Université Mohammed V de Rabat Ecole Nationale Supérieure d'Informatique et d'Analyse et des systèmes.

Rapport de Projet de fin de deuxième année : Système de trading automatique et prédiction de cours à l'aide de Machine Learning

Année Universitaire: 2019/2020.

Supervisé par:
Mr. Mahmoud Nassar

Realisé par:
Aabbar Adnane
Abed Abir

Filière:
Génie Logiciel

Table de matières

Système de trading automatique et prédiction de cours à l'aide de Machine Learning

Realisé par : Aabbar Adnane[a] et Abed Abir[b]
Supervisé par : Mr.Nassar Mahmoud[c]

École Nationale Supérieure d'Informatique et d'Analyse des Systèmes, Maroc.

Abstract
Stock market prediction is the act of trying to determine the future value of a company stock traded on a financial exchange. The successful prediction of a stock's future price will maximize trader's gains. This paper proposes a machine learning model to predict stock market price. The proposed algorithm integrates Long Short-Time Memory(LSTM) model, and optimizes it to give brilliant results for forecasting time series, stock prices in our study. The proposed model was applied and evaluated using both large and small financial datasets, and after a benchmarking of the existing solutions, compared through artificial neural networks with Auto Regressive Integrated Moving Average(ARIMA) class of models given that they perform well on the non seasonal time series. The obtained results show that traders benefit from our model to have a better idea on how the market will look like on a specific day.

KEYWORDS
Stock Trading; Trading Strategy; Algorithmic Trading; Machine Learning ; Technical Analysis; Fundamental Analysis; Investment; Risk Management; Prediction; Forecasting; Backtesting;

CONTACT A. N. Auteur. Email: adnaneaabbar@gmail.com
CONTACT A. N. Auteur. Email: abedabir98@gmail.com
CONTACT A. N. Superviseur. Email: mahmoud.nassar@um5.ac.ma

1. Introduction

Depuis la création de la finance de marché, l'utilisation des moyens humains pour investir sur les marchés diminue d'année en année pour être remplacée par des systèmes informatiques plus fiables que l'homme, appelé « Algotrading » (contraction de l'expression « trading algorithmique »). Ceci est dû au fait que les traders peuvent prendre plusieurs jours avant de réaliser des bénéfices et cela implique de forts risques.

1.1. Problématique

Prédire le rendement du marché boursier est l'une des choses les plus difficiles à faire. Il y a tellement de facteurs impliqués dans la prédiction, les facteurs physiques, le comportement, rationnel et irrationnel des agents économiques. Tous ces aspects se combinent pour rendre les cours des actions volatils et très difficiles à prévoir de manière précise. Par conséquent construire un système qui va aider les traders à prendre une meilleure décision d'acheter ou de se retirer du marché, pourra rencontrer le souci de l'abondance d'informations entraîne des difficultés de traitement des données.

1.2. Objectif

Notre projet tient à préparer un terrain pour les traders en développant un modèle de prédiction des cours des actions d'une entreprise définie, leur permettant de gagner en temps et en énérgie, aulieu de faire une analyse entière à chaque fois, ils peuvent avoir une idée sur l'état des prix des actions dans une date précise, ainsi ils seront en mesure de prendre la décision adéquate à un instant donné, d'investir ou non dans le marché dans cette date.

1.3. Contributions

Après avoir mené une auto-formation au niveau des compétences requises pour bien orienter une étude basé sur le Machine Learning, la récolte de données et leur traitement et analyse, nous avons réussi à construire un modèle qui prédit les futures valeurs d'une série temporelle en se basant sur ses données historiques à l'aide des bibliothèques riches de Python**(Tensorflow, Keras, Sickit-learn..)**. Pour déployer ce modèle, nous avons opté pour une des frameworks de Python qui est Flask, afin de permettre une intéraction plus facile de l'utilisateur avec le modèle.

1.4. Plan

Organization : Le reste du rapport est organisé comme suit : La **section 2** contient une étude du marché, un benchmarking des différentes solutions qui existent au niveau des modèles. Dans la **section 3** nous explicitons notre méthodologie de travail pour réussir cette étude. Dans la **section 4**, les résultats du modèle, les métriques que nous avons réalisé sur les différentes données, suivie par un **section 5** relative à l'optimisation du modèle. Puis vient la **section 6** qui représente les menaces à la validité du sytème de trading établi. Après il y a la **section 7** qui traite le déploiement sous forme d'application web du modèle, suivi par la **section 8** qui explicite la stratégie de trading que nous avons établi, et finalement la **section 9** qui est une conclusion.

2. Benchmarking et étude de l'existant

2.1. Les séries temporelles

Une série temporelle, ou série chronologique, est une suite de valeurs numériques représentant l'évolution d'une quantité spécifique au cours du temps. De telles suites de variables aléatoires peuvent être exprimées mathématiquement afin d'en analyser le comportement, généralement pour comprendre son évolution passée et pour en prévoir le comportement futur. Une telle transposition mathématique utilise le plus souvent des concepts de probabilités et de statistique.
La composante "temps" apporte une information fructueuse; elle permet de parler d'évolution, de prédiction et de stratégie. Cependant cela amène aussi une complexification, un point d'une série temporelle n'a pas beaucoup de sens en isolation aussi la série doit-elle être conçue et traitée comme un tout.

2.2. Réseaux de neurones et modèles

Les réseaux de neurones artificiels sont des systèmes informatiques inspirés des réseaux de neurones biologiques qui constituent les cerveaux biologiques. Ces systèmes apprennent un phénomène en considérant beaucoup d'exemples. Un réseau est basé sur une collection d'unités connectées appelées neurones. Chaque connexion entre deux neurones signifie la possibilité de circulation de signal entre les deux neurones.

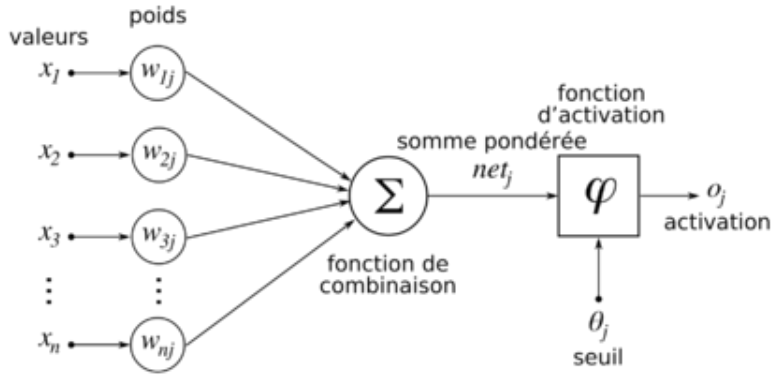


Figure 1.: Réseau de neurones simplifié

Le processus d'apprentissage par réseaux de neurones se passe en deux phases. Lors de la première phase (dite d'apprentissage), il s'agit de déterminer un modèle à partir des données étiquetées. La seconde phase (dite de test) consiste à prédire l'étiquette d'une nouvelle donnée, connaissant le modèle préalablement appris.

2.3. Les modèles ARIMA

Les modèles ARIMA (AutoRegressive Integrated Moving Average) offrent une approche complémentaire de la prévision de séries temporelles. Alors que les différents autre modèles de prédiction sont basés sur une description de la tendance et de la saisonnalité dans les données, les modèles ARIMA visent à décrire les autocorrélations dans les données sans avoir besoin d'avoir une condition de saisonnalité dans les données traitées.

Un modèle ARIMA capture donc une suite de différentes structures temporelles. Il s'agit d'une généralisation de la moyenne mobile auto-régressive à laquelle s'ajoute un processus de différenciation afin de rendre la série temporelle stationnaire. La différenciation se calcule par différences entre les observations consécutives.

2.4. Le RNN-LSTM

Un réseau de neurones récurrents (RNN) est un réseau de neurones artificiels avec des connexions récurrentes, c'est à dire des neurones interconnectés pouvant présenter des cycles. Ces réseaux ont prouvé leur efficacité dans la modélisation de problématiques de prédiction des séries temporelles.
Les réseaux de neurones récurrents LSTM ou mémoire à long terme et à court terme sont les variantes des réseaux de neurones artificiels. Contrairement aux réseaux à action directe où les signaux voyagent uniquement vers l'avant, dans RNN-LSTM, les signaux de données voyagent vers l'arrière et ces réseaux ont les connexions de rétroaction.
Le RNN-LSTM est couramment utilisé dans les prévisions de séries temporelles, car il garde plus longtemps l'information apprise dans les couches précédentes lointaines.

2.5. Résultat du benchmarking

Pour savoir décider sur quel modèle nous allons conduire notre optimisation, nous avons testé les 2 types sur des données historiques des cours de la bourse:

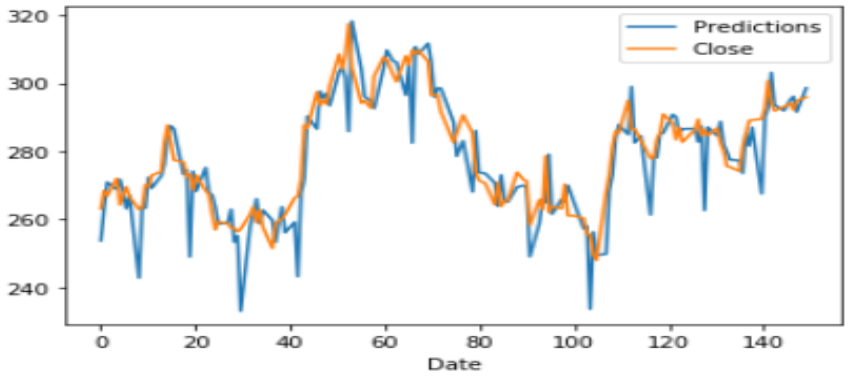


Figure 2.: A : Prédiction avec ARIMA

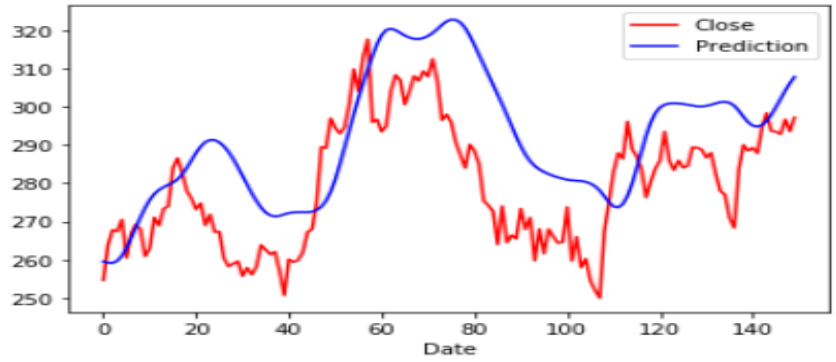


Figure 3.: B : Prédiction avec LSTM

En comparant les deux graphiques de prévision, nous pouvons voir que le modèle ARIMA a prédit des prix de clôture très inférieurs aux prix réels. Mais dans le cas du modèle LSTM, la même prédiction des prix de clôture peut être considérée comme supérieure à la valeur réelle. Mais cette variation peut être observée à quelques endroits dans le graphique et la plupart du temps, la valeur prédite semble être proche de la valeur réelle. Nous pouvons donc conclure que, dans la tâche de prédiction des stocks, le modèle LSTM a surpassé le modèle ARIMA.

Enfin, pour plus de satisfaction, nous allons essayer de trouver l'erreur quadratique moyenne (RMSE) dans la prédiction par les deux modèles:

Table 1.: Comparaison de l'erreur quadratique moyenne

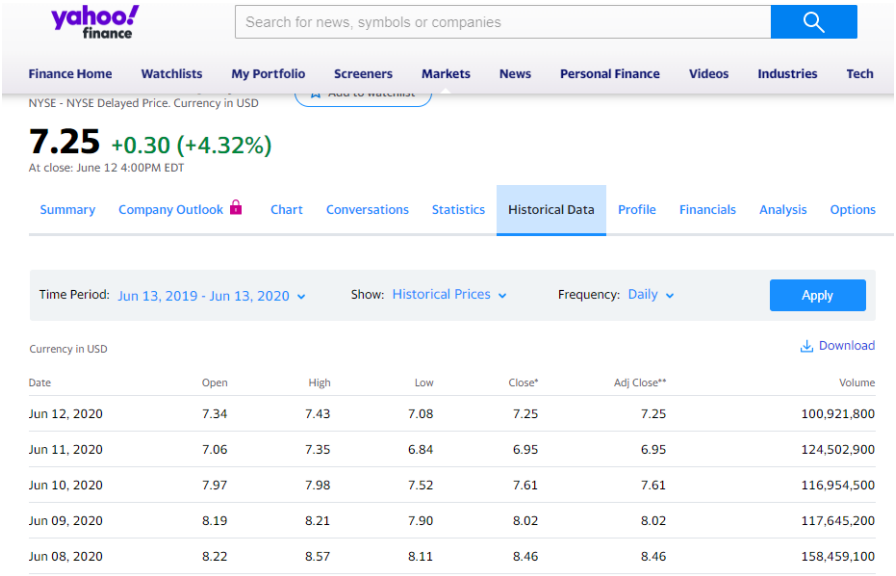
Modèle	RMSE
ARIMA	72.640563493
LSTM	24.923716607

En voyant les RMSE, il est clair maintenant que le modèle LSTM a les meilleures performances dans cette tâche, donc c'est ce modèle avec lequel nous avons choisi de continuer l'optimisation, mais nous allons présenter les résultats réalisés par le modèle ARIMA aussi dans la section 4.

3. Methodologie

3.1. Collecte de données

Le Machine Learning dépend fortement des données, sans données, il est impossible pour une intelligence artificielle d'apprendre. C'est l'aspect le plus crucial qui rend possible la formation à l'algorithme. Peu importe la taille de l'équipe ou la taille de l'ensemble de données, si cet ensemble n'est pas assez bon, tout le projet échouera.
Pour s'assurer de la qualité et de la fiabilité des données nous avons opté pour la plateforme Yahoo Finance qui fournit des informations financières, des données sur les cotations boursières, des communiqués de presse et des rapports financiers. Et toutes les données fournies par Yahoo Finance sont gratuites.



Date	Open	High	Low	Close*	Adj Close**	Volume
Jun 12, 2020	7.34	7.43	7.08	7.25	7.25	100,921,800
Jun 11, 2020	7.06	7.35	6.84	6.95	6.95	124,502,900
Jun 10, 2020	7.97	7.98	7.52	7.61	7.61	116,954,500
Jun 09, 2020	8.19	8.21	7.90	8.02	8.02	117,645,200
Jun 08, 2020	8.22	8.57	8.11	8.46	8.46	158,459,100

Figure 4.: Données historiques de Yahoo! Finance

Nous avons choisi 2 exemples de datasets pour faire notre étude, la différence entre les deux est la taille, le premier contient les données financières de la banque GS - Goldman Sachs des **10 dernières années** résultant en **2265 lignes**, chacune représentant les attributs boursiers d'unne journée pour la banque. Le deuxième est celui de la compagnie GE - General Electric Co. des **55 dernières années** résultant en **14056 lignes**.

3.2. Nettoyage de données

La base de données étant composée principalement de données numériques, la principale action du nettoyage consiste au traitement des valeurs manquantes sur le dataset, l'utilisation de la bibliothèque **Pandas** de Python facilite cette tâche vu les outils prédifinis pour ce genre de traitement. Nous nous assurons que toutes les données sont présentes, lorsqu'une donnée n'existe pas, elle est remplacée par la valeur **null** ou **NaN**.

3.3. Visualisation de données

Une série financière ne se résume pas à une série de prix. Pour une série quotidienne, on dispose d'attributs qui sont le premier prix de la journée **Open**, le dernier prix **Close**, le plus haut prix **High** et le plus bas **Low**. Une autre information importante est le **Volume** de transactions.

Grâce à la bibliothèque de visualisation de Python **Matplotlib**, nous pouvons tracer les 4 premiers attributs:

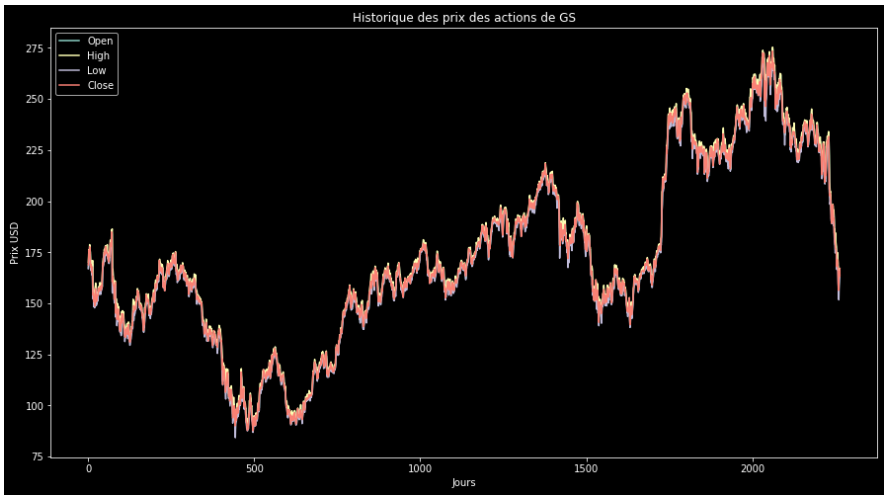


Figure 5.: Visualisation des attributs Open, Close, High, Low

Les prix - Open, Close, Low, High - ne varient pas trop les uns des autres, sauf occasionnellement. Traçons le Volume:

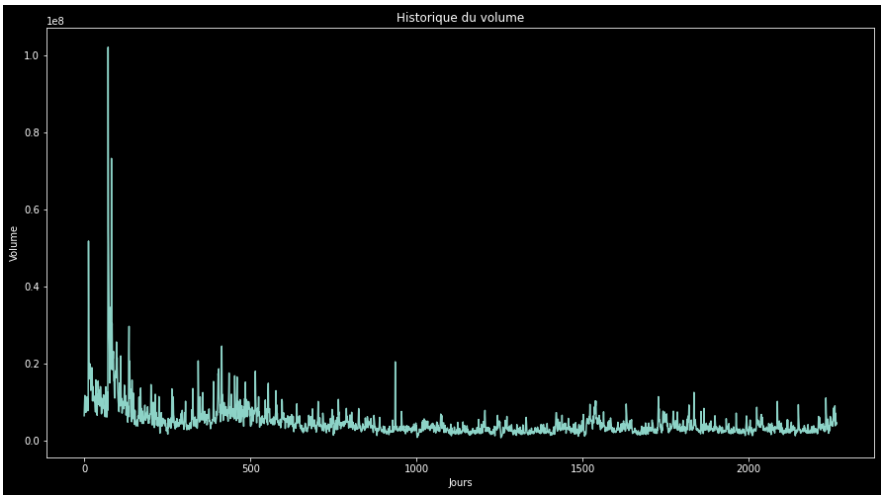


Figure 6.: Visualisation du Volume

Il y a une forte augmentation du nombre de transactions autour du 80ème jour sur la chronologie, qui coïncide avec la chute soudaine du cours des actions just après quelques jours.

3.4. Normalisation des données

Les données ne sont pas normalisées et la plage pour chaque colonne varie, en particulier le volume. La normalisation des données aide l'algorithme à converger, c'est-à-dire à trouver efficacement le minimum local / global. Nous avons utilisé **MinMaxScaler** de **Scikit-learn**. Mais avant cela, nous devons diviser l'ensemble de données en ensembles de données d'apprentissage(training) et de test, nous avons fait une division selon les normes de 80% du dataset pour la phase d'apprentissage et 20% pour la phase de test et validation du modèle, ensuite nous avons redimensionner le dataset pour ne garder que les colonnes qui nous intéressent pour l'étude.

3.5. Conversion des données en séries temporelles

Le modèle RNN-LSTM consomme des entrées au format d'un tableau tridimensionnel.

```
[batch_size, time_steps, features]
```

- **batch_size** : indique le nombre d'échantillons d'entrée que vous souhaitez que votre réseau processe avant de mettre à jour les poids. L'utilisation d'une très petite taille de lot réduit la vitesse de formation et d'autre part, l'utilisation d'une taille de lot trop grande réduit la capacité des modèles à se généraliser à différentes données et consomme également plus de mémoire.
- **time_steps** : définissent le nombre d'unités dans le temps passé que nous souhaitons que notre réseau prenne en considération. C'est-à-dire que nous examinerons **time_steps** jours de données pour prédire le prix du jour **time_steps + 1**.
- **features** : indique le nombre d'attributs utilisés pour représenter chacune des **time_steps**.

Dans cette section nous avons entammé la méthodologie de travail utilisée dans la plupart des projets en Machine Learning, nous avons collecté des datasets, nous l'avons nettoyé, visualisé ses différents attributs, ensuite nous avons normalisé les données des différentes colonnes et divisé notre dataset en 2 parties d'apprentissage et de validation, enfin nous avons converti ces données en séries temporelles représentées au format de l'input nécessaire pour notre modèle LSTM.
Passons à la construction des modèles de prédiction et la phase d'apprentissage et de validation de chacun.

4. Mise en œuvre et Résultats

Après avoir établi notre ensemble de données, vient maintenant la partie du développement de notre modèle de prédiction.

4.1. Construction du modèle

Le LSTM est une variation des réseaux de neurones récurrents, nous avons utilisé une couche de 100 blocks LSTM, un dropout de 0.5 qui a permis de désactiver la moitié des neurones aléatoirement pour accélérer l'apprentissage, une couche qui applique la fonction relu qui est tout simplement la fonction x -> max(x, 0) effectuée élément par élément, et enfin une couche qui applique la fonction sigmoid .
Nous vons laissé le modèle s'entraîner à minimiser la fonction de coût qui est l'erreur en moyenne quadratique MSE à l'aide de l'algorithme d'optimistion RMSprop,ensuite nous avons fixé le pas d'apprentissage à 0.0001 en avançant vers le minimum de cette fonction de coût.
Enfin on a lancé l'apprentissage sur 300 époques.

4.2. Prédiction et visualisation des résultats

4.2.1. Données de grande taille

Dans une première étape on va tester notre modèle sur les données de taille grande(dans notre cas on a utilisé celles de General electrics). On visualise le coût lors de l'apprentissage et lors de la validation.

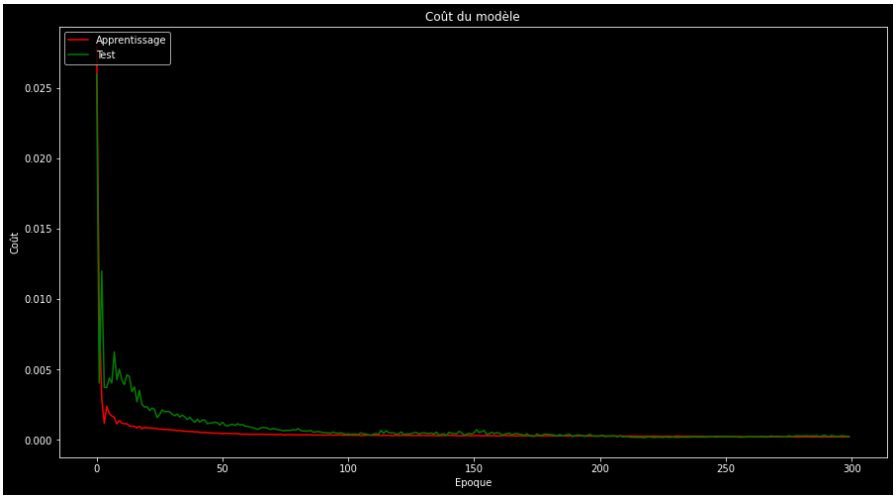


Figure 7.: Coût du modèle pour les données larges

Après les 50 premières époques, le coût de la phase du test rejoint l'ordre de grandeur de la phase d'entrainement.

La prédiction avec le modèle est la suivante :

Figure 8.: Résultat de la prédiction pour les données larges

On peut dire que le modèle est plus ou moins bien, il parvient à prédire la hausse ou la baisse d'une manière exacte, à l'exclusion des premiers jours.

4.2.2. Données de petite taille

Dans cette étape nous allons nous intresser aux données de taille petite(celles de Goldman Sachs) et tester si notre modele va bien predire les prix de cours. On visualise le coût lors de l'apprentissage et lors de la validation. La prédiction avec le modèle est la suivante :

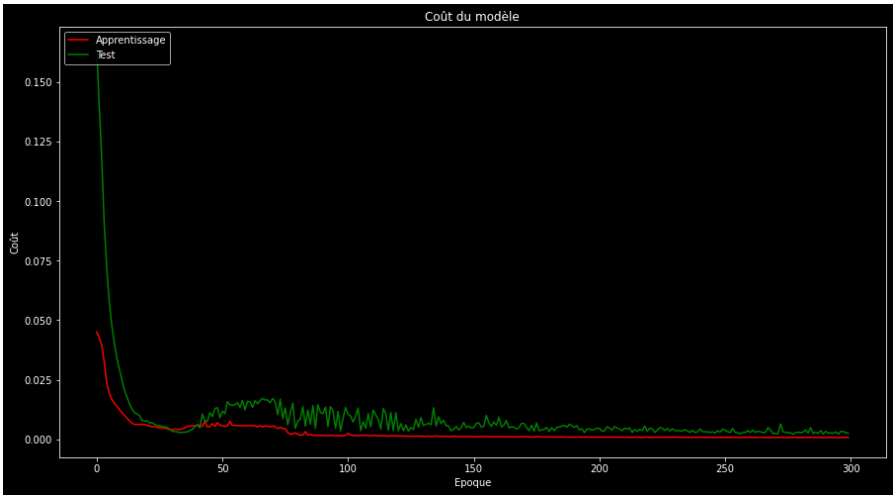


Figure 9.: Coût du modèle pour les données petites

Le coût lors de la phase du test ne suit aucune tendance, il oscille beaucoup autour de celui de la phase d'entrainement.

La prédiction avec le modèle est la suivante :

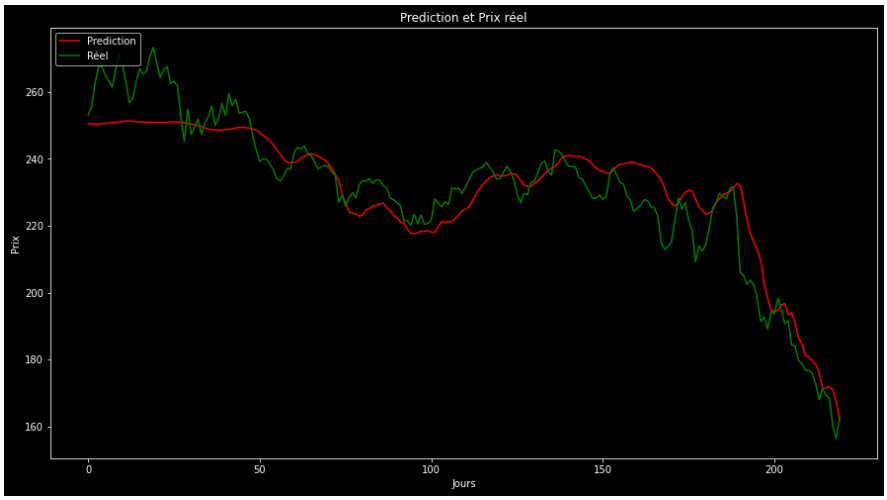


Figure 10.: Résultat de la prédiction pour les données petites

Le résultat montre que le modèle ne se comporte pas du tout bien vis à vis de notre dataset un peu limité en niveau de taille, il parvient à prédire la tendance des actions mais nous avons besoin de beaucoup plus de précisions pour parvenir à un bon résultat. Nous allons donc essayer d'avancer dans l'étude en essayant de rendre notre modele plus optimisé et plus pertinent pour gérer tout type de données.
L'etape suivante sera donc consacrée a l'optimisation.

5. Optimisation

5.1. Augmentation de couches

Afin d'optimiser notre modèle et accélérer l'apprentissage nous avons augmenté le nombre de couches : nous avons jouté 6 couches de neurones. Le code de notre modèle optimisé est le suivant:

```
#Build the LSTM Model

model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape = (x_train.shape[1],1)))
model.add(LSTM(50, return_sequences=False))

model.add(Dense(units = 1024))
model.add(Dense(units = 512))
model.add(Dense(units = 256))
model.add(Dense(units = 128))
model.add(Dense(units = 64))
model.add(Dense(1))
```

Figure 11.: Modèle optimisé

L'un des principaux avantages de l'utilisation d'un **modèle multicouches** par rapport à l'API de niveau inférieur est la possibilité d'enregistrer et de charger un modèle. Un modèle multicouches connaît:

- l'architecture du modèle, permet de recréer le modèle.
- les poids du modèle.
- la configuration de la formation (perte, optimiseur, métriques).
- l'état de l'optimiseur, permet de reprendre l'entraînement.

5.2. Sommaire du modèle

L'optimisation du modèle nous a permis d'augmenter sa capacité, avec 780.169 paramètres.

```
Entrée [16]: model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 60, 50)            10400
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              52224
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_3 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_4 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_5 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 65
=================================================================
Total params: 780,169
Trainable params: 780,169
Non-trainable params: 0
_________________________________________________________________
```

Figure 12.: Sommaire du modèle

5.3. Résultats de l'optimisation

5.3.1. Prédictions

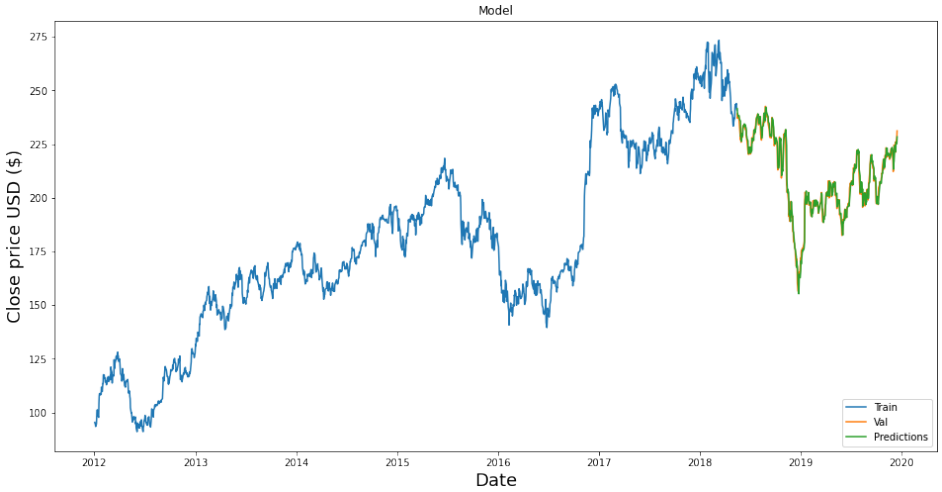


Figure 13.: Résultats de la prédiction

Nous remarquons que notre modèle optimisé parvient à prédire la hausse ou la baisse d'une manière plus ou moins exacte. Le tableau suivant fait une comparaison entre le prix rèel et le prix de prédiction de notre modèle :

Date	Close	Predictions
2018-05-17	239.100006	241.553894
2018-05-18	237.000000	239.634430
2018-05-21	237.699997	237.774200
2018-05-22	238.000000	238.074997
2018-05-23	237.809998	238.244568
...	...	...
2019-12-11	221.190002	222.779053
2019-12-12	226.050003	221.508011
2019-12-13	225.000000	225.907578
2019-12-16	228.039993	225.209930
2019-12-17	231.149994	228.378693

400 rows × 2 columns

Figure 14.: Comparaison du prix rèel et prix de prédiction

Nous remarquons que la majorité des prix prédits sont trés proches du prix réel.

5.3.2. Erreur quadratique moyenne

L'erreur quadratique moyenne indique à quel point une droite de régression est proche d'un ensemble de points. Pour ce faire, il prend les distances des points à la droite de régression (ces distances sont les «erreurs») et les met au carré. La quadrature est nécessaire pour éliminer tout signe négatif. Il donne également plus de poids aux différences plus importantes. L'erreur quadratique moyenne pour notre modèle est la suivante :

```
#root mean squared error

rmse=np.sqrt(np.mean(((predictions- y_test)**2)))

rmse
```

```
3.604530186709846
```

Figure 15.: Erreur quadratique moyenne

Nous remarquons la valeur de l'erreur est proche de 4 donc notre modele est un bon modèle.

5.3.3. Test du modèle

Après la validation de notre modèle l'ètape de tst est necessaire.Pour se faire nous allons faire la prediction du prix de cloture de la bourse pour 'Apple' pour le 28-05-2020 ensuite nous allons le comparer avec le prix rèel de cloture du jour.

```
Entrée [28]:
from datetime import datetime
scaler = MinMaxScaler(feature_range=(0,1))
dataframe = web.DataReader('AAPL', data_source='yahoo', start='2012-01-01', end= '2020-05-27')
new_df = dataframe.filter(['Close'])
last_60_days = new_df[-60:].values
last_60_days_scaled = scaler.fit_transform(last_60_days)
X_test = []
X_test.append(last_60_days_scaled)
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
pred_price = model.predict(X_test)
pred_price = scaler.inverse_transform(pred_price)[0][0]
pred_price

Out[28]: 317.80844

Entrée [26]:
apple_quote2 = web.DataReader('AAPL', data_source='yahoo', start='2020-05-28', end='2020-05-28')
print(apple_quote2['Close'])

Date
2020-05-28    318.25
Name: Close, dtype: float64
```

Figure 16.: Test du modèle

Nous remarquons que le prix predit est très proche du prix rèel avec une erreur de +0.44156.

6. Menaces à la validité

6.1. Validité interne

La validité interne est la mesure dans laquelle une étude établit une relation de cause à effet fiable entre un traitement et un résultat. Elle reflète qu'une étude donnée permet d'éliminer les explications alternatives à une constatation et dépend largement des procédures d'une étude et de la rigueur avec laquelle elle est réalisée.

- **Confondant**: se réfère à une situation dans laquelle les changements dans une variable de résultat peuvent être considérés comme résultant d'une troisième variable liée au traitement que vous avez administré, cette situation ne nous a pas affectés dans notre étude.

- **Biais expérimental**: fait référence à un expérimentateur se comportant différemment avec différents groupes dans une étude, ce qui a un impact sur les résultats de cette étude, dans notre cas, nous nous sommes comportés de la même manière avec nos différents datasets, les différentes séries temporelles financières ont les composants similaires.

- **Test**: se réfère à l'effet de tester à plusieurs reprises les phénomènes en utilisant les mêmes mesures. (essentiellement sur-ajusté ou en anglais **over-fitting**), avec les données de séries temporelles, une attention particulière doit être apportée au fractionnement des données afin d'éviter les fuites de données, les dépendances temporelles doivent être maintenues intactes.

6.2. Validité externe

La validité externe fait référence à la mesure dans laquelle les résultats d'une étude peuvent s'appliquer à d'autres paramètres. En d'autres termes, ce type de validité fait référence à la généralisation des résultats. Si des méthodes de recherche rigoureuses peuvent garantir la validité interne, la validité externe, en revanche, peut être limitée par ces méthodes.

- **Facteurs situationnels**: tels que les jours spéciaux, les stratégies du marketing de l'entreprise, les caractéristiques du marché tels les concurrents et le nombre de mesures utilisées peuvent affecter la généralisabilité des résultats. Dans notre cas il y avait une chute importante dans le volume des transactions faites dans les 2 datasets, plusieurs occurences de ce phénomène pourrait heurter gravement la performance de notre modèle.
- **Biais de sélection**: fait référence au problème des différences entre les groupes dans une étude qui peuvent se rapporter à la variable indépendante, mais dans notre étude, chaque dataset a été traité séparément et les données des séries temporelles financières sont parfaitement hasardeuses à l'exception des cas où les entreprises passent par une crise ou un épanouissement économique qui crée une tendance repérable.

7. Déploiement de l'application

Dans un projet typique de machine learning et de deep learning, nous commençons généralement par définir l'énoncé du problème suivi de la collecte et de la préparation des données, de la compréhension des données et de la construction de modèles, comme nous l'avons fait auparavant pour la prediction de prix de cours et le trading automatique.
Mais, en fin de compte, nous voulons que notre modèle soit disponible pour les utilisateurs finaux afin qu'ils puissent l'utiliser. Cependant, le déploiement de modèles d'apprentissage automatique est complexe.

7.1. Flask

Flask est un framework d'application web écrit en Python. Il dispose de plusieurs modules qui facilitent l'écriture d'applications par un développeur Web sans avoir à se soucier des détails tels que la gestion des protocoles, la gestion des threads, etc.
Flask donne une variété de choix pour développer des applications Web et il nous donne les outils et les bibliothèques nécessaires qui nous permettent de construire une application Web.

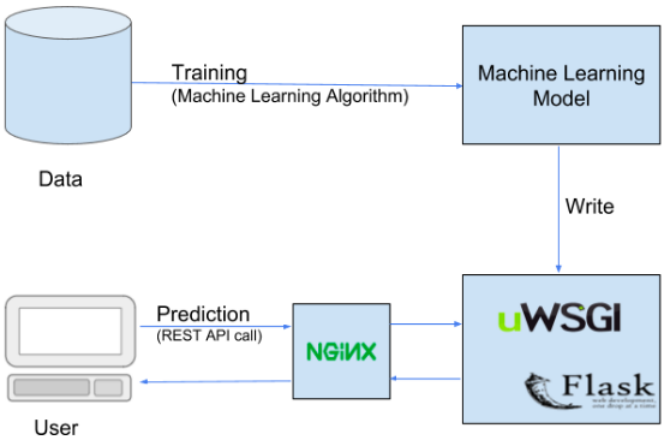


Figure 17.: Pipeline pour le déploiement d'un modèle de Machine Learning

Après le déploiement du modèle sur Flask, et la création de l'interface web pour garantir la facilité d'utilisation pour l'utilisateur final pour saisir le nom de la companie qui sera traitée par le modèle, ensuite le prix de cours pour le lendemain sera predit.

La première page Web de notre application Flask n'est qu'une page d'accueil ou un portail qui demande d'insérer le nom de la companie qui doit ensuite être traité par notre modèle.

7.2. Deploiement

Après l'insertion du nom de la companie comme input, le modèle nous rend le prix de cours pour le lendemain.

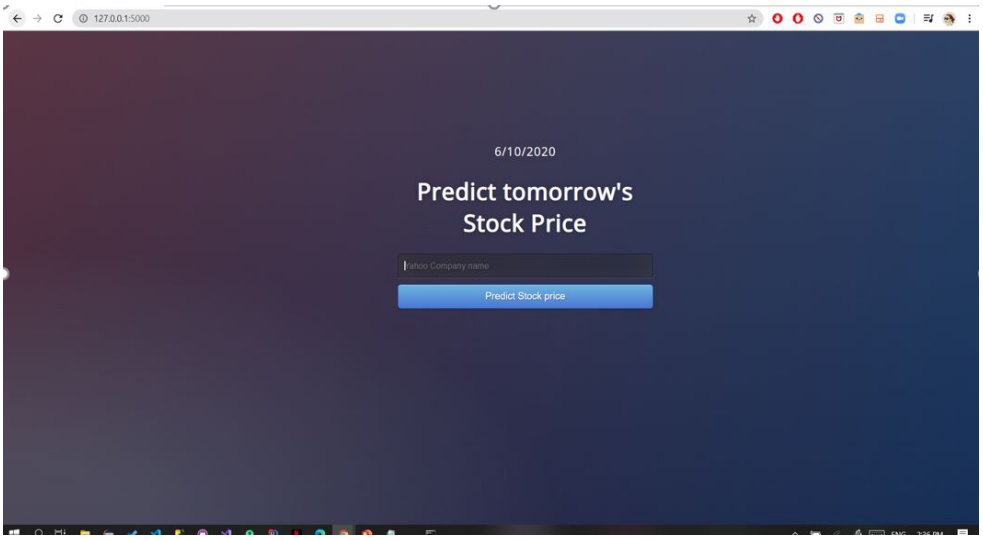


Figure 18.: A : Insertion du nom de l'entreprise

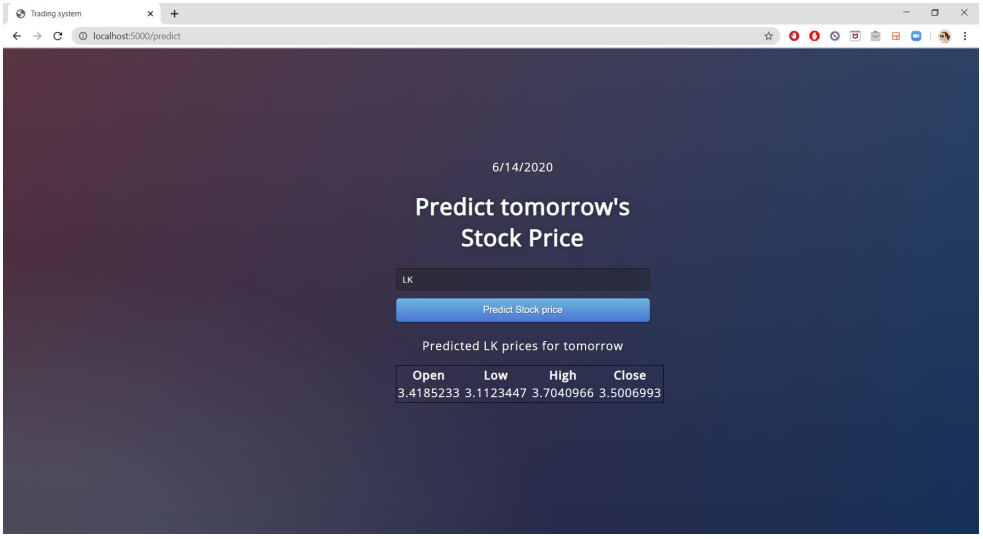


Figure 19.: B : La prédiction des différents prix

8. Stratégie de trading

Tout d'abord une stratégie de trading est un plan fixe pour aller long ou court sur les marchés. En général, il existe deux stratégies de trading communes: la stratégie de **momentum** et la stratégie de **réversion**. Pour notre étude, nous nous sommes concentrés sur le premier type.

La stratégie de **momentum** est également appelée **divergence** ou trading de **tendance**. Lorsque nous suivons cette stratégie, nous le faisons parce que nous pensons que le mouvement d'une quantité continuera dans sa direction actuelle. Autrement dit, nosu pensons que les actions ont une dynamique ou des tendances à la hausse ou à la baisse, que nous pourrons détecter et exploiter.

8.1. Croisement de moyenne mobile

Un des exemples les plus utilisés de cette stratégie est le **croisement de moyenne mobile**. C'est lorsque le prix d'un actif se déplace d'un côté d'une moyenne mobile à l'autre. Ce croisement représente un changement de momentum et peut être utilisé comme un point de prise de décision pour entrer ou sortir du marché. Pour illustrer le rendement de cette stratégie nous l'avons testé sur des données historiques de l'entreprise Apple Inc. En utilisant Pandas, nous avons tracés les moyennes mobiles à 2 périodes d'analyse différentes, une à 30 jours**(SMA30)** et l'autre à 100 jours**(SMA100)**.

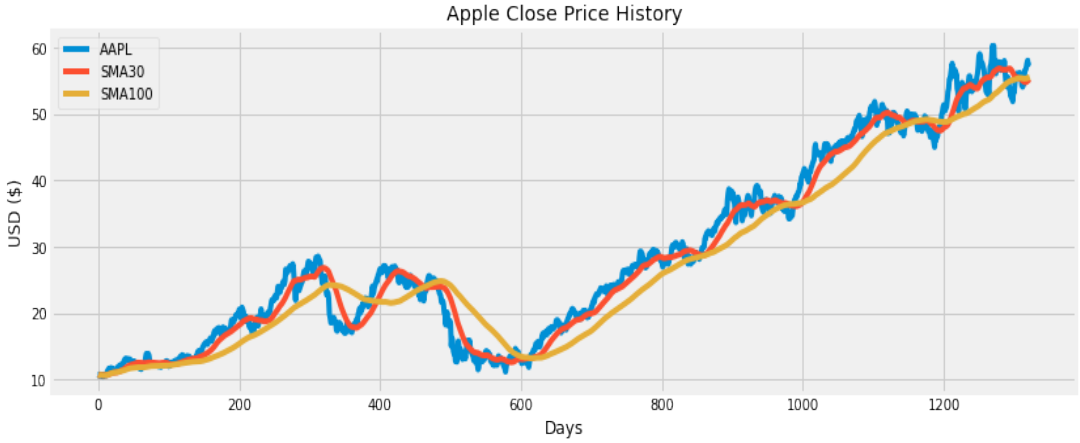


Figure 20.: Visualisation des moyennes mobiles SMA30 et SMA100

Comme nous pouvons le constater sur le graphe, les moyennes mobiles permettent de réduire la quantité de **«bruit»** sur la courbe des prix en bleu. Regardons la direction de la moyenne mobile pour avoir une idée de base de la façon dont le prix évolue. S'il est orienté vers le haut, le prix augmente globalement; incliné vers le bas, et le prix baisse globalement; se déplaçant latéralement, et le prix est probablement dans une fourchette.

8.2. Backtesting de la stratègie

Le backtesting est un élément clé du développement d'un système commercial efficace. Il est accompli en reconstruisant, avec des données historiques, des transactions qui auraient eu lieu dans le passé en utilisant des règles définies par une stratégie donnée. Le résultat propose des statistiques pour mesurer l'efficacité de la stratégie.

La prochaine tâche maintenant est de créer ce qu'on appelle des signaux, des indicateurs qui vont nous montrer quand est-ce qu'il est bénéfique d'acheter des actions ou de vendre ce qu'on possède. Ces signaux se situeront dans les zones de croisement de mobiles moyennes où le momentum va changer.

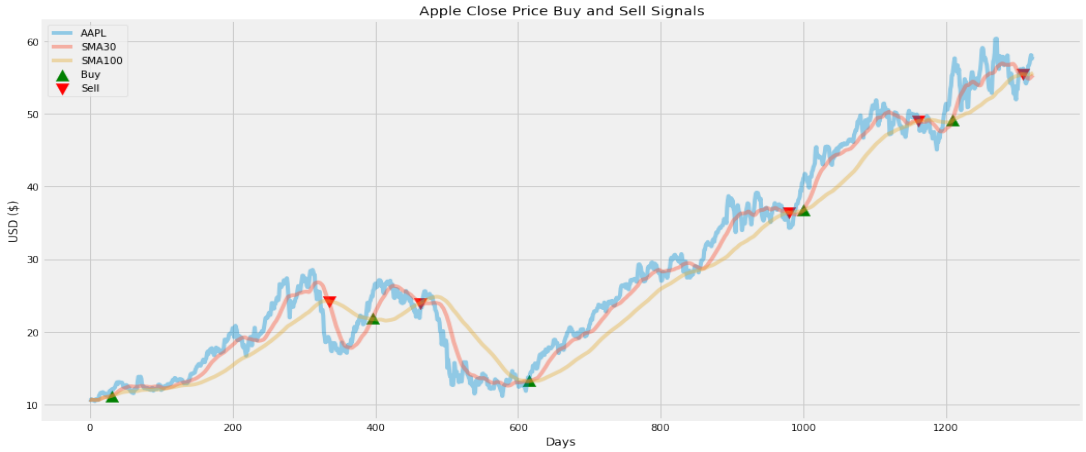


Figure 21.: Placement des signaux sur les points de croisement

Comme nous pouvons le voir les signaux nous suggèrent d'acheter exactement quand le prix est bas, et de vendre après quand le prix augmente remarquablement.

Passons maintenant à l'exploitation de ces signaux, et regardons ensemble les bénéfices que nous pouvons réaliser. Nous avons construit un portefeuille imaginaire et à l'aide de **Pandas** nous avons effectués une simulation de la stratégie si nous suivons exactement ce que les signaux nous proposent de faire. Traçons l'état de notre portefeuille initialisé en départ par 10.000$:

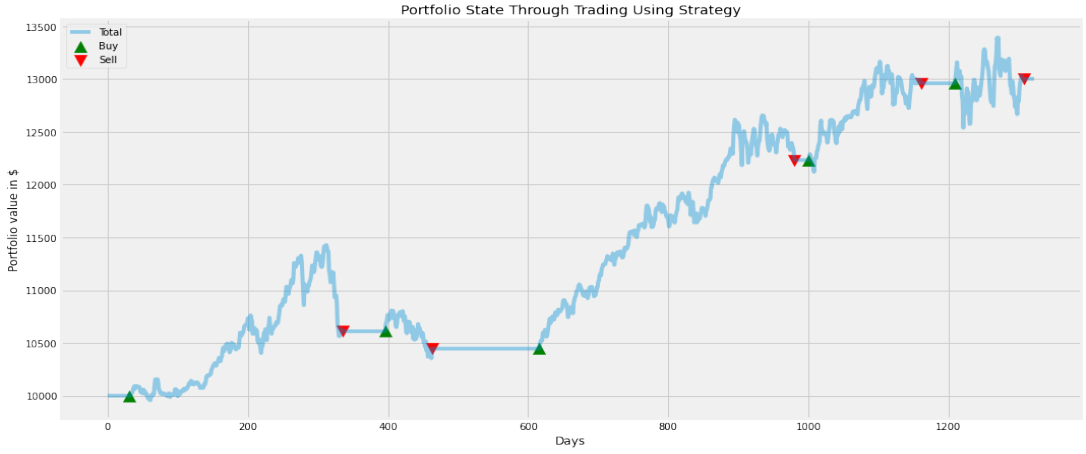


Figure 22.: Placement des signaux sur les points de croisement

Comme nous pouvons le remarquer, en suivant cette stratégie tout au long des **1323 jours** de trading, notre portefeuille a connu plusieurs changements, en globalité on a une marge de gains de 3000$. Ce rendement n'est pas très optimal, même s'il reste non négligeable. Une façon d'améliorer est de changer complètement la stratègie, opter pour une nouvelle, qui propose plusieurs signaux d'achats et de ventes, et qui prend en considération plusieurs autres paramètres, et non seulement les tendances des moyennes mobiles.

9. Conclusion

9.1. Récapitulation

Ce projet vise à la réalisation d'un système de prédiction de prix de cours et de trading automatique. Nous avons construit un système basé sur les techniques d'apprentissage automatique pour prédire les prix de cours des actions boursières et puis décider l'achat et la vente en se basant sur une strategie de trading.

Nous avons réussi la phase du backtesting de notre stratégie, mais cette dernière devrait être améliorée pour faire un plus grand bénéfice.

Comme future aspirations concernant ce projet, nous espérons pouvoir appliquer une meilleure stratégie sur un portefeuille réel, optimiser le modèle à travers plus de réglage d'hyperparamètres. Ensuite nous comptons mettre en oeuvre un bot de trading, lui fournir la stratégie le connecter avec un portefeuille et monitorer ses résultats et puis l'optimiser.

9.2. Discussion

Les informations fournies dans cette mémoire ne sont pas des conseils financiers et aucun des auteurs n'est un professionnel de la finance. Le matériel fourni sur cette mémoire ne doit être utilisé qu'à des fins d'information et ne doit en aucun cas être utilisé pour des conseils financiers. Nous ne faisons aucune déclaration quant à l'exactitude, l'exhaustivité, la pertinence ou la validité des informations. Nous ne serons pas responsables des erreurs, omissions ou pertes, blessures ou dommages résultant de son utilisation.

Assurez-vous de consulter votre propre conseiller financier lorsque vous prenez des décisions concernant votre gestion financière. Les idées et stratégies mentionnées dans cette mémoire ne doivent jamais être utilisées sans avoir d'abord évalué votre situation personnelle et financière, ou sans consulter un professionnel de la finance.

References

[1]Coursera, deeplearning.ai : Deep Learning Specialization
[2]Coursera, deeplearning.ai : Tensorflow in Practice Specialization
[3]Coursera, Hong Kong University of Science and Technology : Python and Statistics for Financial Analysis
[4]Odyssée Tremoulis. Système de trading automatique et prédiction de cours à l'aide de réseaux de neurones.[cs.SE]. 2012
[5]Sepp Hochreiter, Jurgen Schmidhuber - LONG SHORT-TERM MEMORY. 1997
[6]Chenghao Liu1, Steven C. H. Hoi, Peilin Zhao, Jianling Sun - ARIMA Algorithms for Time Series Prediction