1 + \pi_t} - 1$$

Here, R_t is a time t simple return and π_t is the inflation rate

Inflation corrections in practice

- An interesting exercise would be to create a python function which generate this correction automatically from initial data

```
df_all_dates = pd.DataFrame(index=pd.date_range(start='1999-12-31',
                                                end='2010-12-31'))
df = df_all_dates.join(df[['adj_close']], how='left') \
    .fillna(method='ffill') \
    .asfreq('M')
```

Above data are converted in a data.frame with date and value (adj_close)

```
df_cpi = quandl.get(dataset='RATEINF/CPI_USA',
                     start_date='1999-12-01',
                     end_date='2010-12-31')
df_cpi.rename(columns={'Value':'cpi'}, inplace=True)
```

Price index is downloaded to be merged with all data

```
df_merged = df.join(df_cpi, how='left')
```

```
df_merged['simple rtn'] = df_merged.adj_close.pct_change()
df_merged['inflation_rate'] = df_merged.cpi.pct_change()
```

Finally, correction Price index formula is applied:

```
df_merged['real rtn'] = (df_merged.simple rtn + 1) / (df_merged.inflation_rate + 1) - 1
```

Generating volatilities

- Target: To calculate the monthly realized volatilities for an asset using daily returns and then annualize these values
- Formula for computing *realized volatilities*:

$$RV = \sqrt{\sum_{i=1}^T r_t^2}$$

- RVs is frequently used for daily volatility using the intraday returns:
 - Calculate daily log returns
 - Calculate the realized volatility over the months
 - Annualize the values

Generating volatilities in practice

- With the knowledge adquired up to now, it is possible to follow above steps:

```
import pandas as pd
import yfinance as yf

# download data
df = yf.download('AAPL',
                 start='2000-01-01',
                 end='2010-12-31',
                 auto_adjust=False,
                 progress=False)

# keep only the adjusted close price
df = df.loc[:, ['Adj Close']]
df.rename(columns={'Adj Close': 'adj_close'}, inplace=True)

# calculate simple returns
df['log rtn'] = np.log(df.adj_close / df.adj_close.shift(1))

# remove redundant data
df.drop('adj_close', axis=1, inplace=True)
df.dropna(axis=0, inplace=True)

df.head()
```

The use of Python function allows to simplify code and it make it more readable

```
def realized_volatility(x):
    return np.sqrt(np.sum(x**2))
```

apply() allows an efficient way to map a Python function

```
df_rv = df.groupby(pd.Grouper(freq='M')).apply(realized_volatility)
df_rv.rename(columns={'log rtn': 'rv'}, inplace=True)
```

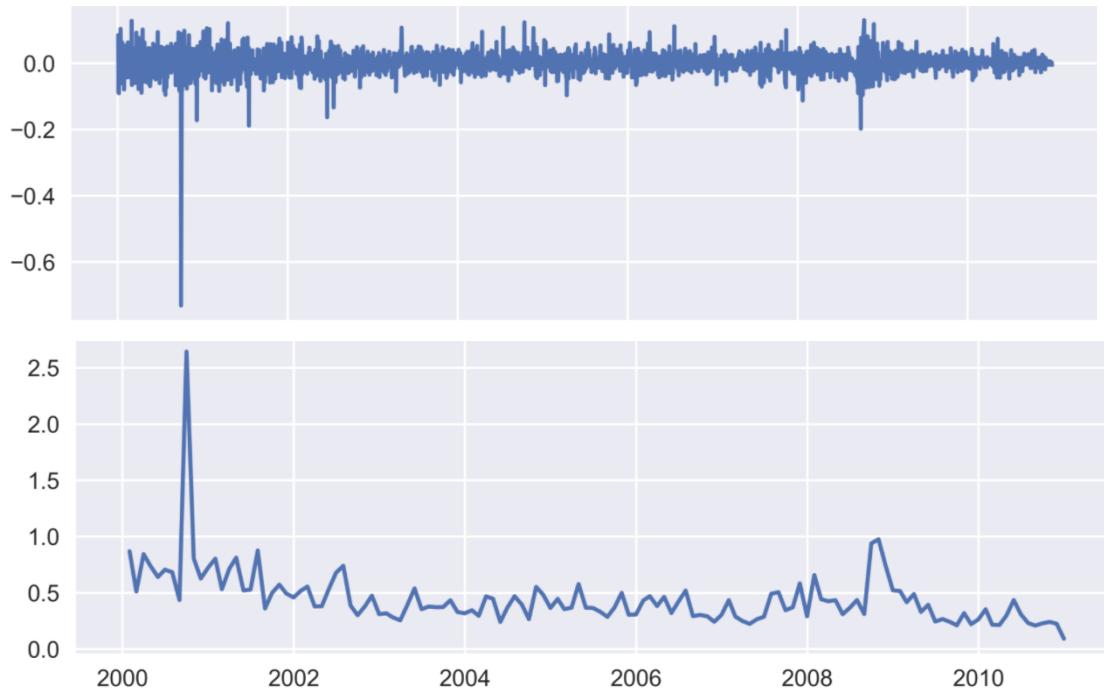
```
df_rv.rv = df_rv.rv * np.sqrt(12)
```

Generating volatilities: visualizations

- Note that we have generated monthly & annualized volatilities

```
fig, ax = plt.subplots(2, 1, sharex=True)
ax[0].plot(df)
ax[1].plot(df_rv)

# plt.tight_layout()
# plt.savefig('images/ch1_im6.png')
plt.show()
```



Questions: Do you think that volatilities during next period were higher or lower? Do a quick verification using spyder What happened in 2001? Draw a time series analysis using that year

Identifying outliers

- There are many methods for identifying anomalous data. Here the simplest will be shown
- After the identification a correction has to be done. Usually, these correction consist of capping or flooring the time series
- The simplest method for outliers-identification is called the 3-sigma method. According to this method, it is detected as outliers all data whose value is higher than:

$$\mu \pm 3 \cdot \sigma$$

- In this kind of analysis is important to make representation in order to see which data would be selected to be modified
- As can be shown, the 3.sigma method is applied dynamically, it is said that after a period (here 20 days), it is analyzed if the following element is higher or lower to the above value

Identifying outliers in practice

- In order to apply this method some simple visualization elements are going to be used here

```
df = yf.download('AAPL',
                 start='2000-01-01',
                 end='2010-12-31',
                 progress=False)

df = df.loc[:, ['Adj Close']]
df.rename(columns={'Adj Close': 'adj_close'}, inplace=True)

df['simple rtn'] = df.adj_close.pct_change()
```

```
def identify_outliers(row, n_sigmas=3):
    ...
    Function for identifying the outliers using the 3 sigma rule.
    The row must contain the following columns/indices: simple_rtn, mean, std.

    Parameters
    -----
    row : pd.Series
        A row of a pd.DataFrame, over which the function can be applied.
    n_sigmas : int
        The number of standard deviations above/below the mean - used for detecting outliers

    Returns
    -----
    0/1 : int
        An integer with 1 indicating an outlier and 0 otherwise.
    ...

x = row['simple_rtn']
mu = row['mean']
sigma = row['std']

if (x > mu + n_sigmas * sigma) | (x < mu - n_sigmas * sigma):
    return 1
else:
    return 0
```

20 days before are considered to compute if a element is or not anomalous

```
df_rolling = df[['simple_rtn']].rolling(window=21) \
    .agg(['mean', 'std'])
df_rolling.columns = df_rolling.columns.droplevel()
```

2. Join the rolling metrics to the original data:

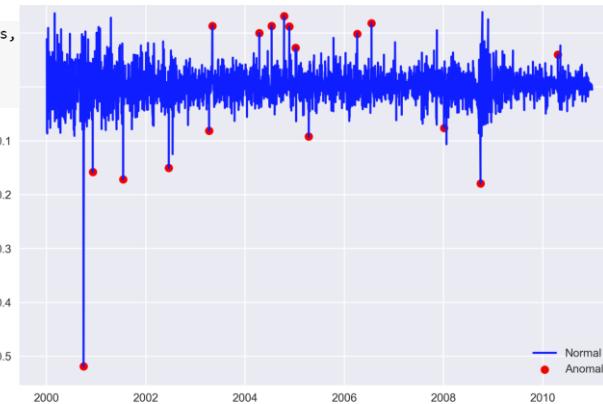
```
df_outliers = df.join(df_rolling)
```

```
df_outliers['outlier'] = df_outliers.apply(identify_outliers,
                                             axis=1)
outliers = df_outliers.loc[df_outliers['outlier'] == 1,
                           ['simple_rtn']]
```

```
fig, ax = plt.subplots()

ax.plot(df_outliers.index, df_outliers.simple_rtn,
        color='blue', label='Normal')
ax.scatter(outliers.index, outliers.simple_rtn,
           color='red', label='Anomaly')
ax.set_title("Apple's stock returns")
ax.legend(loc='lower right')

# plt.tight_layout()
# plt.savefig('images/ch1_im9.png')
plt.show()
```



Normal distribution of returns

- Some hypothesis are often supposed over the behavior of returns
- Depend on the class of time series model to be used, some hypothesis can be more relevant than other
- One usual hypothesis is that returns have to follow a normal distribution. In this case, some known test has to be applied
- One relevant fact is: when this kind of analysis is applied, time structure is broken and all data are analyzed without ordering. In this case histograms and another diagrams are key tools to be used

Normal distribution of returns in practice I

- Some graph analyses are developed

```
import pandas as pd
import numpy as np
import yfinance as yf
import seaborn as sns
import scipy.stats as scs
import statsmodels.api as sm
import statsmodels.tsa.api as smt
```

2. Download the S&P 500 data and calculate the returns:

```
df = yf.download('^GSPC',
                 start='1985-01-01',
                 end='2018-12-31',
                 progress=False)

df[['Adj Close']].rename(columns={'Adj Close': 'adj_close'})
df['log_rtn'] = np.log(df.adj_close/df.adj_close.shift(1))
df = df[['adj_close', 'log_rtn']].dropna(how = 'any')
```

1. Calculate the Normal PDF using the mean and standard deviation of the observed returns:

```
r_range = np.linspace(min(df.log_rtn), max(df.log_rtn), num=1000)
mu = df.log_rtn.mean()
sigma = df.log_rtn.std()
norm_pdf = scs.norm.pdf(r_range, loc=mu, scale=sigma)
```

2. Plot the histogram and the Q-Q Plot:

```
fig, ax = plt.subplots(1, 2, figsize=(16, 8))

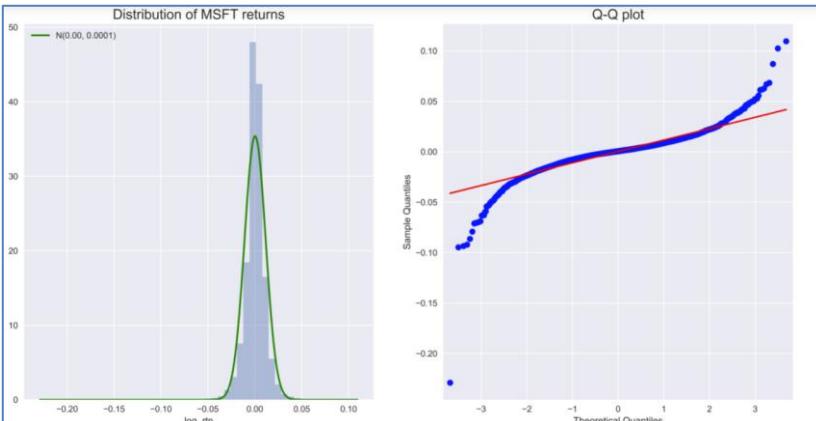
# histogram
sns.distplot(df.log_rtn, kde=False, norm_hist=True, ax=ax[0])
ax[0].set_title('Distribution of MSFT returns', fontsize=16)
ax[0].plot(r_range, norm_pdf, 'g', lw=2,
           label=f'N({mu:.2f}, {sigma**2:.4f})')
ax[0].legend(loc='upper left');

# Q-Q plot
qq = sm.qqplot(df.log_rtn.values, line='s', ax=ax[1])
ax[1].set_title('Q-Q plot', fontsize = 16)

# plt.tight_layout()
# plt.savefig('images/ch1_im10.png')
plt.show()
```

Normal distribution of returns in practice II

Questions: Does this distribution follow a normal one?



```
jb_test = scs.jarque_bera(df.log rtn.values)

print('----- Descriptive Statistics -----')
print('Range of dates:', min(df.index.date), '-', max(df.index.date))
print('Number of observations:', df.shape[0])
print(f'Mean: {df.log rtn.mean():.4f}')
print(f'Median: {df.log rtn.median():.4f}')
print(f'Min: {df.log rtn.min():.4f}')
print(f'Max: {df.log rtn.max():.4f}')
print(f'Standard Deviation: {df.log rtn.std():.4f}')
print(f'Skewness: {df.log rtn.skew():.4f}')
print(f'Kurtosis: {df.log rtn.kurtosis():.4f}')
print(f'Jarque-Bera statistic: {jb_test[0]:.2f} with p-value: {jb_test[1]:.2f}')
```

Jarque – Bera test is a classical one which allows through a number to test if a distribution of data follows a theoretic normal distribution

```
----- Descriptive Statistics -----
Range of dates: 1985-01-02 - 2018-12-28
Number of observations: 8569
Mean: 0.0003
Median: 0.0006
Min: -0.2290
Max: 0.1096
Standard Deviation: 0.0113
Skewness: -1.2624
Kurtosis: 28.0111
Jarque-Bera statistic: 282076.61 with p-value: 0.00
```

Residual auto-correlation

- A time series model should detect all pattern in data
- If any pattern exists among data non detected by the model, it is usual that residual shows structure along the time
- Those residual structures can be detected by the total and partial autocorrelations that will be analyzed in a posterior chapter
- To accomplish with hypothesis of a stationary time series, non-total and any partial autocorrelation has to exist

$$E(\hat{\rho}_u) = 0$$

$$\text{Var}(\hat{\rho}_u) = \frac{1}{T}$$

$$\text{cov}(\hat{\rho}_u; \hat{\rho}_{u+h}) = 0 \quad \forall h \neq 0$$

$$S_t = \varphi_1^1 S_{t-1}$$

$$S_t = \varphi_1^2 S_{t-1} + \varphi_2^2 S_{t-2}$$

...

$$S_t = \varphi_1^k S_{t-1} + \varphi_2^k S_{t-2} + \cdots + \varphi_k^k S_{t-k}$$

$$\text{cov}(\varphi_i^i; \varphi_{i+h}^{i+h}) = 0 \quad \forall h \neq 0$$

Residual auto-correlation in practice I

- In spite of the long formulas for total and partial autocorrelation. Python allows their computation with two very simple formulas: *acf* and *pacf*

```
N_LAGS = 50
SIGNIFICANCE_LEVEL = 0.05

2. Run the following code to create ACF plot of log returns:

acf = smt.graphics.plot_acf(df.log rtn,
                             lags=N_LAGS,
                             alpha=SIGNIFICANCE_LEVEL)

# plt.tight_layout()
# plt.savefig('images/ch1_im13.png')
plt.show()
```



Residual auto-correlation in practice II

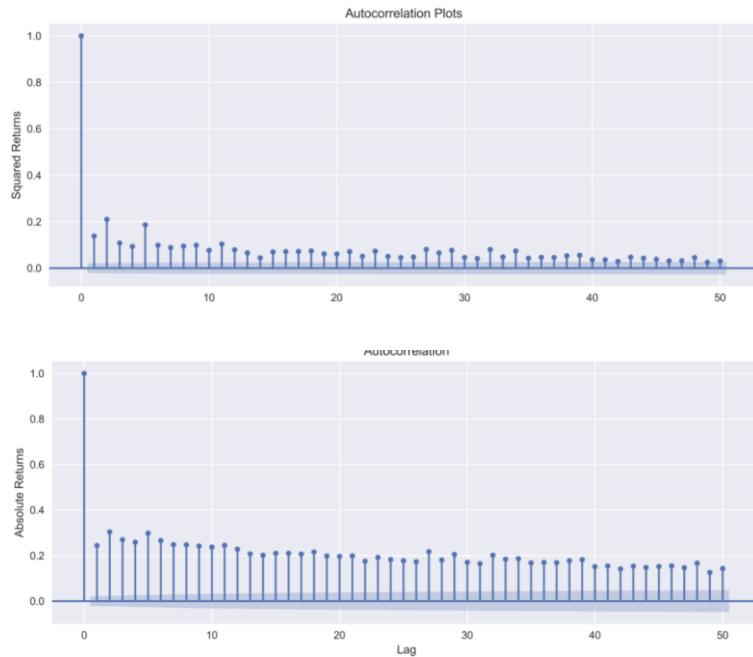
- However, if residual are powered to 2, a curious effect happens with the correlation

```
fig, ax = plt.subplots(2, 1, figsize=(12, 10))

smt.graphics.plot_acf(df.log rtn ** 2, lags=N_LAGS,
                      alpha=SIGNIFICANCE_LEVEL, ax = ax[0])
ax[0].set(title='Autocorrelation Plots',
           ylabel='Squared Returns')

smt.graphics.plot_acf(np.abs(df.log rtn), lags=N_LAGS,
                      alpha=SIGNIFICANCE_LEVEL, ax = ax[1])
ax[1].set(ylabel='Absolute Returns',
           xlabel='Lag')

# plt.tight_layout()
# plt.savefig('images/ch1_im14.png')
plt.show()
```



Questions: What is happening? Which concept is squared residual related with?

SIMPLE TIME SERIES VISUALIZATIONS

Plain representations with *plotly* a powerful interactive library

- Plotly allows powerful and simply time series representations. But this library allows a huge graph variety

<https://plotly.com/python/>

Plotly Python Open Source Graphing Library



Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

- Nowadays, its installation in Python is easy with *pip install*

```
import cufflinks as cf
from plotly.offline import iplot, init_notebook_mode

# set up settings (run it once)
cf.set_config_file(world_readable=True, theme='pearl',
                   offline=True)

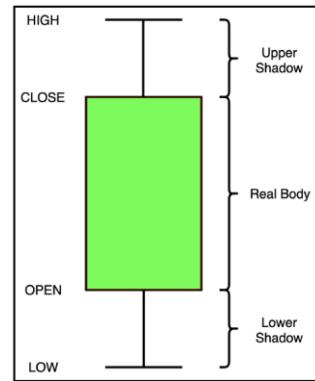
# initialize notebook display
init_notebook_mode()
```

```
!!!!!! You must apply: pip install chart_studio
df.iplot(subplots=True, shape=(3,1), shared_xaxes=True, title='MSFT time series')
```



Candlestick chart

- It is a chart used to describe the movements of a given security's Price
- It combines daily the OHLC information: Open, high, low and close
- A candlestick chart can be bullish if *close prize > open prize*



- On the contrary, if *open prize > close prize*, the candlestick will be a bearish
- This kind of graph is used often by traders in their platform

Properties of Candlestick chart

- Candlestick dates back to 18th century Japanese rice traders
- This chart offers more information than traditional *open-high, low-close* bars or simple lines that connect the dots of *closing prices*
- Candlesticks build pattern that predict *Price direction* once completed. Proper color, as can be seen, adds depth to this colorful technical tool
- There are various candlestick pattern used to determine Price direction and momentum, including three-line strike, two black gapping, three black crowns, evening star and abandoned baby

Candlestick chart: some patterns

- Three line Strike (see <https://www.investopedia.com/articles/active-trading/092315/5-most-powerful-candlestick-patterns.asp>)



Each bar posts a lower low and closes near the intrabar low. The 4th bar opens even lower but reverses in a wide-range outside bar that closes above the high of the first candle in the series. According to Bulkovski, this reversal predict higher prices with a 83% accuracy rate

- Two black gapping



The bearish two black gapping appears after a notable top in an uptrend, with a gap down that yields 2 black bars posting lower lows. This pattern predict that the decline will continue to even lower lows, it predict lower prices with a 68% accuracy rate

Candlestick chart in practice

- The Python implementation of this graph is totally direct

```
import pandas as pd
import yfinance as yf
```

2. Download the adjusted prices from Yahoo Finance:

```
df_twtr = yf.download('TWTR',
                      start='2018-01-01',
                      end='2018-12-31',
                      progress=False,
                      auto_adjust=True)
```



```
import cufflinks as cf
from plotly.offline import iplot, init_notebook_mode
```

```
init_notebook_mode()
```

2. Create the candlestick chart using Twitter's stock prices:

```
qf = cf.QuantFig(df_twtr, title="Twitter's Stock Price",
                  legend='top', name='TWTR')
```

3. Add volume and moving averages to the figure:

```
qf.add_volume()
qf.add_sma(periods=20, column='Close', color='red')
qf.add_ema(periods=20, color='green')
```

Here two additional lines have been added using *close prices*. *Simple Moving Average* and *Exponential Moving Average* using 20 past to compute these averages