Dauphine | PSL
UNIVERSITÉ PARIS

Python Quantitative

Analyse Quantitative et Machine Learning : Optimisation de stratégie de trading à l'aide d'indicateurs issus d'une étude

Université Paris-Dauphine – PSL
Janvier 2025

Giraud-Liansot Thomas
Régi Théo
Negre Arthur

Introduction

Nous avons, avant toute chose, sélectionné une étude sur des indicateurs techniques pour le trading afin de mener une analyse approfondie de cette étude à travers un environnement de machine Learning.

Le papier de recherche que nous avons étudié et testé à travers notre modèle de machine Learning propose une étude comparative du MACD et tend à développer un indicateur plus avancé et robuste : le VPVMA. En effet, la recherche étudiée s'inscrit dans une démarche de compréhension et d'amélioration des stratégies de trading technique.

De ce fait, en exploitant des outils avancés de machine Learning, nous allons analyser la performance des indicateurs étayée par le papier de recherche. En effet, l'objectif principal du papier de recherche est de comprendre quels sont les indicateurs qui peuvent augmenter la sensibilité des signaux de trading pour capturer des opportunités de marchés tout en réduisant des erreurs potentielles. Par ailleurs, la ligne directrice de la recherche est d'abord d'analyser le MACD avec les règles de signaux associés à cet indicateur bien connu, puis de tenter d'améliorer celui-ci en le combinant à d'autres indicateurs techniques comme le RSI, MFI et les bandes de Bollinger. Le projet tend ensuite à développer un nouvel indicateur qui permet d'améliorer la performance globale en prenant en compte le volume de trading et la volatilité des prix. En effet, le papier de recherche étudié souhaite améliorer l'indicateur MACD ainsi que ses compères en développant un indicateur plus élaboré : le VPVMA, qui a pour but d'accroître la sensibilité et la précision des signaux de trading, tout en réduisant les faux signaux.

Ainsi, nous allons dans un premier temps, nous attarder sur la conception dans notre modèle des indicateurs décrit dans le papier de recherche, en étant le plus fidèle possible à ce qui est décrit dans celui-ci. A la suite de la création de ces indicateurs déjà plus ou moins connu, nous allons tenter de recréer l'indicateur VPVMA dans notre environnement de machine Learning afin qu'il soit, une nouvelle fois, au plus proche possible du papier de recherche. Cet indicateur tire parti des données de prix, de volumes, et de volatilité pour fournir des signaux plus adaptés aux dynamiques des marchés financiers.

Cela nous permettra par la suite d'effectuer une analyse quantitative sur les indicateurs ainsi déterminés dans le papier de recherche afin d'évaluer leur performance dans un environnement de machine Learning et leur capacité à améliorer ou non la précision de notre modèle. L'objectif étant de tester les indicateurs du papier de recherche, ainsi que l'indicateur VPVMA ayant pour but d'améliorer au mieux les signaux de trading. Par ailleurs, cette étude aura pour but de déterminer les outils les plus pertinents et performants pour optimiser des stratégies de trading par le biais de notre environnement de Machine Learning.

Partie Indicateur :

Indicateur MACD

Le MACD (Moving Average Convergence Divergence) est un indicateur technique basé sur les moyennes mobiles. L'objectif principal du MACD est de détecter les changements dans la force, la direction, la durée et la dynamique d'une tendance sur les marchés financiers. Le MACD est conçu sur deux moyennes mobiles exponentielles (EMA) : une courte (*short_period*) et une longue (*long_period*). Par ailleurs, le *signal_period* instancié à 9 dans le papier de recherche et dans notre code permet le calcul de la ligne de signal, soit la différence entre la EMA longue et courte.

```
class IndicatorMACD(Indicator):  8 usages
    def __init__(self, short_period=12, long_period=26, signal_period=9):
        super().__init__()
```

L'indicateur MACD repose sur les calculs des EMA pour chaque nouvelle donnée de prix. L'indicateur repose donc sur la formule suivante :

$$EMA(t) = \alpha \times price(t) + (1 - \alpha) \times EMA(t-1) \text{ avec } \alpha = 2 / (\text{période} + 1).$$

Ainsi, l'indicateur que nous avons implémenté produit une EMA courte et longue pour chaque nouveau prix. Ces EMA sont mises à jour dynamiquement, garantissant une sensibilité accrue à l'évolution des tendances de prix.

```
def calculate_macd(self, mid_price: float):  1 usage
    if self.__short_ema is None:
        self.__short_ema = mid_price
        self.__long_ema = mid_price
    else:
        alpha_short = 2 / (self.short_period + 1)
        alpha_long = 2 / (self.long_period + 1)
        self.__short_ema = alpha_short * mid_price + (1 - alpha_short) * self.__short_ema
        self.__long_ema = alpha_long * mid_price + (1 - alpha_long) * self.__long_ema
```

Afin de calculer la ligne MACD, nous faisons la différence entre l'EMA courte et l'EMA longue. Cette différence est stockée dans notre variable *__macd_line*. De cette façon, nous pouvons calculer à partir de cette base, la ligne de signal qui correspond à une moyenne mobile de la ligne MACD, calculée sur un nombre de périodes prédéfinis en input de l'indicateur dans la variable *signal_period*.

```
# Limiter la taille de __macd_line
if len(self.__macd_line) > self.signal_period:
    self.__macd_line.pop(0)

if len(self.__macd_line) >= self.signal_period:
    self.__signal_line = sum(self.__macd_line) / self.signal_period
    self.__histogram = macd_line_value - self.__signal_line
```

Par conséquent, la différence entre la ligne MACD et cette moyenne mobile sur la ligne MACD (la ligne de signal), nous permet de construire l'histogramme.

$$Hist = MACD - Signal$$

Avec cet indicateur, différentes règles de signalisations existent. Celles-ci sont détaillés dans le papier de recherche.

- Le "signal line crossover" permet, dès lors que la ligne MACD croise la ligne de signal à la hausse, de signaler un ACHAT ou une VENTE dans le cas contraire.
- Le "zero crossover", quant à lui, signale un ACHAT dès lors que la ligne MACD passe au-dessus de zéro, et vente lorsqu'elle passe en dessous.
- L'« histogramme trends » dégage un signal d'achat lorsque l'histogramme montre une croissance soutenue et un signal de vente lorsqu'il montre une baisse.

Ainsi, les différents signaux que peut générer cet indicateur à la suite de ses différents résultats, peuvent améliorer notre modèle de Machine Learning. En effet, les résultats sont ressortis à travers notre code grâce à l'usage de *get_current_value (),* tout comme les autres indicateurs. Cette fonction permet de retourner les valeurs actuelles du MACD, soit la ligne MACD (denier point), ligne de signal et l'histogramme. Ces données sont ainsi utilisées par notre modèle pour tester l'efficacité de l'indicateur MACD dans la prédiction des tendances de marché et peut, si l'indicateur est robuste, améliorer notre environnement de prédiction.

Indicateur RSI :

L'objectif de notre projet est de suivre au plus près le papier de recherche étudié avec les indicateurs qui y sont proposés. Ainsi nous avons, à la suite du MACD, implémenté un nouvel indicateur : le RSI. Le RSI est un indicateur technique utilisé pour évaluer si un actif est suracheté ou survendu. Il oscille entre 0 et 100 avec des zones typiques de surachat (>70) et de survente (<30). C'est un oscillateur de Momentum qui mesure la force relative des hausses et des baisses sur une période donnée.

Les formules de bases du RSI sont les suivantes :

$$RSI = 100 - \frac{100}{1 + RS}$$

$$RS_t = \frac{Average\ Gain_t}{Average\ Loss_t}$$

La formule Average Gain ou Loss montre que le premier gain ou perte moyen correspond à la moyenne des gains ou pertes de l'actif sur les 14 premières périodes (par défaut). Par ailleurs, ces moyennes sont calculées de manière pondérée en utilisant la moyenne précédente et la valeur actuelle. Pour ce faire et afin de rester au plus proche de notre papier de recherche, nous avons codé notre RSI en fonction de ces équations.

```
# Mise à jour des moyennes pondérées après la première période
self._avg_gain = (self._avg_gain * (self.__period - 1) + gain) / self.__period
self._avg_loss = (self._avg_loss * (self.__period - 1) + loss) / self.__period
```

Nous allons par ailleurs, dans notre fonction *get_current_value(),* procéder au calcul du ratio RS qui correspond à la division des gains moyens par les pertes moyennes (comme indiqué dans la formule du papier de recherche ci-dessus). Ainsi, le ratio RS nous permet de calculer, in fine, le RSI, qui est renvoyé par la méthode *get_current_value()* afin que notre modèle puisse déterminer si nous sommes dans une zone de surachat ou de survente.

```
rs = avg_gain / avg_loss
rsi = 100 - (100 / (1 + rs))
print(f"Calculated RSI: {rsi}, Avg Gain: {avg_gain}, Avg Loss: {avg_loss}")
return rsi
```

Indicateur Bandes de Bollinger :

Cet indicateur implémente le concept des bandes de Bollinger, un outil classique de l'analyse technique qui est repris dans le papier de recherche. Les bandes de Bollinger fonctionnent autour de moyennes mobiles simples (SMA) sur une période donnée et sont augmentées/diminuées d'un multiple de l'écart type. Dans le code, l'indicateur est enrichi par l'ajout d'une largeur de Bandes de Bollinger (BBW) et par le calcul de 2 SMA (courtes et longues) sur la largeur des bandes. Ces calculs permettent de capter des signaux basés sur les divergences entre les bandes courtes et longues, comme soulignés dans le papier.

Par conséquent, en suivant le papier de recherche, nous avons calculé le *upper_band* et le *lower_band*. Ces bandes permettent de modéliser correctement les bandes autour de la moyenne mobile.

```
    lower_band = moving_average - self.__multiplier * std_dev
    upper_band = moving_average + self.__multiplier * std_dev
```

La largeur des bandes ou Bollinger Bandwidth (BBW) est calculée comme suit dans notre code ainsi que dans le papier de recherche, sachant que la Middle Band, ici, est une moyenne mobile sur les prix de clôtures et sur une période prédéfinie en input de la fonction (comme mentionné dans le papier de recherche). Cependant, dans notre code, nous avons remplacé les prix de clôture par le *mid-price,* étant donné que nous ne disposons pas de données spécifiques sur les prix de clôture. Le mid-price est calculé comme étant une moyenne entre le prix bid et le prix ask.

$$BBW = \frac{Upper\ Band - Lower\ Band}{Middle\ Band}$$

Cela nous permet ensuite de calculer la SMA courte et longue sur la largeur des bandes (BBW). Ces deux moyennes mobiles permettent de détecter des divergences ou des croisements dans la largeur des bandes, comme suggéré dans le papier. Ainsi, ces croisements peuvent être interprété comme des signaux de changement de tendance ou de volatilité dans notre modèle de machine Learning.

Indicateur Money Flow Index (MFI):

Cet indicateur implémenté à notre code est une version adaptée de l'indice de Flux Monétaire (Money Flow index, MFI). Le MFI est conçu pour identifier des situations de Momentum sur le marché en combinant les prix et les volumes. Il est souvent utilisé pour détecter des zones de surachat et de survente ainsi que pour anticiper des retournements de tendance potentielle, tout comme l'indicateur RSI a tendance à le faire (voir précédemment).
Le papier de recherche décrit une version du MFI basé sur les volumes exécutés pour estimer les flux monétaires entrants et sortants. Cependant, notre code ne dispose pas directement des volumes exécutés. C'est pourquoi, nous sommes passés par une approche différente, tout en gardant une proximité avec le travail de recherche étudié, en estimant les flux monétaires à partir des variations de volumes visibles dans le carnet d'ordres.

De plus, notons que la formule présentée dans le papier de recherche est la suivante :

$$Money\ Flow = Tyipical\ Price * Volume$$

$$Typical\ Price = \frac{High + Low + Close}{3}$$

Par ailleurs, nous avons pris un prix médian calculé à partir d'une moyenne sur le prix bid et ask issus du carnet d'ordres à la place de la formule *Typical price*. Cela permet de garder un reflet de la dynamique de marché en temps réel. Le volume total, quant à lui, est obtenu en additionnant les quantités Bid et Ask issus du carnet d'ordres.

```
if self._order_book.get_current_snapshot(True) and self._order_book.get_current_snapshot(False):
    # Calculate the mid-price
    mid_price =  (self._order_book.get_best_price(True) + self._order_book.get_best_price(False))/ 2

    # Calculate the total volume in the order book
    total_volume = self._order_book.get_best_quote(True).get_amount() + self._order_book.get_best_quote(False).get_amount()
```

Par la suite, le code va calculer les flux monétaires positifs et négatifs en fonction des variations de volumes entre deux périodes consécutives. Si une variation de volume est positive, elle contribue au flux monétaire positif et si elle est négative elle contribue au flux monétaire négatif.

Par conséquent, une fois les flux négatifs et positifs cumulés sur la période, le ratio des flux monétaires est calculé :

```
# Avoid division by zero
if negative_money_flow == 0:
    self.current_mfi = 100.0
else:
    money_flow_ratio = positive_money_flow / negative_money_flow
    self.current_mfi = 100.0 - (100.0 / (1 + money_flow_ratio))
```

Ainsi, nous avons calculé le MFI qui, tout comme le RSI, nous permet de dégager des zones de surachat et de survente sur le marché.

Indicateur Parabolic Stop and Reverse (SAR):

Cet indicateur est conçu pour détecter les tendances du marché en fonction de la dynamique des prix et de leur rapidité de variation. Il se base sur la notion du facteur d'accélération (AF), qui évolue dynamiquement en fonction des points extrêmes atteint par les prix (« Extreme price »). Une nouvelle fois, cet indicateur a pour but de détecter des signaux de renversement de tendances. En effet, lorsque le prix d'un actif augmente, la valeur du SAR suit une valeur d'accélération donnée qui augmente avec le prix des actifs, et inversement si le prix diminue.

Ainsi, notre indicateur suit la logique tel qu'énoncé avec les formules suivantes issus du papier de recherche :

$$SAR_t = SAR_{t-1} + \alpha * (EP - SAR_{t-1}) \text{ uptrend}$$

$$SAR_t = SAR_{t-1} - \alpha * (SAR_{t-1} - EP) \text{ downtrend}$$

Lors de la première mise à jour, le SAR est initialisé avec le prix issu de la *quote* entrante, et le facteur d'accélération, quant à lui, est réglé à sa valeur initiale (AF), défini par α ci-dessus et initialisé à 0.02 dans le papier de recherche et dans notre code également.

Lorsque la tendance est à la hausse (« up »), le SAR est mis à jour en ajoutant le produit du facteur d'accélération de l'écart entre le point extreme et le SAR actuel.

```
# Always update SAR based on the current trend
if self.current_trend == "up":

    # Check for new high and adjust EP and AF
    if price > self.extreme_point:
        self.extreme_point = price
        self.af = min(self.af + 0.02, self.max_AF)
        # Update SAR for up trend
        self.current_sar += self.af * (self.extreme_point - self.current_sar)
```

Comme nous pouvons le voir ci-dessus, si un nouveau point haut est atteint, le point extreme est mis à jour, et l'AF augmente de manière dynamique, dans la limite définie par *max_AF* (le maximun étant initialisé à 0.2 dans le papier de recherche et dans notre code).

Par ailleurs, si les prix passent sous le SAR, une inversion de tendance est détectée. Le SAR devient alors égal au dernier point extrême, la tendance passe à « down » et les paramètres du facteur d'accélération sont réinitialisés (à 0.02 comme nous pouvons le voir-ci dessous) afin de commencer une nouvelle tendance.

```
# Check for trend reversal
if price < self.current_sar:
    # Reverse trend to "down"
    self.current_trend = "down"
    self.current_sar = self.extreme_point
    self.extreme_point = price
    self.af = 0.02
```

Par conséquent, de manière contraire à ce que nous venons d'énoncer, lorsque la tendance est détectée à la baisse (« down »), le SAR diminue en fonction de l'AF et de l'écart entre le SAR et le point extrême jusqu'à arriver à une nouvelle tendance haussière.

Ainsi, le SAR détecte les renversements lorsque le prix croise un point extrême, en cohérence avec la méthodologie du papier de recherche. Les points extrêmes et l'AF, dans le cas d'un croisement, sont réinitialisés pour s'adapter à la nouvelle tendance.

Indicateur Volume Price Volume Moving Average

Nous en venons au dernier indicateur, et pas des moindres, de notre papier de recherche qui a pour but d'améliorer l'efficacité des autres méthodes générés, notamment celle du MACD. En effet, cet indicateur est conçu pour capturer des dynamiques de prix et de volumes en utilisant des moyennes pondérées. Cet indicateur inclut 2 moyennes mobiles comme le MACD et permet de dégager un signal en se basant sur des croisements entre la ligne VPVMA et une ligne de signal calculée à partir de cette dernière.

Comment fonctionne, dans le papier de recherche, l'indicateur et comment nous l'avons implémenté dans notre code ?
L'indicateur se montre relativement robuste en vue des differents calculs intermédiaires qu'il nécessite.

En effet, le calcul du VPVMA suit une logique basée sur la différence entre deux moyennes pondérées ajustés par la volatilité, à savoir E*SVMap* et *ELVMap*. Afin d'arriver à cette étape, nous devons d'abord calculer le *Short/Long Volume-Weighted Moving average* qui calcule la moyenne pondérée sur une période courte (12 dans notre code) et longue (26 dans notre code). Or, tout comme nous l'avons vu précédemment, ces formules se base sur le *Typical Price* (voir ci-dessous) que l'on a dû approximer étant donné que nous n'avions pas de prix haut, bas et close dans notre *order_book*. Ainsi, tout comme vu précédemment, nous avons choisi le *mid-price*, correspondant à la moyenne du prix *Bid* et du prix *Ask*, afin de s'approcher le plus possible de la méthode détaillée par le papier de recherche.

$$Typical\ Price(TP) = \frac{High + Low + Close}{3}$$

$$SVWMA = \frac{\sum_i^n TP_i * V_i}{\sum_i^n V_i}\ where\ n = short\ term\ period$$

$$LVWMA = \frac{\sum_i^n TP_i * V_i}{\sum_i^n V_i} where\ n = long\ term\ period$$

Nous avons ainsi pu dans un premier temps calculer le *SVWMA* et *LVWMA*. Ces valeurs, comme on peut le voir dans les formules ci-dessus, sont calculées en pondérant les prix par les volumes.

```
# Calcul SVWMA et LVWMA
self.svwma = self.calculate_svwma(self.fast_period) if len(self.prices) >= self.fast_period else None
self.lvwma = self.calculate_svwma(self.slow_period) if len(self.prices) >= self.slow_period else None
```

```
def calculate_svwma(self, period):  2 usages
    if len(self.prices) < period:
        print(f"Pas assez de données pour SVWMA (period: {period}).")
        return None
    weighted_price_sum = sum([p * v for p, v in zip(self.prices[-period:], self.volumes[-period:])])
    volume_sum = sum(self.volumes[-period:])
    return weighted_price_sum / volume_sum if volume_sum != 0 else None
```

La suite du calcul se fait par le biais de la volatilité quotidienne que l'on a dû une nouvelle fois approximer comme étant l'écart-type des *mid-price* dans notre modèle.

$$Daily\ Volatility(DV) = Std(High, Low, Close, Open)$$

Nous pouvons grâce à cette dernière formule arriver à notre dernière étape primordiale pour le calcul du VPVMA et de la ligne de signal. D'abord, nous commençons par établir la Enhanced Short/Long Volume Moving Average Product (*ESVMap* et *ELVMap*) qui correspond à une moyenne mobile du produit de la SVWMA, respectivement LVWMA, et de la volatilité. Ainsi, dans notre code, et dans le papier de recherche, on calcule les points énoncés ci-dessus en utilisant une EMA (Exponential Moving Average). La formule est, comme nous l'avons vu précédemment, la suivante :

$$EMA(t) = \alpha \times price(t) + (1 - \alpha) \times EMA(t-1) \text{ avec } \alpha = 2/(periode+1).$$

Nous avons intégré ce calcul de l'EMA dans notre code à travers une fonction disponible dans notre indicateur afin de pouvoir calculer les *ESVMap* et *ELVMap* directement en appelant cette fonction dans *incoming_quote.*

```
def calculate_ema(self, values, period):  2 usages
    if len(values) < period:
        print(f"Pas assez de données pour EMA (period: {period}, values: {len(values)}).")
        return None
    alpha = 2 / (period + 1)
    ema = values[0]
    for value in values[1:]:
        ema = alpha * value + (1 - alpha) * ema
    return ema
```

La fonction prend en compte en entrée l'historique des résultats de *SVWMA X DV* et *LVWMA X DV* ainsi que la période de temps, longue ou courte, afin de ressortir le résultat des EMA courtes et longues sur ces valeurs.

```
# Calcul ESVMap et ELVMap
if self.svwma and self.lvwma and self.dv:
    self.esvmap_history.append(self.svwma * self.dv)
    self.elvmap_history.append(self.lvwma * self.dv)

    if len(self.esvmap_history) >= self.fast_period:
        self.esvmap = self.calculate_ema(self.esvmap_history, self.fast_period)

    if len(self.elvmap_history) >= self.slow_period:
        self.elvmap = self.calculate_ema(self.elvmap_history, self.slow_period)
```

Ceci nous permet de calculer les *ESVMap* et *ELVMap*, comme nous l'indique le calcul issu du papier de recherche. Les EMA permettent ainsi de lisser ces valeurs sur les périodes de temps définies (*period* dans la fonction). Ainsi, une fois réalisé, ceci nous permet de calculer le fameux VPVMA, qui résulte de la différence entre le *ESVMap et le ELVMap.* Ce résultat est par la suite stocké dans une liste *self.vpvma_histogram* qui va être utilisé afin de calculer la ligne de signal qui correspond à une moyenne glissante des valeurs récentes VPVMA calculés sur une période de signal (qui est égale à 9 dans notre code et dans le papier de recherche) entrée en input de la classe *IndicatorVPVMA.*

```
# Calcul VPVMA et signal_line
if self.esvmap is not None and self.elvmap is not None:
    self.vpvma = self.esvmap - self.elvmap
    self.vpvma_histogram.append(self.vpvma)

    if len(self.vpvma_histogram) >= self.signal_period:
        self.signal_line = np.mean(self.vpvma_histogram[-self.signal_period:])
```

En effet, le code intègre une moyenne arithmétique sur les n dernières valeurs (où n = *signal_period*) inclus dans la liste qui regroupe toutes les VPVMA calculés. Ensuite, à travers la variable *self.signal_line* où a été effectué le calcul, ainsi que les VPVMA calculés et le band width en input de l'indicateur, un signal de trading peut être dégagé en fonction de ce que nous pouvons voir à la *Table 8* issu du papier de recherche ci-dessous. Ainsi, ce signal (d'achat ou de vente) est issu des paramètres initialement entrés en input de notre indicateur ainsi que des valeurs qui auront été calculés à travers celui-ci.

Table 8: Parameters and trading signal identification rule for the VPVMA indicator

Indicator	**Parameter**	**Trading signal identification rule**
VPVMA	$Window_{fast} = 12$ $Window_{slow} = 26$ $window_{sign} = 9$ bandwidth= 0.1	Buy: (VPVMAt > (1+band width) * VPVMASt) & (VPVMAt-1 <=VPVMASt-1) Sell: (VPVMAt < (1-band width *2) VPVMASt) & (VPVMAt-1 <= VPVMASt-1)

Par conséquent, comme nous l'indique le papier de recherche, ces différents calculs rend l'indicateur VPVMA plus complexe contrairement à l'indicateur MACD initialement conçu. En effet, *"It takes price factor, time factor, volume factor and price volatility factor into account so that the sensitive moment of price and time trend can be captured correctly".* En prenant en compte des éléments comme la volatilité quotidienne et la moyenne mobile pondérée par les volumes, le VPVMA vise à mieux refléter les conditions de marché et à offrir une meilleure prédiction des opportunités de trading.

Ainsi, après avoir détaillés et intégrés nos indicateurs issus du papier de recherche dans notre modèle de Machine Learning, nous allons pouvoir détailler l'utilisation de ces indicateurs par notre réseau de neurones et comment ceux-ci peuvent affecter favorablement ou non notre environnement de code.

Partie 2 : La mise en place des indicateurs dans notre modèle de Machine Learning

Pour commencer, nous avons implémenté les indicateurs cités précédemment, avec plusieurs variantes, en modifiant les inputs. A ce stade, les résultats seront issus de la configuration suivante :

indicators_set_up.py > ...

```python
from indicator_RSI import IndicatorRSI
from indicator_parabolic_stop_reverse import IndicatorSAR
from indicator_money_flow_index import IndicatorMoneyFlowIndex
from indicator_VPVMA import IndicatorVPVMA

# Chosen set of indicators
INDICATORS: tuple = (  IndicatorMACD(6, 13, 4),
                       IndicatorMACD(12, 26, 9),
                       IndicatorMoneyFlowIndex(14),
                       IndicatorMoneyFlowIndex(28),
                       IndicatorBollingerBands(9, 2, 9, 18),
                       IndicatorBollingerBands(12, 2, 12, 24),
                       IndicatorSAR(0.01, 0.2, -0.2, 12),
                       IndicatorSAR(0.02, 0.2, -0.2, 12),
                       IndicatorRSI(7),
                       IndicatorRSI(14),
                       IndicatorMovingAverageOnPrice(9),
                       IndicatorMovingAverageOnPrice(12),
                       IndicatorBestBidOfferVariance(),
                       IndicatorQuantityOfQuotesInBook(),
                       IndicatorMovingAverageOnAmount(9),
                       IndicatorVPVMA(12,26,9,0.1),
                    )
```

Les différents indicateurs sont détaillés avec leur description dans le tableau en *Annexe 1*. Ce tableau représente notre premier mapping des indicateurs avec leurs indices qui ressortent dans les features.

Pour commencer, nous obtenons les résultats de fiabilité du modèle suivants :

```
Test accuracy: 48.13%. Goal: 100%.
1/1 - 0s - 92ms/step
```

EUR_USD first accuracy test.

```
Test accuracy: 50.0%. Goal: 100%.
1/1 - 0s - 40ms/step
```

XAU_USD first accuracy test.

Nous allons maintenant regarder la corrélation entre chaque features pour comprendre si certains indicateurs sont redondants. Nous entendons par là, que l'on cherche à comprendre si certains indicateurs émette des signaux similaires dans le modèle, et si par conséquent, il serait possible de se passer de certaines variables pour éviter de créer du bruit et de la confusion dans le modèle de Machine Learning. Nous commençons par cette étape, car nous nous attendons à ce que des variables comme la SMA des Bandes de Bollinger et les Moving Averages on Prices influencent le modèle de manière similaire.
Nous extrayons les données du «.pkl » générés et générons une matrice de corrélation entre les features, avec un scrypt python. Nous observerons si les résultats sont les mêmes sur les séries EUR_USD (*Annexe 2*) et sur les séries de données XAUUSD (*Annexe 3*).

Analyse de *l'Annexe 2* :

Nous pouvons remarquer premièrement que les features 6 (MFI avec 14 periodes), 7 (MFI avec 28 périodes) et 28 (Quantity_of_quotes_in_book) ne nous ont renvoyés aucun résultat. En regardant dans le détail, nous avons observé que la variance de valeurs renvoyés par ces features est égale à 0 avec le set de données USD_EUR utilisé. Pour les deux indicateurs MFI, il est possible que les périodes d'observations ne soient pas appropriées, soit car ne conviennent pas à la période d'observation du

cours (LookBack_Time), soit car ne gardent pas assez de valeurs en mémoire pour permettre des calculs pertinents. Il conviendra de tester différents paramètres d'entrés (de longueur) et différents LookBack_Time pour observer si cela se résout. Pour la quantité de quotes en book, il est possible que cette quantité ne soit pas assez volatile pour générer des résultats dans le modèle.

Des résultats sont flagrants car certains indicateurs ont une corrélation très forte entre eux :
La MACD Line et MACD Signal Line sont quasiment interprétées de manière similaire par le modèle. Les deux indicateurs MACD calculés sur des périodes différentes sont également très corrélés, mais pas de manière quasi parfaite comme la MACD Line et MACD Signal Line pour une même période. Il serait pertinent de retester avec deux périodes différentes, mais en augmentant l'écart. Néanmoins, il semble pertinent de garder uniquement la Signal Line ou la MACD Line pour une période. L'histogramme est légèrement négativement corrélé avec les deux autres valeurs de l'indicateur MACD. Il semble donc pertinent de le garder.

Les indicateurs MACD et RSI sont très fortement corrélés entre eux. Augmenter la période d'observation du RSI (période MACD inchangée), amplifie d'ailleurs ce phénomène. En analyse technique, il est courant que les « traders » techniques utilisent les deux ensembles, l'un pour confirmer le signal de l'autre. Pour implémenter dans un modèle de Machine Learning, il serait préférable d'éviter ce cumul.

On remarque également que l'histogramme de l'indicateur MACD et la Signal Line de la VPVMA se recoupent dans leurs résultats. Néanmoins, il semblerait qu'une MACD (12,26,9) et un indicateur VPVMA (12, 26, 9, 0.1) soit moins corrélés (avec même une corrélation négative de -0.5) qu'avec un MACD (6, 13, 4).

Si l'on observe les Bandes de Bollinger, un premier résultat (qui semblait déjà évident), est que les moyennes mobiles sur les prix et la moyenne mobile des Bandes de Bollinger sont redondants. Il faut donc garder un seul des deux indicateurs. Au sein des mêmes indicateurs Bandes de Bollinger, on retrouve une forte corrélation de la Band Width avec les valeurs de bandes hautes et basses. On peut éliminer ces deux dernières variables et ne garder que la variable Bands Width comme feature pour le modèle. L'indicateur Stop and Reverse semble générer exactement les mêmes signaux que les variables moyenne mobile, borne haute et borne basse des bandes de Bollinger. Il est pertinent ici de faire un choix entre les indicateurs.

Finalement, on remarque que l'indicateur Variance Best Bid Best Offer est modérément à fortement corrélé avec la BB Width, Short et Long SMA. Etonnamment, les corrélations de Variance Best Bid est toujours plus faible que les corrélations de Variance Best Offer. Enfin, il semble qu'augmenter les périodes d'entrées dans les Bandes de Bollinger ne déclenche pas systématiquement des résultats différents. Il serait intéressant de faire retourner le modèle en augmentant la différence de temps.

Il convient de vérifier si ces résultats sont confirmés par l'Annexe 3, tableau similaire sur les données XAU_USD.

Analyse de l'Annexe 3 :

Au vu des résultats de l'Annexe 3, nous pouvons commencer par observer que l'on récupère des valeurs pour les indicateurs MFI. La série de données XAU_USD semble être plus adaptée à cet indicateur, possiblement en raison de différences de prix plus grandes dans les périodes observées ou de volumes plus volatils. Nous observons d'ailleurs les résultats suivants :

	Feature_6	Feature_7	Feature_28
count	11712.0	11712.0	11712.0
mean	100.0	100.0	4.0
std	0.0	0.0	0.0
min	100.0	100.0	4.0
25%	100.0	100.0	4.0
50%	100.0	100.0	4.0
75%	100.0	100.0	4.0
max	100.0	100.0	4.0

```
Feature_6     0
Feature_7     0
Feature_28    0
dtype: int64
Feature_6     0.0
Feature_7     0.0
Feature_28    0.0
```

	Feature_6	Feature_7	Feature_28
count	22811.000000	22811.000000	22811.0
mean	52.019533	50.435261	2.0
std	13.021744	8.265801	0.0
min	3.225840	1.886738	2.0
25%	45.454558	46.383336	2.0
50%	50.819724	50.120458	2.0
75%	55.964063	53.571525	2.0
max	100.000000	100.000000	2.0

```
Feature_6     0
Feature_7     0
Feature_28    0
dtype: int64
Feature_6     169.565805
Feature_7      68.323474
Feature_28      0.000000
```

Features 6/7/28 .describe, .isna().sum() et .var on EUR_USD (Image gauche), XAU_USD (Image droite)

Ainsi, nous voyons que les indicateurs semblaient bloqués à une valeur de 100 lors du passage sur le jeu de données EUR_USD, ce qui n'est plus le cas sur le jeu de données XAU_USD. Nous observons d'ailleurs des variances significatives. Néanmoins, nous ne retrouvons pas de résultat sur l'indicateur Quantity of quotes in book.

Dans un second temps, on confirme l'interprétation similaire par le modèle pour l'histogramme MACD et l'indicateur VPVMA (Variables VPVMA Value et VPVMA Signal Line), mais la décorrélation Signal Line et MACD avec les deux variables VPVMA. Cependant, cette série de donnée tend à rapprocher les résultats de 0. Les indicateurs MACD et RSI produisent des résultats similaires sur cette nouvelle série de données également. Il semble bien pertinent de faire ici un choix.

On confirme la corrélation BB Width avec les long/short SMA, ainsi que la corrélation entre ces trois variables et l'indicateur Variance Best Bid Best Offer (plus fortes avec XAU_USD qu'avec EUR_USD). Finalement, si l'on croise les résultats obtenus précédemment avec les nouveaux, on observe que Long et Short SMA sont redondants avec les deux variables de la VPVMA.

Pour résumer, avec ces premiers résultats, pour éviter de générer du bruit dans le modèle, nous devons choisir :

- Au sein de l'indicateur MACD, on ne doit garder uniquement la MACD Line ou la Signal Line. Il est possible de garder une des deux et l'histogramme.
- Il est préférable de ne garder uniquement l'indicateur MACD ou l'indicateur RSI. Les deux produisent des signaux très similaires.
- L'histogramme MACD et la Signal Line de la VPVMA sont redondants entre eux.
- Les Bandes de Bollinger incluant une moyenne mobile prix, nous supprimerons ces dernières de nos indicateurs. Nous ferons un choix entre la BB Width et les Upper et Lower Bands. Il faut noter également la redondance entre les Bandes de Bollinger et l'indicateur SAR. On finit par noter que la BB Width pourrait remplacer l'indicateur Variance Best Bid Best Offer.

Pour faire ces premiers choix, nous allons utiliser les résumés en suivant qui présentent l'influence de chaque variable sur les labels du modèle :

CONCLUSIONS SUR LES PREMIERS RÉSULTATS

- Concernant la MACD Line et Signal Line, on remarque des résultats très différents entre les deux séries de données. Il semble que la MACD Line donne des signaux très légèrement plus fiable que la Signal Line au modèle (respectivement 0.030 contre 0.026 et 0.039 contre 0.037 sur l'EUR_USD, mais -0.018 contre -0.015 et -0.028 contre -0.030 sur XAU_USD).
- Nous n'incluront plus les Moyennes Mobiles sur les prix lors des prochains tests car génèrent les mêmes signaux que celles des Bandes de Bollinger, avec les mêmes impacts sur les Labels. En suivant le même résonnement, nous ne conserverons pas l'indicateur SAR.
- Finalement, nous conserveront la BB Width plutôt que la Variance Best Bid Offer, les bornes supérieures et inférieures des Bandes de Bollinger

Indices	Corr Labels EUR_USD	Corr Labels XAU_USD
0	0.030	-0.018
1	0.026	-0.015
2	0.015	-0.010
3	0.039	-0.028
4	0.037	-0.030
5	-0.008	0.006
6		0.035
7		0.015
8	0.093	-0.102
9	0.093	-0.102
10	0.093	-0.102
11	-0.013	-0.002
12	-0.010	0.012
13	0.000	0.016
14	0.093	-0.102
15	0.093	-0.102
16	0.093	-0.102
17	-0.015	0.003
18	-0.004	0.014
19	-0.004	0.016
20	0.091	
21	0.092	
22	0.035	-0.022
23	0.038	-0.022
24	0.093	-0.102
25	0.093	-0.102
26	-0.008	0.013
27	0.010	0.019
28		
29	0.006	-0.062
30	-0.022	-0.001
31	-0.024	0.000

Normalisation et pré-processing :

Après l'étape précédente, nous avons maintenant relancé le calcul des features-labels avec le fichier set_up_indicator comme suit :

```
# Chosen set of indicators
INDICATORS: tuple = (
                        IndicatorMACD(24, 52, 18),
                        IndicatorMACD(12, 26, 9),
                        IndicatorMoneyFlowIndex(14),
                        IndicatorMoneyFlowIndex(28),
                        IndicatorMovingAverageOnAmount(9),
                        IndicatorBollingerBands(9, 1, 9, 18),
                        IndicatorBollingerBands(20, 2, 10, 50),
                        IndicatorVPVMA(12,26,9,0.1),
                    )
```

Nous avons récolté un nouveau set de features et labels. On remarque que les features sortent sur des échelles très différentes. Certaines features ont des valeurs comprises entre 0 et 100 (comme l'indicateur MFI), d'autres sur une échelle de prix (comme les movings average), nous obtenons par exemple ce type de tableau (en colonne les features, en row les valeurs d'observations).

	3	4	5	6	7
0	-0.245803	40.8014	52.4386	203.333	2723.65
1	-0.206881	42.1847	52.626	203.333	2723.64
2	-0.172672	43.6391	53.1118	240	2723.65
3	-0.113846	49.8239	50.0552	210	2723.68
4	-0.0942054	43.7194	50.4915	246.667	2723.7
5	-0.0804271	44.8175	50.9169	210	2723.72
6	-0.0713137	45.8769	50.4786	210	2723.73
7	-0.0643867	43.8888	47.3265	230	2723.74

Pour faciliter la compréhension et l'apprentissage du modèle, nous décidons d'appliquer une normalisation des données. Pour ce faire, nous utiliserons les fonctions de « scikit-learn » et ses « preprocessing tools ». Nous déterminons si certains indicateurs sont distribués normalement avec un test de Jacque-Bera (*Annexe 4*). Nous observons qu'aucun indicateur retournant une variance significative n'est normalement distribué, ce qui semble tout à fait normal dans un contexte où les données affichent une importante volatilité. Par conséquent, nous choisirons de normaliser les données avec un MinMaxScaler de Scikit-learn. Nous avons construit une fonction normalize_features qui est implémentée comme suit :

```
normalized_features_labels = normalize_features(processed_features_labels)

total_lines_features_labels = len(processed_features_labels[0][0])
```

```
def normalize_features(processed_features_labels):
    """
    Fonction pour normaliser les valeurs de features labels grâce à la classe
    """
    normalized_features_labels = processed_features_labels
    normalizer = FeatureNormalization(processed_features_labels[0][1])
    normalized_features = normalizer.run_normalization()
    normalized_features_labels[0][1] = normalized_features

    return normalized_features_labels
```

La classe prend en entrée les processed_features_labels, transforme les features (en [0][1]) dans un format compatible et les normalise grâce à la classe FeatureNormalization que nous avons créé. Pour voir l'implémentation en détail, il convient de se référer aux codes. Mais le run_normalization résume le process.

```python
class FeatureNormalization:
    def __init__(self, features):
        self.__features = features
        self.__minmax_scaler = MinMaxScaler()

    def run_normalization(self):
        self._convert_to_array()     # Convert to NumPy array
        self._debug_features()       # Inspect the feature array
        self._clean_features()       # Clean invalid values
        self._validate_cleaning()    # Validate cleaned features
        self.__fit()                 # Fit the scaler
        normalized_features = self.__transform()  # Transform the features
        return self._convert_to_tuples(normalized_features)
```

Dans l'ordre, nous passons le features en « numpy array » pour que ce soit un format intelligible par Scikit-learn, puis nous examinons s'il n'y a pas de valeurs NaN ou Infinie. Nous supprimons ces valeurs et en informons l'utilisateur. On valide que le cleanning est bien effectué, et finalement, nous normalisons les données avant de les repasser en format tuple pour les modèles.

Cette partie n'a pas permis d'améliorer le modèle, nous évoquerons ce qui le bloque dans la prochaine partie. Néanmoins, normaliser les données devraient améliorer l'efficacité du modèle dans le cas où le problème majeur qui empêche les modèles d'apprendre venait à être résolu.

Classements de données déséquilibrées :

Il convient maintenant de passer à la phase entrainement du (des) modèle(s). Nous lançons une première fois le process pour obtenir des valeurs témoins de référence pour noter notre amélioration, ou non, et trouver les points cruciaux sur lesquels travailler. Nous utiliserons pendant cette prochaine phase le Data Set EUR_USD, car fait ressortir plus de valeurs pour les features et labels pour notre entrainement de modèle. Autrement dit, nous générons plus de signaux sur ce set de données que sur les données XAU_USD.

Nous obtenons le même résultat d'Accuracy que précédemment (**48.13%**) et notons la « confusion matrix » suivante :

```
Confusion matrix and metrics:
[[[158   0]
  [163   0]]

 [[  0 146]
  [  0 175]]]
```

Nous comprenons dans la matrice de corrélation qu'il y a un problème majeur. Le modèle classifie mal les données labels, (la class 1 correspondrait aux signaux BUY et SELL, la class 0 aux signaux [0,0] et [1,1], les DO NOTHING). Ce phénomène bloque l'apprentissage du modèle, et ce dès les premiers calculs sur les EPOCH. Nous avons d'abord tenté de réduire et modifier le nombre d'EPOCH, sans résultat. Par conséquent, nous décidons de chercher à résoudre le problème des données déséquilibrées.

Dans un premier temps, nous voulons confirmer ce déséquilibre. Pour ce faire, nous suivrons la guideline TensorFlow appliquée à nos données :

https://www.tensorflow.org/tutorials/structured_data/imbalanced_data ?hl=fr

Nous avons commencé par utiliser le code suivant, pour obtenir les résultats associés :

```python
storage_path = "C:/Users/theor/Desktop/M2/python_qa/project_pythonQA/new_version/machine_learning_backtester-main/f
#storage_path = "C:/Users/theor/Desktop/M2/python_qa/project_pythonQA/new_version/resultats/copies_models (premier
name = "2025-01-11-06-48_0.pkl"
storage_path += name

#Getting features and labels to numpy arrays
storage = FeaturesLabelsStorage()
restored_data = storage.restore_ready_features_labels(file_name=storage_path)

if not restored_data:
    raise ValueError("No data was restored. Check the storage path.")

(labels, features), _, _ = restored_data
features = np.array(features)
labels = np.array(labels)

#Classification of labels
labels_flattened = labels.reshape(-1, 2)
class_map = {
    (0, 1): "SELL",
    (1, 0): "BUY",
    (0, 0): "AMBIGUOUS",
    (1, 1): "AMBIGUOUS"
}

mapped_labels = [class_map[tuple(row)] for row in labels_flattened]

#Results
unique, counts = np.unique(mapped_labels, return_counts=True)
class_counts = dict(zip(unique, counts))
print("Class Counts:", class_counts)

neg = class_counts.get("AMBIGUOUS", 0)
pos = class_counts.get("BUY", 0) + class_counts.get("SELL", 0)
total = neg+pos

if total > 0:
    print('Examples:\n    Total: {}\n    Positive: {} ({:.2f}% of total)\n'.format(
        total, pos, 100 * pos / total))
else:
    print("No BUY or SELL examples found.")
```

```
Class Counts: {'AMBIGUOUS': 16165, 'BUY': 7822, 'SELL': 11149}
Examples:
    Total: 35136
    Positive: 18971 (53.99% of total)
```

Classification :

Univers	Count	% in univers
Tout : (Ambiguous + BUY + SELL)	35 136	100%
Ambiguous ([0,0] ou [1,1])	16 165	47.01%
BUY	7 822	22.26%
SELL	11 149	31.73%
Positives : (BUY + SELL)	18 971	100%
BUY	7 822	41.23%
SELL	1 1149	58.76%

Nous en déduisons plusieurs choses. Premièrement, les signaux « ambigus », correspondant aux labels [1,1] ou [0,0] semblent représenter une grande partie des résultats, alors les signaux « clairs » représentent 53.99% des labels. Ensuite, même si nous remarquons moins de signaux BUY que de signaux SELL, nous notons que la répartition est à peu près équilibrée. Alors que le premier constat peut causer le problème d'apprentissage, le second constat ne devrait pas bloquer le modèle. Il faut noter que les features et labels obtenus proviennent de calcul avec un LOOK_BACK_TIME de 2 minutes.

Diminuer celui-ci renforçait le déséquilibre, avec proportionnellement plus de valeurs ambigus (comme sur l'exemple ci-dessous), et ne renforce pas l'Accuracy du modèle.

```
Class Counts: {'AMBIGUOUS': 63898, 'BUY': 369, 'SELL': 1556}
Examples:
    Total: 65823
    Positive: 1925 (2.92% of total)
```

Cette conclusion est renforcée par le fait que l'on essaye de classifier de manière binaire avec notre modèle :

```
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=hp.Float('learning_rate', min_value=1e-4, max_value=1e-2,
    loss=keras.losses.BinaryCrossentropy(),
    metrics=[keras.metrics.BinaryAccuracy()]
)
```

Mais changer la loss function par *CategoricalCrossentropy()* n'améliore pas les résultats.

On commence notre tentative de résolution de ce problème avec un modèle simple, compilé de la manière suivante :

```
model.fit(x=train_dataset, epochs=constants.EPOCHS_COUNT)
```

```
model = keras.Sequential(
    [
        # Input layer: 1xN
        keras.layers.Input(shape=(input_vector_length,)),
        keras.layers.Dense(57, name="layer1", kernel_regularizer=keras.regularizers.l2(0.01)),
        keras.layers.BatchNormalization(),
        keras.layers.Dropout(0.4),
        keras.layers.Dense(57, name="layer2", kernel_regularizer=keras.regularizers.l2(0.01)),
        keras.layers.BatchNormalization(),
        keras.layers.Dropout(0.4),
        keras.layers.Dense(114, name="layer3", kernel_regularizer=keras.regularizers.l2(0.01)),
        keras.layers.BatchNormalization(),
        keras.layers.Dropout(0.4),
        keras.layers.Dense(output_vector_length, activation="softmax",  name="output_layer"),
    ]
)
```

```
model.compile(optimizer=optimizer,
              loss=keras.losses.BinaryCrossentropy(),
              metrics=[keras.metrics.BinaryAccuracy()])
```

Le choix de 57 et 114 neurones correspond à nombre d'indicateurs * 3 pour 57 et nombre d'indicateurs * 6 pour 114.

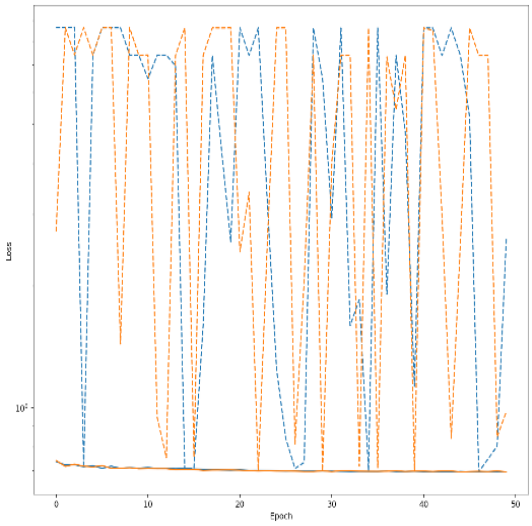
<u>Définition du biais initial :</u>

On sait que l'ensemble des données sont déséquilibrées, nous cherchons le biais initial comme sur le tutoriel. Nous obtenons une Loss Initiale de 8.8565 et un biais de 0.16, après recalcul, nous diminuons la Loss à 0.9668:

```
Loss: 8.8565
Initial Bias:  [0.16006309]
First layer: layer1
Loss: 0.9668
```

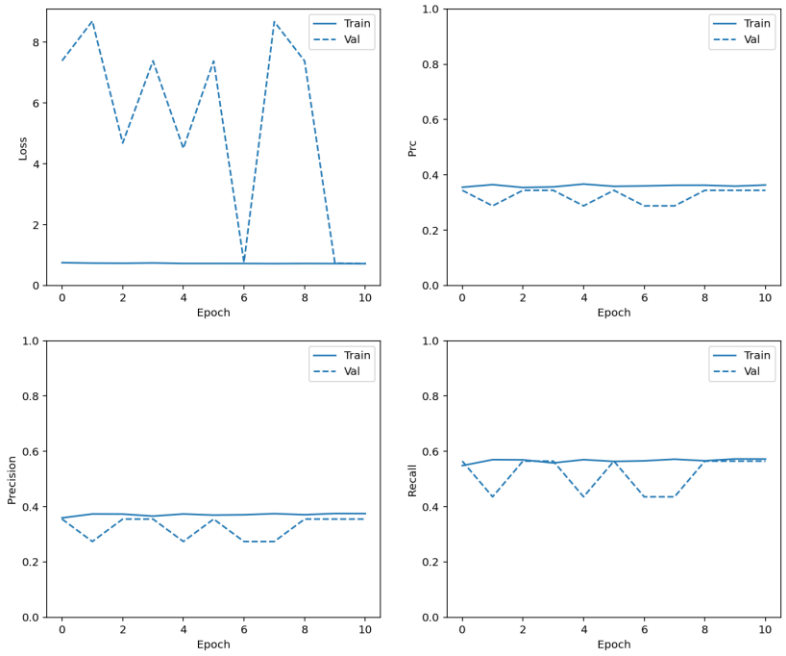
Nous sauvegardons les poids ainsi obtenus, et nous générons deux modèles avec les mêmes paramètres, seul le biais initial est ajouté.

```python
#CONFIRMATION OF BIAIS INITIALIZATION:
model = load_model(path_to_model)
model.load_weights(initial_weights)
#Loading 0 biais:
first_layer = model.layers[0]
weights, biases = first_layer.get_weights()
biases += 0                                    #Zero initial_bia
first_layer.set_weights([weights, biases])
zero_bias_history = model.fit(
    train_features,
    train_labels,
    batch_size=BATCH_SIZE,
    epochs=constants.EPOCHS_COUNT,
    validation_data=(val_features, val_labels),
    verbose=0)

#Model with Initial Bias to compare with
model = load_model(path_to_model)
model.load_weights(initial_weights)
#Loadind the biais:
first_layer = model.layers[0]
weights, biases = first_layer.get_weights()
biases += initial_bias
first_layer.set_weights([weights, biases])
careful_bias_history = model.fit(
    train_features,
    train_labels,
    batch_size=BATCH_SIZE,
    epochs=constants.EPOCHS_COUNT,
    validation_data=(val_features, val_labels),
    verbose=0)
```



De notre côté, le graph ne permet pas de conclure. Nous notons néanmoins que l'implémentation d'un biais initial peut aider le modèle à réduire la perte (image précédente). Nous lisons une forte variabilité des valeurs de pertes, indiquant que le modèle n'arrive pas à se stabiliser autour d'une valeur. Il semble que le modèle forme une sorte d'explosion dans le gradient d'apprentissage.

Nous nous demandons également pourquoi la val_loss à tendance à exploser dans le temps, et ne pas rester stable et converger vers une valeur finale. En ressortant diverses valeurs de l'history, nous obtenons :



Le graphique sur les pertes semble suggérer un problème d'« overfitting » ou de manque de généralisation. Sur ces deux possibilités, nous penchons vers la seconde option. Le graphique « PrC » fait, quant à lui, ressortir des valeurs stables et des courbes plates, indiquant que le modèle ne réussis pas à différencier les classes. Il semble qu'il se concentre uniquement sur les signaux « ambigus ». Une problématique se trouve donc bien dans le problème de déséquilibre entre les classes, avec des

signaux « ambigus » qui sont générés en grande quantité comparé aux signaux SELL and BUY. Le modèle n'arrive pas à suffisamment différencier les deux catégories, et ainsi s'entraine mal.

Calcul du poids des classes pour compenser la surreprésentation des signaux ambigus comme classe :

```
#CALCUL DES POIDS de Classes
# Scaling by total/2 helps keep the loss to a similar magnitude
# The sum of the weights of all examples stays the same.
weight_for_0 = (1 / neg) * (total / 2.0)
weight_for_1 = (1 / pos) * (total / 2.0)

class_weight = {0: weight_for_0, 1: weight_for_1}

print('Weight for class 0: {:.2f}'.format(weight_for_0))
print('Weight for class 1: {:.2f}'.format(weight_for_1))
```

```
In [24]: weight_for_0 = (1 / neg) * (total / 2.0)
    ...: weight_for_1 = (1 / pos) * (total / 2.0)
    ...:
    ...: class_weight = {0: weight_for_0, 1: weight_for_1}
    ...:
    ...: print('Weight for class 0: {:.2f}'.format(weight_for_0))
    ...: print('Weight for class 1: {:.2f}'.format(weight_for_1))
    ...:
Weight for class 0: 0.52
Weight for class 1: 17.10

In [25]:
```

Nous cherchons à calculer des poids à donner au modèle en input pour qu'il surinterprète les labels class 1 et sous interprète ceux de class 0. Un poids supérieur à 1 demande au modèle de surinterpréter, et inférieur à 1 de sous interpréter. Nous lisons qu'il faudrait donner 0.52 à la class 0 et 17.10 à la classe 1. Or, après passage des poids à ce modèle, nous n'obtenons pas directement résultats plus cohérents.

Une possibilité serait également de changer le type de loss function et metric lors de la compilation. Un test est fait avec :

```
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=hp.Float('learning_rate', min
    loss=keras.losses.CategoricalCrossentropy(),
    metrics=[keras.metrics.CategoricalAccuracy()]
)
return model
```

Où nous obtenons :

```
Test accuracy: 72.9%. Goal: 100%.
1/1 - 0s - 27ms/step
```

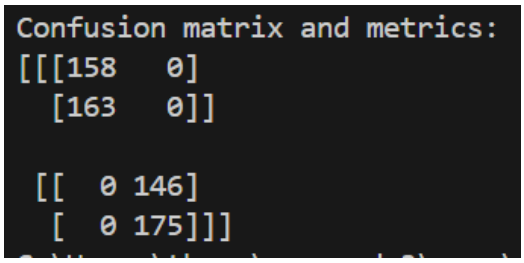
```
Confusion matrix and metrics:
[[[  0 158]
  [  0 163]]

 [[146   0]
  [175   0]]]
```

Bien que l'Accuracy soit élevée, la confusion matrix fait toujours ressortir ce problème de classification des labels. Nous ne pouvons conclure à une solution sur cette base. Le problème majeur de notre modèle n'étant pas résolu.

<u>Construction d'une classe FeaturesLabelModificator :</u>

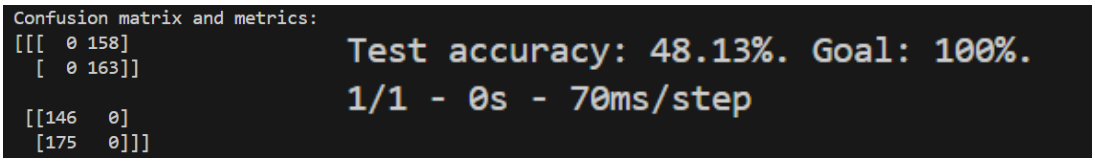
Avant de chercher à définir les paramètres optimaux avec les méthodes d'optimisation. Nous souhaitons « débloquer » l'apprentissage du modèle en rééquilibrant les classes. Ainsi, nous implémentons diverses techniques pour modifier les features et labels avant de les donner au(x) modèle(s). L'ensemble de ces fonctionnalités sont regroupées dans une classe FeaturesLabelsModificator. En fonction d'un choix utilisateur (par ajustement de la variable FEATURE_LABEL_MODIFICATION_STRATEGY, dans constants.py), on laisse à l'utilisateur le choix d'une stratégie qui pourrait répondre au problème de classification. On regroupe dans cette classe, un calcul des poids à donner au modèle, la méthode SMOTE pour générer des labels/features, ainsi qu'un nettoyage complet des signaux ambigus liée aux labels [1,1].

Avec le choix « class_weights » qui cherche à calculer des poids à transmettre au modèle, nous obtenons la confusion matrix suivante :

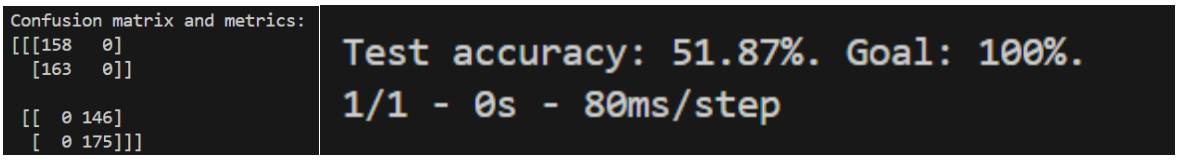


On remarque que les classes dedans sont inversés par rapport aux précédentes, mais que cela ne résout pas notre problématique, ni n'améliore l'accuracy.

Avec le choix « SMOTE », nous revenons à la confusion matrix de base, et notre accuracy rebaisse légèrement :



Avec le choix « map_label », rien de concluant non plus :

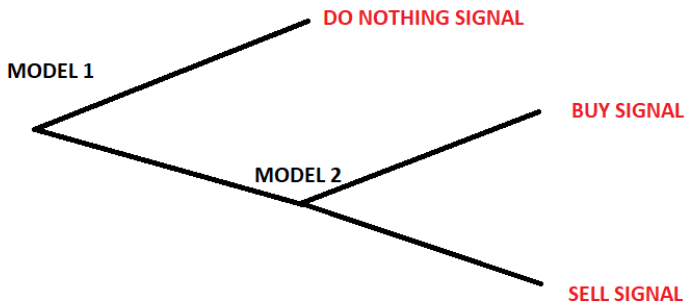


Avec ces solutions, nous restons bloqués sur une valeur de précision, avec une classification qui nous semble problématique.

Nous pouvons cependant retenir quelques points clefs que nous pourrions utiliser dans un modèle où l'apprentissage se verrait débloqué :

- Une normalisation doit être appliquée aux données, et pour cela une classe et des fonctions ont été créé. Ces fonctionnalités sont directement utilisables par le code. Aucun de nos indicateurs sont distribués normalement (et c'est très rare pour des séries financières). Par conséquent, nous appliquons la méthode MinMaxScaler de Scikit.
- Un biais initial pourrait être donné à la première couche du réseau de neurone pour faire commencer le modèle avec une perte plus faible, lui permettant de converger vers un résultat plus précis. Ce biais est calculé par un rapport des labels signaux (BUY + SELL)/(DO NOTHING = [0,0] + [1,1]). Le biais n'est pas directement implémenté dans ce code, car une dernière piste non implémentée et non testée sera discutée à la suite.
- Trois méthodes pour contrer le déséquilibre de classification sont implémentées, avec un calcul des poids de classes optimaux, une méthode SMOTE pour regénérer synthétiquement des échantillons positifs, et une méthode de mapping des signaux [1,1], par essence très ambigus car suggère un achat et une vente simultanée, en signaux [0,0]. Finalement, dans les tests de modèles simple (get_model_prototype_simple), nous avons testé une fonction de perte focal_loss avec différents paramètres Betas et Alpha sans succès pour forcer à punir d'avantage la mauvaise classification des signaux ambigus comme signaux positifs.
- Bien que l'entrainement du modèle fut impossible, toutes ces démarches ont permis de valider quelques bases citées précédemment pour l'entrainement d'un modèle efficient. Finalement, il serait possible de rendre l'architecture plus rapide et plus précise avec un système à deux

étages. Un premier modèle classifierait en sortie TAKE ACTION or DO NOTHING, et un second BUY or SELL :



Ce type d'architecture n'a pas été implémentée et testée pour une raison de temps, et de besoin de modifier de nombreuses parties de code. Pour ce faire, il aurait été nécessaire d'implémenter un double calcul des features et labels, différenciant deux séries pour chacun des modèles (avec une série pour le modèle 1 qui regroupe toutes les features/labels, et une pour le modèle 2 avec uniquement les labels et features lors de BUY/SELL SIGNAL).

Conclusion

Malgré nos différentes tentatives pour implémenter des solutions, nous ne parvenons pas à résoudre les problèmes de classification des labels fournies au modèle. Celui-ci bloque sur une accuracy de **51.87%** au maximum. Des tentatives de créer un nouveau modèle depuis zéro, ou de modifier de nombreuses variables dans constants.py sont restées infructueuses. Nous ne pouvons donc malheureusement pas conclure sur la viabilité d'une potentielle stratégie et sur la capacité de l'indicateur VPVMA (ainsi que celles des autres indicateurs techniques) pour fournir des signaux de tradings cohérents à un modèle de Machine Learning.

Il n'est pas impossible que l'ensemble de nos problèmes viennent du fait que les indicateurs de trading technique ne sont pas cohérents ou ne reflètent pas des mouvements de marchés avant que ceux-ci arrivent.

Pour conclure, il existe deux possibilités vis-à-vis de l'impossibilité d'entrainer le modèle de machine learning. Une mauvaise compréhension de l'architecture et des méthodes d'intégrations des features et labels au modèle de machine learning, générant ainsi une problématique de classification efficace des labels par le modèle car celles-ci ne sont pas correctement préprocessées. La deuxième possibilité est la limite des indicateurs de trading technique qui ne reflètent pas du tout ou pas suffisamment la réalité des mouvements sous-jacents sur les marchés des devises et de l'or. Nous souhaitons rappeler que l'étude prise pour introduire ce projet ne conclut pas sur une méthode ou stratégie très performante. Il est possible que les rendements positifs présentés par l'auteur en fin de papier de recherche découlent plus d'un bon « money management » (placement des Stop Loss et Take Profits) qui statistiquement, même avec 50% de chances de prévisions, peut générer des gains.

Annexe 1 : Mapping des indicateurs par rapport aux indices Features :

Le tableau suivant récapitule les Indices Features par rapport aux indicateurs intégrés dans un premier temps.

Indice Features	Indicateur	Description Indicateur	Variable de l'indicateur
0	MACD	MACD_6_13_4	MACD Line
1	MACD	MACD_6_13_4	Signal Line
2	MACD	MACD_6_13_4	Histogram
3	MACD	MACD_12_26_9	MACD Line
4	MACD	MACD_12_26_9	Signal Line
5	MACD	MACD_12_26_9	Histogram
6	MFI	MFI_14	Money Flow Index
7	MFI	MFI_28	Money Flow Index
8	Bande Bollinger	BOLL_9_2_BBW_9_18	Lower band
9	Bande Bollinger	BOLL_9_2_BBW_9_18	Moving Average
10	Bande Bollinger	BOLL_9_2_BBW_9_18	Upper Band
11	Bande Bollinger	BOLL_9_2_BBW_9_18	BB Width
12	Bande Bollinger	BOLL_9_2_BBW_9_18	Short SMA
13	Bande Bollinger	BOLL_9_2_BBW_9_18	Long SMA
14	Bande Bollinger	BOLL_12_2_BBW_12_24	Lower band
15	Bande Bollinger	BOLL_12_2_BBW_12_24	Moving Average
16	Bande Bollinger	BOLL_12_2_BBW_12_24	Upper Band
17	Bande Bollinger	BOLL_12_2_BBW_12_24	BB Width
18	Bande Bollinger	BOLL_12_2_BBW_12_24	Short SMA
19	Bande Bollinger	BOLL_12_2_BBW_12_24	Long SMA
20	Stop and Reverse	SAR_0.01_0.2_-0.2_12	SAR Value
21	Stop and Reverse	SAR_0.02_0.2_-0.2_12	SAR Value
22	Relative Strengh Index	RSI_7	RSI Value
23	Relative Strengh Index	RSI_14	RSI Value
24	Moving Avg on Price	MA_PX_9	Moving Average value
25	Moving Avg on Price	MA_PX_12	Moving Average value
26	Variance Best Bid/Offer	VAR_BBID_BOFFER_10	Var Best Bid
27	Variance Best Bid/Offer	VAR_BBID_BOFFER_10	Var Best Offer
28	Quantity of quotes in book	QTY_QUOTES_ORD_BOOK_{}	Quantity Quotes in Book
29	Moving Avg on Amount	MA_AMT_9	Moving Average Amount
30	Volume Price weigthted Avg	Volume Price-Weighted Moving Average (12,26,9,0.1)	VPVMA value
31	Volume Price weigthed Avg	Volume Price-Weighted Moving Average (12,26,9,0.1)	Signal Line VPVMA

Annexe 2 : Matrice de corrélation entre les features pour les données EUR_USD :

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1.00	0.98	-0.02	0.74	0.53	0.45			0.03	0.03	0.03	0.01	0.02	0.03	0.03	0.03	0.03	0.01	0.03	0.05	0.00	0.00	0.59	0.86	0.03	0.03	0.06	0.00		0.00	-0.01	0.00
1	0.98	1.00	-0.24	0.78	0.59	0.37			0.03	0.03	0.03	-0.01	0.01	0.03	0.03	0.03	0.03	0.00	0.02	0.05	0.00	0.00	0.41	0.81	0.03	0.03	0.00	0.00		0.00	-0.01	-0.01
2	-0.02	-0.24	1.00	-0.27	-0.33	0.32			-0.01	-0.01	-0.01	0.08	0.04	0.02	-0.01	-0.01	-0.01	0.06	0.03	0.00	0.01	0.00	0.77	0.11	-0.01	-0.01	0.28	0.00		0.02	0.02	0.01
3	0.74	0.78	-0.27	1.00	0.96	-0.26			0.03	0.03	0.03	-0.01	0.00	0.02	0.03	0.03	0.03	0.00	0.01	0.05	-0.01	0.00	0.27	0.60	0.03	0.03	0.01	0.00		-0.01	-0.50	-0.51
4	0.53	0.59	-0.33	0.96	1.00	-0.51			0.03	0.03	0.03	-0.01	0.00	0.01	0.03	0.03	0.03	-0.01	0.00	0.04	-0.01	0.00	0.11	0.41	0.03	0.03	-0.01	0.00		-0.01	-0.62	-0.62
5	0.45	0.37	0.32	-0.26	-0.51	1.00			0.00	0.00	0.00	0.02	0.03	0.02	0.00	0.00	0.00	0.02	0.03	0.00	0.01	0.00	0.46	0.42	0.00	0.00	0.06	0.00		0.01	0.61	0.62
6																																
7																																
8	0.03	0.03	-0.01	0.03	0.03	0.00			1.00	1.00	1.00	0.04	0.07	0.10	1.00	1.00	1.00	0.05	0.10	0.12	1.00	1.00	0.01	0.02	1.00	1.00	0.02	0.09		0.03	0.01	0.01
9	0.03	0.03	-0.01	0.03	0.03	0.00			1.00	1.00	1.00	0.05	0.08	0.11	1.00	1.00	1.00	0.06	0.10	0.12	1.00	1.00	0.01	0.02	1.00	1.00	0.03	0.09		0.03	0.01	0.01
10	0.03	0.03	-0.01	0.03	0.03	0.00			1.00	1.00	1.00	0.06	0.08	0.11	1.00	1.00	1.00	0.07	0.10	0.13	1.00	1.00	0.01	0.02	1.00	1.00	0.03	0.10		0.03	0.01	0.01
11	0.01	-0.01	0.08	-0.01	-0.01	0.02			0.04	0.05	0.06	1.00	0.64	0.45	0.04	0.05	0.06	0.86	0.43	0.29	0.05	0.05	0.06	0.04	0.05	0.05	0.48	0.44		0.00	0.09	0.05
12	0.02	0.01	0.04	0.00	0.00	0.03			0.07	0.08	0.08	0.64	1.00	0.80	0.07	0.08	0.08	0.83	0.87	0.59	0.08	0.08	0.04	0.04	0.08	0.08	0.45	0.70		0.01	0.15	0.07
13	0.03	0.03	0.02	0.02	0.01	0.02			0.10	0.11	0.11	0.45	0.80	1.00	0.10	0.11	0.11	0.59	0.94	0.87	0.11	0.11	0.04	0.04	0.11	0.11	0.39	0.70		0.01	0.17	0.12
14	0.03	0.03	-0.01	0.03	0.03	0.00			1.00	1.00	1.00	0.04	0.07	0.10	1.00	1.00	1.00	0.05	0.09	0.12	1.00	1.00	0.01	0.02	1.00	1.00	0.02	0.09		0.03	0.01	0.01
15	0.03	0.03	-0.01	0.03	0.03	0.00			1.00	1.00	1.00	0.05	0.08	0.11	1.00	1.00	1.00	0.06	0.10	0.12	1.00	1.00	0.01	0.02	1.00	1.00	0.03	0.09		0.03	0.01	0.01
16	0.03	0.03	-0.01	0.03	0.03	0.00			1.00	1.00	1.00	0.06	0.08	0.11	1.00	1.00	1.00	0.07	0.10	0.13	1.00	1.00	0.01	0.02	1.00	1.00	0.03	0.10		0.03	0.01	0.01
17	0.01	0.00	0.06	0.00	-0.01	0.02			0.05	0.06	0.07	0.86	0.83	0.59	0.05	0.06	0.07	1.00	0.61	0.41	0.06	0.06	0.05	0.05	0.06	0.06	0.50	0.56		0.00	0.11	0.05
18	0.03	0.02	0.03	0.01	0.00	0.03			0.10	0.10	0.10	0.43	0.87	0.94	0.09	0.10	0.10	0.61	1.00	0.77	0.10	0.10	0.04	0.04	0.10	0.10	0.44	0.79		0.02	0.19	0.11
19	0.05	0.05	0.00	0.05	0.04	0.00			0.12	0.12	0.13	0.29	0.59	0.87	0.12	0.12	0.13	0.41	0.77	1.00	0.12	0.12	0.03	0.04	0.12	0.12	0.30	0.57		0.01	0.13	0.13
20	0.00	0.00	0.01	-0.01	-0.01	0.01			1.00	1.00	1.00	0.05	0.08	0.11	1.00	1.00	1.00	0.06	0.10	0.12	1.00	1.00	0.00	0.00	1.00	1.00	0.03	0.09		0.03	0.01	0.01
21	0.00	0.00	0.00	0.00	0.00	0.00			1.00	1.00	1.00	0.05	0.08	0.11	1.00	1.00	1.00	0.06	0.10	0.12	1.00	1.00	0.00	0.00	1.00	1.00	0.03	0.09		0.03	0.01	0.01
22	0.59	0.41	0.77	0.27	0.11	0.46			0.01	0.01	0.01	0.06	0.04	0.04	0.01	0.01	0.01	0.05	0.04	0.03	0.00	0.00	1.00	0.63	0.01	0.01	0.21	0.01		0.01	0.01	0.01
23	0.86	0.81	0.11	0.60	0.41	0.42			0.02	0.02	0.02	0.04	0.04	0.04	0.02	0.02	0.02	0.05	0.04	0.04	0.00	0.00	0.63	1.00	0.02	0.02	0.13	0.01		0.00	-0.02	-0.02
24	0.03	0.03	-0.01	0.03	0.03	0.00			1.00	1.00	1.00	0.05	0.08	0.11	1.00	1.00	1.00	0.06	0.10	0.12	1.00	1.00	0.01	0.02	1.00	1.00	0.03	0.09		0.03	0.01	0.01
25	0.03	0.03	-0.01	0.03	0.03	0.00			1.00	1.00	1.00	0.05	0.08	0.11	1.00	1.00	1.00	0.06	0.10	0.12	1.00	1.00	0.01	0.02	1.00	1.00	0.03	0.09		0.03	0.01	0.01
26	0.06	0.00	0.28	0.01	-0.01	0.06			0.02	0.03	0.03	0.48	0.45	0.39	0.02	0.03	0.03	0.50	0.44	0.30	0.03	0.03	0.21	0.13	0.03	0.03	1.00	0.34		0.00	0.08	0.03
27	0.00	0.00	0.00	0.00	0.00	0.00			0.09	0.09	0.10	0.44	0.70	0.70	0.09	0.09	0.10	0.56	0.79	0.57	0.09	0.09	0.01	0.01	0.09	0.09	0.34	1.00		0.01	0.17	0.10
28																																
29	0.00	0.00	0.02	-0.01	-0.01	0.01			0.03	0.03	0.03	0.00	0.01	0.01	0.03	0.03	0.03	0.00	0.02	0.01	0.03	0.03	0.01	0.00	0.03	0.03	0.00	0.01		1.00	0.01	0.00
30	-0.01	-0.01	0.02	-0.50	-0.62	0.61			0.01	0.01	0.01	0.09	0.15	0.17	0.01	0.01	0.01	0.11	0.19	0.13	0.01	0.01	0.01	-0.02	0.01	0.01	0.08	0.17		0.01	1.00	0.99
31	0.00	-0.01	0.01	-0.51	-0.62	0.62			0.01	0.01	0.01	0.05	0.07	0.12	0.01	0.01	0.01	0.05	0.11	0.13	0.01	0.01	0.01	-0.02	0.01	0.01	0.03	0.10		0.00	0.99	1.00

Annexe 3 : Matrice de corrélation entre les features pour les données XAU_USD :

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1.00	0.97	0.17	0.84	0.61	0.68	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.73	0.89	0.01	0.01	0.00	0.03		-0.01	0.00	0.00
1	0.97	1.00	-0.09	0.89	0.69	0.57	0.01	0.00	0.02	0.02	0.02	0.01	0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.58	0.85	0.02	0.01	0.01	0.03		-0.01	0.00	0.00
2	0.17	-0.09	1.00	-0.16	-0.32	0.47	0.01	-0.01	-0.02	-0.02	-0.02	0.00	-0.02	-0.01	-0.02	-0.02	-0.02	-0.01	-0.02	-0.01	0.00	0.00	0.62	0.22	-0.02	-0.02	-0.02	-0.02		0.01	-0.02	-0.01
3	0.84	0.89	-0.16	1.00	0.94	0.17	0.01	0.01	0.03	0.03	0.03	0.00	0.01	0.00	0.03	0.03	0.03	0.00	0.01	0.00	0.00	0.00	0.43	0.69	0.03	0.03	0.00	0.02		0.00	-0.16	-0.18
4	0.61	0.69	-0.32	0.94	1.00	-0.17	0.00	0.01	0.04	0.04	0.03	0.00	0.00	0.00	0.03	0.03	0.03	0.00	0.01	0.00	0.00	0.00	0.18	0.46	0.04	0.03	-0.01	0.02		0.01	-0.21	-0.25
5	0.68	0.57	0.47	0.17	-0.17	1.00	0.01	-0.01	-0.01	-0.01	-0.01	0.01	0.01	0.00	-0.02	-0.02	-0.02	0.02	0.00	0.00	0.00	0.00	0.74	0.67	-0.01	-0.02	0.01	0.02		-0.02	0.17	0.20
6	0.01	0.01	0.01	0.01	0.00	0.01	1.00	0.39	0.10	0.10	0.10	-0.02	-0.04	-0.04	0.10	0.10	0.10	-0.03	-0.04	-0.03			0.02	0.01	0.10	0.10	-0.02	-0.03		-0.02	-0.01	0.00
7	0.00	0.00	-0.01	0.01	0.01	-0.01	0.39	1.00	0.12	0.12	0.12	-0.03	-0.03	-0.03	0.12	0.12	0.12	-0.03	-0.02	-0.02			0.00	0.00	0.12	0.12	-0.03	-0.03		0.02	-0.01	0.00
8	0.01	0.02	-0.02	0.03	0.04	-0.01	0.10	0.12	1.00	1.00	1.00	0.04	0.06	0.08	1.00	1.00	1.00	0.05	0.07	0.09			0.00	0.01	1.00	1.00	0.03	0.04		0.04	0.01	0.02
9	0.01	0.02	-0.02	0.03	0.04	-0.01	0.10	0.12	1.00	1.00	1.00	0.06	0.07	0.09	1.00	1.00	1.00	0.06	0.08	0.10			-0.01	0.01	1.00	1.00	0.04	0.05		0.03	0.01	0.02
10	0.01	0.02	-0.02	0.03	0.03	-0.01	0.10	0.12	1.00	1.00	1.00	0.08	0.08	0.10	1.00	1.00	1.00	0.08	0.09	0.10			-0.01	0.01	1.00	1.00	0.05	0.06		0.03	0.02	0.02
11	0.01	0.01	0.00	0.00	0.00	0.01	-0.02	-0.03	0.04	0.06	0.08	1.00	0.64	0.47	0.04	0.06	0.08	0.88	0.42	0.33	0.00	0.00	-0.04	-0.02	0.06	0.06	0.59	0.46		-0.08	0.16	0.05
12	0.01	0.01	-0.02	0.01	0.00	0.01	-0.04	-0.03	0.06	0.07	0.08	0.64	1.00	0.80	0.06	0.07	0.09	0.85	0.87	0.63	0.00	0.00	-0.03	-0.02	0.07	0.07	0.81	0.78		-0.11	0.34	0.17
13	0.00	0.00	-0.01	0.00	0.00	0.00	-0.04	-0.03	0.08	0.09	0.10	0.47	0.80	1.00	0.08	0.09	0.10	0.61	0.94	0.89	0.00	0.00	-0.03	-0.02	0.09	0.09	0.69	0.77		-0.13	0.43	0.30
14	0.01	0.01	-0.02	0.03	0.03	-0.02	0.10	0.12	1.00	1.00	1.00	0.04	0.06	0.08	1.00	1.00	1.00	0.04	0.07	0.09			-0.01	0.00	1.00	1.00	0.03	0.04		0.04	0.01	0.01
15	0.01	0.01	-0.02	0.03	0.03	-0.02	0.10	0.12	1.00	1.00	1.00	0.06	0.07	0.09	1.00	1.00	1.00	0.06	0.08	0.10			-0.01	0.00	1.00	1.00	0.04	0.05		0.04	0.01	0.02
16	0.01	0.01	-0.02	0.03	0.03	-0.02	0.10	0.12	1.00	1.00	1.00	0.08	0.09	0.10	1.00	1.00	1.00	0.08	0.09	0.11			-0.01	0.00	1.00	1.00	0.06	0.06		0.03	0.02	0.02
17	0.01	0.01	-0.01	0.00	0.00	0.02	-0.03	-0.03	0.05	0.06	0.08	0.88	0.85	0.61	0.04	0.06	0.08	1.00	0.62	0.46	0.00	0.00	-0.03	-0.02	0.06	0.06	0.75	0.63		-0.09	0.24	0.09
18	0.00	0.01	-0.02	0.01	0.01	0.00	-0.04	-0.02	0.07	0.08	0.09	0.42	0.87	0.94	0.07	0.08	0.09	0.62	1.00	0.80	0.00	0.00	-0.03	-0.02	0.08	0.08	0.79	0.86		-0.12	0.45	0.28
19	0.00	0.00	-0.01	0.00	0.00	0.00	-0.03	-0.02	0.09	0.10	0.10	0.33	0.63	0.89	0.09	0.10	0.11	0.46	0.80	1.00	0.00	0.00	-0.02	-0.02	0.10	0.10	0.56	0.64		-0.12	0.34	0.32
20	0.00	0.00	0.00	0.00	0.00	0.00						0.00	0.00	0.00				0.00	0.00	0.00			0.00	0.00			0.00	0.00				
21	0.00	0.00	0.00	0.00	0.00	0.00						0.00	0.00	0.00				0.00	0.00	0.00			0.00	0.00			0.00	0.00				
22	0.73	0.58	0.62	0.43	0.18	0.74	0.02	0.00	0.00	-0.01	-0.01	-0.04	-0.03	-0.03	-0.01	-0.01	-0.01	-0.03	-0.03	-0.02	0.00	0.00	1.00	0.73	-0.01	-0.01	-0.04	-0.01		0.01	-0.02	-0.01
23	0.89	0.85	0.22	0.69	0.46	0.67	0.01	0.00	0.01	0.01	0.01	-0.02	-0.02	-0.02	0.00	0.00	0.00	-0.02	-0.02	-0.02	0.00	0.00	0.73	1.00	0.01	0.00	-0.03	0.00		0.01	-0.01	-0.01
24	0.01	0.02	-0.02	0.03	0.04	-0.01	0.10	0.12	1.00	1.00	1.00	0.06	0.07	0.09	1.00	1.00	1.00	0.06	0.08	0.10			-0.01	0.01	1.00	1.00	0.04	0.05		0.03	0.01	0.02
25	0.01	0.01	-0.02	0.03	0.03	-0.02	0.10	0.12	1.00	1.00	1.00	0.06	0.07	0.09	1.00	1.00	1.00	0.06	0.08	0.10			-0.01	0.00	1.00	1.00	0.04	0.05		0.04	0.01	0.02
26	0.00	0.01	-0.02	0.00	-0.01	0.01	-0.02	-0.03	0.03	0.04	0.05	0.59	0.81	0.69	0.03	0.04	0.06	0.75	0.79	0.56	0.00	0.00	-0.04	-0.03	0.04	0.04	1.00	0.91		-0.08	0.38	0.19
27	0.03	0.03	-0.02	0.02	0.02	0.02	-0.03	-0.03	0.04	0.05	0.06	0.46	0.78	0.77	0.04	0.05	0.06	0.63	0.86	0.64	0.00	0.00	-0.01	0.00	0.05	0.05	0.91	1.00		-0.09	0.44	0.25
28																																
29	-0.01	-0.01	0.01	0.00	0.01	-0.02	-0.02	0.02	0.04	0.03	0.03	-0.08	-0.11	-0.13	0.04	0.04	0.03	-0.09	-0.12	-0.12			0.01	0.01	0.03	0.04	-0.08	-0.09		1.00	-0.03	-0.02
30	0.00	0.00	-0.02	-0.16	-0.21	0.17	-0.01	-0.01	0.01	0.01	0.02	0.16	0.34	0.43	0.01	0.01	0.02	0.24	0.45	0.34			-0.02	-0.01	0.01	0.01	0.38	0.44		-0.03	1.00	0.94
31	0.00	0.00	-0.01	-0.18	-0.25	0.20	0.00	0.00	0.02	0.02	0.02	0.05	0.17	0.30	0.01	0.02	0.02	0.09	0.28	0.32			-0.01	-0.01	0.02	0.02	0.19	0.25		-0.02	0.94	1.00

Annexe 4 : Résultat des tests de normalités (avant début du preprocessing) :

	Feature	p-stat	p-value	Normality
0	Feature_0	0.982963621	5.62177E-45	No
1	Feature_1	0.984166511	9.23455E-44	No
2	Feature_2	0.969572295	3.24097E-55	No
3	Feature_3	0.972027755	1.1571E-53	No
4	Feature_4	0.942295209	8.96855E-68	No
5	Feature_5	0.95274088	1.03063E-63	No
6	Feature_6	0.825407742	1.27359E-92	No
7	Feature_7	0.778022203	1.55966E-98	No
8	Feature_8	0.937676147	2.25216E-69	No
9	Feature_9	0.937319116	1.71055E-69	No
10	Feature_10	0.937295032	1.67919E-69	No
11	Feature_11	0.846685592	1.68675E-89	No
12	Feature_12	0.894696874	8.79254E-81	No
13	Feature_13	0.935747286	5.17545E-70	No
14	Feature_14	0.937694993	2.28519E-69	No
15	Feature_15	0.93726208	1.63722E-69	No
16	Feature_16	0.93726853	1.64535E-69	No
17	Feature_17	0.867194801	4.10549E-86	No
18	Feature_18	0.906649578	4.39934E-78	No
19	Feature_19	0.945458321	1.29431E-66	No
20	Feature_20		1	Yes
21	Feature_21		1	Yes
22	Feature_22	0.967731113	2.58812E-56	No
23	Feature_23	0.979840079	7.75732E-48	No
24	Feature_24	0.937319116	1.71055E-69	No
25	Feature_25	0.93726208	1.63722E-69	No
26	Feature_26	0.597656952	5.5958E-114	No
27	Feature_27	0.593396103	2.9106E-114	No
28	Feature_28	1	1	Yes
29	Feature_29	0.978714338	8.85162E-49	No
30	Feature_30	0.514736391	4.41E-119	No
31	Feature_31	0.341155571	1.1812E-127	No

Annexe 5 : Indicateurs différents du papier de recherche mis en place dans notre environnement de Machine Learning

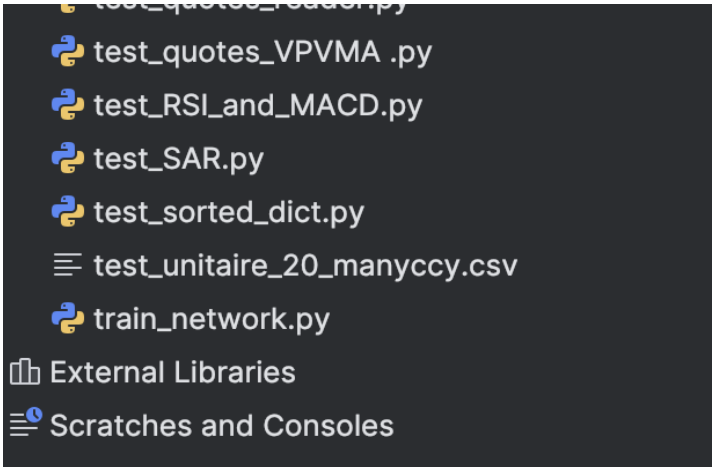
Nous avons à la suite de l'analyse du papier de recherche sélectionné, décidé de créer d'autres indicateurs afin de les implémenter dans notre environnement de Machine Learning (cela nous permet d'avoir un choix d'indicateur un peu plus large dans notre dossier où se trouve l'entièreté du code).

- Indicateur VAROC (Volatility-Adjusted Rate of Change), qui est une variation du ROC (rate of change), ajustée en fonction de la volatilité des prix sur une période donnée.

- Indicateur ADX afin d'évaluer la force d'une tendance haussière ou baissière. Il est souvent utilisé pour distinguer les phases de tendance et de consolidation.

Annexe 6 : Les tests unitaires

Chaque indicateur créé donne lieu à plusieurs tests unitaires conçus par nos soins essayant de valider le bon fonctionnement de nos indicateurs. Ils se trouvent à l'emplacement de tous les tests unitaires et sont nommés de manière claire et précise afin de comprendre ce qu'ils testent (un exemple de nom de nos fichiers tests ci-dessous).



Principale référence : Papier de recherche

- Pat Tong Chio*, June 18, 2022: "A comparative study of the MACD-base trading strategies: evidence from the US stock market"*
- ***Lien**:https://www.researchgate.net/publication/361557923_A_comparative_study_of_the_MACD-base_trading_strategies_evidence_from_the_US_stock_market*

Référence Complémentaire:

- Thierry Tshilonda, *https://www.ig.com/fr/strategies-de-trading/qu_est-ce-que-l_indicateur-cci-et-comment-lutiliser-en-trading---221209*
- Thierry Tshilonda, *https://www.ig.com/fr/strategies-de-trading/qu_est-ce-que-l_indicateur-adx-et-comment-lutiliser-pour-trader--221111*
- Thierry Tshilonda, *https://www.ig.com/fr/strategies-de-trading/qu_est-ce-que-l_indicateur-rsi-et-comment-lutiliser-en-trading---230511*
- *Thomas Giraud, https://www.lynxbroker.fr/bourse/trading/analyse-technique/indicateurs-techniques/indicateur-rsi/#:~:text=Le%20RSI%20(Relative%20Strength%20Index,le%20niveau%2080%20comme%20seuil.*
- *https://www.degiro.fr/connaissances/strategies/analyse-technique/indice-macd*
- *https://www.avatrade.fr/education/technical-analysis-indicators-strategies/macd-trading-strategies*
- *https://www.jedha.co/formation-ia/feature-engineering*
- *https://invivoo.com/blog/ameliorer-performance-prediction-obtenue-machine-learning*
- *https://www.axi.com/fr-ma/blog/education/trading-indicators*
- *https://www.lynxbroker.fr/bourse/trading/analyse-technique/indicateurs-techniques/8-indicateurs-populaires-analyse-technique/*
- *https://www.polytechnique-insights.com/tribunes/economie/trading-a-haute-frequence-quelles-performances-et-quels-risques-pour-les-marches-financiers/#:~:text=Le%20trading%20%C3%A0%20haute%20fr%C3%A9quence,rapidement%20aux%20%C3%A9v%C3%A8nements%20du%20march%C3%A9.*