UNIVERSITÉ DE THIES

UFR DES SCIENCES ECONOMIQUES ET SOCIALES

UFR DES SCIENCES ET TECHNOLOGIQUES

Master Science Des Données et Applications

Par

ALMAMY YOUSSOUF LY

Projet Apprentissage Par Renforcement : Application sur le trading

Professeur : Mr. Allaya

Année universitaire 2020-2021

Table des matières

I. Introduction

L’objectif principal de l’intelligence artificielle est le développement de fonctions informatiques associées à l’intelligence humaine, telle que le raisonnement, l’apprentissage et la résolution de problèmes, qui peuvent particulièrement dans notre étude être utiles pour les marchés financiers. Commerce et investissement sur le marché ne prend rien d'autre qu'une série de raisonnements et de calculs, basés sur les données et résolvant le problème de la prévision de l'orientation future des cours boursiers actuels. L'analyse fondamentale et technique manuelle est en train de se démoder de nos jours. L'application de la technologie d'apprentissage automatique dans le commerce ou le marché boursier est utilisé afin que le système apprenne automatiquement la complexité du commerce et améliore ses algorithmes pour une meilleure rentabilité. Au cours de la dernière décennie, il semblait y avoir une utilisation du portefeuille des commerçants, afin que chacun puisse gagner ses bénéfices. Mais, avec l'aide de l'IA, on peut parfaitement analyser les points de données sous-jacents présentés très rapidement et avec précision.

En utilisant de tels points de données, nous pouvons analyser les tendances actuelles du marché et former des modèles à grande vitesse, qui sont les deux éléments nécessaires généralement utilisés pour le trading intelligent. En utilisant les gros titres des chaînes d'information et des sources d'information, les critiques des médias sociaux et les commentaires présents sur d'autres plateformes, l' IA peut analyser l'action en effectuant une analyse des sentiments sur ces données. L'apprentissage automatique stocke généralement les résultats et les paramètres qui ont donné ces résultats et peut mieux analyser le marché boursier.

Les données aident souvent à trouver une meilleure solution, en particulier dans les activités basées sur les probabilités et basées sur le sentiment, comme les transactions boursières. L'IA peut évaluer et analyser des milliers d'actions en quelques instants, et donc cette technologie ajoute encore plus de vitesse au trading. Aujourd'hui, chaque milliseconde compte, et avec l'IA comme moyen de trading automatisé, c'est une merveille.

C’est dans ce contexte que nous allons appliquer un algorithme d’apprentissage par renforcement avec deux modèles différents sur les données de prix GME historiques afin de mieux appréhender le fonctionnement du trading.

II. Définition des concepts

A. Apprentissage par renforcement

Les deux principales parties du cadre d'apprentissage par renforcement sont :

L'Environnement : il s'agit d'un « terrain de jeu » ou d'un marché dans notre cas, qui peut nous dire ce qui se passe en ce moment et quelle sera notre récompense à l'avenir si nous agissons dès maintenant

L'Agent : un « joueur », qui interagit avec l'environnement et apprend à maximiser les récompenses à long terme en effectuant différentes actions dans différentes situations

Graphiquement, il peut être représenté par le schéma suivant :

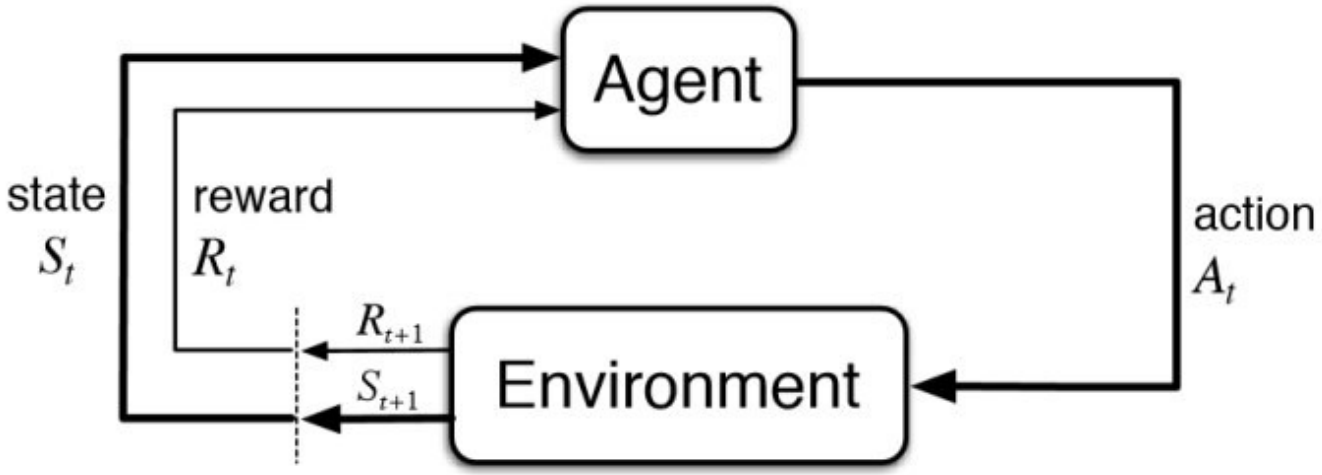


Figure 1 : Représentation de l'apprentissage par renforcement

B. Open AI Gym Anytrading

AnyTrading est une collection Open Source d'environnements OpenAI Gym pour les algorithmes de trading basés sur l'apprentissage par renforcement. Les algorithmes de trading sont principalement implémentés sur la base de deux des plus grands marchés présents : FOREX et Stock. AnyTrading vise à fournir des environnements Gym pour améliorer et faciliter la procédure de développement et de test d'algorithmes basés sur l'apprentissage par renforcement dans le domaine du trading sur le marché. Ceci est obtenu en l'implémentant sur trois environnements Gym : TradingEnv, ForexEnv et StocksEnv. AnyTrading peut vous aider à connaître les tendances du marché boursier et à effectuer des analyses puissantes, en fournissant des informations approfondies pour des décisions basées sur les données.

C. A2C

C’est un algorithme typique d'acteur critique. A2C utilise des copies du même agent travaillant en parallèle pour mettre à jour les gradients avec différents échantillons de données. Chaque agent travaille indépendamment pour interagir avec le même environnement.

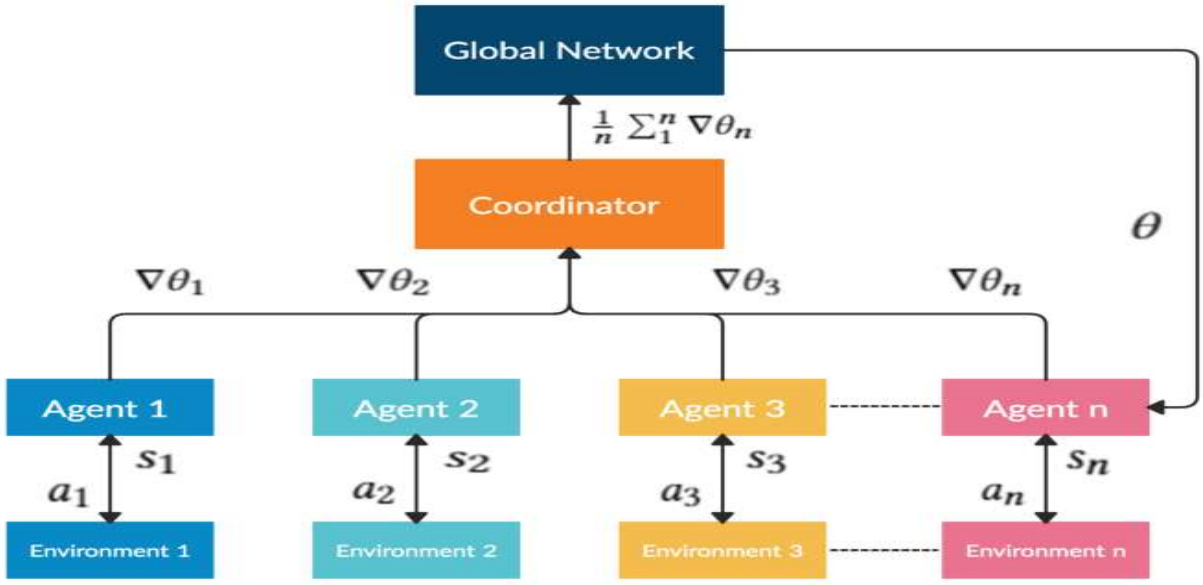


Figure 2 : Visuel de fonctionnement du modèle A2C

D. PPO

Il a été introduit pour contrôler la mise à jour du gradient de politique et garantir que la nouvelle politique ne sera pas trop différente de la précédente.

E. RSI (Relative Strengh Index)

L'indice de force relative (RSI) est un indicateur de dynamique utilisé dans l'analyse technique qui mesure l'ampleur des changements de prix récents pour évaluer les conditions de surachat ou de survente du prix d'une action ou d'un autre actif. Le RSI est affiché sous forme d'oscillateur (un graphique linéaire qui se déplace entre deux extrêmes) et peut avoir une lecture de 0 à 100.

F. SMA (Simple Moving Average)

Une moyenne mobile simple (SMA) est une moyenne mobile arithmétique calculée en additionnant les prix récents, puis en divisant ce chiffre par le nombre de périodes dans la moyenne de calcul.

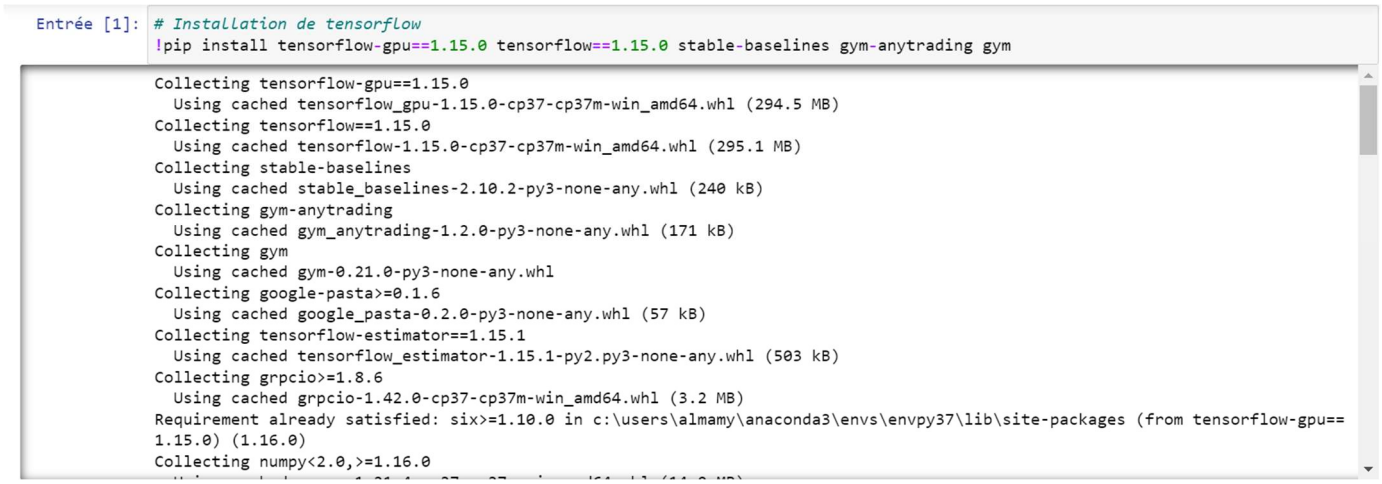
G. OBV (On Balance Volume)

Le volume au bilan fournit un total cumulé du volume de négociation d'un actif et indique si ce volume entre ou sort d'un titre ou d'une paire de devises donné. L'OBV est un total cumulé de volume (positif et négatif).

III. Application de l'algorithme

Nous mettrons en œuvre un modèle de trading de marché basé sur l'apprentissage par renforcement, dans lequel nous créerons un environnement de trading à l'aide d'OpenAI Gym AnyTrading. Nous utiliserons les données de prix GME historiques, puis nous entraînerons et évaluerons notre modèle à l'aide d'agents d'apprentissage par renforcement et de l'environnement de Gym.

A. Installation de la bibliothèque



La première étape essentielle serait d'installer la bibliothèque nécessaire. Pour ce faire, nous pouvons exécuter la ligne de code suivante :
!pip install tensorflow-gpu==1.15.0 tensorflow==1.15.0 stable-baselines gym-anytrading gym
Stable-Baselines nous fournira l'algorithme d'apprentissage par renforcement et Gym Anytrading nous fournira notre environnement de trading.

B. Importation des dépendances

Installons maintenant les dépendances requises pour créer un cadre de base pour nos modèles et nous allons utiliser l'algorithme d'apprentissage par renforcement A2C et PPO pour créer nos modèles de trading de marché.

Entrée [1]:
```
# Gym stuff
import gym
import gym_anytrading

# Stable baselines - rl stuff
from stable_baselines.common.vec_env import DummyVecEnv
from stable_baselines import A2C, PPO2


# Processing libraries
import time
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

C. Traitement de notre ensemble de données

Maintenant, avec la configuration de notre pipeline, chargeons et prétraitons nos données GME Market.

Entrée [71]:
```
# Chargement des données
def loadfile(path):
    df = pd.read_csv(path)
    return df

path = 'data/gmedata.csv'
loadfile(path)
```

Out[71]:

	Date	Open	High	Low	Close	Volume
0	03/12/2021	275.00	295.50	262.27	264.50	25,845,900
1	03/11/2021	241.64	281.50	232.60	260.00	28,312,490
2	03/10/2021	269.43	348.50	172.00	265.00	71,570,570
3	03/09/2021	217.71	249.85	208.51	246.90	39,099,328
4	03/08/2021	154.89	210.87	146.10	194.50	63,565,621
...	...	...	...	...	...	...
246	03/20/2020	4.08	4.08	3.65	3.76	7,722,194
247	03/19/2020	3.71	4.20	3.55	4.19	5,039,539
248	03/18/2020	4.10	4.25	3.50	3.77	3,651,709
249	03/17/2020	4.40	4.65	4.11	4.23	3,562,210
250	03/16/2020	3.93	4.57	3.90	4.37	4,866,696

251 rows × 6 columns

Entrée [72]:
```
# Types des variables
df = loadfile(path)
df.dtypes
```

Out[72]:
```
Date       object
Open      float64
High      float64
Low       float64
Close     float64
Volume     object
dtype: object
```

Entrée [73]:
```
# Convetir la columne date en format date
df['Date'] = pd.to_datetime(df['Date'])
df.dtypes
```

Out[73]:
```
Date      datetime64[ns]
Open             float64
High             float64
Low              float64
Close            float64
Volume            object
dtype: object
```

Entrée [75]:
```
# Transformer la colonne date en colonne indexe
df.set_index('Date', inplace = True)
df.head()
```

Out[75]:

Date	Open	High	Low	Close	Volume
2021-03-12	275.00	295.50	262.27	264.5	25,845,900
2021-03-11	241.64	281.50	232.60	260.0	28,312,490
2021-03-10	269.43	348.50	172.00	265.0	71,570,570
2021-03-09	217.71	249.85	208.51	246.9	39,099,328
2021-03-08	154.89	210.87	146.10	194.5	63,565,621

Entrée [76]:
```
# Trier les données par ordre ascendant
df.sort_values('Date', ascending=True, inplace=True)
df.head()
```

Out[76]:

Date	Open	High	Low	Close	Volume
2020-03-16	3.93	4.57	3.90	4.37	4,866,696
2020-03-17	4.40	4.65	4.11	4.23	3,562,210
2020-03-18	4.10	4.25	3.50	3.77	3,651,709
2020-03-19	3.71	4.20	3.55	4.19	5,039,539
2020-03-20	4.08	4.08	3.65	3.76	7,722,194

Nous allons maintenant transmettre les données et créer notre environnement de gym pour que notre agent s'y entraine plus tard.

Entrée [77]:
```
# Transmission des données et création de notre modéle
env = gym.make('stocks-v0', df = df, frame_bound = (5, 100), window_size = 5)
```

La définition du paramètre de taille de fenêtre spécifiera le nombre de références de prix précédentes que notre bot de trading aura afin qu'il puisse décider de faire une transaction.

D. Testons notre environnement

Maintenant, avec notre configuration de modèle, testons notre environnement de base et déployons notre agent d'apprentissage par renforcement.

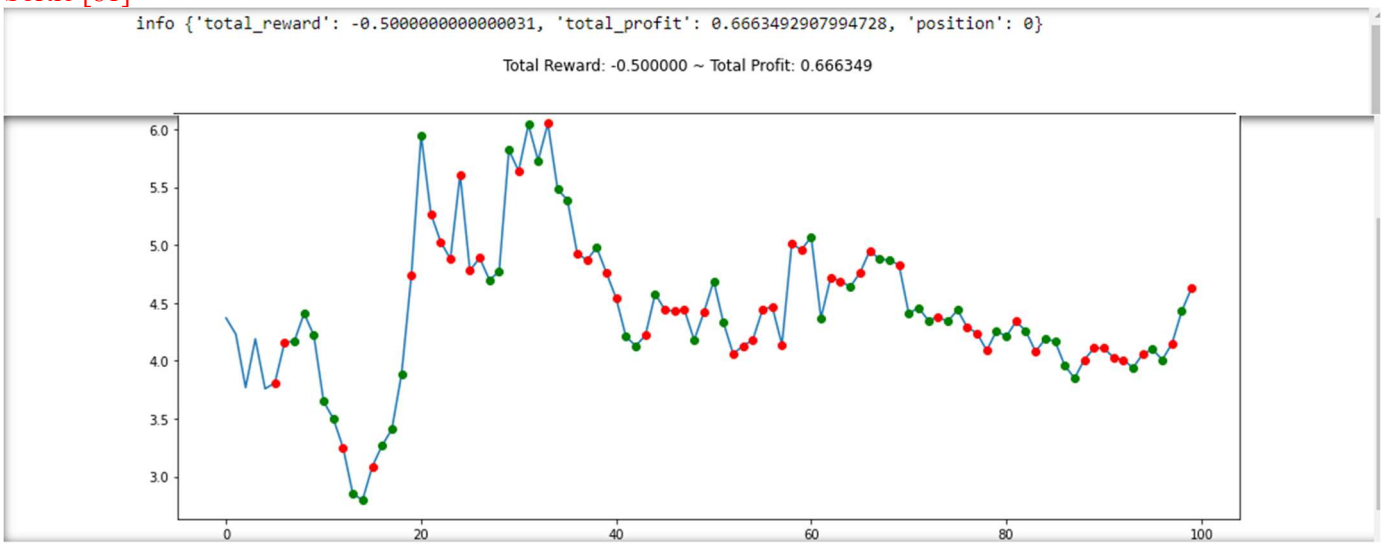
Entrée [80]:
```
# Configuration de l'environnement de l'agent
state = env.reset()
while True:
    action = env.action_space.sample()
    n_state, reward, done, info = env.step(action)


    if done:
        print("info", info)
        break
plt.figure(figsize=(15,6))
plt.cla()

env.render_all()
plt.show()
```

Sortie [81]

```
info {'total_reward': -0.5000000000000031, 'total_profit': 0.6663492907994728, 'position': 0}
```



Comme nous pouvons le voir, notre agent RL a acheté et vendu des actions au hasard. Notre marge bénéficiaire semble être supérieure à 1, nous pouvons donc déterminer que notre agent nous a rapporté des bénéfices grâce aux transactions qu'il a effectuées. Mais il s'agissait d'étapes aléatoires, maintenant nous allons créer nos propres indicateurs techniques financiers personnalisés pour pouvoir les utiliser dans nos deux modèles d'apprentissage par renforcement (A2C et PPO). Et nous allons l'utiliser pour négocier les actions de Gamestop ($GME). Nous allons d'abord calculer l'indice de force relative (RSI), la moyenne mobile simple (SMA) et les indicateurs de volume d'équilibre (OBV) et les utiliser ensuite pour entraîner nos deux modèles d'apprentissage par renforcement (A2C et PPO) alimentés par des lignes de base stables enfin d'obtenir de meilleurs résultats.

E. Former notre environnement

Configurons notre environnement pour former notre agent d'apprentissage par renforcement sur les deux modèles (A2C et POO).

Entrée [88]:
```
def add_signals(env):
    start = env.frame_bound[0] - env.window_size
    end = env.frame_bound[1]
    prices = env.df.loc[:,'Low'].to_numpy()[start:end]
    signal_features = env.df.loc[:, ['Low', 'Volume', 'SMA', 'RSI', 'OBV']].to_numpy()[start:end]
    return prices, signal_features
```

Entrée [89]:
```
class MyCustomEnv(StocksEnv):
    _process_data = add_signals

env2 = MyCustomEnv(df=df, window_size=12, frame_bound=(5, 100))
```

Nous venons de créer notre environnement en ajoutant une superposition personnalisée pour être en mesure d'apporter de nouveaux signaux. Ensuite, nous avons créé la fonction add_signals à laquelle on a passé en argument notre environnement existant. Finalement on a créé la classe MyCustomEnv à laquelle nous avons fourni comme argument l'environnement d'actions existant.

Entrée [94]:
```
# Environnement du modèle
#env_maker = lambda: gym.make('stocks-v0', df = df, frame_bound=(5, 100), window_size=5)
env_maker = lambda: env2
env = DummyVecEnv([env_maker])
```

Entrée [95]:
```
# Entrainement du modèle A2C
def train_A2C(env_train, timesteps):
    start = time.time()
    model = A2C('MlpPolicy', env_train, verbose=1)
    model.learn(total_timesteps=timesteps)
    end = time.time()
    print("Training time (A2C):", (end-start)/60,"minutes")
    return model
env_train = env
timesteps=10000
train_A2C(env, timesteps)
```

```
-----------------------------------
| explained_variance | -8.59    |
| fps                | 22       |
| nupdates           | 1        |
| policy_entropy     | 0.693    |
| total_timesteps    | 5        |
| value_loss         | 0.201    |
-----------------------------------
-----------------------------------
| explained_variance | -16.6    |
| fps                | 561      |
| nupdates           | 100      |
```

Entrée [96]:
```
# Entrainement du modèle PPO
def train_PPO(env_train,  timesteps):
    start = time.time()
    model = PPO2('MlpPolicy', env_train, verbose = 1)
    model.learn(total_timesteps=timesteps)
    end = time.time()
    print("Training time (PPO):", (end-start)/60,"minutes")
    return model
env_train = env
timesteps=10000
train_PPO(env, timesteps)
```

```
---------------------------------------
| approxkl            | 0.00016329416 |
| clipfrac            | 0.0           |
| explained_variance  | -0.686        |
| fps                 | 267           |
| n_updates           | 1             |
| policy_entropy      | 0.6929473     |
| policy_loss         | -0.004442195  |
| serial_timesteps    | 128           |
| time_elapsed        | 0             |
| total_timesteps     | 128           |
| value_loss          | 0.21867694    |
---------------------------------------
---------------------------------------
| approxkl            | 4.271244e-05  |
```

F. Evaluation des deux modèles

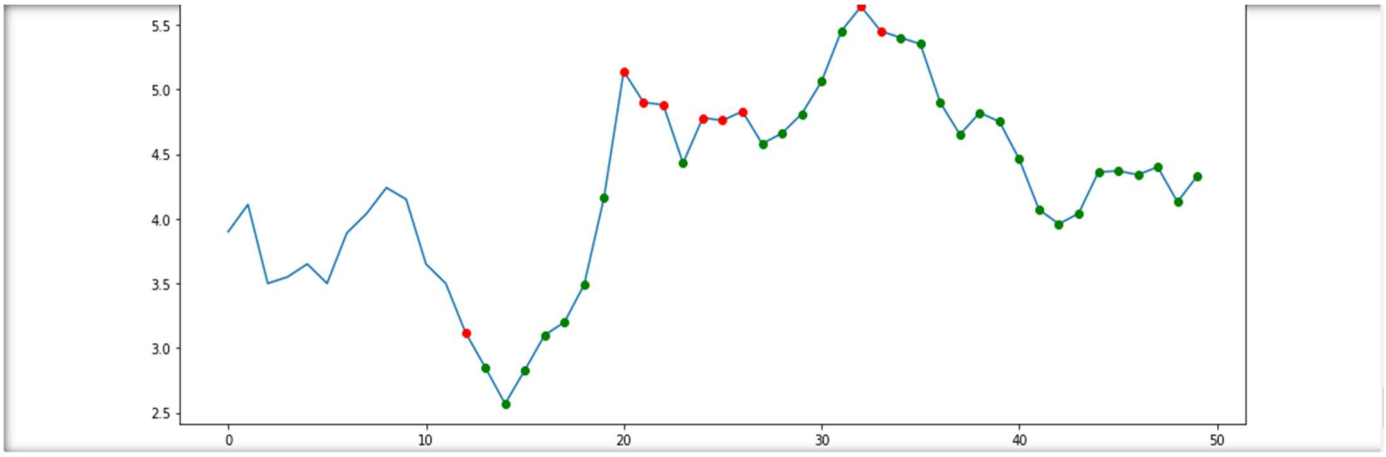
On définit une fonction evalmodel enfin d'évaluer nos deux modèles et éventuellement en déduire le plus performant.

Entrée [97]:
```
# Fonction d'évaluation des modèles
def evalmodel(envname, modelname):
    obs=env.reset()
    while True:
        obs = obs[np.newaxis, ...]
        action, _states = model.predict(obs)
        obs, rewards, done, info = env.step(action)
        if done:
            print("info", info)
            break
    plt.figure(figsize=(15,6))
    plt.cla()
    env.render_all() #allows us to render al of the different trdes that are happening or that have happened as part of our trad
    # particular environment; red dots aare shorts and the green are dots are going long
    plt.show()
```

On évalue le modèle A2C en utilisant l'environnement créé avec nos indicateurs techniques financiers à savoir RSI, OBV et SMA.

Entrée [98]:
```
# Evaluation du modèle A2C
#env = gym.make('stocks-v0', df=df, frame_bound=(90, 110), window_size=5)
env = env2
model = train_A2C(env, timesteps)
evalmodel(env, model)
```

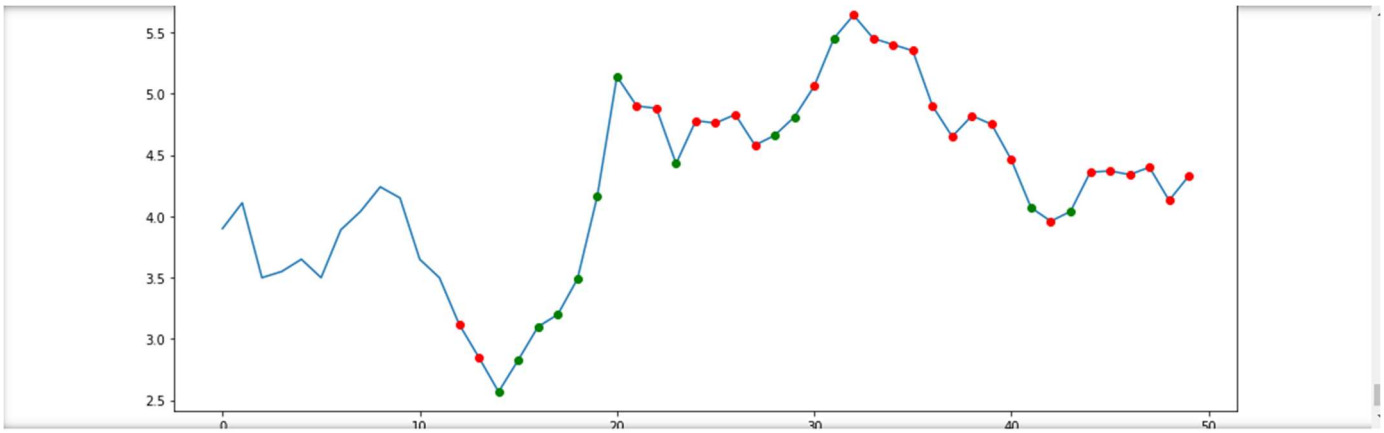
```
Wrapping the env in a DummyVecEnv.
---------------------------------
| explained_variance | -6.89    |
| fps                | 23       |
| nupdates           | 1        |
| policy_entropy     | 0.693    |
| total_timesteps    | 5        |
| value_loss         | 1.6      |
---------------------------------
---------------------------------
| explained_variance | -22.3    |
| fps                | 619      |
| nupdates           | 100      |
| policy_entropy     | 0.693    |
| total_timesteps    | 500      |
| value_loss         | 0.541    |
---------------------------------
---------------------------------
| explained_variance | 0.293    |
```



On évalue le modèle PPO en utilisant l'environnement créé avec nos indicateurs techniques financiers à savoir RSI, OBV et SMA.

Entrée [99]:
```
# Evaluation du modèle PPO
#env = gym.make('stocks-v0', df=df, frame_bound=(90, 110), window_size=5)
env = env2
model = train_PPO(env, timesteps)
evalmodel(env, model)
```

```
Wrapping the env in a DummyVecEnv.
--------------------------------------
| approxkl           | 0.00013689866 |
| clipfrac           | 0.0           |
| explained_variance | -1.35         |
| fps                | 275           |
| n_updates          | 1             |
| policy_entropy     | 0.6929909     |
| policy_loss        | -0.0031444947 |
| serial_timesteps   | 128           |
| time_elapsed       | 0             |
| total_timesteps    | 128           |
| value_loss         | 0.54714924    |
--------------------------------------
--------------------------------------
| approxkl           | 0.0001320728  |
| clipfrac           | 0.0           |
| explained_variance | -0.642        |
| fps                | 848           |
```



Comme nous pouvons le voir avec nos deux modèles ci-dessus, notre agent qualifié effectue maintenant de bien meilleures transactions et beaucoup moins de transactions aléatoires, ce qui nous donne des bénéfices en même temps beaucoup plus de conscience quant au moment d'acheter (en rouge ou position courte) et le bon moment pour vendre l'action (en vert ou position longue).

Qui plus est on note pour le modèle A2C plus de position longue que short alors qu'on observe le contraire dans PPO. Notre agent apprend plus rapidement avec A2C qu'avec PPO à cause de sa courte durée d'exécution. Nous observons une perte de valeur (value loss) croissante pour A2C ce qui est une indication de bonnes performances exploratoires ce qui signifie que l'agent effectue un bon apprentissage.

D'un autre côté, on engrange plus de récompense côté A2C avec un total reward de 3.7 contre 3.48 pour PPO. On constate le contraire concernant les profits totaux où celui de PPO (2.21734) est supérieur à celui de A2C (1.809186).

D'après les différents résultats obtenus par nos deux modelés nous pouvons conclure que A2C est plus performant que PPO.

En résumé, au cours de l'étude, nous avons ajouté un prétraitement des données en cherchant les valeurs manquantes, trier les données, calculer les indicateurs financiers afin de garantir un meilleur apprentissage pour notre agent pour finalement ajouter le modèle PPO dans le but de comparer ses performances par rapport à A2C.

IV. Conclusion

En somme, nous avons essayé de comprendre le fonctionnement de l'intelligence artificielle sur le trading enfin de tirer parti sur l'art d'acheter et de vendre des parts de marché. Nous avons également créé deux modèles d'apprentissage par renforcement grâce auquel notre agent qualifié peut acheter et vendre des actions, nous réservant simultanément de ce fait des bénéfices.

Référence bibliographique

1. https://medium.com/geekculture/first-steps-before-applying-reinforcement-learning-for-trading-579a5b0299a1
2. https://github.com/AminHP/gym-anytrading
3. https://github.com/nicknochnack/Reinforcement-Learning-for-Trading-Custom-Signals
4. https://towardsdatascience.com/deep-reinforcement-learning-for-automated-stock-trading-f1dad0126a02
5. https://towardsdatascience.com/value-based-methods-in-deep-reinforcement-learning-d40ca1086e1
6. Reinforcement Learning An Introduction 2nd Edition, Richard S. Sutton and Andrew G. Barto© 2014, 2015