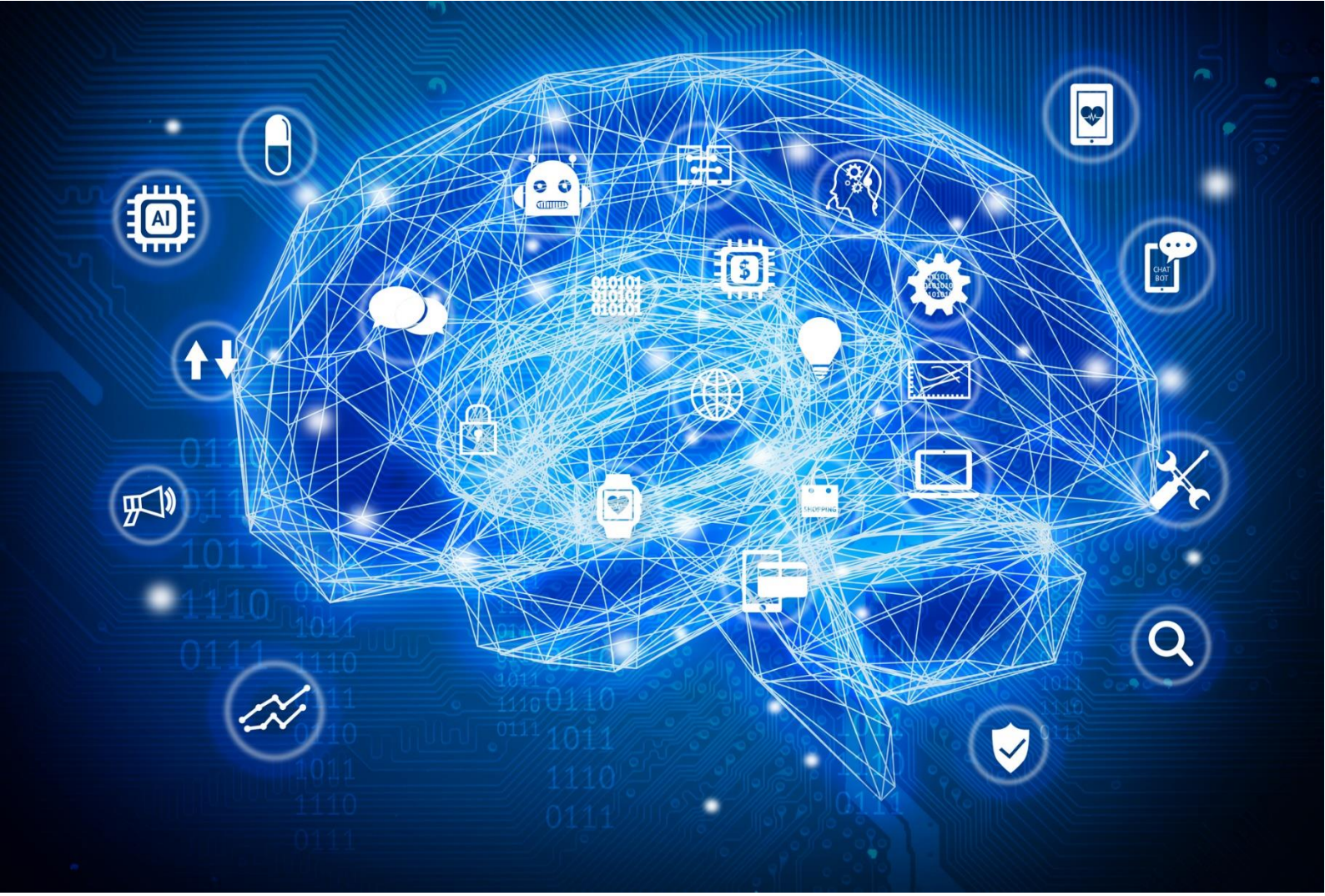

16 novembre 2022

Rapport d'analyse

PREVISIONS DES COURS D'AIRBUS GRACE AU MACHINE LEARNING SUR PYTHON

BRUNET ALEXANDRE – M1 APE SEP

Résumé

Le CAC 40 est le principal indice boursier de la Bourse de Paris mesurant la valeur des actions émises par de nombreuses entreprises sur le marché financier. Les cours de ces actions évoluent continuellement, positivement ou négativement.

Le problème que nous nous posons ici c'est d'établir des prévisions les plus fiables possibles. Cependant, les cours ont une dimension assez incertaine, donc nous devons créer un modèle reposant sur des outils statistiques qui apprenne des données que l'on récolte afin d'en dégager des prévisions quant aux cours futurs. Parmi toutes les entreprises du CAC 40, nous allons intéresser ici au seul cas d'Airbus, un des leaders mondiaux de construction d'aéronautiques.

La démarche repose sur l'hypothèse selon laquelle l'information fondamentale accessible au public dans le passé a des relations prédictives avec les rendements futurs des actions.

Nous avons obtenu certains résultats qui suivent la tendance à la hausse sur la fin de l'échantillon. A partir de ces résultats, nous avons pu établir des prévisions. Ces dernières montrent les cours des actions d'Airbus ont une tendance à la hausse avec le temps.

Sommaire

I/ Introduction

La prévision du cours boursier est un sujet important pour les investisseurs. Elle est utilisée comme outil d'aide à la décision éclairant les investisseurs sur le meilleur moment pour acheter ou vendre des actions.

Les cours des actions du CAC 40 (de l'indice CAC 40) peuvent se trouver sur le site internet de Boursorama. On y trouve beaucoup d'indicateurs pour chaque entreprises qui sont nombreuses elles aussi. Pour notre projet, nous allons nous concentrer sur l'entreprise Airbus en prenant seulement la variation des cours de ses actions et aussi la date et l'heure auxquelles les données des cours ont été récolté.

L'objectif du projet est de mettre en place un outil d'aide à la décision permettant de guider les décisions d'un investisseur. Pour cela, nous avons à note disposition de nombreux outils statistiques dont le célèbre modèle de régression. Mais étant donné qu'il en existe plusieurs, nous avons fait le choix d'en comparer deux : la régression linéaire simple et la régression polynomiale. Autrement dit, nous avons évaluer s'il était plus pertinent de prendre en compte qu'une seule variable (régression linéaire qui est en fait une régression polynomiale à un seul degrés) ou plusieurs (régression polynomiale).

Dans un premier temps, nous avons récolter les données, puis nous les avons manipulés pour qu'elles soient utilisables pour notre modèle. En plus de cela, nous avons pris seulement les éléments nécessaires. Dans un deuxième temps, nous avons principalement utilisé la libraire « Sckit-Learn » qu'offre Python dans le but de créer un modèle de régression le plus fiable possible et ainsi fournir des prédictions qui se rapproche le plus possible de la réalité. Dans un dernier temps, nous avons obtenus des résultats grâce au modèle utilisé.

II/ Collecte des données

Dans cette partie, nous nous concentrerons exclusivement sur le processus de collecte des données. Tout d'abord, la première chose que nous devions faire c'était bien évidemment la création d'un crawler qui viendrais extraire les données du site de Boursorama concernant les cours des actions, sous l'indice du CAC 40. La page internet où l'on les récupère est celle-ci : « https://www.boursorama.com/bourse/actions/cotations/?quotation_az_filter%5Bmarket%5D=1rPCAC&quotation_az_filter%5Bletter%5D=&quotation_az_filter%5Bfilter%5D=&pagination_603981177= ».

Remarque : Sur cette page, nous pouvions voir qu'il y a 2 pages, car trop il y a trop d'entreprise pour toutes les mettre dans une seule page. Mais comme nous avions fait le choix de seulement nous concentrer sur le cas d'Airbus, nous nous sommes contentés de prendre la première page.

Pour récupérer les données que l'on voulait, nous avions eu besoin de la fonction « requests » permettant d'aller vers la page internet et de la fonction « BeautifulSoup » du package « bs4 » permettant de récupérer les données de cette page. Mais cela ne s'arrête pas là. En effet, si nous n'avions fait que récupérer en une seule fois les données, nous en aurions que pour un temps t. Or, pour construire un modèle statistique, nous avons besoin d'une assez grand échantillon. Autrement dit, nous devons répéter l'extraction des données autant de fois que l'on veut. Ceci explique l'intervention de la fonction « time » qui nous a servi dans une boucle « tant que » pour récupérer les données jusqu'à un temps que l'on aura choisis. Ensuite, les informations que nous avions récoltées ont été stocker dans une liste. Nous avions enregistré tout ceci dans une fonction aux variables correspondantes d'une part à l'itération à laquelle la fonction se répète et d'autre part, le temps total de la répétition de la fonction.

L'étape suivante était de manipuler les données pour qu'elles soient traitables. La seconde fonction a pour but de placer les données à leur « place ». Notre méthode d'organisation se base sur l'état des données après leur extraction directe grâce au crawler. Là relevait le plus gros des problèmes qui a mis un certain temps à se résoudre (3 semaines environ). Une fois résolu, nous avions réfléchis s'il était nécessaire de prendre toutes les variables, la réponse est non. Nous nous intéressions en effet qu'aux cours des entreprises (et à leur nom, évidemment), mais nous étions d'abord tentés de seulement prendre la variable correspondante. Seulement, c'était impossible étant donné la méthode de manipulation choisie, donc nous étions contraints de prendre toutes les variables jusqu'aux cours (variable

« Dernier » uniquement). En plus de cela, nous avons créé 2 variables supplémentaires qui sont l'heure et la date à laquelle chaque données sont extraites. Finalement, la fonction ne retournait que les éléments réellement utiles pour nous (donc pas la variable « Dernier »).

À la suite de cela, nous avions défini une fonction telle qu'elle mette en forme les données dans une DataFrame, c'est alors que la fonction « pandas » est intervenu tout naturellement. L'objectif de cette fonction était de créer une DataFrame avec les noms des variables concernant les noms, les cours (Ex_rt = « Exchange rate ») ainsi que les heures et les dates (où l'on a utilisé la fonction « time » une nouvelle fois). Puis, à cette DataFrame « initialisée », on y ajoute notre jeu de données.

Pour finir, la fonction finale a été construite pour rassembler toutes les autres fonctions précédemment expliquées qui retourne une DataFrame contenant toutes les informations des entreprises, leurs cours, etc. Cette DataFrame nous l'avons enregistré sous le format csv (Excel) pour que les données soient utilisées ultérieurement. Avec ce fichier csv, nous avions enlevé toutes les entreprises, excepté Airbus. Ensuite, nous avions exécuté des manipulations pour enregistrer dans un autre fichier csv les cours d'Airbus ainsi que la date et l'heure auxquelles les cours ont été extraites du site de Boursorama. Jusque-là, nous n'avions pas parler de l'itération et le délais de répétition. Car en fait, cela se décidait dans cette fonction-ci. Nous avions alors choisi une itération de 30 secondes car les cours des actions se rafraîchissent toutes les 30 secondes. Et puis nous avions choisis de répéter la fonction pendant 8h30 (soit 30600 secondes car on devait l'exprimer en secondes) car les cours commencent à varier à 9h et finissent de varier à 17h30.

III/ Modélisation

Dans cette partie, nous allons nous concentrer sur le choix du modèle ainsi que son application.

Mais avant de choisir le modèle et pour simplifier le traitement des données, nous avons fait le choix de supprimer la colonne « Timestamp » pour la remplacer avec une colonne contenant une liste de chiffre allant de 1 au nombre d'observation total (1016 dans notre cas). Nous pouvons effectivement faire cette modification car le modèle ne prendra pas réellement en compte les dates. Une fois cette modification faite, nous avons redimensionner les deux colonnes de notre table de données, en les transformant d'abord en array avec la fonction « numpy ». L'utilité est simple : si nous ne nous redimensionnons pas les données, elles ne seront pas traitées correctement par la suite.

Ensuite est venu du choix du modèle que l'on utilisera pour faire des prédictions. Nous avions pensé au modèle de prédiction le plus connu : la régression, mais il en existe plusieurs (linéaire, polynomiale, logistique, etc.), toutes utilisent la fonction « LinearRegression » du package « sklearn.linear_model ». Naturellement, nous avons initialement prévu d'utiliser la régression linéaire simple (car une seule variable explicative). Cependant, si on calcule le R^2 de ce modèle, en utilisant la fonction « r2_score » du package « sklearn.metrics », et étant donné les données récupérées grâce au crawler, nous avons vu qu'il était de 50% environ. Autrement dit, 50,32% de la variance est expliquée par le modèle, ce qui est très peu. Donc la solution était d'utiliser le modèle de régression polynomiale. Ce modèle fonctionne comme la régression linéaire mais la différence est que plusieurs degrés sont pris en compte.

Remarque : en réalité, la régression linéaire est une régression polynomiale mais à un seul degrés.

Donc nous avons comparé ces deux modèles sur 2 critères : le R^2 et la courbe de régression (si la courbe suit bien la dispersion des points ou pas). Mais avant cela, nous devions faire le choix du degrés de la régression polynomiale. Nous avions testé 2 degrés puis 10 degrés grâce à la fonction « PolynomialFeatures » du package « sklearn.preprocessing », mais le R^2 des deux cas n'était pas satisfaisant. Nous sommes arrivés à choisir le degrés 6 étant donné que le R^2 y est le plus élevé (nous pouvions prendre un degrés 7 avec un R^2 un tout petit plus élevé mais la différence est tellement faible entre le degrés 6 et le degrés 7 que cela ne valait pas le coup de l'utiliser).

Après avoir choisi le degrés de la régression polynomiale, il fallait visualiser les 2 régressions (figure 1), grâce à la fonction python « matplotlib.pyplot ». Le résultats est sans appel, le choix régression polynomiale se révélait être le beaucoup plus pertinent (figure 2) car la courbe suivait beaucoup mieux les points des données que la courbe de la régression linéaire. Cela semblait être un choix pertinent même sans visualiser la courbe car le R^2 de la régression polynomiale de degrés 6 était de 85% environ.

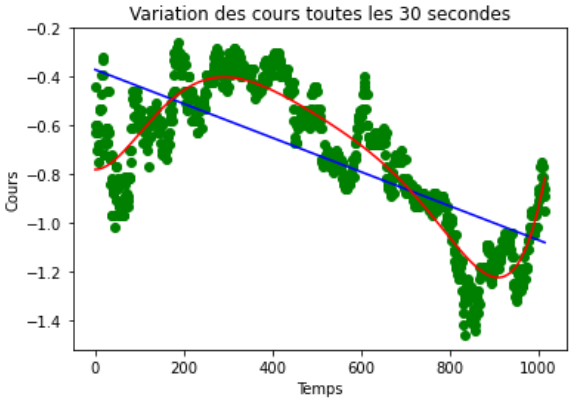

***Figure 1**: Régression linéaire et polynomiale*

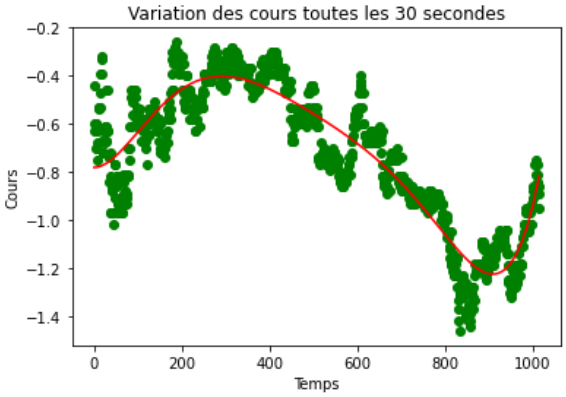

***Figure 2**: Régression polynomiale seule*

Après les évaluations des 2 modèles, vient l'utilisation du modèle de régression polynomiale. Pour cela, il nous faut diviser notre échantillon en une partie apprentissage et une partie test. 80% de l'échantillon (les premières observations) a été utilisé pour l'apprentissage du modèle et 20% de l'échantillon (les dernières observations) a été utilisé pour tester le modèle et ainsi voir si le test ressemblait bien à la réalité. Cette division a pu se faire grâce à la fonction « train_test_split » du package « sklearn.model_selection ».

IV/ Résultats et perspectives

Maintenant, nous allons nous concentrer sur les résultats qu'a obtenu le modèle que l'on a construit.

D'abord si nous revenus sur les valeurs prédites pour les 20% des dernières observations (échantillon-test), nous pouvons dire que la différence entre les valeur des prédictions et les réelles valeurs peut varier au plus de quelques dixièmes. Donc à partir de là, nous pouvons nous dire que le modèle reste assez correct. D'ailleurs il ne faut pas s'attendre à mieux car, rappelons-le, le R^2 de ce modèle est de 85,21%, c'est correct, certes, mais c'est loin de refléter parfaitement les réelles observations. A partir de ces tests assez concluants, nous pouvions utiliser le Machine Learning pour effectuer des prévisions.

Nous avons fait des prédictions sur toute la journée suivant la collecte des données, en l'occurrence, le 09/11/2022 mais pas à une intervalle de 30 secondes, mais à une intervalle de 30 minutes. La raison est simple : les prédictions suivent toujours une tendance croissante avec le temps. Donc il était inutile de donner des prédictions toutes les 30 secondes. Nous obtenons les résultats se trouvant dans le tableau 1.

Pour trouver les prédictions pour le lendemain, il a donc fallu donner des valeurs en abscisses (des dates) mais étant donné que ces dates sont sous le format de 1, 2, 3, etc. la première valeur de prédictions était donc le nombre total d'observations + 1, donc 1017. Et comme les cours commencent à varier à partir de 9h, la valeur 1017 fait référence à la première heure du jour suivante de l'extraction des observations (donc le 09/11/2022 à 9h00). Comme nous voulons une itération de 30 minutes, nous devons ajouter 60 à la valeur précédente jusqu'à atteindre 2032 (2*1016, c'est-à-dire le jour suivant l'extraction à 17h30). Donc les valeurs sont 1017, 1077, 1137… 2032.

Ensuite, au lieu d'avoir ces nombres qui ne veulent rien dire en tant que colonne de la DataFrame finale qui retourne les prédictions, nous avons créé une liste contenant toutes les heures (les temps-clés) à laquelle correspondent les prédictions. Nous avons concaténer les prédictions avec les temps-clés pour retourner une DataFrame (tableau 1) beaucoup plus claire et parlante avant de l'enregistrer en tant que fichier csv pour un potentiel futur traitement.

Key-times	**Prédictions pour le jour suivant la récolte des données (arrondis au millième près)**
9:00	-0.796
9:30	0.096
10:00	1.865
10:30	4.988
11:00	10.099
11:30	18.028
12:00	29.827
12:30	46.813
13:00	70.603
13:30	103.157
14:00	146.826
14:30	204.393
15:00	279.130
15:30	374.844
16:00	495.939
16:30	647.467
17:00	835.196
17:30	1044.654

Tableau 1: *Prédictions pour le 9/11/2022*

Cette augmentation exponentielle s'explique par le fait que les dernières observations (voir figure 2) traduisent une augmentation rapide, ce qui a « faussé » la courbe de régression qui a pensé que les valeurs allaient très vite et indéfiniment augmentées.

V/Conclusion

Pour conclure nous pouvons dire que ce projet a nécessité la mobilisation de plusieurs compétences. D'abord la collecte des données et faire en sorte qu'elles soient traitables et sélection des informations importantes sans les altérer. Puis évaluation et modélisation d'un algorithme permettant de s'emparer des données pour les résumer au mieux et ainsi réaliser des prédictions les plus fiables possibles.

Nous avons donc choisi d'utiliser le modèle de régression polynomiale (de degrés 6), un modèle à la fois certes moins connu que le modèle de la régression linéaire mais bien plus pertinente car il prend en compte l'évolution des cours avec un seul échantillon de données.

Cependant, même si l'idée du modèle de régression polynomiale paraissait assez pertinente (même avec une variance expliquée assez élevée), les prédictions du jour J+1 (pour le lendemain), sont incohérentes car les cours vont jusqu'à plus de 1040% en fin de journée. Si nous ne regardions pas cette énorme incohérence, nous pouvions conseiller les investisseurs en leur disant d'investir en fin de journée. Mais personne ne conseillerait des investisseurs dans cette situation. De ce fait, le modèle choisi est bon mais il faut d'abord changer le degrés qui peut ne pas être le meilleur selon les échantillons, et s'assurer que le R^2 soit au moins supérieur à 95% pour être plus efficace.