End-Of-Year Project Report

Comparison of 3 Algorithmic Trading Strategies on the MetaQuotes 5 Trading Platform

Prepared By :

Aymen Tlili

Assigned to Group :

2nd year IDSD 1

Academic Term:

2021-2022

- ❖ Supervising Teacher: Bassem Ben Hamed
- ❖ Examining Teacher : Mohamed Neji

Dedicated Thanks

I would like to begin this Report by Thanking everyone who helped make this possible and allowed it to come Fruitation.

These outstanding people have been pillars to this project's success and it's growth from a simple Idea that had to be re-iterated multiple times into a full-on industry scale applicable product.

These outstanding individuals believed in me and my vision and gave me ample opportunity and chance to explore my own limitations and push the boundaries of my own learning while maintaining a healthy environment to communicate and work together.

I have failed more than anyone in my promotion and took on "risky projects" in hopes to fine-tune my own estimations and push myself to see what's possible and what's not and every time I failed or felt insecure these individuals have been there for me .

In this regard , I would like to Thank Mr.Bassem Ben Hamed ,professor in Applied Mathematics at ENET'Com,Co-founder & Data Scientist at DataCamp Training & Consulting, Senior Instructor Huawei Artificial Intelligence,Instructor CDOSS Machine Learning and Deep Learning for his Patience with me , his openness to new project Ideas and for always driving me to find solutions best suited to my problem and taking a fundamental approach in seeing things . As caring as Mr Bassem is , he always sought to make us adopt a "Do it Yourself Approach" while balancing the usage of ready software when needed.

I would like to Thank my Family , for lifting up the burdens of life and easing me into adulthood by monitoring me closely and making sure I was never overwhelmed physically and emotionally and always balanced life and work properly. Specifically , my brother being the person to introduce me to Trading platforms and showing me how the world market connected all into a single network of relationships and currency values agreed upon wherever man has set foot and for picking my curiosity into this domain.Also my Father , who was a senior professor in Economical Geography that studied the movement of wealth across countries for his great storytelling and linking events seamlessly.

I would like to thank all the Article writers, Book authors and publishers and online Forum discussion members for making sure their concepts were understandable , reproducible and Open-Source .I'm most grateful to belong to a community of programmers where everyone develops their solution and shares it with the world because while it's crucial for learning to do it yourself ,it's also an (off-policy method) of learning from Others . much like civilizations learning from older historians .

Preface :Choice of the Project

This Project was meant to be a stepping stone for something bigger, an entire product that culminates in efficient trading strategies that Econmics experts can use .

This project served to create a framework that didn't necessarily focus on making the best models but of just making things work . Connecting the pieces together and building a small but effective pipeline hosted on my local machine in hopes of amending the lack of experience that we encounter as academics when making our projects where we focus on applying what was seen in class but not how it would turn out in the outside world in the hands of a user who shouldn't be peaking "under the hood'

I had chosen this project because it merged several key aspects :
Database management , Handling time-indexed data , combining traditional ML and Deep learning models with subgroups of the supervised ML models, automated bots through scripts in a fast changed environment and thus Key Aspects like Scalability , incremental change and Dynamic programming all became the essence of this project.

The MetaQuotes platform served as a great simple interface to send orders with a simple GUI ,C programming Language support, MQL being their own Language and several connectors to outside API like those of Pthon .

All in all , I started this project because it sounded fun and it was something I was keen on exploring for a long time .

Table Of Contents :

Technologies used :

IDE: jupyter notebook , VS Code , MS word, notepad++

Database Management : InfluxDB , MySQL

API: Weights & Biases sweeps, MetaApi , Tensorflow ,Google Drive Synching

Python 3.6 and dependencies mentioned in requirements.txt

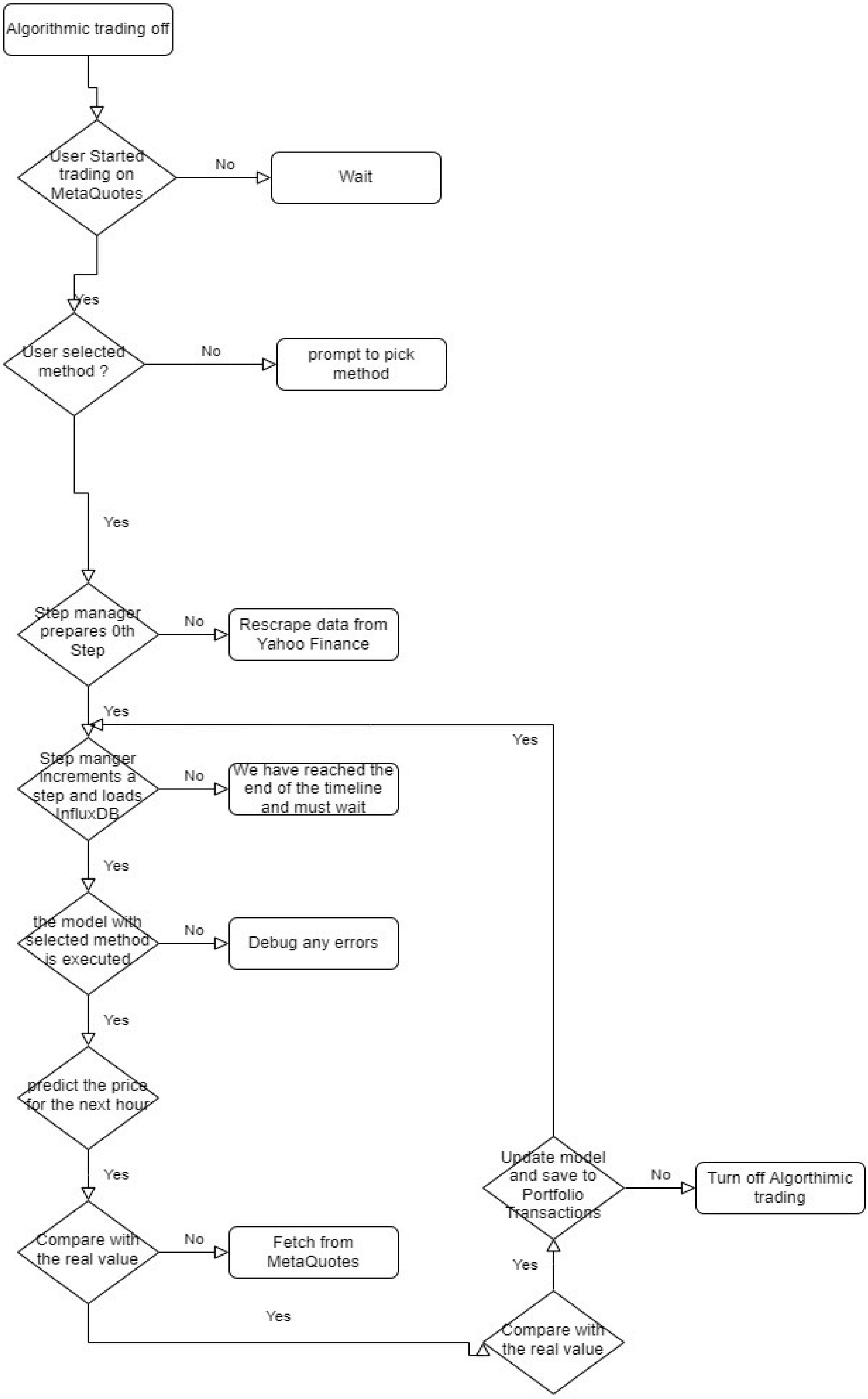
Go syntax for data Query in influxDB

MetaQuotes 5 trading platform

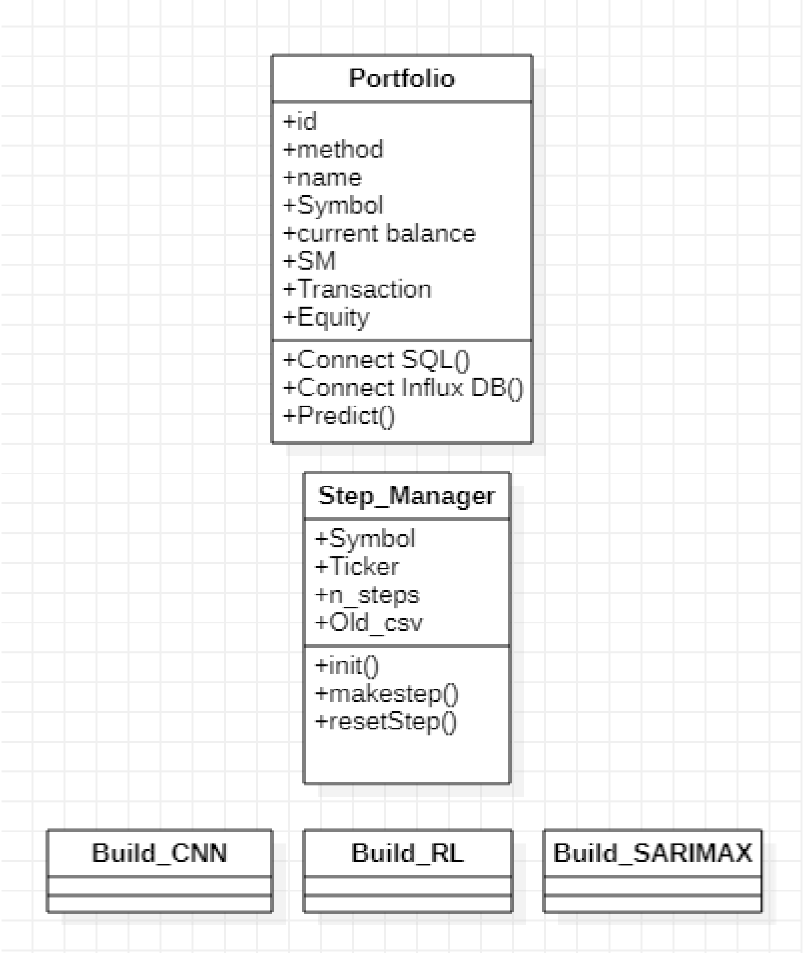
Chapter 1 :
Project Preparation and Pipeline Outline

1)UML Design and system architecture :

--Activity Flowchart


Algorithmic trading off
User Started trading on MetaQuotes
No
Wait
Yes
User selected method ?
No
prompt to pick method
Yes
Step manager prepares 0th Step
No
Rescrape data from Yahoo Finance
Yes
Yes
Step manger increments a step and loads InfluxDB
No
We have reached the end of the timeline and must wait
Yes
the model with selected method is executed
No
Debug any errors
Yes
predict the price for the next hour
Yes
Compare with the real value
No
Fetch from MetaQuotes
Yes
Compare with the real value
Yes
Update model and save to Portfolio Transactions
No
Turn off Algorthimic trading

- Class Hierarchy



The association in this Class diagram have not been filled yet since the interactions will depend heavily on Apache Airflow API and thus we use this just to mention the attributes and operations wish to execute with said classes that will be automated and done repeatedly at each step increment with saving to the appropriate database per mentioned the above flowchart .

The design concept is periodic and will run every hour to perform necessary updates to maintain scalability and not diverge from wanted performance .

We also use the Step manager atomically to create the 0^{th}-step or the starting point and digress from there , it's useful to remember that the incremental steps have to start somewhere and in this case it's 11-01 midnight for all 3 pairs of currency exchange rates to be studied.

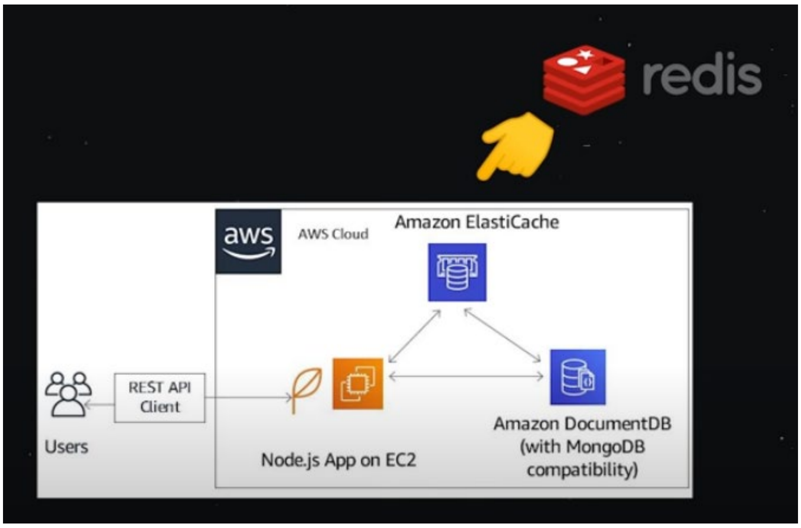
2)Setting up the environment :

a) introduction to inflexDB and database paradigms:

1)Key-Value paradigm : redis, memcached, Fcd

Simple interactions Read and write key value pairs using SET and GET commands

For redis and memcached : all the data is held in the machine's memory as opposed to other Databses that keep it in the disc(HDD or other) limiting the amount of data we can stor but makes the data access extremely fast with the tradeoff of no queries or joins preferred for caching and Leaderboards



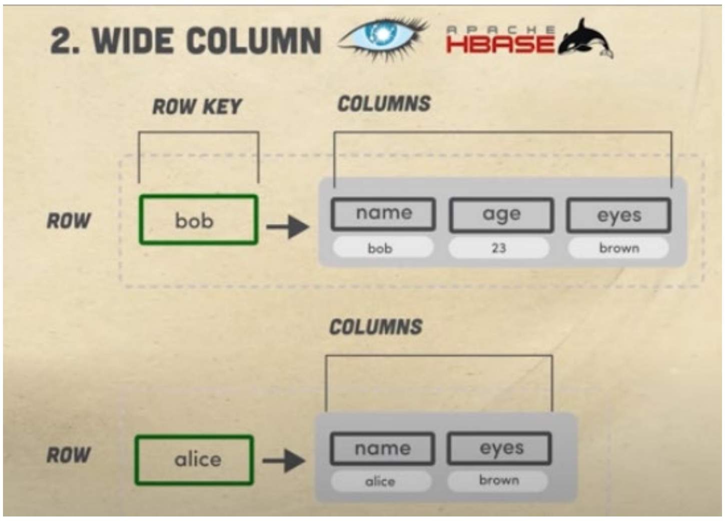
2)wide_column databases:Cassandra , HBase

Taking a key-value database and adding a second dimension to it with the outer dimension holding a key space which will hold column families and each column family is a set of ordered rows.tradeoff : no scheme so it can't handle unstructured data (CQL) easy to scale up and replicate across multiple nodes. Decentralized and can scale horizontally

Best for : Time-Series

Historical Records

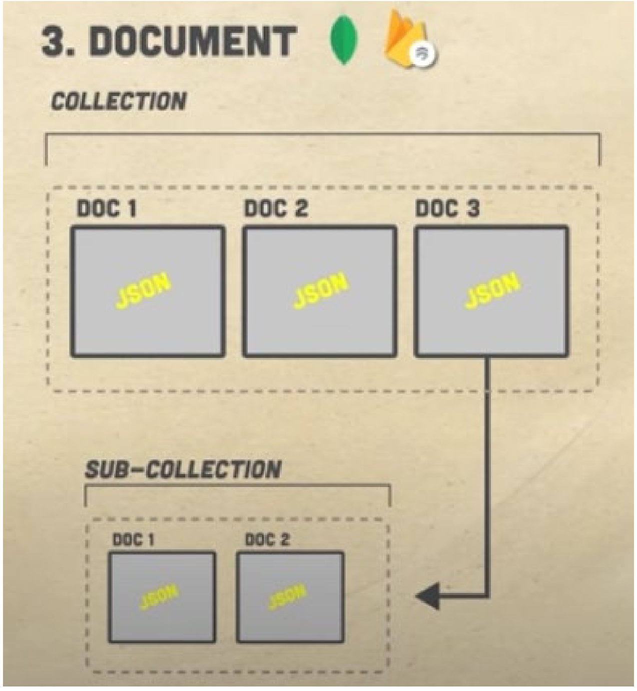
High write, low read



3) Document Oriented Database : MongoDB, firestore

We have documents , each document is a container for Key value Pairs , they are unstructured and don't require a schema . The documents are grouped together in schemas. Fields within a collection can be indexed and collections can be organized into a logical hierarchy allowing relational data retrieval . read and writes are denormalized and are more general purposed that the last paradigms .

They are easy to use for apps/ games but fail to represent graphs and connections updated often

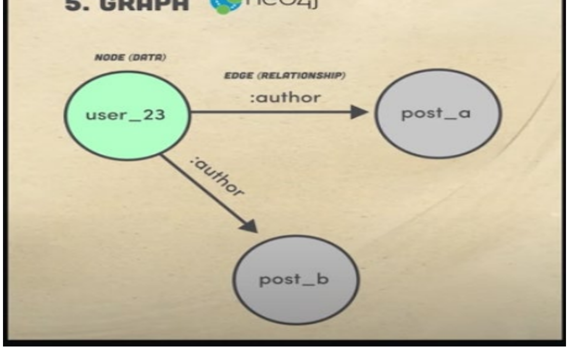


4)Relational database : MySQL , Postgres

Invented by Codd who backed up the theory with Maths and enough theory to make the paradigm last for over 50 years and was the reason for the creation of SQL that enabled fast joins and queries. It's compared to a warehouse of parts with IDs and the relations of parts connect to create the Query organizing data in it's smallest potential form .This requires a schema of the data base and they are ACID:atomicity , consistency ,isolation and durability . this maintains data and safeguards it but at the cost of scalability . they are not good for unstructured data .

5) Graph Database :

The relation itself is an of a graph . this makes setting up a many to many relationship quiet easy and avoids joins and middle man tables . the performance is much better and has clear syntax . Graph databases can be alternatives to SQL if we have Graph of social networks,fraud detection or Recommendation systems

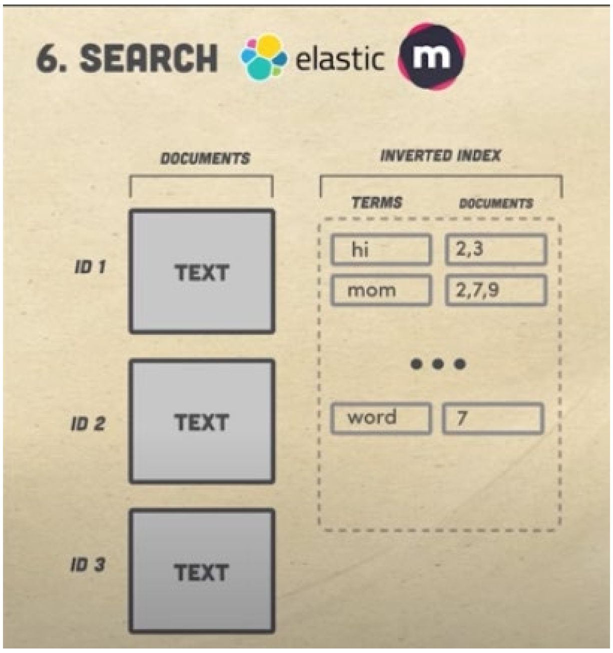


6)Full text search engines :

A user provides a small amount of text and the DB must return the most relevant results ranked in the proper order from a huge amount of data . most of the databases here are based on top of the apache Lucere project

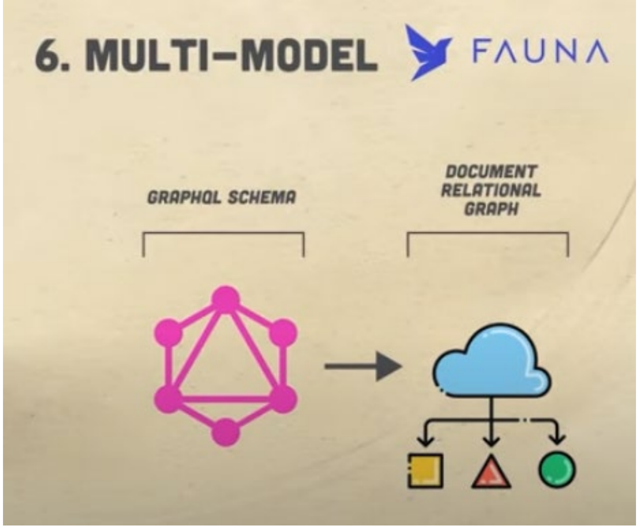
With cloud based options like algolia , MeiliSearch .

Very similer to document based , we start with an index and we add data objects to it . the difference is databse will analyze all the text in the document and creat an index of the searchable terms . when querying , only scan the index making it very fast



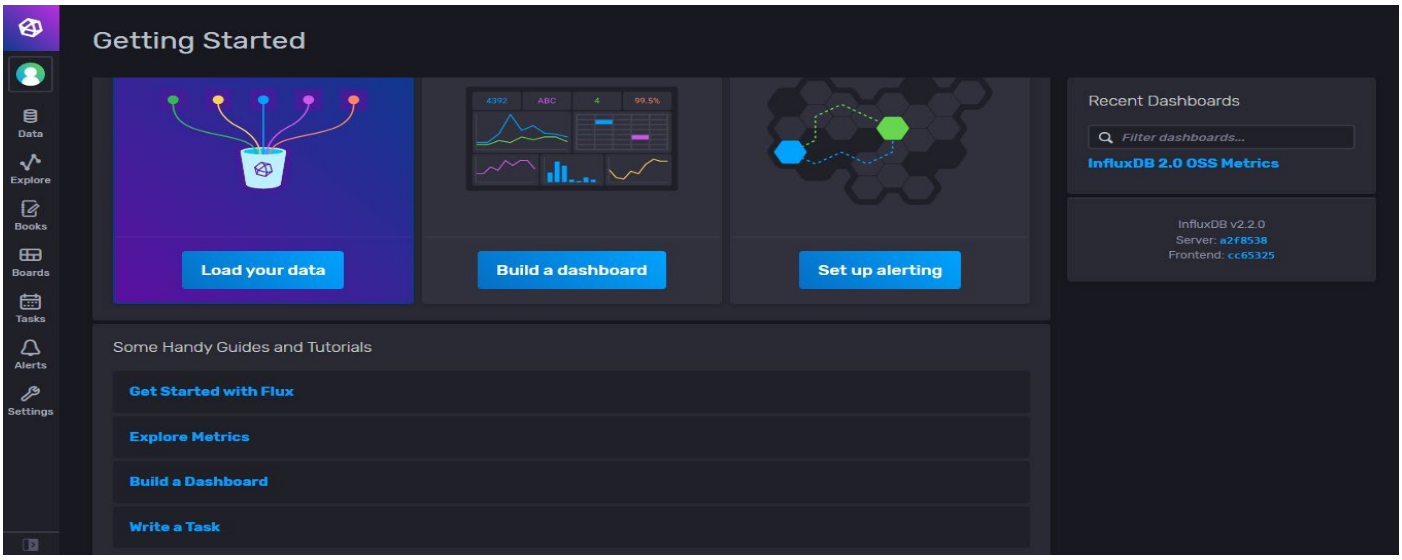
7) Multi-Model database :

Front-End developers care about the data they can consume , a json where schemas and data replication is applied or sharding . we only describe how we want to access the data using GraphQL an example is a user and post model where auser can have multiple posts . the DB automatically created indexed documents and determining how to best use the known paradigms to handle the data input in the backend surcomventing all the obstavles

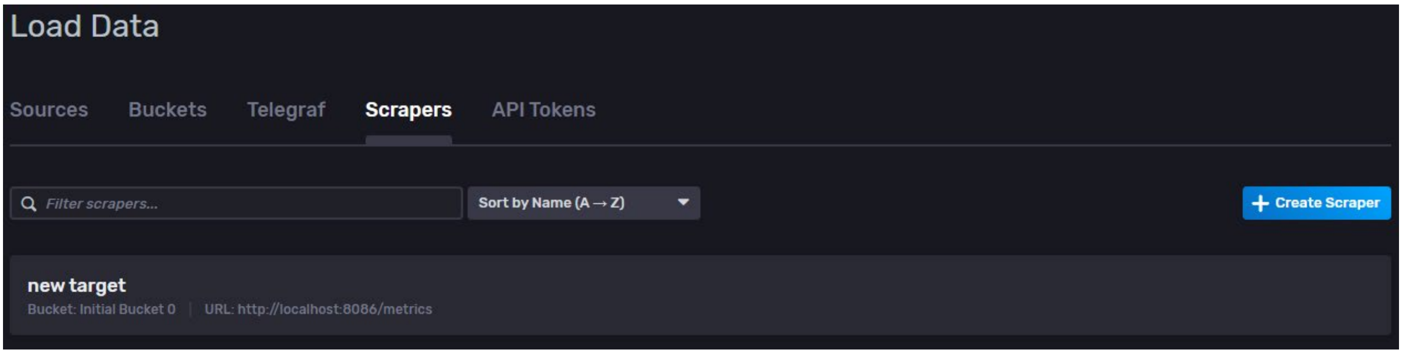


Influx DB : A Time series DataBase

Influx DB allows us to create scalable analytics quickly and to configure scrapers and backets to contain the results of scheduled tasks . the results of our steps and CSV files made along the way will be stored in this database



We can manage multiple backets and control the target input and output of each scraper we defined , here we see it ointing to a metrics source that exists on my local machine but can be retargeted to yahooFinance's API and scraped csv's



new target
Bucket: Initial Bucket 0 | URL: http://localhost:8086/metrics

I relied in my work on this documentation to advance my queries , while written in Go-Like language , the statements themselves are pretty straightforward and similer to SQL in syntax.
Herein are the refrenced operations :
https://docs.influxdata.com/influxdb/v1.8/query_language/explore-data/

The Basics:	Configure Query Results:	General Tips on Query Syntax:
The SELECT statement	ORDER BY time DESC	Time Syntax
The WHERE clause	The LIMIT and SLIMIT clauses	Regular Expressions
The GROUP BY clause	The OFFSET and SOFFSET clauses	Data types and cast operations
The INTO clause	The Time Zone clause	Merge behavior
		Multiple statements
		Subqueries

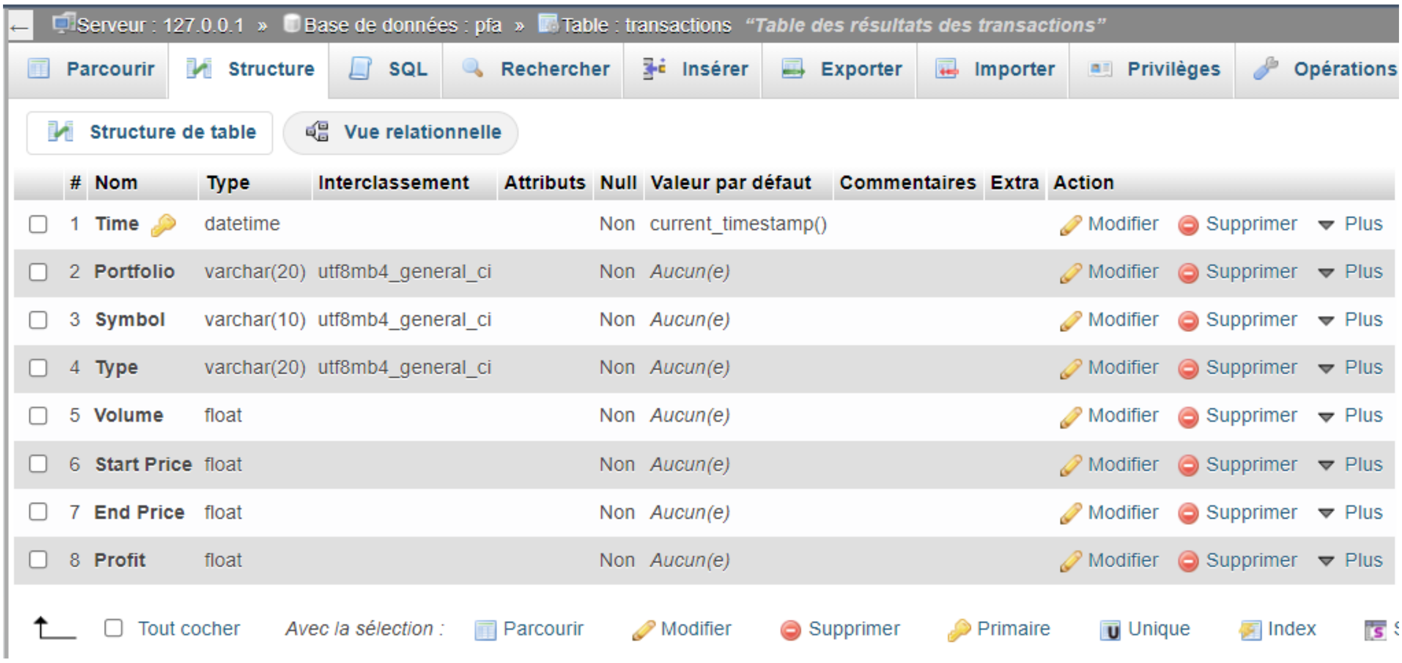
b)making an SQL table to store Transactions :

I opted for a Transactional approached based on SQL even though all transactions are timestamped seeing that the operations required are easily done and I personally more comfortable applying them in my python Code .

The Schema for the corresponding table is :

(Time DateTime,Profolio Varchar(20) , Symbol Varchar(20) , Type Varchar (20)

Volume float , Start Price float ,End Price float ,Profit)



	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
□	1	Time	datetime			Non	current_timestamp()			Modifier Supprimer Plus
□	2	Portfolio	varchar(20)	utf8mb4_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
□	3	Symbol	varchar(10)	utf8mb4_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
□	4	Type	varchar(20)	utf8mb4_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
□	5	Volume	float			Non	*Aucun(e)*			Modifier Supprimer Plus
□	6	Start Price	float			Non	*Aucun(e)*			Modifier Supprimer Plus
□	7	End Price	float			Non	*Aucun(e)*			Modifier Supprimer Plus
□	8	Profit	float			Non	*Aucun(e)*			Modifier Supprimer Plus

Time is our primary Key indicating when the transaction was committed

Portfolio serves to identify the name of the Economic agent in action , note that in code we have mentioned the agent’s name and a pseudo id to keep track of all them.

Symbols are the Key Values of this dictionary {"GBPAUD":"GBPAUD=X","JPYCHF":"JPYCHF=X", "USDEUR":"EUR=X"} which allows us to choose which ticker from yahoo finance to pick and assign to said portfolio .

Type of the transaction includes Buy , Sell , Hold as strings of characters

Volume is the percentage of out balance at the time that was invested into this transaction

Start Price indicates the starting price when launching the transaction

End Price logs the price for the above mentioned symbol after 1 hour of the starting prices’ start

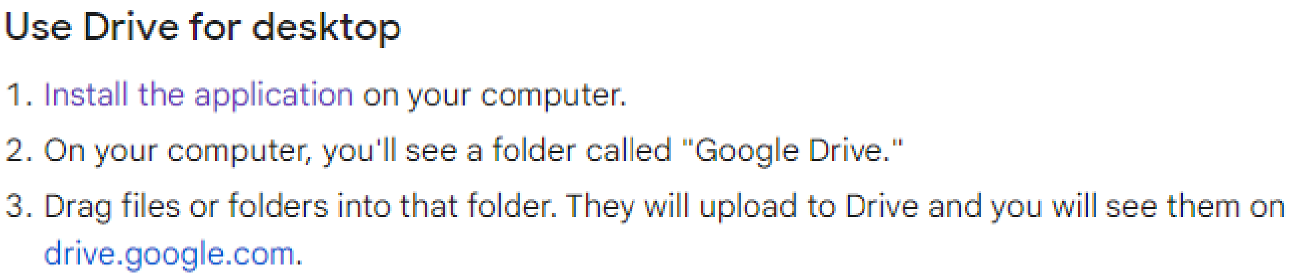
Profit is the actual amount of equity made given by(End Price –Start Price)*Volume*Balance at StartPrice

With buy or sell or hold being the transaction that led to said profit

c)setting up backup and save to Google Drive

To make sure that our scraped CSV files are preserved and that each step incremented is automatically stored . we avoid rescraping the yahoo finance each time to avoid being blocked off . while the API is free it’s best to allow server traffic to be healthy during our traffic . Note that when experimenting with stepes using 5 minutes of increment the local machine's hard drive didn't have the storage space to hold the data . A different approach was to store it in a serialized form but we ended up relaxing the incremental update by up to 1 hour .

Setting up Google Drive File Synch and backup for the data folder follows 3 easy steps :



Use Drive for desktop

1. Install the application on your computer.
2. On your computer, you'll see a folder called "Google Drive."
3. Drag files or folders into that folder. They will upload to Drive and you will see them on drive.google.com.

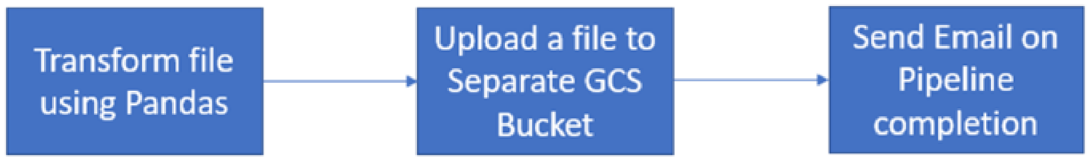
d)Setting up apache Airflow

What are Directed Acyclic Graphs, or DAGs?

DAGs, or Directed Acyclic Graphs, have nodes and edges. DAGs should not contain any loops and their edges should always be directed.

In short, a DAG is a data pipeline and each node in a DAG is a task. Some examples of nodes are downloading a file from GCS (Google Cloud Storage) to Local, applying business logic on a file using Pandas, querying the database, making a rest call, or uploading a file again to a GCS bucket.

Visualizing DAGs



Transform file using Pandas

Upload a file to Separate GCS Bucket

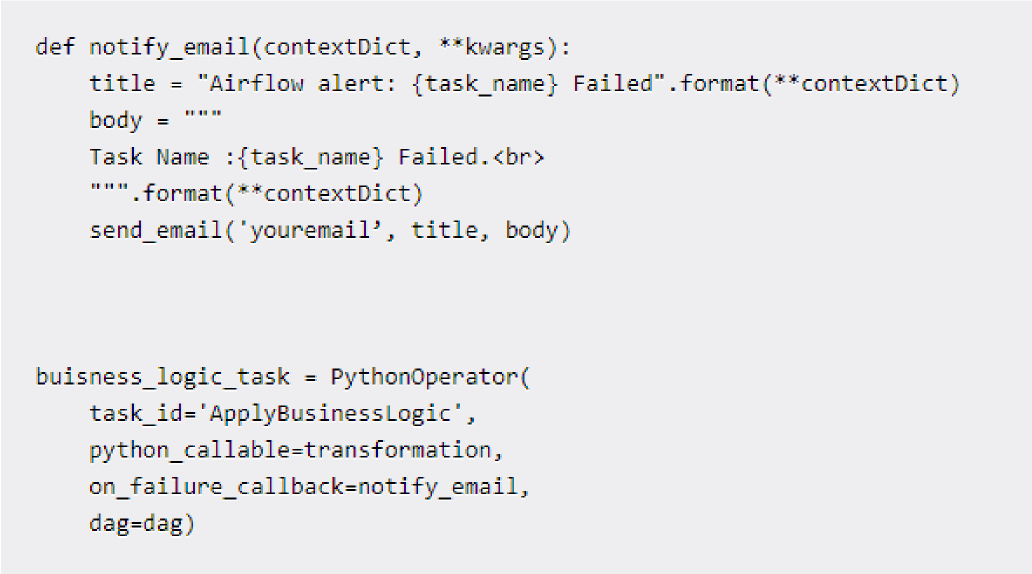
Send Email on Pipeline completion

Correct DAG with no loops

You can schedule DAGs in Airflow using the schedule_interval attribute. By default it's "None" which means that the DAG can be run only using the Airflow UI.
You can schedule the DAG to run once every hour, every day, once a week, monthly, yearly or whatever you wish using the cron presets options (@hour, @daily, @weekly, @hourly, @monthly, @yearly).
If you need to run the DAG every 5 mins, every 10 mins, every day at 14:00, or once on a specific day like every Thursday at 10:00am, then you should use these cron-based expressions.
*/5 * * * * = Every 5 minutes
0 14 * * * = Every day at 14:00

How to Create Your First DAG

The example DAG we are going to create consists of only one operator (the Python operator) which executes a Python function.



```
def notify_email(contextDict, **kwargs):
    title = "Airflow alert: {task_name} Failed".format(**contextDict)
    body = """
    Task Name :{task_name} Failed.<br>
    """.format(**contextDict)
    send_email('youremail’, title, body)



buisness_logic_task = PythonOperator(
    task_id='ApplyBusinessLogic',
    python_callable=transformation,
    on_failure_callback=notify_email,
    dag=dag)
```

This allows us to automate the steps between :

-Incrementing the steps

-updating our models

-pushing and pulling data from the appropriate databases

e)installing necessary packages

for packaging or future deployment on docker we store our current requirements

```
pip freeze > requirements.txt
```

To finalize our setup get the proper package versions by using

```
$ pip install -r requirements.txt
```

to install said dependencies and launch our project with the correct versions

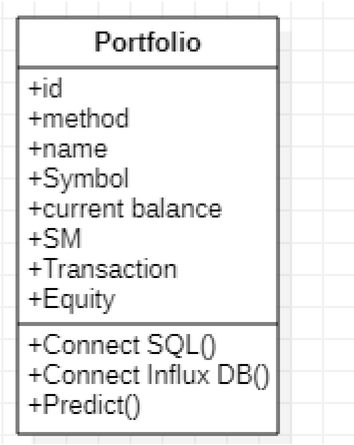
3)Preparing the incremental data Scraping

To prepare the incremental data scraping we prepare the training set lasting from 2021-11-01 00:00:00+01:00 To 2022-04-29 23:59:00+01:00 up to which we save the files under their own separate csvs for each symbol Then we proceed to create the 0th step in a jupyter notebook manually given the training data .

Now we establish an entire eco-system for handling the steps by creating a class of Step_Managers than handles all operations related to incrementing and adding to the the dataframe in question and save / deleting files when decrementing . Note that this class doesn’t save the operation to the database so we have to access it’s dataframe object to collect the measurements and transactions to be done.

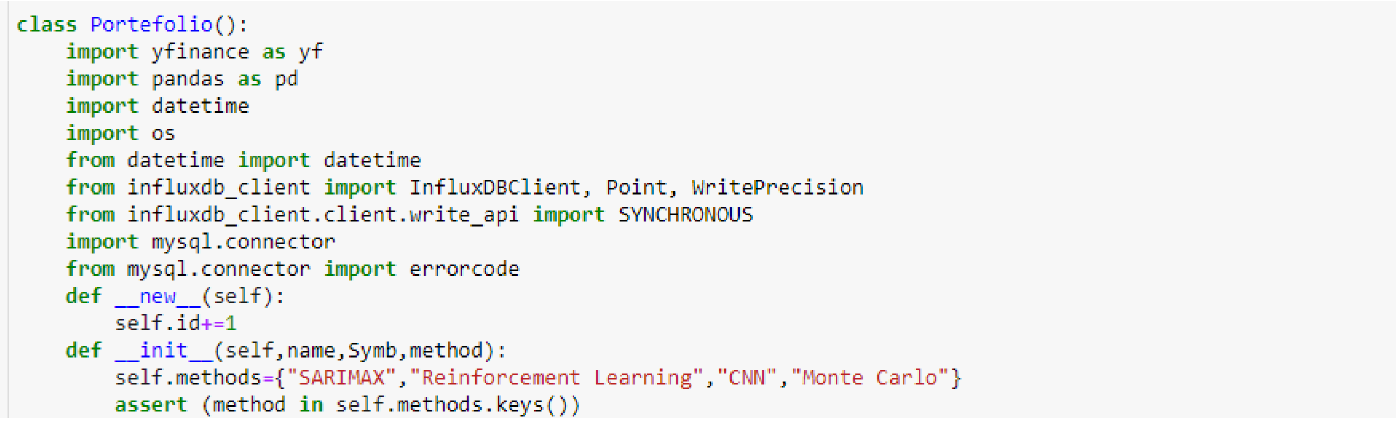
The Step manager is also endowed with a reset_step()method that might require administrative rights to execute . be wary when using it as it relies on relative path to decide what to delete and may raise errors if not conforming to the path tree described along it.

4)creating a portfolio Class :



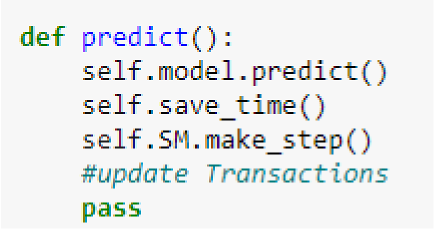
The main object that we'll be manipulating the the portfolio instance which holds all historical data and connectors to our different databases upon instantiation it takes in a number of parameters among which is the method of algorithmic trading , the name and the symbol in question .

The predict function gets a hold of the dataframe containing our data , passes the content to the model then passes the output to the transaction table and within 1 hour the real price is announced and the profit is calculated terminated the treatment to be done on the Transaction table and passing the new dataframe to Influx DB for dashboard visualizations .



```
class Portefolio():
    import yfinance as yf
    import pandas as pd
    import datetime
    import os
    from datetime import datetime
    from influxdb_client import InfluxDBClient, Point, WritePrecision
    from influxdb_client.client.write_api import SYNCHRONOUS
    import mysql.connector
    from mysql.connector import errorcode
    def __new__(self):
        self.id+=1
    def __init__(self,name,Symb,method):
        self.methods={"SARIMAX","Reinforcement Learning","CNN","Monte Carlo"}
        assert (method in self.methods.keys())
```

Note that all importation need to accompany every instance of the class to make sure that it's extraporable to other projects . while a more efficient approach would be to assign these imprs at the top of the .py file or the jupyter notebook when needed



```
def predict():
    self.model.predict()
    self.save_time()
    self.SM.make_step()
    #update Transactions
    pass
```

The transactions' update requires knowledge of the future value and thus we must wait 1hr for it to be generated in the market after which we scrape it directly and calculate the profit for the next model update

Chapter II:
Modeling and Model choice explanation

II.Modeling and Model choice explanation:

1)Time series Analysis & forecast models family:ARIMA

a) An overview of SARIMAX components

- The AR(p)
- The MA(q)
- The Integration(d)
- The Seasonal component (P,D,Q)
- The exogenous variables

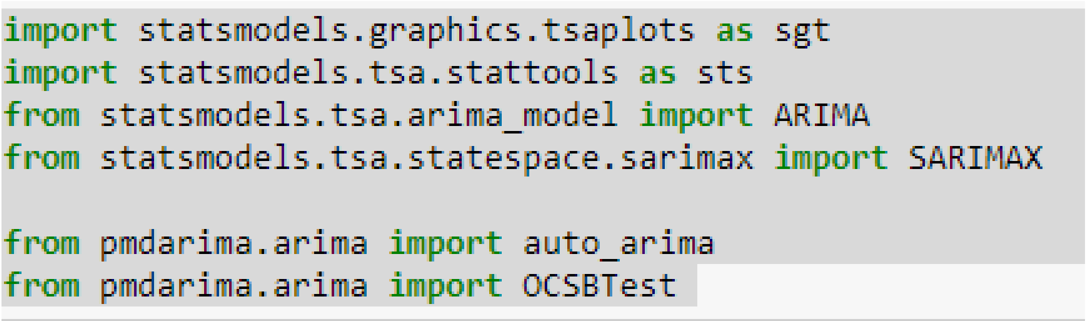
$$\underset{\text{differenced time series}}{y'_t} = \overset{\text{intercept}}{c} + \underbrace{\varphi_1 y'_{t-1} + \ldots + \varphi_p y'_{t-p}}_{\text{lagged values}} + \underbrace{\theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t}_{\text{lagged errors}}$$

The Formula for Arima(p,1,q) model

The AR(p):

The AutoRegressive component describes the timestep to be predicted by using the values of previous timeSteps . note that not all time steps need to be included and that only the significant ones at a threshold we define (5% usually) are necessary . the p denotes the number after which we can assume all the lagged values are insignificant and that values too far in the past won't affect the future .

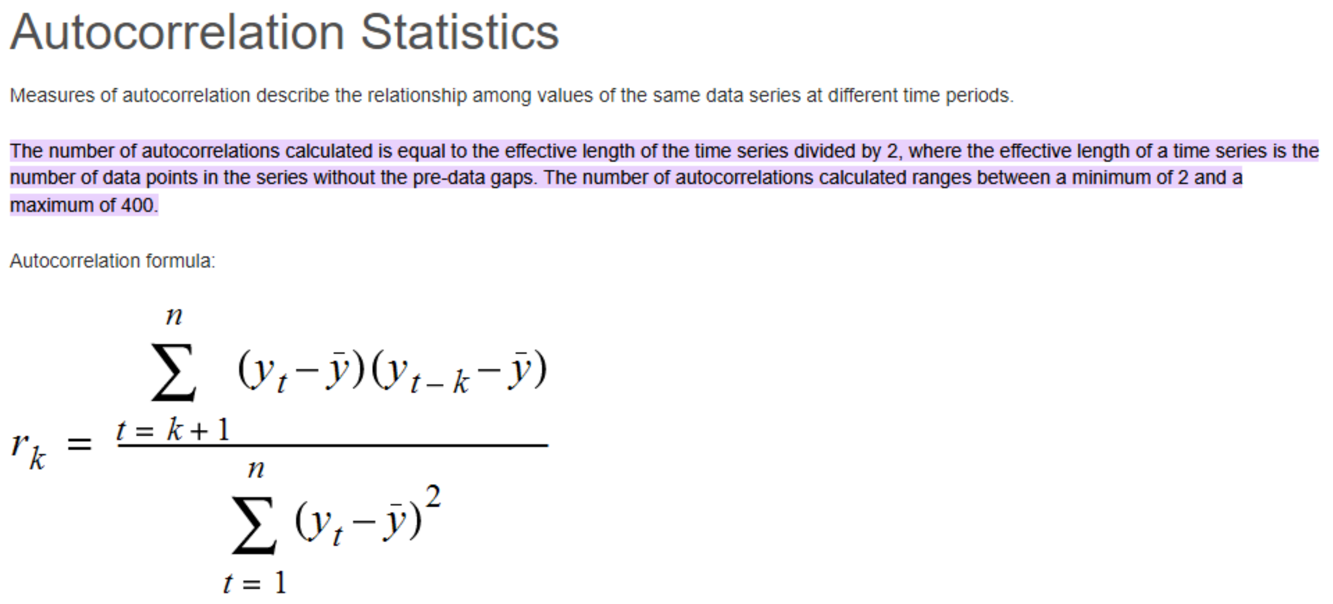
The ARIMA family of models tries to reach the best log-likelihood while minimizing the complexity thus metrics like AIC are used to judge said complexity



```
import statsmodels.graphics.tsaplots as sgt
import statsmodels.tsa.stattools as sts
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX

from pmdarima.arima import auto_arima
from pmdarima.arima import OCSBTest
```

The p in a AR(p) model is determined using the ACF plot and counts towards the trend component of the time-series



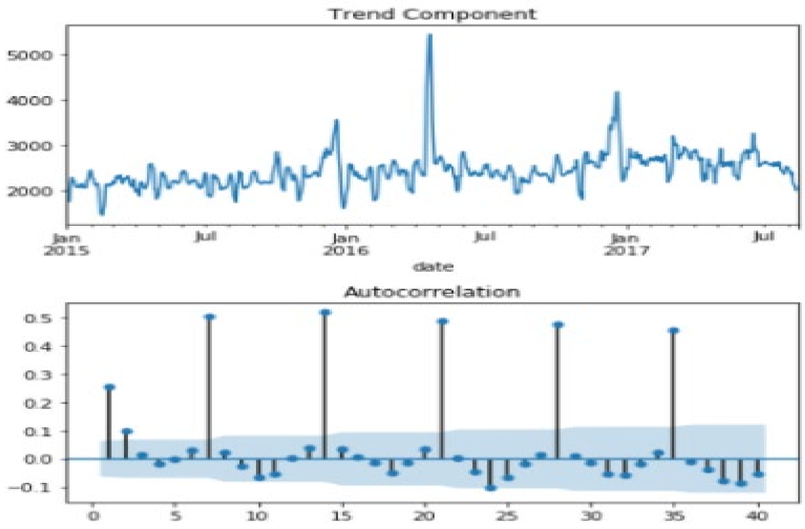
Autocorrelation Statistics

Measures of autocorrelation describe the relationship among values of the same data series at different time periods.

The number of autocorrelations calculated is equal to the effective length of the time series divided by 2, where the effective length of a time series is the number of data points in the series without the pre-data gaps. The number of autocorrelations calculated ranges between a minimum of 2 and a maximum of 400.

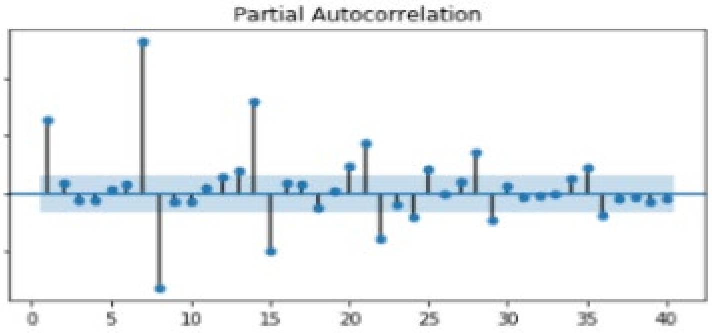
Autocorrelation formula:

$$r_k = \frac{\sum_{t=k+1}^{n} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{n} (y_t - \bar{y})^2}$$



The MA(q):

The Moving average Component of the model takes into account the variation of the past errors and exponentially smoothes them out to help prepare our model for shocks and best decribes the volatile part of the model where if cancel out all these components we con’t reach a high enough LogLikelihood when the data is noisy or ubruptly volatile.
This Component is calculated using the Partial Autocorrelation function .



Partial Autocorrelation Function

For regression of y on x_1, x_2, x_3, x_4, the partial correlation between y and x_1 is

$$\frac{cov(y, x_1 | x_2, x_3, x_4)}{\sqrt{var(y|x_2,x_3,x_4) \cdot var(x_1|x_2,x_3,x_4)}}$$

This can be calculated as the correlation between the residuals of the regression of y on x_2, x_3, x_4 with the residuals of x_1 on x_2, x_3, x_4.

For a time series, the h^{th} order partial autocorrelation is the partial correlation of y_i with y_{i-h}, conditional on $y_{i-1}, \ldots, y_{i-h+1}$, i.e.

$$\frac{cov(y_i, y_{i-h} | y_{i-1}, \ldots, y_{i-h+1})}{\sqrt{var(y_i|y_{i-1},\ldots,y_{i-h+1}) \cdot var(y_{i-h}|y_{i-1},\ldots,y_{i-h+1})}}$$

The first order partial autocorrelation is therefore the first-order autocorrelation.

The partial autocorrelations can be calculated as in the following alternative definition.

■ The Integration(d)

Also referred to as the differencing component where we try to induce stationarity on the time-series.

Stationarity is when the mean , variance and autocorrelations all stop depending on the time variable, what this gives us is a set of stable parameters to build our model upon and link all the component .

This can be achieved by choosing a point in the time-series and using it as a reference to the rest of the lagged values i.e we calculate the difference between those points and that specific point .this is in fact equivalent to the reverse operation of derivation : Integration but performed on each segment [y0 : yx] separately

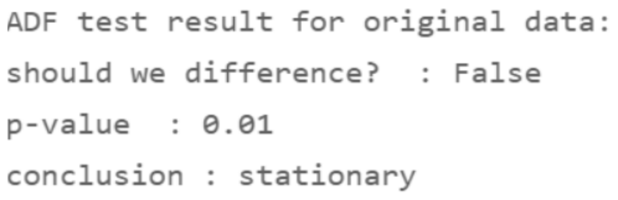
which yield deltaX=yx-yx0

$$F_X(x_{t_1+\tau}, \ldots, x_{t_n+\tau}) = F_X(x_{t_1}, \ldots, x_{t_n}) \quad \text{for all } \tau, t_1, \ldots, t_n \in \mathbb{R} \text{ and for all } n \in \mathbb{N} \quad \text{(Eq.1)}$$

$$F_{XY}(x_{t_1}, \ldots, x_{t_m}, y_{t'_1}, \ldots, y_{t'_n}) = F_{XY}(x_{t_1+\tau}, \ldots, x_{t_m+\tau}, y_{t'_1+\tau}, \ldots, y_{t'_n+\tau}) \quad \text{for all } \tau, t_1, \ldots, t_m, t'_1, \ldots, t'_n \in \mathbb{R} \text{ and for all } m, n \in \mathbb{N} \quad \text{(Eq.5)}$$

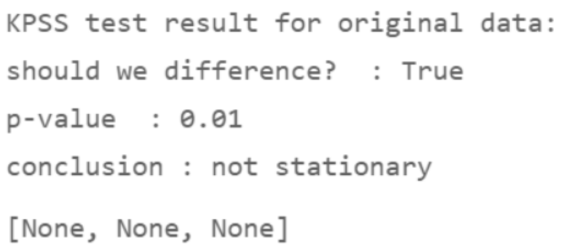
Several test allow to make sure Stationarity is achieved:

(AD-fuller test)



```
ADF test result for original data:
should we difference?  : False
p-value  : 0.01
conclusion : stationary
```

(KPSS test)



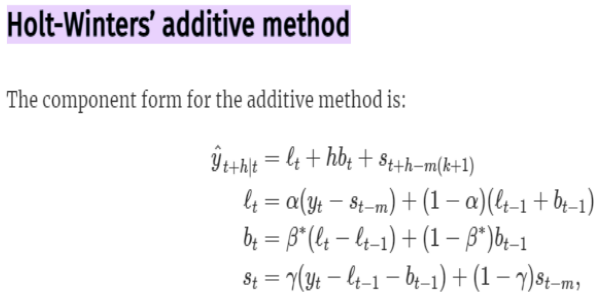
```
KPSS test result for original data:
should we difference?  : True
p-value  : 0.01
conclusion : not stationary
[None, None, None]
```

Among many others

“A stationary time series is one whose properties don’t depend on the time at which the series is observed.” (Hyndman: [8.1 Stationarity and differencing | Forecasting: Principles and Practice (2nd ed) (otexts.com)]())

- The Seasonal component (P,D,Q)

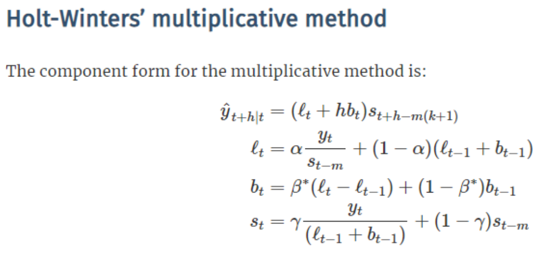
Some natural/economic phenomena occur in cyclic patterns that is, they will repeat themselves over the course of a year (each summer for example for AC sails) and so Holt-Winters model (both additive and multiplicative) take account of this season trend that the time-series may present and propose the following decomposition :



Holt-Winters’ additive method

The component form for the additive method is:

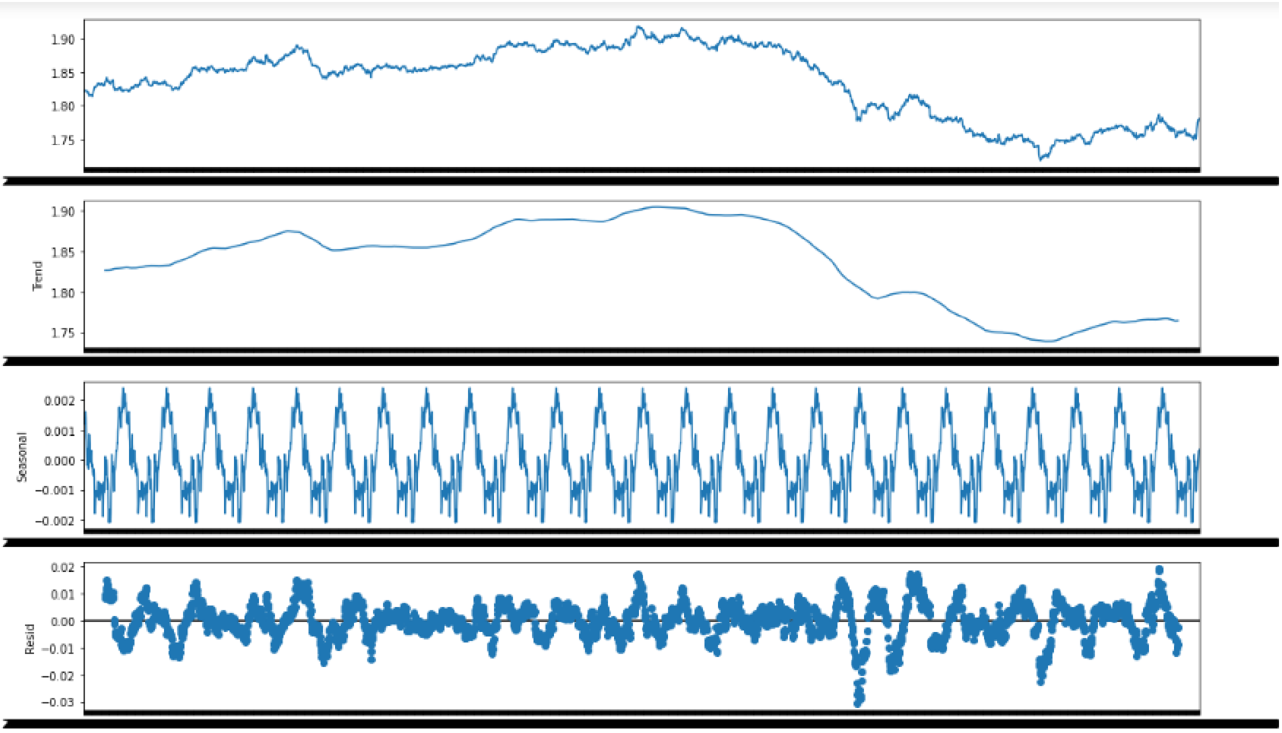
$$\begin{aligned}\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)}\\ \ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})\\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}\\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},\end{aligned}$$



Holt-Winters’ multiplicative method

The component form for the multiplicative method is:

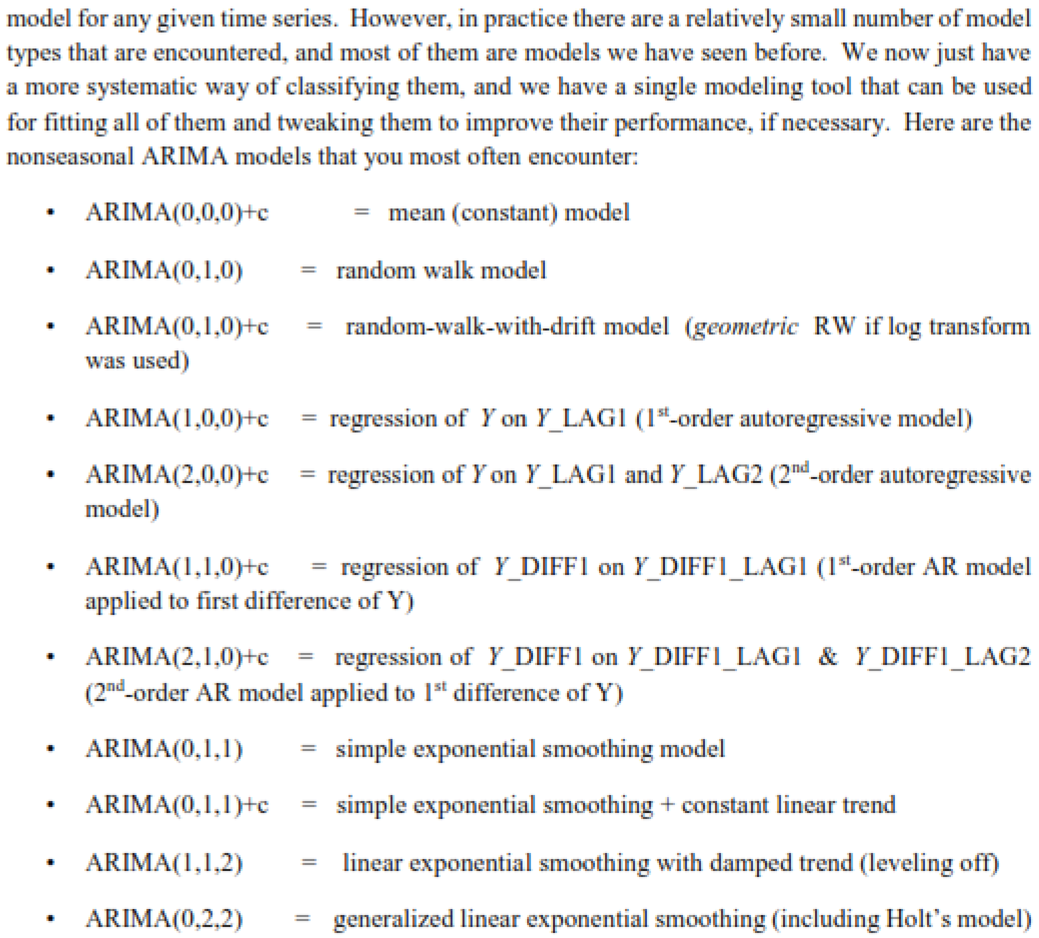
$$\begin{aligned}\hat{y}_{t+h|t} &= (\ell_t + hb_t)s_{t+h-m(k+1)}\\ \ell_t &= \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})\\ b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}\\ s_t &= \gamma\frac{y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m}\end{aligned}$$



Exogenous Variables:

Exogenous variables are outsider variables(time series or others) that help us detect the irregularities of the outside world they could be the result of sentiment analysis or just the opening , high , low exchange rates for a specific symbol when taking the closing rate as the endogenous variable (the one being studied)

Here's a cheat sheet for most commonly named **ARIMA** models:



model for any given time series. However, in practice there are a relatively small number of model types that are encountered, and most of them are models we have seen before. We now just have a more systematic way of classifying them, and we have a single modeling tool that can be used for fitting all of them and tweaking them to improve their performance, if necessary. Here are the nonseasonal ARIMA models that you most often encounter:

- ARIMA(0,0,0)+c = mean (constant) model
- ARIMA(0,1,0) = random walk model
- ARIMA(0,1,0)+c = random-walk-with-drift model (*geometric* RW if log transform was used)
- ARIMA(1,0,0)+c = regression of *Y* on *Y*_LAG1 (1st-order autoregressive model)
- ARIMA(2,0,0)+c = regression of *Y* on *Y*_LAG1 and *Y*_LAG2 (2nd-order autoregressive model)
- ARIMA(1,1,0)+c = regression of *Y*_DIFF1 on *Y*_DIFF1_LAG1 (1st-order AR model applied to first difference of Y)
- ARIMA(2,1,0)+c = regression of *Y*_DIFF1 on *Y*_DIFF1_LAG1 & *Y*_DIFF1_LAG2 (2nd-order AR model applied to 1st difference of Y)
- ARIMA(0,1,1) = simple exponential smoothing model
- ARIMA(0,1,1)+c = simple exponential smoothing + constant linear trend
- ARIMA(1,1,2) = linear exponential smoothing with damped trend (leveling off)
- ARIMA(0,2,2) = generalized linear exponential smoothing (including Holt's model)

b)Examples of Estimators

knowing the information criterion : **information_criterion** : str, optional (default='aic')

finding the orders necessary for each component become the results of solving the associated equation :

- newton' for Newton-Raphson
- 'nm' for Nelder-Mead
- 'bfgs' for Broyden-Fletcher-Goldfarb-Shanno (BFGS)
- 'lbfgs' for limited-memory BFGS with optional box constraints
- 'powell' for modified Powell's method
- 'cg' for conjugate gradient
- 'ncg' for Newton-conjugate gradient
- 'basinhopping' for global basin-hopping solver

c)Rules to follow

Identifying the order of differencing and the constant:

- Rule 1: If the series has positive autocorrelations out to a high number of lags (say, 10 or more), then it probably needs a higher order of differencing.
- Rule 2: If the lag-1 autocorrelation is zero or negative, or the autocorrelations are all small and patternless, then the series does *not* need a higher order of differencing. If the lag-1 autocorrelation is -0.5 or more negative, the series may be overdifferenced. **BEWARE OF OVERDIFFERENCING.**
- Rule 3: The optimal order of differencing is often the order of differencing at which the standard deviation is lowest. (Not always, though. Slightly too much or slightly too little differencing can also be corrected with AR or MA terms. See rules 6 and 7.)
- Rule 4: A model with no orders of differencing assumes that the original series is stationary (among other things, mean-reverting). A model with one order of differencing assumes that the original series has a constant average trend (e.g. a random walk or SES-type model, with or without growth). A model with two orders of total differencing assumes that the original series has a time-varying trend (e.g. a random trend or LES-type model).
- Rule 5: A model with no orders of differencing normally includes a constant term (which allows for a non-zero mean value). A model with two orders of total differencing normally does not include a constant term. In a model with one order of total differencing, a constant term should be included if the series has a non-zero average trend.

Identifying the numbers of AR and MA terms:

- Rule 6: If the partial autocorrelation function (PACF) of the differenced series displays a sharp cutoff and/or the lag-1 autocorrelation is positive--i.e., if the series appears slightly "underdifferenced"--then consider adding one or more AR terms to the model. The lag beyond which the PACF cuts off is the indicated number of AR terms.
- Rule 7: If the autocorrelation function (ACF) of the differenced series displays a sharp cutoff and/or the lag-1 autocorrelation is negative--i.e., if the series appears slightly "overdifferenced"--then consider adding an MA term to the model. The lag beyond which the ACF cuts off is the indicated number of MA terms.
- Rule 8: It is possible for an AR term and an MA term to cancel each other's effects, so if a mixed AR-MA model seems to fit the data, also try a model with one fewer AR term and one fewer MA term--particularly if the parameter estimates in the original model require more than 10 iterations to converge. **BEWARE OF USING MULTIPLE AR TERMS AND MULTIPLE MA TERMS IN THE SAME MODEL.**
- Rule 9: If there is a unit root in the AR part of the model--i.e., if the sum of the AR coefficients is almost exactly 1--you should reduce the number of AR terms by one and increase the order of differencing by one.
- Rule 10: If there is a unit root in the MA part of the model--i.e., if the sum of the MA coefficients is almost exactly 1--you should reduce the number of MA terms by one and reduce the order of differencing by one.
- Rule 11: If the long-term forecasts* appear erratic or unstable, there may be a unit root in the AR or MA coefficients.

Identifying the seasonal part of the model:

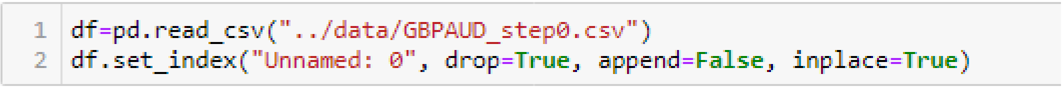
- Rule 12: If the series has a strong and consistent seasonal pattern, then you must use an order of seasonal differencing (otherwise the model assumes that the seasonal pattern will fade away over time). However, never use more than one order of seasonal differencing or more than 2 orders of total differencing (seasonal+nonseasonal).
- Rule 13: If the autocorrelation of the appropriately differenced series is positive at lag s, where s is the number of periods in a season, then consider adding an SAR term to the model. If the autocorrelation of the differenced series is negative at lag s, consider adding an SMA term to the model. The latter situation is likely to occur if a seasonal difference has been used, which should be done if the data has a stable and logical seasonal pattern. The former is likely to occur if a seasonal difference has not been used, which would only be appropriate if the

seasonal pattern is not stable over time. You should try to avoid using more than one or two seasonal parameters (SAR+SMA) in the same model, as this is likely to lead to overfitting of the data and/or problems in estimation.

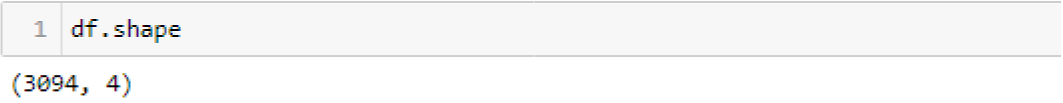
***A caveat about long-term forecasting in general:** linear time series models such as ARIMA and exponential smoothing models predict the more distant future by making a series of one-period-ahead forecasts and plugging them in for unknown future values as they look farther ahead. For example, a 2-period-ahead forecast is computed by treating the 1-period-ahead forecast as if it were data and then applying the same forecasting equation. This step can be repeated any number of times in order to forecast as far into the future as you want, and the method also yields formulas for computing theoretically-appropriate confidence intervals around the longer-term forecasts. However, the models are identified and optimized based on their one-period-ahead forecasting performance, and rigid extrapolation of them may not be the best way to forecast many periods ahead (say, more than one year when working with monthly or quarterly business data), particularly when the modeling assumptions are at best only approximately satisfied (which is nearly always the case). If one of your objectives is to generate long-term forecasts, it would be good to also draw on other sources of information during the model selection process and/or to optimize the parameter estimates for multi-period forecasting if your software allows it and/or use an auxiliary model (possibly one that incorporates expert opinion) for long-term forecasting.

d) preprocessing & Modeling :

importing the data



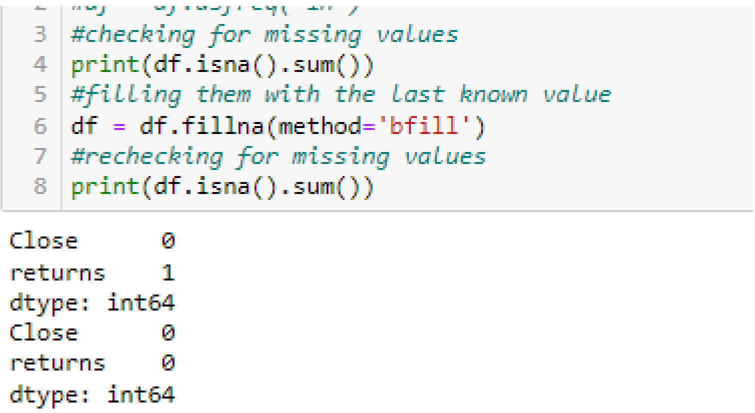
```
df=pd.read_csv("../data/GBPAUD_step0.csv")
df.set_index("Unnamed: 0", drop=True, append=False, inplace=True)
```



```
df.shape
```

```
(3094, 4)
```

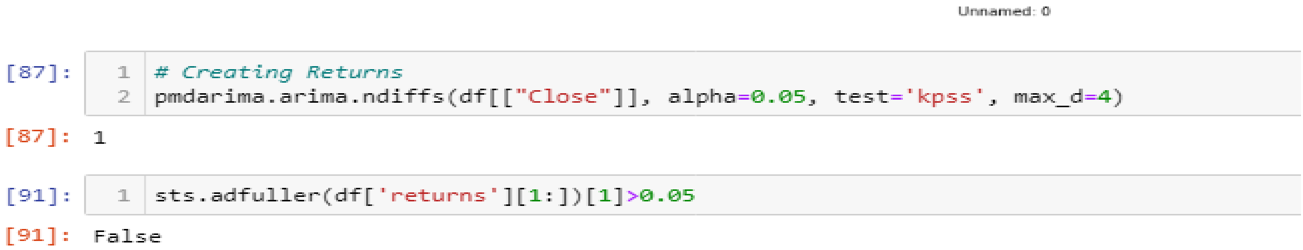
Checking for missing values



```
#checking for missing values
print(df.isna().sum())
#filling them with the last known value
df = df.fillna(method='bfill')
#rechecking for missing values
print(df.isna().sum())
```

```
Close      0
returns    1
dtype: int64
Close      0
returns    0
dtype: int64
```

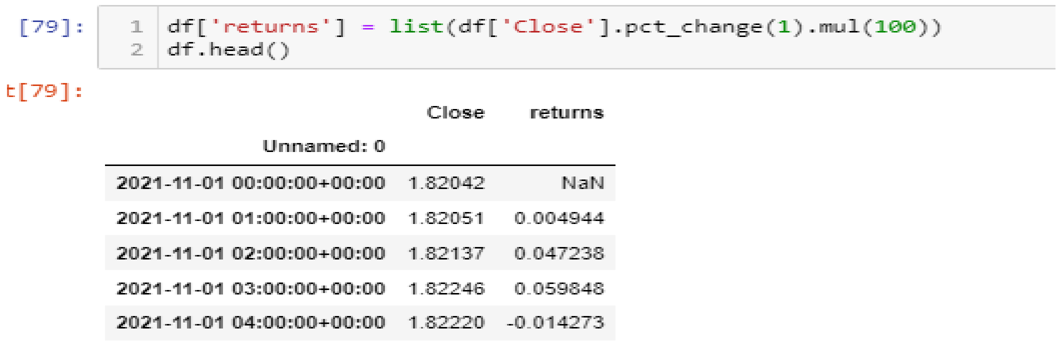
Checking for stationarity :



```
[87]: # Creating Returns
      pmdarima.arima.ndiffs(df[["Close"]], alpha=0.05, test='kpss', max_d=4)
[87]: 1
```

```
[91]: sts.adfuller(df['returns'][1:])[1]>0.05
[91]: False
```

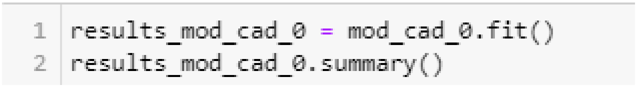
Differencing



```
[79]:  df['returns'] = list(df['Close'].pct_change(1).mul(100))
       df.head()
```

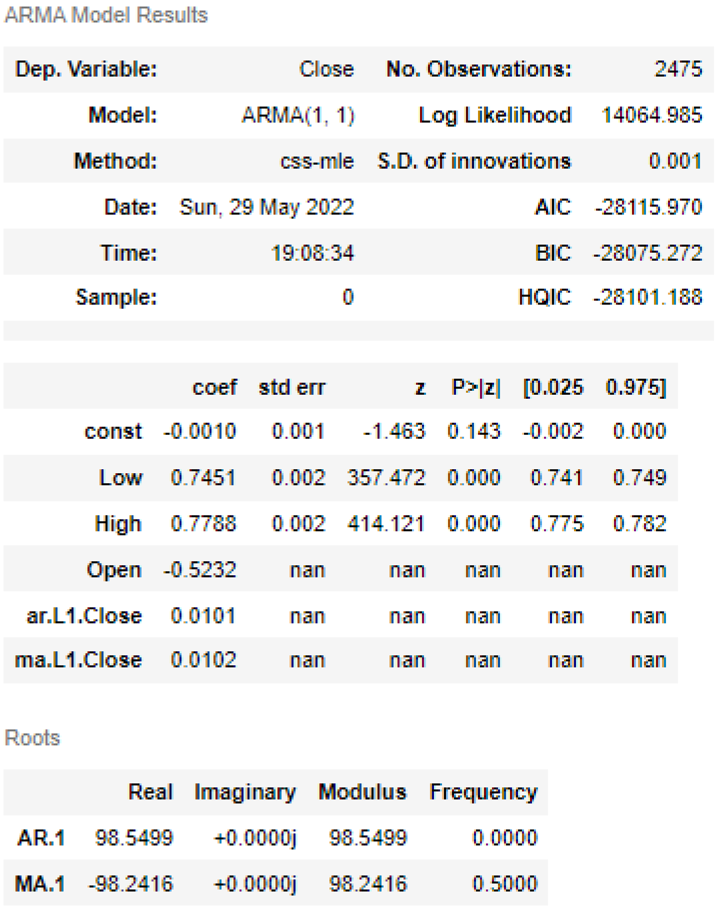
t[79]:

	Close	returns
Unnamed: 0		
2021-11-01 00:00:00+00:00	1.82042	NaN
2021-11-01 01:00:00+00:00	1.82051	0.004944
2021-11-01 02:00:00+00:00	1.82137	0.047238
2021-11-01 03:00:00+00:00	1.82246	0.059848
2021-11-01 04:00:00+00:00	1.82220	-0.014273



```
results_mod_cad_0 = mod_cad_0.fit()
results_mod_cad_0.summary()
```

Yields:



ARMA Model Results

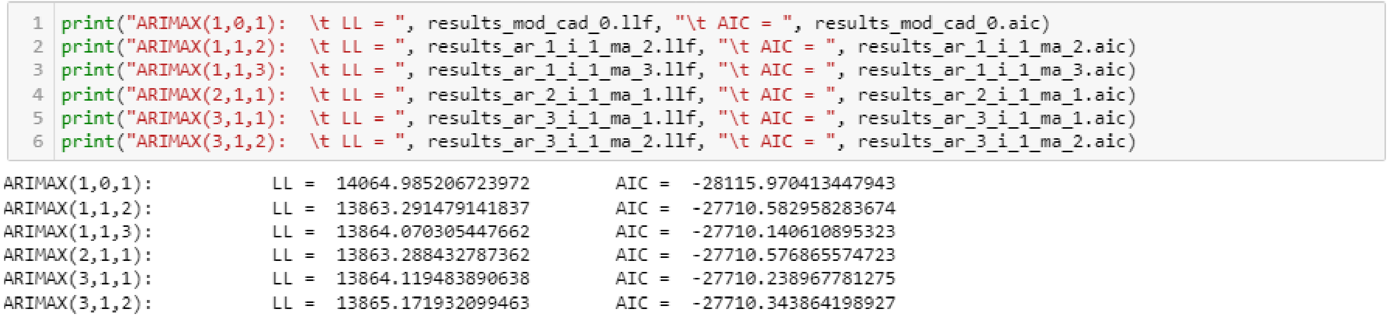
Dep. Variable:	Close	No. Observations:	2475
Model:	ARMA(1, 1)	Log Likelihood	14064.985
Method:	css-mle	S.D. of innovations	0.001
Date:	Sun, 29 May 2022	AIC	-28115.970
Time:	19:08:34	BIC	-28075.272
Sample:	0	HQIC	-28101.188

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0010	0.001	-1.463	0.143	-0.002	0.000
Low	0.7451	0.002	357.472	0.000	0.741	0.749
High	0.7788	0.002	414.121	0.000	0.775	0.782
Open	-0.5232	nan	nan	nan	nan	nan
ar.L1.Close	0.0101	nan	nan	nan	nan	nan
ma.L1.Close	0.0102	nan	nan	nan	nan	nan

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	98.5499	+0.0000j	98.5499	0.0000
MA.1	-98.2416	+0.0000j	98.2416	0.5000

But then comparing to more complicated models we notice significant improvement of the NLL:



```
print("ARIMAX(1,0,1):  \t LL = ", results_mod_cad_0.llf, "\t AIC = ", results_mod_cad_0.aic)
print("ARIMAX(1,1,2):  \t LL = ", results_ar_1_i_1_ma_2.llf, "\t AIC = ", results_ar_1_i_1_ma_2.aic)
print("ARIMAX(1,1,3):  \t LL = ", results_ar_1_i_1_ma_3.llf, "\t AIC = ", results_ar_1_i_1_ma_3.aic)
print("ARIMAX(2,1,1):  \t LL = ", results_ar_2_i_1_ma_1.llf, "\t AIC = ", results_ar_2_i_1_ma_1.aic)
print("ARIMAX(3,1,1):  \t LL = ", results_ar_3_i_1_ma_1.llf, "\t AIC = ", results_ar_3_i_1_ma_1.aic)
print("ARIMAX(3,1,2):  \t LL = ", results_ar_3_i_1_ma_2.llf, "\t AIC = ", results_ar_3_i_1_ma_2.aic)
```

```
ARIMAX(1,0,1):          LL =  14064.985206723972        AIC =  -28115.970413447943
ARIMAX(1,1,2):          LL =  13863.29147914183 7       AIC =  -27710.582958283674
ARIMAX(1,1,3):          LL =  13864.070305447662        AIC =  -27710.140610895323
ARIMAX(2,1,1):          LL =  13863.288432787362        AIC =  -27710.576865574723
ARIMAX(3,1,1):          LL =  13864.119483890638        AIC =  -27710.238967781275
ARIMAX(3,1,2):          LL =  13865.171932099463        AIC =  -27710.343864198927
```

We take the Arimax (2,1,1) model seeing that it holds the smalled Log-Likelihood

In the rest of the script we continue and calculate volatility with Egarch model and try to quantify the bias that people have towards buying as opposed to being too afraid to sell when it's time to sell.

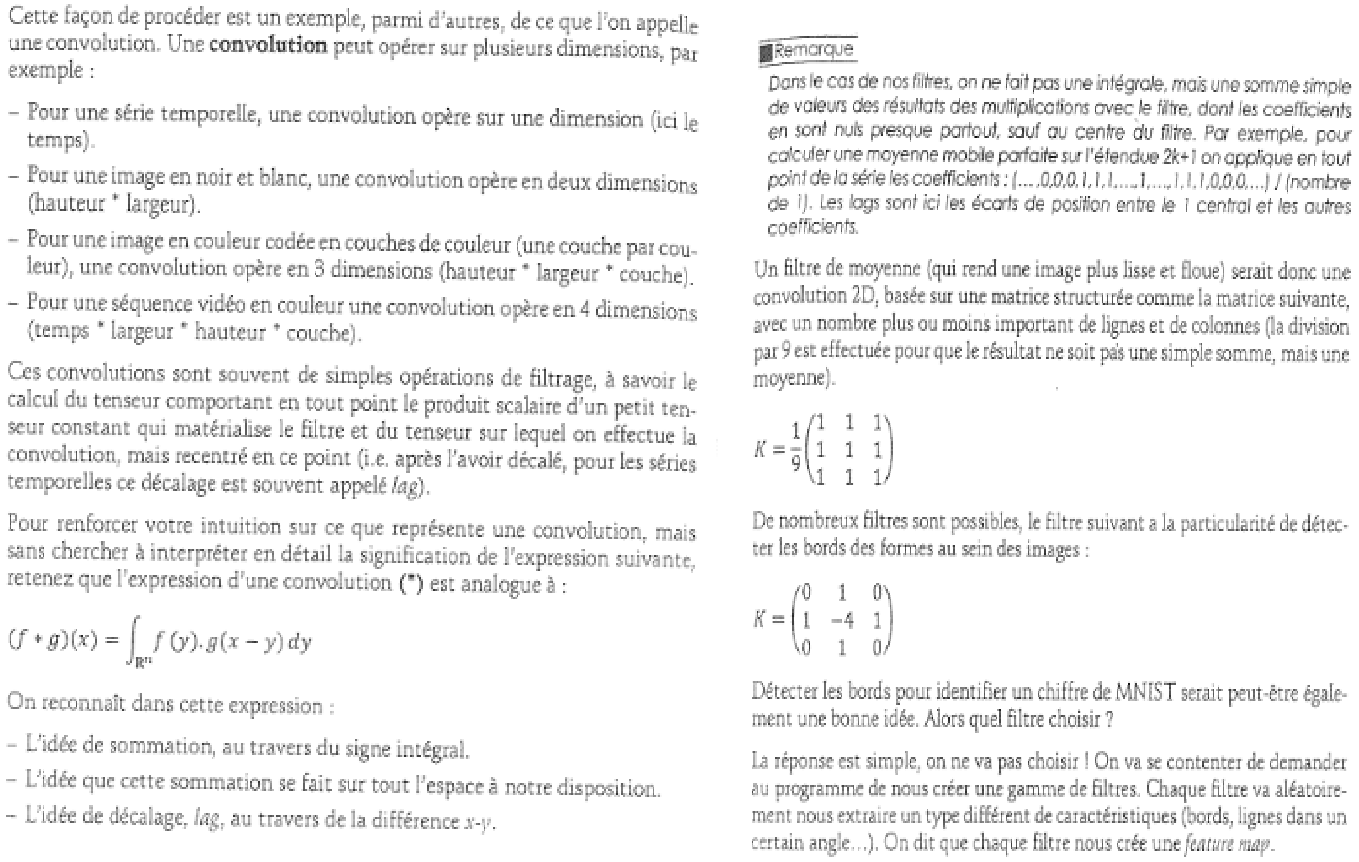
In fact this work can be systematically achieved by the auto_arima function which we will call on each step of incremental step in our Apache airflow DAGs

2)1D Convolutional neural network for predictive price forecasting and pattern matching

a) The regression Task

while ConvNets may serve various tasks when the dense layer just before our output doesn't include any activation function we are said to perform the task of regression seeing that the Convolutional part extracts features while the Multi perceptron layers(Dense) apply a more complicated form of regression . the output of said layers allows us to predict the values of stocks after having detected and used deep learning to reach the non-linearities within and thus we take on the problem of price forecast as being a regression problem where the features are engineered by the convolutional nets and the labels are the specific timesteps we want to predict , in our case 1h into the future.

b)the generic architecture for the Neural Network



Cette façon de procéder est un exemple, parmi d'autres, de ce que l'on appelle une convolution. Une **convolution** peut opérer sur plusieurs dimensions, par exemple :

- Pour une série temporelle, une convolution opère sur une dimension (ici le temps).
- Pour une image en noir et blanc, une convolution opère en deux dimensions (hauteur * largeur).
- Pour une image en couleur codée en couches de couleur (une couche par couleur), une convolution opère en 3 dimensions (hauteur * largeur * couche).
- Pour une séquence vidéo en couleur une convolution opère en 4 dimensions (temps * largeur * hauteur * couche).

Ces convolutions sont souvent de simples opérations de filtrage, à savoir le calcul du tenseur comportant en tout point le produit scalaire d'un petit tenseur constant qui matérialise le filtre et du tenseur sur lequel on effectue la convolution, mais recentré en ce point (i.e. après l'avoir décalé, pour les séries temporelles ce décalage est souvent appelé *lag*).

Pour renforcer votre intuition sur ce que représente une convolution, mais sans chercher à interpréter en détail la signification de l'expression suivante, retenez que l'expression d'une convolution (*) est analogue à :

$$(f * g)(x) = \int_{\mathbb{R}^n} f(y).g(x-y)\,dy$$

On reconnaît dans cette expression :

- L'idée de sommation, au travers du signe intégral.
- L'idée que cette sommation se fait sur tout l'espace à notre disposition.
- L'idée de décalage, *lag*, au travers de la différence x-y.

Remarque

Dans le cas de nos filtres, on ne fait pas une intégrale, mais une somme simple de valeurs des résultats des multiplications avec le filtre, dont les coefficients en sont nuls presque partout, sauf au centre du filtre. Par exemple, pour calculer une moyenne mobile parfaite sur l'étendue 2k+1 on applique en tout point de la série les coefficients : (....0,0,0,1,1,1,...,1,...,1,1,1,0,0,0,...) / (nombre de 1). Les lags sont ici les écarts de position entre le 1 central et les autres coefficients.

Un filtre de moyenne (qui rend une image plus lisse et floue) serait donc une convolution 2D, basée sur une matrice structurée comme la matrice suivante, avec un nombre plus ou moins important de lignes et de colonnes (la division par 9 est effectuée pour que le résultat ne soit pas une simple somme, mais une moyenne).

$$K = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

De nombreux filtres sont possibles, le filtre suivant a la particularité de détecter les bords des formes au sein des images :

$$K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Détecter les bords pour identifier un chiffre de MNIST serait peut-être également une bonne idée. Alors quel filtre choisir ?

La réponse est simple, on ne va pas choisir ! On va se contenter de demander au programme de nous créer une gamme de filtres. Chaque filtre va aléatoirement nous extraire un type différent de caractéristiques (bords, lignes dans un certain angle...). On dit que chaque filtre nous crée une *feature map*.

In my work however I opted to using the BRU activation function instead of the Relu seeing that it produced 67% compared to relu's 27% on the Cifar 10 dataset.

The same article discussing the Bionodal root unit activation function is surpassing the traditional Relu when addressing the non-linearities of natural Brownian mouvements (similer to currency exchange rates) provides text book implementation to creating our own gradients and Activation functions with support of the tensorflow api and Keras' backend

c)the different possible activation functions and their performance on benchmarks

BRU:

```
Epoch 039: Loss: 1.807, Accuracy: 68.474%
Time taken for epoch 95.40356135368347 sec
```

ELU:

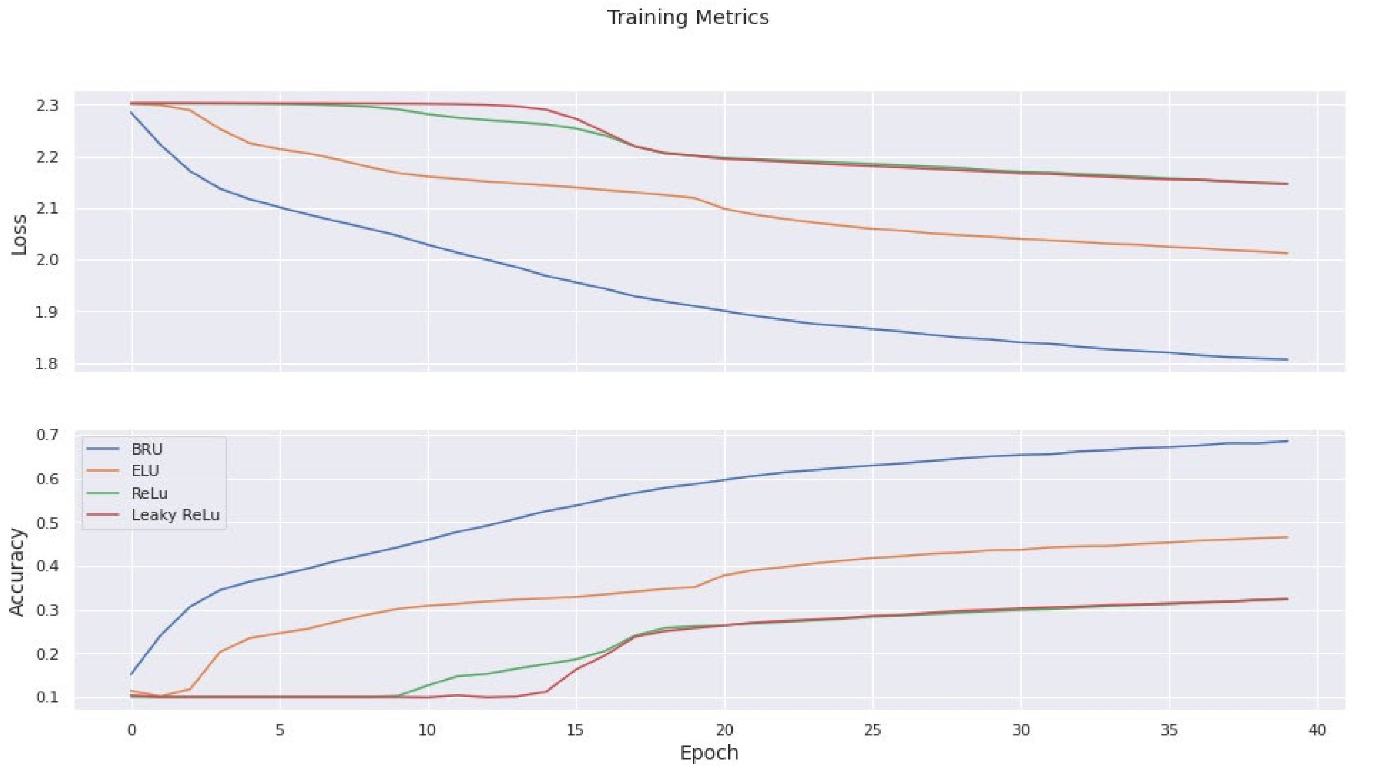
```
Epoch 039: Loss: 2.012, Accuracy: 46.544%
Time taken for epoch 49.870466232299805 sec
```

Relu:

```
Epoch 039: Loss: 2.147, Accuracy: 32.462%
Time taken for epoch 48.71515703201294 sec
```

Leaky RELU:

```
Epoch 039: Loss: 2.146, Accuracy: 32.334%
Time taken for epoch 49.46626281738281 sec
```



d)Using weights and Biases API to Explore the hyper-parameter space

By wrapping our training session in the API provided by weights and Biases we are not only able to see the hardware requirements for training (CPU usage , RAM , GPU memory, writing to HDD) along with modeling metrics(val_accuracy , mse) we can also visit all possible combinations of hyper parameters to visualize a plane of the different combinations with respect to our accuracy and thus deciding which combination fits best to our data.

To perform this search , there are 3 strategies after enumerating all the parameters and potention values:

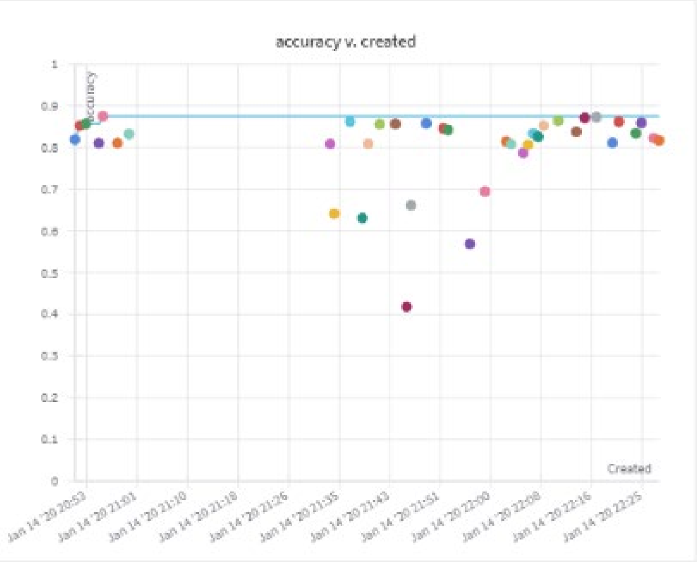
-Random : purely stochastic (good for a large nombre of combinations , may not find the optimal one)

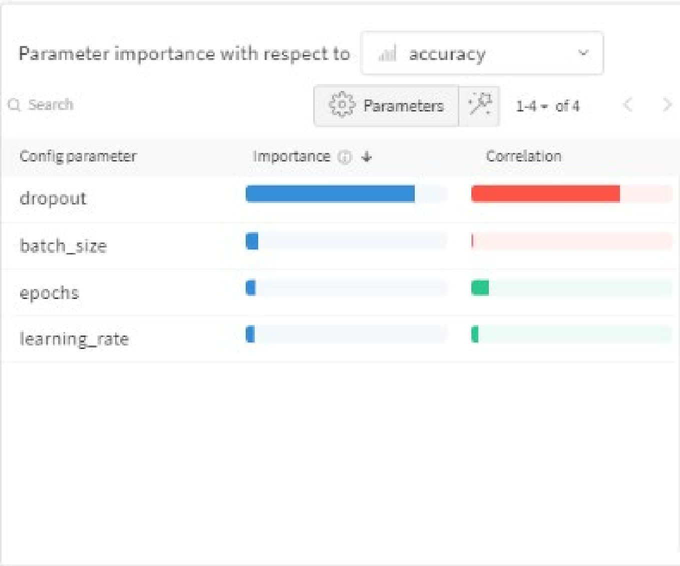
-Grid Search: Searches every combination one by one . exhaustive but yields exact results

-according to a distribution : usually we set the estimator function of these hyperparameters to a uniform distribution when we have no idea what to expect . but with A priori knowledge we can skew the distribution as a log transform and favour some combinations over others .

In Practice, it's best to out line the min, max values and perform a random search then take the good subspace we found and perform an exhaustive Grid Search

By conforming to the API and running multiple training sessions :



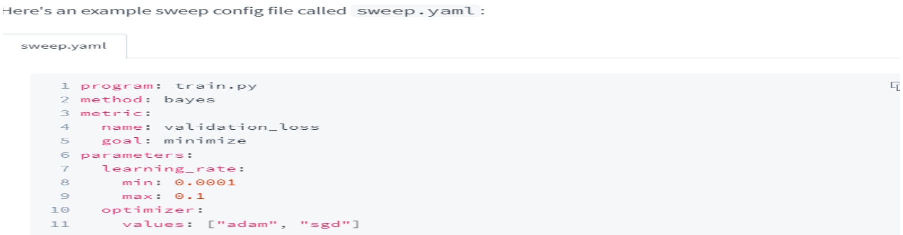


Config parameter	Importance ⓘ ↓	Correlation
dropout		
batch_size		
epochs		
learning_rate		

The paramaters need to be passed as a Yaml or dictionary to the W&B API

These results show that dropout is the highest correlated value with maximizing accuracy but the fact that the bar is red signifies that the sweep detected that dropout is inversely correlated to accuracy meaning , a decrease of dropout should in icrease accuracy

While on the other hand , the learning rate is positively correlated with improving accuracy but only a small insignificant amount we should prioritize improving the other parameters before passing to the next .

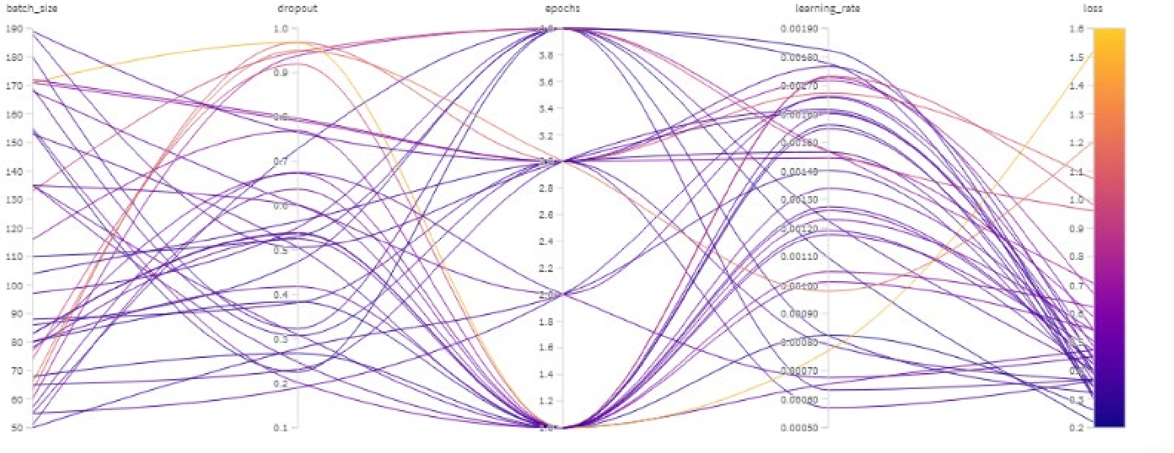


Here's an example sweep config file called `sweep.yaml`:

sweep.yaml

```yaml
program: train.py
method: bayes
metric:
  name: validation_loss
  goal: minimize
parameters:
  learning_rate:
    min: 0.0001
    max: 0.1
  optimizer:
    values: ["adam", "sgd"]
```

We can figure out which combination failed when taking what values by observing this graph and retracing the blue purple values on the loss Axis.



e)Other model improvements:

i)MaxGroupPooling

ii)Batch normalization

iii)Differencing / MinMaxScaling

iv)Data augmentation

v)full padding

3)Reinforcement Learning model:

a)Model free or Model based

Model-Free or Model-Based

The first major decision that you have to make is whether you have an accurate model of the environment. Model-based algorithms use definitive knowledge of the environment they are operating in to improve learning. For example, board games often limit the moves that you can make, and you can use this knowledge to (a) constrain the algorithm so that it does not provide invalid actions and (b) improve performance by projecting forward in time (for example, if I move here and if the opponent moves there, I can win). Human-beating algorithms for games like Go and poker can take advantage of the game's fixed rules. You and your opponent can make a limited set of moves. This limits the number of strategies the algorithms have to search through. Like expert systems, model-based solutions learn efficiently because they don't waste time searching improper paths.

Model-free algorithms can, in theory, apply to any problem. They learn strategies through interaction, absorbing any environmental rules in the process. This is not the end of the story, however. Some algorithms can learn models of the environment at the same time as learning optimal strategies. Several new algorithms can also leverage the potential, but unknown actions of other agents (or other players). In other words, these agents can learn to counteract another agent's strategies.

Algorithms such as these tend to blur the distinction between model-based and model-free, because ultimately you need a model of the environment somewhere. The difference is whether you can statically define it, whether you can learn it, or whether you can assume the model from the strategy.

-------- > we opt toward a model-free design

b)Discrete or Continuous Actions

Actions within an environment can take many forms: you could vary the amount of torque to apply to a motor controller, decide whether to add a banana to a shopping basket, or buy millions of dollars of stock on a trade.
Some actions are binary: a stop/go road sign has precisely two classes. Other times you might have categories that you can encode into binary actions. For example, you could quantize the throttle control of a vehicle into three binary actions: none, halfpower, and full power.

But often actions require more finesse. When you drive a car you rotate the steering wheel over an infinite number of angles. If you made it discrete, pain would ensue. How would you like it if your bus driver turned the steering wheel in 90-degree increments?

The weight or length of time of an action might also be important. In Super Mario Bros. the longer you hold the jump button the higher Mario jumps. You could make time part of the problem and have a continuous action that represents the amount of time to hold a button, for example. Or you could make it discrete and ensure you repeatedly poll the agent to see if it should continue to perform the action. If you can argue that the length of time of the action is decoupled from performing the action, they could be separate variables.

RL algorithms should handle both binary and continuously variable actions. But many algorithms are limited to one.

--------- >we opt toward Discrete Actions

c)Optimization methods

In general, you build models (which may or may not contain linear equations) and train them using an optimization method. RL has the same problem. You want to build an agent that is able to produce a solution given a goal. Precisely how it does this creates another fundamental theme in RL.

One way is to try as many actions as possible and record the results. In the future we can guide the agent by following the strategy that led to the best result. These are called value-based algorithms.

Another way is to maintain a model and tweak the parameters of the model to tend toward the actions that produced the best result. These are called policy-based algo-rithms.

To help us understand this, imagine a two-dimensional grid with a cliff toward the south. Your task is to design a robot that will repeatedly test each square and learn that there is a cost associated with falling off the cliff. If you used a value-based algo-rithm, and if you converted the strategy into words, it would say “do not step off a cliff.” A policy-based algorithm would say “move away from the cliff.” A subtle but important difference.

Value- and policy-based algorithms are currently the most studied and therefore the most popular. But imitation-based algorithms, where we optimize the agent to mimic the actions of an expert, can work well when we are trying to incorporate human guidance. Any other algorithms that don’t fit into any of these classes may spawn new methodologies in the future.

d)Reward Engineering

i)reward shaping

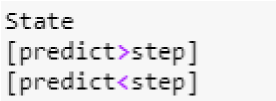
we will simply shape the reward as being the profit made and nothing else . surely for faster learning we should give profitable actions more rewards but at this time seeing that the end state space is Binary we might include the amount of variation along the way and here again : the exogenous variables high and low as well as open all help lead to the best performing reward engineering

ii)discounting

when we want to be greedy and reach a certain a Q-value as fast as possible we’d assign a decaying factor gamma to each reward making it’s value drop in the future (like inflation) so –like any ration economical agent- we rush towards the maximum Q-value /best policy in as few agent timesteps as possible in order not to “miss out”

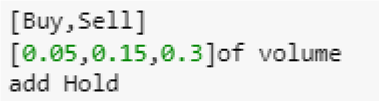
e)Environment Description

States=



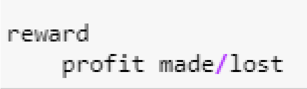
```
State
[predict>step]
[predict<step]
```

Actions=



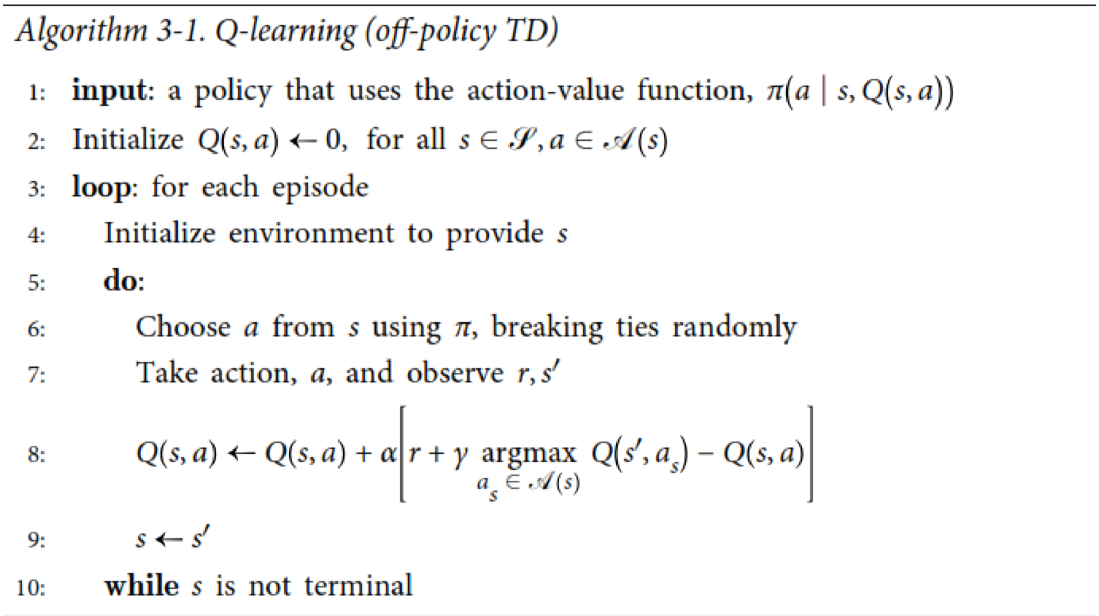
```
[Buy,Sell]
[0.05,0.15,0.3]of volume
add Hold
```

Rewards=



```
reward
    profit made/lost
```

f)Q Learning : an off-policy efficient solution



Algorithm 3-1. Q-learning (off-policy TD)

```
input: a policy that uses the action-value function, $\pi(a \mid s, Q(s,a))$
Initialize $Q(s,a) \leftarrow 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
loop: for each episode
    Initialize environment to provide $s$
    do:
        Choose $a$ from $s$ using $\pi$, breaking ties randomly
        Take action, $a$, and observe $r, s'$
        $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \operatorname*{argmax}_{a_s \in \mathcal{A}(s)} Q(s', a_s) - Q(s,a)\right]$
        $s \leftarrow s'$
    while $s$ is not terminal
```

g) the Markov decision Process

A graph of simple MDP (case of restorcking with probability p)

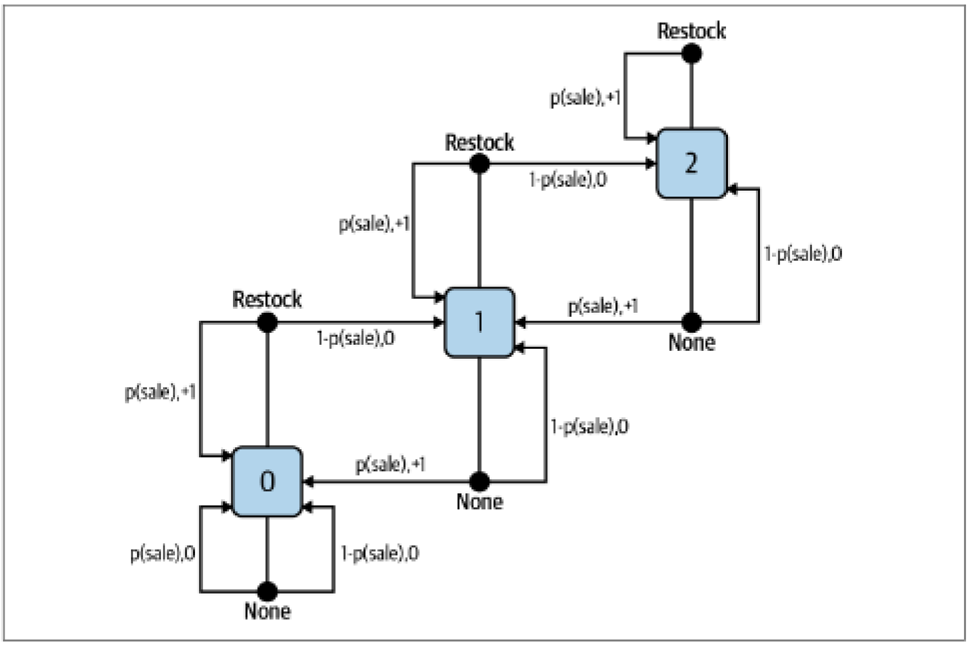
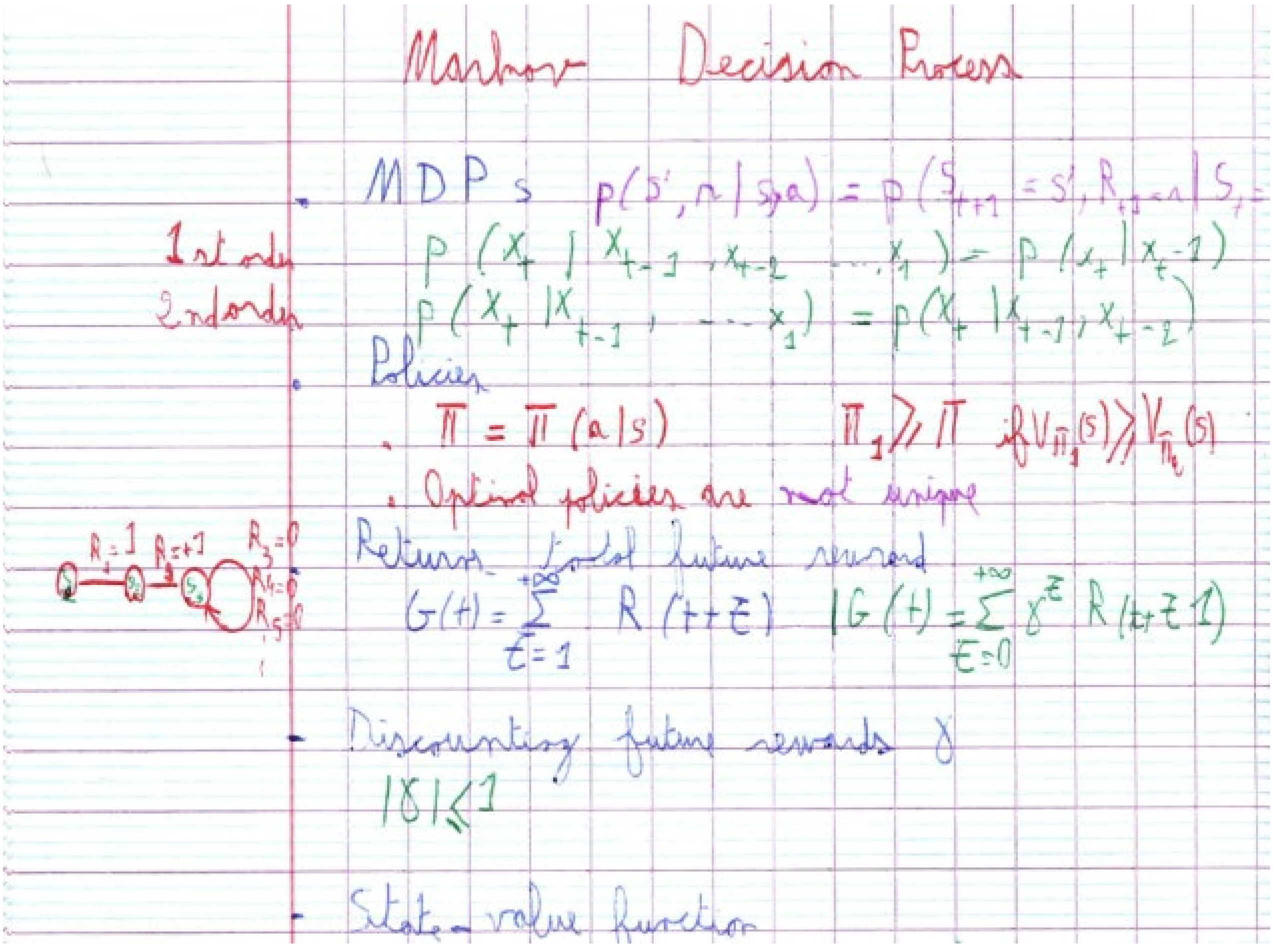


Figure 2-5. A graph representing the transition probabilities of the simple inventory problem.

h)RL Overview



$R_1=1$
$R_2=+1$
$R_3=0$
$R_4=0$
$R_5=0$
S_1
S_2
S_3

Markov Decision Process

- MDPs $p(s', r \mid s, a) = p(S_{t+1} = s', R_{t+1} = r \mid S_t =$

1st order $p(X_t \mid X_{t-1}, X_{t-2}, \ldots, X_1) = p(X_t \mid X_{t-1})$

2nd order $p(X_t \mid X_{t-1}, \ldots X_1) = p(X_t \mid X_{t-1}, X_{t-2})$

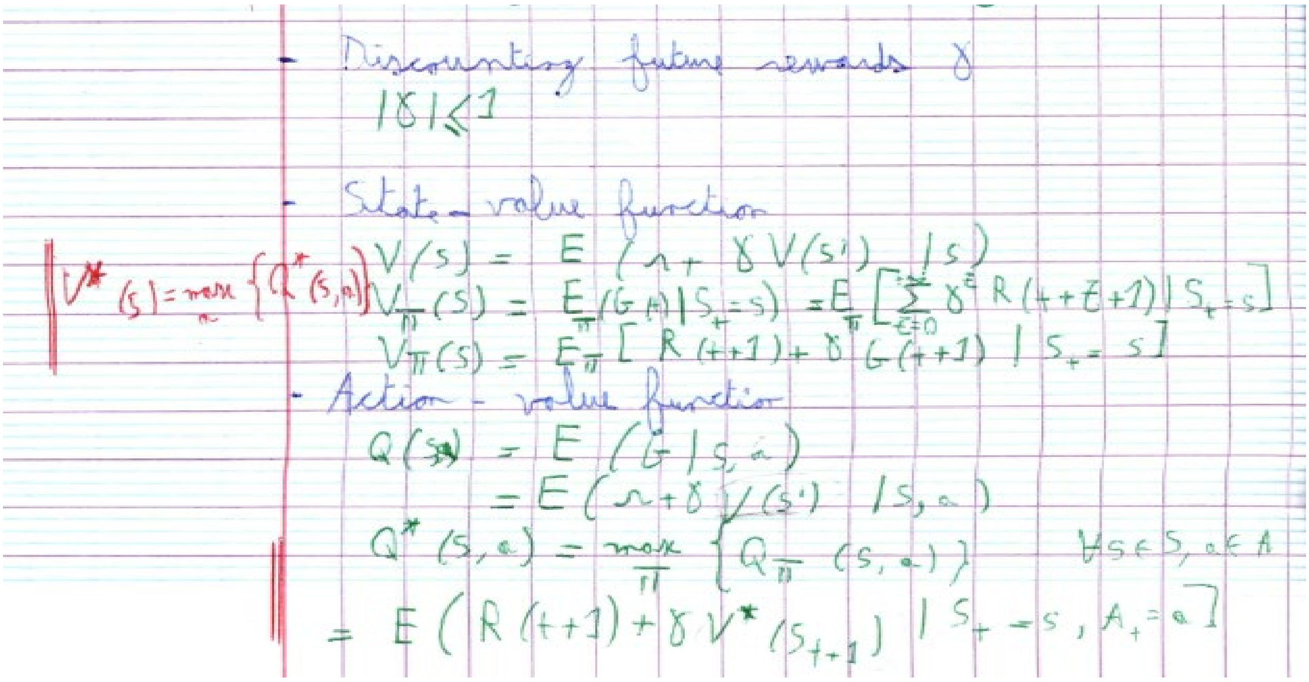
- Policies
  - $\pi = \pi(a \mid s)$ $\pi_1 \geqslant \pi$ if $V_{\pi_1}(s) \geqslant V_{\pi_2}(s)$
  - Optimal policies are not unique
- Return total future reward

  $G(t) = \sum_{\tau=1}^{+\infty} R(t+\tau)$ $\mid G(t) = \sum_{\tau=0}^{+\infty} \gamma^{\tau} R(t+\tau+1)$

- Discounting future rewards γ

  $|\gamma| \leqslant 1$

- State-value function



- Discounting future rewards γ

  $|\gamma| \leqslant 1$

- State-value function

$V^*(s) = \max_a \{Q^*(s,a)\}$

$V(s) = E(r + \gamma V(s') \mid s)$

$V_{\pi}(s) = E_{\pi}(G(t) \mid S_t = s) = E_{\pi}\left[\sum_{\tau=0}^{+\infty} \gamma^{\tau} R(t+\tau+1) \mid S_t = s\right]$

$V_{\pi}(s) = E_{\pi}[R(t+1) + \gamma G(t+1) \mid S_t = s]$

- Action-value function

$Q(s,a) = E(G \mid s, a)$

$= E(r + \gamma V(s') \mid s, a)$

$Q^*(s,a) = \max_{\pi} \{Q_{\pi}(s,a)\}$ $\forall s \in S, a \in A$

$= E(R(t+1) + \gamma V^*(S_{t+1}) \mid S_t = s, A_t = a]$

Chapter III:
Creating an expert adviser on the MetaQuotes Trading platform

III)Creating an expert adviser on the MetaQuotes Trading platform

1)Introducing the platform

Established in 2000, MetaQuotes is considered as one of the leading developers of software applications for brokerages, banks, and exchanges. The company's representative offices are located in many countries around the world.

MetaQuotes

The company has developed a series of popular software products, from a simple FX Charts platform to the MetaTrader 5 multi-asset trading platform.

MetaQuotes' first product FX Charts was released in the year 2000. This easy-to-use platform allowed performing trading operations and analyzing currency pair quotes in the Forex market. FX Charts was a truly revolutionary and highly functional product available at a reasonable price as opposed to its expensive and low-performance counterparts. The new product quickly gained a favorable market share and was implemented in several brokerage firms.

2)Creating the EA that will handle Algorithmic trading

Explination of the various parts of an Expert Adviser program are described here :

https://www.mql5.com/en/articles/100

The top part (Header) of the code is where the property of the EA is defined. You can see that here are the values you filled in the MQL5 Wizard in figure 3.

In this section of the code, you can define additional parameters like *description* (brief text description of the EA), declare constants, include additional files or import functions.

When a statement begins with a # symbol, it is called a preprocessor directive and it does not end with a semicolon ‘;’ other example of preprocessor directives includes:

#define :

The **#define** directive is used for a declaration of constants. It is written in the form

#define *identifier token_string*

What this does is substitute every occurrence of *identifier* in your code with the value *token_string*.

it will replace every occurrence of COMPANY_NAME with the string "MetaQuotes Software Corp." or it will replace every occurrence of ABC with the char (or integer) 100 in your code.

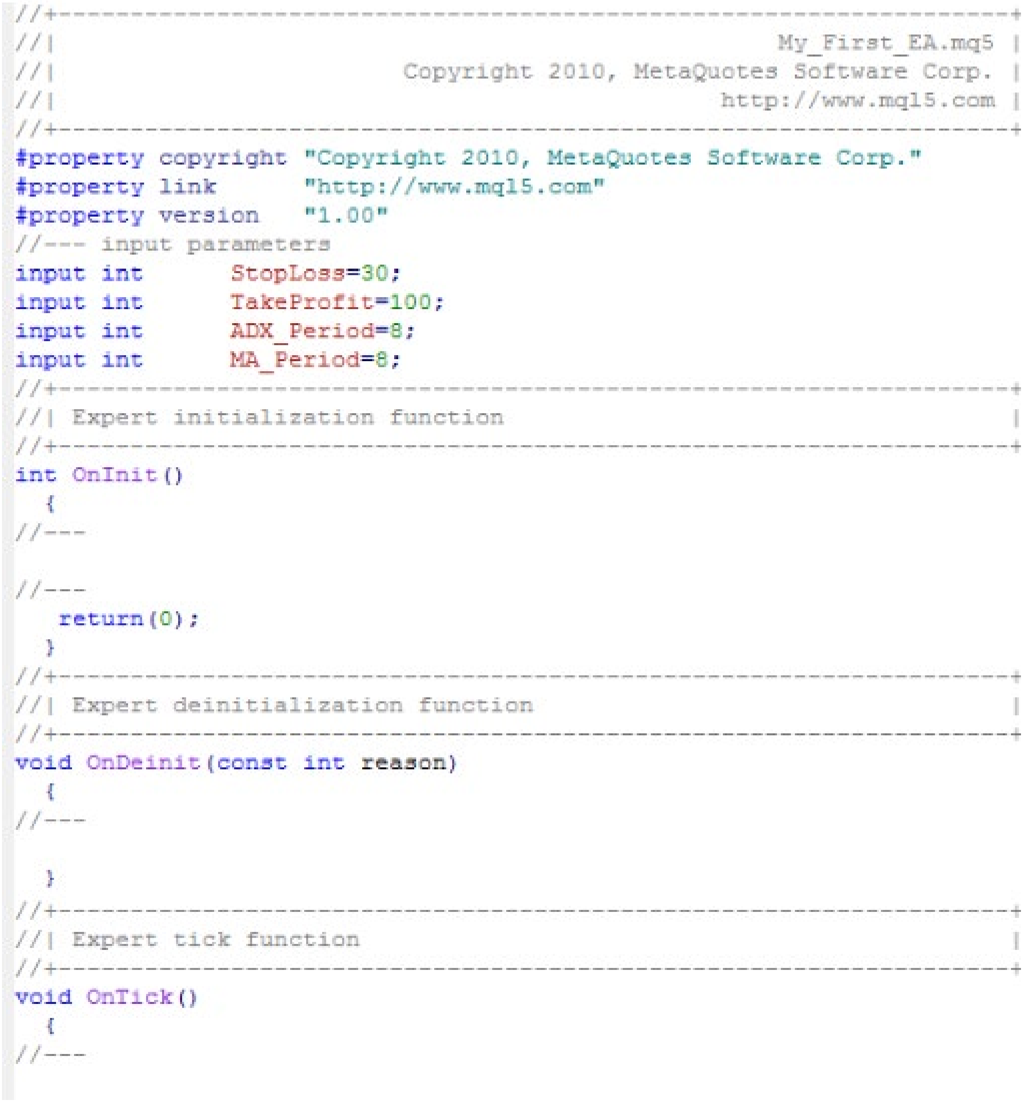
You can read more about the preprocessor directives in the MQL5 Manual. Let us now continue with our discussion.

The second part of the header of our code is the input parameters section

We specify all parameters, which will be used in our EA at this section. These include all variables that will be used by all the functions we will be writing in our EA.

Variables declared at this level are called Global Variables because they are accessible by every function in our EA that may need them. The input parameters are parameters that can only be changed outside of our EA. We can also declare other variables which we will manipulate in the course of our EA but will not be available outside of our EA in this section.

Next is the EA initialization function. This is the first function that is called when the EA is launched or attached to a chart and it is called only once.



```
//+------------------------------------------------------------------+
//|                                                  My_First_EA.mq5 |
//|                        Copyright 2010, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2010, MetaQuotes Software Corp."
#property link      "http://www.mql5.com"
#property version   "1.00"
//--- input parameters
input int      StopLoss=30;
input int      TakeProfit=100;
input int      ADX_Period=8;
input int      MA_Period=8;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---

//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---

  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
```

On Deinit:

This section is the best place to make some important checks in order to make sure our EA works very well.
We can decide to know if the chart has enough bars for our EA to work, etc.
It is also the best place to get the handles we will be using for our indicators (ADX and Moving Average indicators).
The OnDeinit function is called when the EA is removed from the chart.
For our EA, we will release the handles created for our Indicators during the initialization in this section.

OnTick :

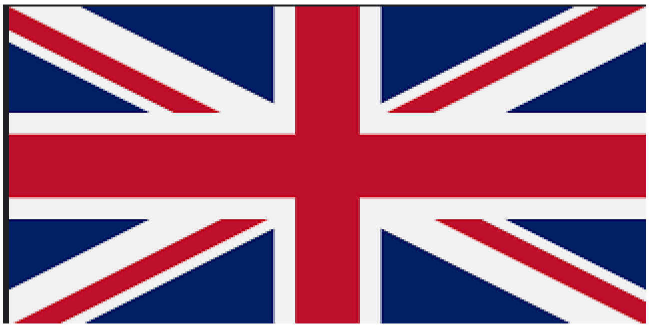
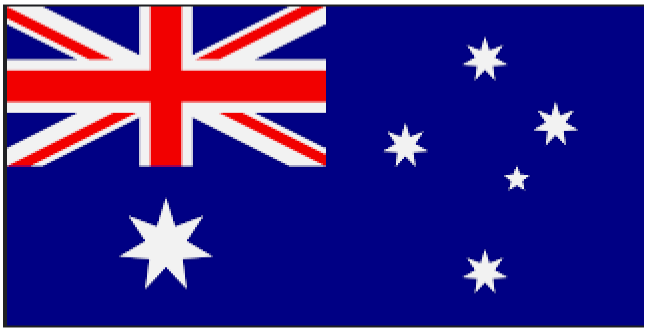
This function process the NewTick event, which is generated when a new quote is received for a symbol.

Note, that Expert Advisor cannot perform trade operations if the use of Expert Advisors in the client terminal is not allowed (Button "Auto Trading").

3)Choice of the Symbols

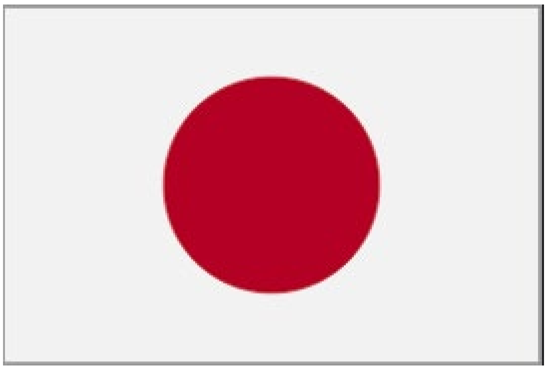
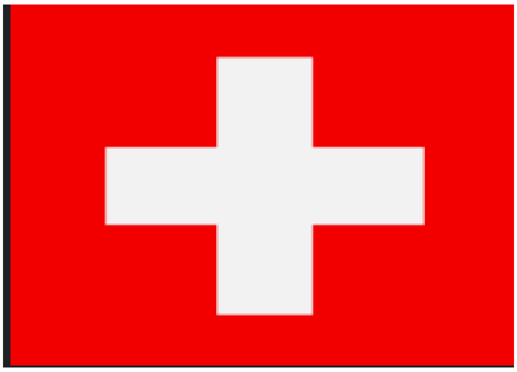
{"GBPAUD":"GBPAUD=X",

Great English speaking Countries with opposite pasts of stability

"JPYCHF":"JPYCHF=X",

One is an economical powerhouse of production and the other is a known fiscal hideout with political neutrality

"USDEUR":"EUR=X"

The most commonly currencies in the first world with heavy shares of the world market and deciding political weight

}

4) MetaApi for placing orders and fetching data:

About the Solution

Today companies have understood the interest of digital and data sharing, or even the integration of API's, but they are still reluctant to open their data, fearing for their technology. They're not going to help other companies' teams integrate their API.

Based on this observation, Meta-API is a solution offering IT teams a simple and effective solution to gather data from different actors in the same place and process information to extract only the necessary, thus creating new uses adapted to their products while ensuring stability and scalability through monitoring the proper transmission of information and automating the correction of problems coming from the instability of market API's.

Here are some key features:

- Authentication: we handle all kinds of authentication: OAuth 2, API key, headers, etc.;
- More than 23,000 API's waiting to connect via our public connectors;
- Import your own API's using the Open API Specifications standard;
- A fully integrated development environment with dedicated interface to manage connectors and a JavaScript / TypeScript code editor;
- Deploy your code in one click and in a few seconds;
- Automatic monitoring of your code and all third-party connectors and APIs you use.

IV)Room for improvement

- ✓ Streamz to automate the work and python script preparation we did with Apache Airflow
- ✓ Dask to make computations Synchronized and updates after every step more effecient
- ✓ Save summary to folder with Google drive
- ✓ Using LSTMs instead of SARIMAX
- ✓ Using Monte Carlo methods instead of TD-lambda (Q-learning)

V)Conclusion :

In this project , while facing difficulties and aspects we got to compare the implementation and the creation of a pipeline to solve the problem of Algorithmic trading using machine learning techniques.(Influx db, Apache Airflow)

We arrived to a point where the infrastructure for the models is ready and more familiarity with the tools needed was important .

We verified that the Arima model family was powerful on it's own and even overfits with exogenous variables and taking the seasonal component into account , we also tried to remedy the difference if bias of 'seller's fear' pertaining to currency owners refusing to sell in prospects of losing out.

We viewed the CNN model as a pattern-matching application capable of extracting feature maps from even 1D inputs (which is mathematically easier than working on 2 or 3 D tensors) , we also experimented with different Activation functions and found that the BRU excelled in dealing with Brownian movement .

We successfully defined the environment for our Reinforcement agent as well as chose a suitable algorithm capable of reaching our Goal.

Most importantly , last but not least we had experience building pipelines and connecting pieces of code when working with unfamiliar APIs.

In Finality , this project is open for more implementation and work on my part and I will carry it out to the end but the framework we established so far allowed us to take bigger better steps into the future.

❖ References and webography :

- 7 Database paradigms : https://www.youtube.com/watch?v=W2Z7fbCLSTw
- Explore data using InfluxQL : https://docs.influxdata.com/influxdb/v1.8/query_language/explore-data/
- Why and how to make a requirements.txt file

https://boscacci.medium.com/why-and-how-to-make-a-requirements-txt-f329c685181e

- use Google Drive for Desktop

https://support.google.com/drive/answer/10838124?visit_id=637894194561384862-4101061013&rd=1

- how to setup apache Airflow

https://www.freecodecamp.org/news/how-to-use-apache-airflow-to-manage-workflows/?fbclid=IwAR3-jwzHMCQRsErz6JYEV9D1dMnr-9cx7IVka5SFBoTonGSSZwRNSDRkt_Y

- why does stationarity matter https://boscacci.medium.com/why-and-how-to-make-a-requirements-txt-f329c685181e
- Tensorflow et Keras: L'intelligence artificielle appliqué à la robotique humanoide
- Oreilly , Reinforcement Learning Applications of intelligent Agents
- Bionodal-root-units https://github.com/marek-kan/Bionodal-root-units/blob/master/BRU_ConV.ipynb
- Rules when forecasting ARIMA https://people.duke.edu/~rnau/arimrule.htm

(This list is not exhaustive , please see attached file)