Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Mecánica

Predicción de acciones de bolsa

Autor:

Muñiz Conde, César

Marta Herráez Sánchez

ENSAM París

Valladolid, septiembre, 2022

TFG REALIZADO EN PROGRAMA DE INTERCAMBIO

TÍTULO:	Prédiction de stock boursier
ALUMNO:	César Muñiz Conde
FECHA:	02/02/2022
CENTRO:	École Nationale Supérieure d'Arts et Métiers – Campus de Paris
UNIVERSIDAD:	École Nationale Supérieure d'Arts et Métiers
TUTOR:	Pascal Caestecker

Resumen

Este proyecto forma parte del curso de especialización del tercer año del programa "*Grande École*" francés en la *École Nationale d'Arts et Métiers.*

El objetivo del proyecto es analizar un problema de predicción en las acciones de bolsa. Para ello, se trabajará sobre la previsión de una acción propuesta por la empresa CFM en colaboración con la plataforma Data Challenge.

En el transcurso del proyecto, primero se prepararán los datos que se utilizan en los algoritmos de predicción. En segundo lugar, se intentarán modelar varios algoritmos que permitan realizar la predicción. Estas dos etapas han sido iterativas, con numerosas modificaciones y nuevas propuestas, sabiendo que cada vez que se realizaba una modificación, podía tener un gran impacto en los resultados finales.

Por último, se presentan y comentan los resultados, se presenta la solución elegida y algunas áreas de mejora.

Palabras clave: Machine Learning, predicción, algoritmo, acciones, datos.

Summary

This project is part of the third-year specialization course of the French "Grande École" programme at the École Nationale d'Arts et Métiers.

The objective of the project is to analyze a prediction problem in stock exchanges. To this end, work will be done on the anticipation of an action proposed by CFM in collaboration with the Data Challenge platform.

During the project, the data used in the prediction algorithms will first be prepared. Secondly, we will try to model several algorithms that allow us to make the prediction. These two stages have been iterative, with numerous modifications and new proposals, knowing that each time a modification was made, it could have a great impact on the final results.

Finally, the results are presented and commented, the chosen solution and some areas of improvement are presented.

Keywords: Machine Learning, prediction, algorithm, actions, data.

Executive Summary

Ce projet s'inscrit dans le cadre du projet d'expertise en 3e année du Programme Grande École à l'École d'Arts et Métiers, plus précisément en lien avec les cours de Machine Learning assuré par Mme. NGUYEN Angie.

L'objectif du projet est d'analyser une problématique de prédiction. Pour cela, on a choisi d'aborder la prédiction de stock boursier proposé par l'entreprise CFM en collaboration avec la plateforme Data Challenge.

Au cours du projet, on a essayé premièrement de préparer les données qui ont été postérieurement utilisés dans les algorithmes de prédiction. Deuxièmement, on a essayé de modéliser plusieurs algorithmes nous permettant de réaliser la prédiction. Ces deux étapes ont été itératifs, des nombreuses modifications et nouvelles propositions ont été réalisées, tout en sachant que chaque fois qu'une modification été fait en amont, cela pouvait avoir un grand impact dans les résultats finaux.

Finalement, on expose et commente les résultats, la solution retenue et en propose quelques axes d'amélioration.

Table de matières

Table des figures

1. Introduction

Ce projet s'inscrit dans le cadre du projet PJE9 du programme académique « Programme Grande Ecole ENSAM ». Le projet relève le défi proposé par *Capital Fund Management* (CFM) une entreprise de gestion d'actifs en collaboration avec *Challenge Data* une plateforme appartenant à l'ENS et Collège de France.

Le défi est le suivant :

Dans de nombreuses bourses, à la fin d'une journée de négociation, une vente aux enchères a lieu pour chaque action. Chaque action est alors échangée à un prix unique, en fonction de l'intérêt que les participants du marché portent à l'enchère.

Il est avantageux d'effectuer certaines opérations pendant cette vente aux enchères plutôt que pendant la négociation continue qui précède, car les coûts de négociation sont généralement moins élevés.

- En outre, certains participants du marché (*day traders, market makers*...) préfèrent ne pas détenir d'actions pendant la nuit, car des événements peuvent affecter le cours de l'action entre la clôture du marché et le cours à l'ouverture le lendemain (élections, annonces de sociétés, etc.), ce qui peut entraîner une perte. Pour ces participants du marché, la vente aux enchères est la dernière chance de limiter un tel risque, car c'est l'occasion de se débarrasser de leurs stocks restants et de ne pas en détenir pendant la nuit.
- Pour les acteurs du marché qui souhaitent au contraire détenir un nombre déterminé d'actions avant la fin de la journée (gestionnaires d'actifs, etc.), la vente aux enchères est leur dernière chance d'atteindre cet objectif. Ceci est en principe important, car ils ont optimisé ce nombre d'actions à détenir. Par exemple, s'ils prévoient que le prix d'une action devrait augmenter, ils ont intérêt à acheter le plus d'actions possible, dans les limites fixées par le montant qu'ils peuvent investir et par le risque financier qu'ils sont prêts à prendre.

Les participants du marché peuvent donc vouloir estimer le nombre attendu d'actions disponibles lors de l'enchère, car cela leur permet d'évaluer le nombre d'actions qu'ils peuvent espérer acheter ou vendre lors de cette dernière opportunité d'échange financièrement avantageuse.

L'objectif du projet est de prédire le volume (valeur totale des actions échangées) disponible pour les enchères, pour 900 actions sur environ 350 jours.

2. Organisation du projet

Le projet « Prédiction de Stock Boursier » a été réalisé en binôme par :

- Daniel MARTINEZ-ZABALETA
- César MUÑIZ CONDE

La date de début du projet a été le 6 octobre 2021 et il a été présenté le 2 février 2022 avec une durée de 18 semaines.

Limites du projet :

Pendant la réalisation du projet, on a trouvé différents obstacles qui nous ont fait ralentir le déroulement normal du travail. Les deux limites principales du projet ont été le temps limité pour sa réalisation et la capacité de nos ordinateurs pour soutenir le code informatique.

Le limite du temps nous a conditionné à ne pas arriver à une meilleure solution finale, car ce projet a été intégré à la fois sur le Programme Grande École ENSAM en troisième année, avec la limite d'être rendu pour le mois de février.

La capacité de nos ordinateurs nous a fait reconsidérer quelques solutions de résolution du problème. Effectivement, des nombreuses propositions de calculer et/ou optimiser le résultat non pas été capables d'être retenues. Malgré ça, d'autres algorithmes plus simples ont été proposés.

Une troisième limite a été le moment où on a reçu les cours de Machine Learning dans le PGE. La formation a été réalisé vers la fin du semestre, au moment où la plupart du projet avait déjà été réalisé. Ainsi, nous n'avons pas pu capitaliser cette connaissance depuis le début du projet.

Outils utilisés :

- Microsoft Teams (collaboration)
- Pack office
- Python (IDE Spyder) + libraries
- Git + Source tree (Collaboration, contrôle des versions et contrôle de l'historique du projet)
- Google collab

Organisation :

Le diagramme de Gant [Annexe I] montre l'avancement réalisé sur le projet au long des 18 semaines ainsi que les différentes tâches dont on a divisé le travail et le responsable (s) de la mise en œuvre.

3. Intérêt du projet

Ce projet suit le programme académique Programme Grande Ecole ENSAM pour la troisième année en expertise, et requiert des compétences acquises pendant les années précédentes à l'Ecole.

Pendant le déroulement du projet, la combinaison de compétences stratégiques et techniques seront utilisés, ainsi que celles d'innovation.

Le principal intérêt du projet est l'apprentissage en termes de Machine Learning et concepts financiers.

On devra donc réaliser des analyses prédictives en exploitant les données issues d'une grande base de données et traitées par des algorithmes statistiques et des techniques de Machine Learning, afin de prédire la variable objective en se basant sur le passé.

Les analyses prédictives sont réalisées à partir de plusieurs disciplines et technologies afin de permettre aux entreprises de prévoir les tendances et résultats financiers de demain.

L'intelligence artificielle et le Machine Learning constituent sans aucun doute le niveau supérieur de l'analyse de données, les systèmes informatiques étant désormais capables d'apprendre en permanence des données captées par les entreprises afin de prédire intelligemment les besoins des consommateurs, les tendances futures de tel ou tel marché et bien plus encore. Un tel niveau d'expertise ne pouvant être atteint que par des systèmes informatiques cognitifs, capables de comprendre des données non structurées et structurées, afin d'en extraire des analyses prédictives affinées au fur et à mesure de chaque interaction.

C'est pourtant une technique très en vogue et demandée par les entreprises.

4. Données

Pour ce projet, les données nécessaires à l'étude sont fournies par Challenge Data. La prédiction du volume d'enchères pour un stock et un jour donné peut être faite sur la base des 126 colonnes d'entrée suivantes :

- **pid** : un Product ID, qui représente un stock.

- **day** : jour de l'échantillon de données, sous forme d'un nombre entier. L'ordre est chronologique, le jour 0 venant avant le jour 1, etc.

- **abs_ret** (n de 0 à 60) : valeurs absolues des rendements des actions (variation relative des prix) entre le dernier prix connu (typiquement le prix au début de la période n) et la fin de la période n (en pourcentage), où les périodes couvrent une bonne partie de la journée de négociation, ne se chevauchent pas et ont la même durée. Le rendement n=0 vient avant le rendement n=1, etc.

- **rel_vol** : comme abs_retn, mais représente le volume de l'action négociée comme une fraction du volume négocié pendant la période couverte (ainsi, leur somme est égale à 1, sur une journée). Les périodes sont les mêmes que pour les rendements.

- **LS** et **NLV** : deux quantités associées aux transactions du jour pour le titre en question. Leur nature n'est pas divulguée pour ce défi.

Les données de sortie contiennent, pour une action donnée et un jour donné, le logarithme naturel du volume d'enchères (= valeur totale des actions échangées), en tant que fraction du volume total dans les 61 périodes données. Ainsi, si le volume des enchères représente 10 % du volume négocié sur l'ensemble des périodes d'une journée, la cible est log 0,10 = -2,30...

Les 900 actions présentes dans les données d'entraînement et de test sont les mêmes : il est donc en principe possible de concevoir des prédictions personnalisées pour chaque action.

Les données d'entraînement contiennent des informations sur environ 800 jours différents, tandis que les données de test nécessitent des prédictions de volume d'enchères pour environ 350 jours.

En outre, les données de test correspondent à des jours qui viennent après ceux des données d'entraînement. Une difficulté réside dans le fait que les volumes d'enchères peuvent évoluer dans le temps (par exemple en devenant relativement plus grands et plus importants au fil du temps), mais nous ne voyons que ce à quoi ressemblent les volumes d'enchères passés (formation).

5. Traitement des données

Les suivantes étapes nous permettent dans un premier temps de nous familiariser avec la nature des différents données et validé dans deuxième temp la qualité des données. Cette étape est indispensable car les résultats des algorithmes et en conséquence du projet, dépendent fortement de l'entrée. De plus, la programmation des algorithmes ne supporte pas les valeurs du type NaN, infinie etc…

5.1 Exploration des données

Premièrement on vérifie la consistance des données d'une même variable au long de toute la base de données. Le résultat de l'exploration a été le suivant :

```
In [3]: train_input.dtypes
Out[3]:
ID              int64
pid             int64
day             int64
abs_ret0      float64
abs_ret1      float64
                ...
rel_vol58     float64
rel_vol59     float64
rel_vol60     float64
LS            float64
NLV           float64
Length: 127, dtype: object
```

Figure 1 : Exploration des données

Puis, on analyse quelques statistiques (*Voir Figure 2*). Pour l'ensemble de variables, on a extrait le nombre de valeurs, la moyenne, la déviation standard, le minimum et maximum de la distribution et la valeur correspondant au 25%, 50% et 75% de la distribution quand la distribution est ordonnée en ordre croissante.

```
In [39]: df_all_input["target"].describe()
Out[39]:
count    684482.000000
mean         -1.958691
std           0.909245
min          -7.137686
25%          -2.513789
50%          -1.948804
75%          -1.395366
max           3.580919
Name: target, dtype: float64
```

Figure 2 : Exploration des variables

Représenter les distributions de certaines variables, voir LS et NLV du *train input* ou le *Target* de la distribution output nous a aider à mieux comprendre les données. Les distributions des variables sont les suivantes :

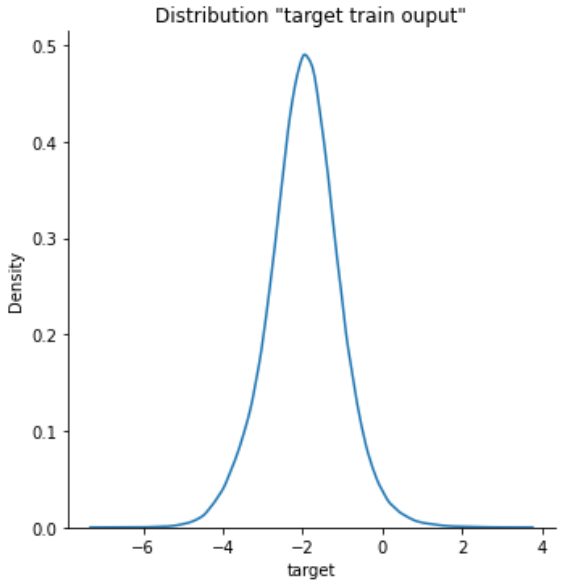


Figure 3 : Distribution "Target train output"

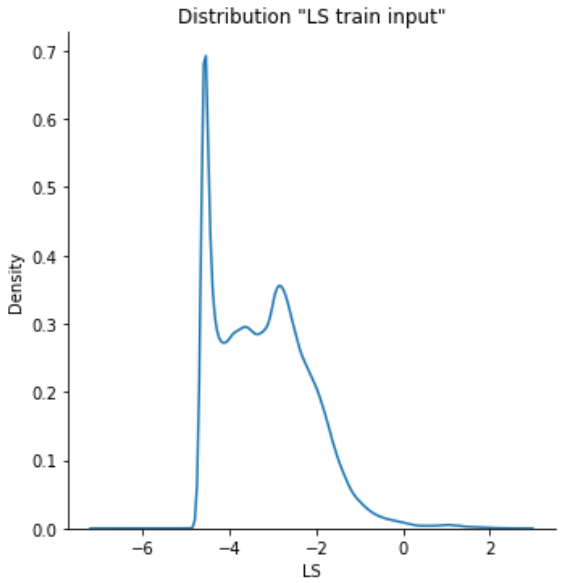


Figure 4 : Distribution "LS train output"

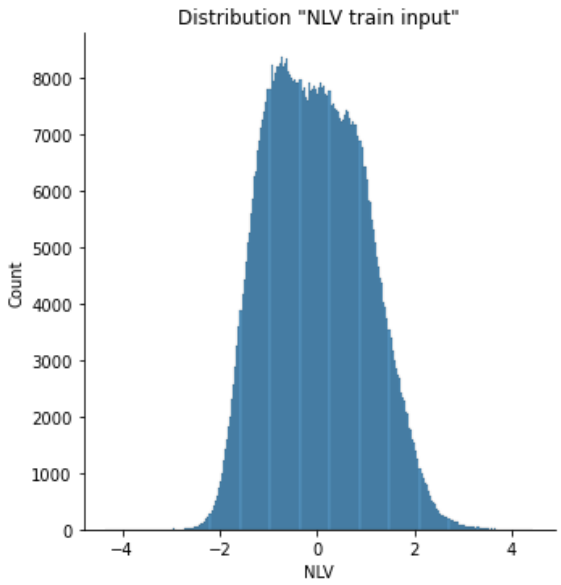


Figure 5 : Distribution "NLV train input"

Pour la distribution du Target, on c'aperçoit qu'elle est normale. Pour le NLV en *input*, même si on ne connaît pas l'origine et la base de la variable, elle a une distribution qui se ressemble beaucoup à celle du *target* en *output*. Pourtant, il y a un potentiel *feature* à exploiter pour prédire le *target* à partir de cette variable NLV.

On a aussi analysé la matrice de corrélations pour ces trois variables. Le résultat montre qu'il n'y a pas une forte corrélation entre elles. Avec la deuxième matrice de corrélations, on cherche à analyser la corrélation entre la variable objectif et le volume relatif. On observe que même si la corrélation est petite, à la fin de la journée il est possible d'apprécier que celle-ci est un peu plus forte qu'au début.

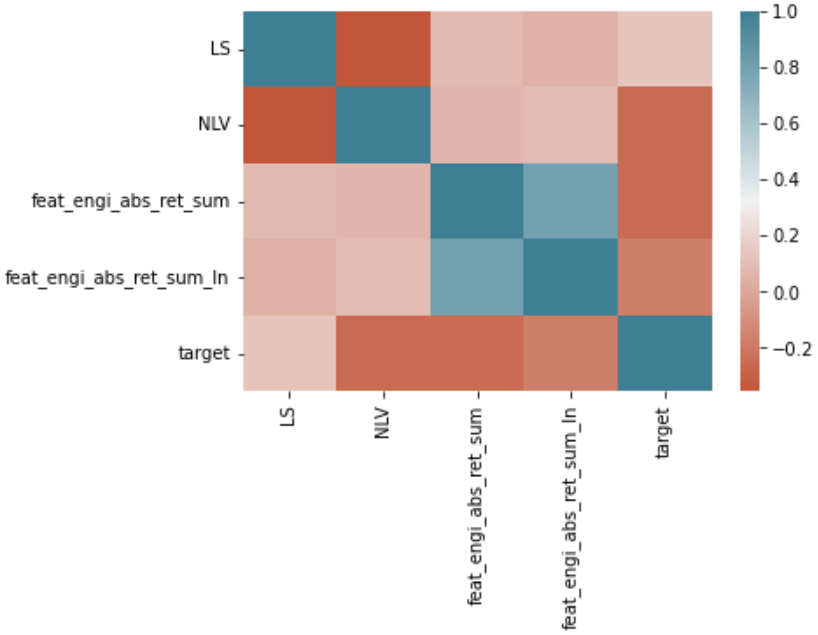


Figure 6 : Matrice de corrélations

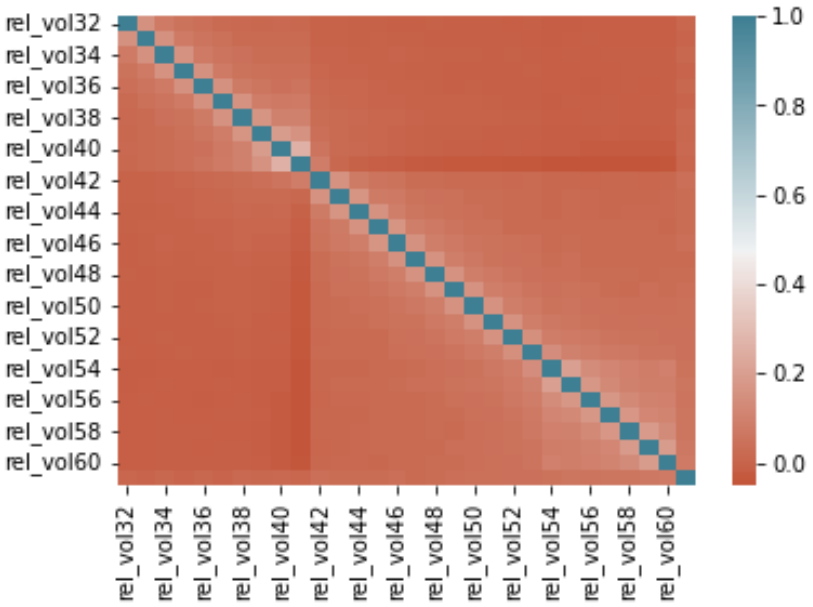


Figure 7 : Matrice de corrélations pour le volume rélatif

La distribution suivante montre la moyenne pour les variables abs_ret___ tout au long de la journée. La moyenne est beaucoup plus élevée au début qu'à la fin, et elle commence à acquérir un caractère constant vers la valeur 0.10.

On observe que la variation relative des prix a une tendance décroissante pendant le jour, cela peut être expliquer à cause de la grande volatilité, nouvelles informations et nouvelles positions prises pendant que le marché est fermé.

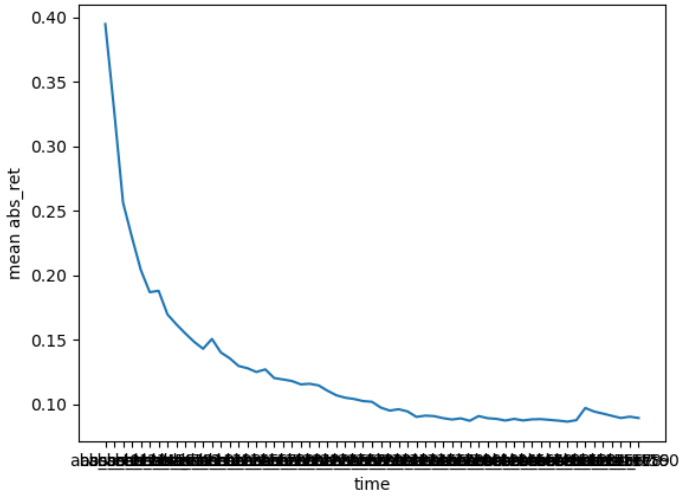


Figure 8 : Variation relative des prix

On calcule aussi la même distribution mais pour la moyenne du volume relatif, et on obtient une tendance similaire à la précédente.

La tendance est décroissante pendant le jour jusqu'à la fermeture du marché, où le volume a tendance augmenter. La tendance générale peut être expliquer de la même façon que dans le cas précédent. De plus, l'augmentation avant la fermeture du marché peut être expliquer par la nécessité de fermer des positions.

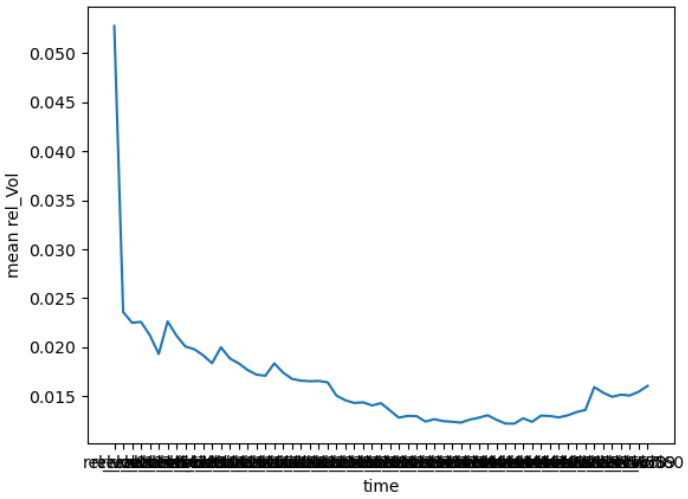


Figure 9 : Évolution de la moyenne pour le volume relatif

5.2 Nettoyage

Après une exploration de la base de données, la plupart des données initiale a été validé ; le problème principal à résoudre est la substitution des valeurs NaN. Pour cela, on a étudié les différentes options dont on dispose. Le deuxième problème rencontré a été l'existence des valeurs aberrantes dans certaines variables.

```
In [23]: print(train_input.isnull().sum())
ID              0
pid             0
day             0
abs_ret0     4996
abs_ret1    27779
             ...
rel_vol58   31479
rel_vol59   30558
rel_vol60   29959
LS              0
NLV             0
Length: 127, dtype: int64
```

Figure 10 : Exploration valeurs NaN

5.2.1 Valeurs NaN du volume relatif (rel_vol)

Le « rel_vol » représente le volume de l'action négociée comme une fraction du volume négocié pendant la période couverte. Ainsi, on vérifie si leur somme est égale à 1, sur l'ensemble d'une journée.

```
def feat_engi_rel_vol_sum (data_input):
    col_num_init = data_input.columns.get_loc("rel_vol0") #get the column number
    col_num_fin = data_input.columns.get_loc("rel_vol60")
    data_input['feat_engi_rel_vol_sum']=data_input.iloc[:,\
                        col_num_init:col_num_fin+1].sum(axis=1) #sum all the columns from 0 to 60 included
    return data_input
```

Figure 11 : Code de la fonction « feat_engi_rel_vol_sum »

```
In [9]: train_input['feat_engi_rel_vol_sum'].describe()
Out[9]:
count    6.844820e+05
mean     1.000000e+00
std      2.629818e-15
min      1.000000e+00
25%      1.000000e+00
50%      1.000000e+00
75%      1.000000e+00
max      1.000000e+00
Name: feat_engi_rel_vol_sum, dtype: float64
```

Figure 12 : Résultats statistiques de la fonction « feat_engi_rel_vol_sum"

On peut observer dans l'image ci-dessus que la somme pour toutes lignes est 1 et par conséquent, les valeurs NaN des colonnes respectifs peut être substituer directement par la valeur zéro.

5.2.2 Valeurs NaN du retour absolue (abs_ret)

Ce deuxième est bien plus compliqué. Au contraire que pour le cas précèdent, ce n'est pas possible de poser une inéquation ou égalité qui doit être vérifier et qui nous permet de calculer les valeurs NaN. Pourtant, ce problème est très récurrent dans le monde du traitement de données, à continuation on proposera quelques méthodes (plus ou moins complexes) pour résoudre le problème.

5.2.2.1 Substitution des valeurs NaN par des zéros

Cette méthode consiste en remplir les valeurs Nan de toutes les colonnes abs_ret par des zéros. Le remplissage par des zéros peut être valable, car due à certaines situations

externes, le stock absolu n’est pas actif ou n’est pas en vente et donc sa valeur est égale à zéro.

C’est une méthode facile à implémenter dans le programme et nous permet de ne pas devoir éliminer observations.

5.2.2.2 Substitution des valeurs NaN par interpolation linéaire

Cette méthode permet de remplir les valeurs NaN des colonnes abs_ret par une valeur entre l’instant précèdent et le postérieure. Dans certains cas où le première et/ou le dernière valeur est NaN, on substitue la valeur par zéro. Si les valeurs NaN sont dues à une perte de l’information, l’interpolation est une approximation pour remplir les données vides. Cependant, si les NaN sont dus à de interruptions du stock dans le marché où d’autres évènement qui rendent illiquides l’action, l’approche de substituer les NaN par des zéros est plus pertinente.

5.2.2.3 Substitution des valeurs NaN par la valeur de la moyenne

La moyenne est utilisée pour les distributions avec un faible nombre de valeurs atypiques. Si les données comparées sont pour la plupart uniformes, il est possible d’utiliser sans risque l'agrégateur de moyenne. Toutefois, si l'ensemble des chiffres comporte des valeurs atypiques, il est recommandable d’utiliser la médiane ou de filtrer les valeurs qui faussent les résultats. Dans notre cas, le filtrage des valeurs aberrants peut s’avérer nécessaire.

Cette méthode peut être assez puissante pour donner un bon résultat si elle est bien appliquée. En revanche, le danger de cette méthode est de créer une distribution encore plus biaisée et peu fiable si la moyenne n’est pas représentative, et de ne pas avoir une base de données complète du au filtrage.

A travers cette méthode, on cherche d’approximer toutes les valeurs manquantes (NaN) aux valeurs moyennes de chaque PID. Pour que cette méthode soit applicable, il est nécessaire qu’il n’y aille pas des valeurs « hors moyenne », c’est-à-dire, des valeurs très différents qui nous fassent avoir une distribution biaisée et qui affectent à la valeur de la moyenne.

Premièrement, on a filtré ces valeurs extrêmes. On a utilisé la méthode des percentiles pour éliminer les pics de valeurs et le résultat a été le suivant (Fig 13 - 16). Pour mieux montrer l’effet de ce filtrage, on prendra comme exemple les valeurs abs_ret0 pour le PID : 360 (nombre suffisant de valeurs pour voir clairement l’effet du percentile).

Sur la première image on voit le spectre de valeurs sans appliquer le filtrage. Il est possible de différencier au moins 8 valeurs qui sont très loin de la moyenne, et beaucoup de valeurs qui sortent de la marge 0 – 0,8 dans la colonne abs_ret0, où est placé la plupart des valeurs. Le reste d'images montrent un filtrage progressif selon le percentile appliqué : 99%, 97% et 95%.

À partir du percentile 97% il est plus difficile de trouver des pics de valeurs, et la valeur de la moyenne sera plus fiable au moment de le remplacer par les valeurs Nan.

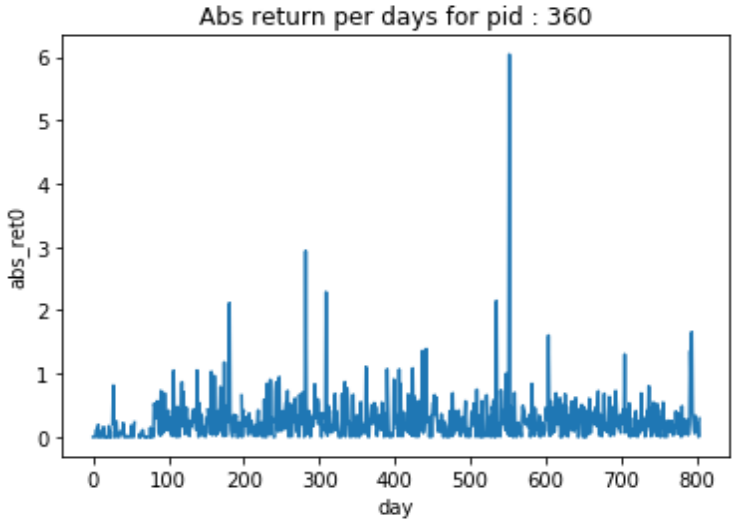


Figure 13 : Retour absolu par jour pour le PID 360

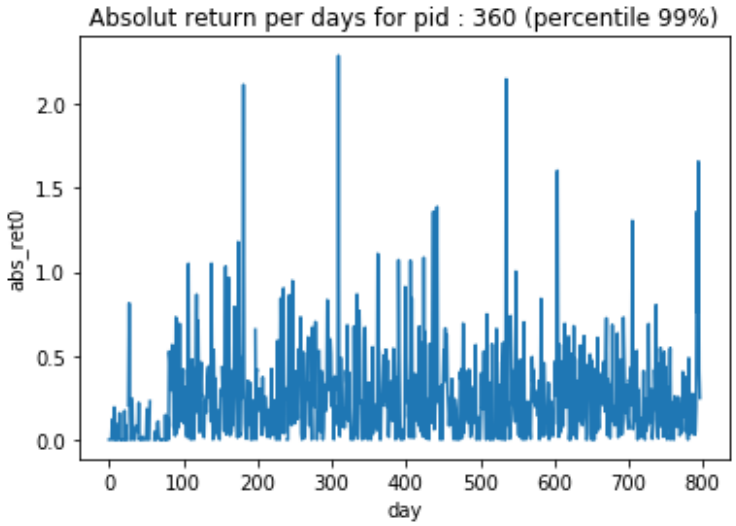


Figure 14 : Retour absolu par jour pour le PID 360 avec le percentile 99%

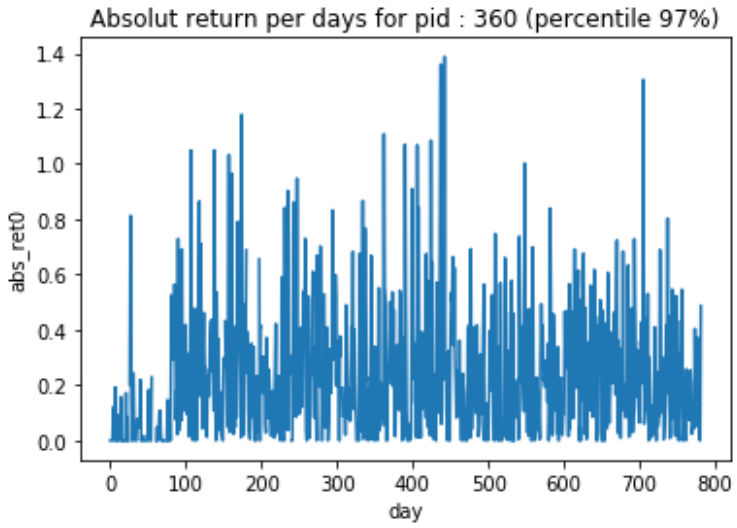


Figure 15 : Retour absolu par jour pour le PID 360 avec le percentile 97%

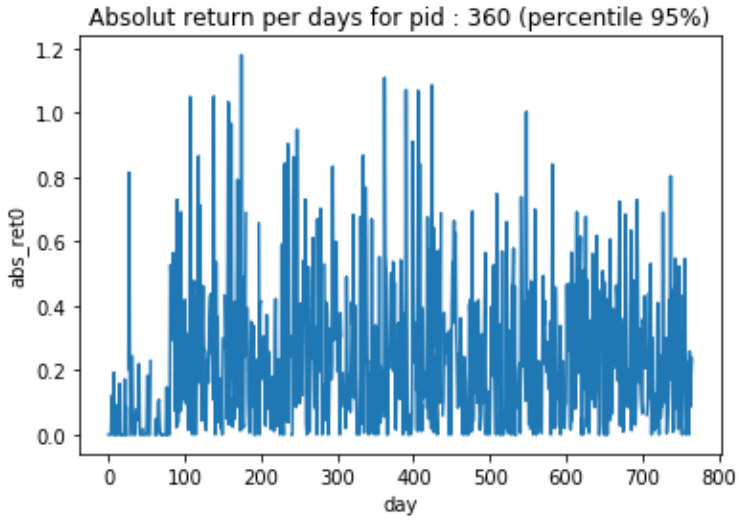


Figure 16 : Retour absolu par jour pour le PID 360 avec le percentile 95%

Une fois les valeurs sont filtrées, le programme calculera pour chaque PID la moyenne de chaque abs_ret. On aura donc un vecteur de 61 valeurs moyennes. Pour optimiser le temps de traitement de données, le programme calcule la moyenne de ces 61 valeurs en obtenant une moyenne globale pour chaque PID. Cette valeur globale sera celle qu'on remplacera par les valeurs Nan.

En résumé, on aura 900 différentes valeurs moyennes à utiliser correspondant aux 900 différents PID.

5.2.2.4 Substitution des valeurs NaN par la valeur de la médiane

La moyenne et la médiane jouent un rôle similaire dans la compréhension de la tendance centrale d'un ensemble de données. La moyenne a traditionnellement été une mesure populaire du point médian d'un ensemble ; toutefois, elle présente l'inconvénient d'être

influencée par des valeurs distinctes qui sont beaucoup plus élevées ou plus faibles que le reste des valeurs. C'est pourquoi la médiane est une meilleure mesure du point médian dans les cas où un petit nombre de valeurs atypiques pourrait fausser considérablement la moyenne.

L'application de cette méthode est potentiellement intéressante, parce qu'elle nous permet d'avoir une mesure centrale en évitant les valeurs aberrantes. Aucune opération de filtrage est requise pour mettre en œuvre la méthode. Le risque à assumer est de ne pas avoir une bonne précision dans la valeur qui remplira les NaN, puisque la médiane prend la valeur centrale de la distribution lorsqu'elle est ordonnée.

5.2.2.5 Substitution des valeurs NaN par l'algorithme MICE

MICE est l'abréviation de l'algorithme « *Multivariate Imputation By Chained Equations* », une technique qui permet de substituer les valeurs manquantes d'un ensemble de données en examinant les données d'autres colonnes et en essayant d'estimer la meilleure prédiction pour chaque valeur manquante [5].

L'avantage de cette méthode est que chaque valeur est différente et indépendamment calculé pour chaque valeur manquante NaN. De cette manière, on peut utiliser comme alternative une méthode qui n'assigne pas la même valeur à tous les valeurs NaN qui appartient au même PID.

L'algorithme MICE peut être décomposé en quatre étapes générales :

- Étape 1 : Un remplissage simple, telle que celle de la moyenne, est effectuée pour chaque valeur manquante dans l'ensemble de données. Cette substitution par la moyenne peut être considéré comme une première approximation.
- Étape 2 : les valeurs de la moyenne « substituts » pour une variable (« var ») sont rétablies comme manquantes.
- Étape 3 : Les valeurs observées de la variable « var » à l'étape 2 sont régressées sur les autres variables du modèle d'imputation, qui peut ou non être constitué de toutes les variables de l'ensemble de données. Autrement dit, « var » est la variable dépendante dans un modèle de régression et toutes les autres variables sont des variables indépendantes dans le modèle de régression.
- Étape 4 : Les valeurs manquantes pour « var » sont ensuite remplacées par des prédictions (imputations) du modèle de régression. Lorsque « var » est ensuite utilisée comme variable indépendante dans les modèles de régression pour d'autres variables, les valeurs observées et ces valeurs imputées seront utilisées.
- Étape 5 : Les étapes 2 à 4 sont ensuite répétées pour chaque variable pour laquelle des données manquent. Le passage par chacune des variables constitue une itération ou « cycle ». A la fin d'un cycle, toutes les valeurs manquantes ont été

remplacées par des prédictions issues de régressions qui reflètent les relations observées dans les données.

- Étape 6 : Les étapes 2 à 4 sont répétées pendant un certain nombre de cycles, les imputations étant mises à jour à chaque cycle.

MICE part du principe que, compte tenu des variables utilisées dans la procédure de substitution, les données manquantes sont des données au hasard (MAR ou *Missing At Random*), ce qui signifie que la probabilité qu'une valeur soit manquante dépend uniquement des valeurs observées et non des valeurs non observées.

Ainsi, on peut prédire que l'algorithme MICE peut ne pas être suffisamment adapté à notre cas. Néanmoins, on a considéré que notre base de données est assez large pour pouvoir approximer les valeurs manquantes à celles de sa même colonne, selon PID, et trouver une valeur potentiellement proche à la valeur réelle.

- Résultats MICE :

Bien que la méthode fonctionne bien, elle prend beaucoup de temps pour traiter toute la base de données dans nos ordinateurs. C'est pourquoi on n'a pas pu obtenir un résultat final complet de la méthode. On a constaté ce fait en essayant la méthode avec les 10 premières PID, d'où on a obtenu un temp de 1482 secondes pour les traiter. Ce résultat veut dire que pour traiter toute la base de données il faudrait plus de 37 heures, ce qui dépasse les limites de nos ordinateurs.

5.2.2.6 Eliminer les observations avec des valeurs NaN

Cette méthode, la plus simple, permet d'éliminer toutes les observations qui ont des valeurs vides. Cependant, cela réduit d'une façon importante la quantité des données (presque le 40%), d'où, cette méthode ne va pas être considérée. Les images suivantes montrent le résultat avant et après l'élimination des valeurs NaN.
Avant :

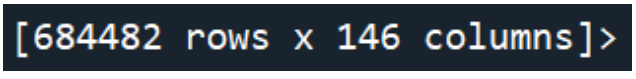


Figure 17 : Nombre de colonnes et rangées avant filtrage

Après :

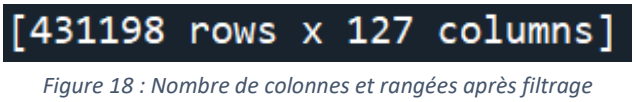


Figure 18 : Nombre de colonnes et rangées après filtrage

5.3 Feature Engineering

Pour améliorer la prédiction des algorithmes on a implémenté des nouvelles variables, qui ont pour but nous aider dans la prédiction :

a) **feat_engi_abs_ret_sum** : Additionne tous les rendements absolus d'une action donnée pour un jour donné. Cela permet d'introduire une mesure de la volatilité du prix.

Pour ce faire, il prend comme point de départ les données saisies (*data_input*), puis les positions initiale et finale de la série de colonnes "abs_retXX".

Enfin, il effectue la somme de chaque rendement absolu entre cet intervalle de colonnes pour une action donnée et un jour donné.

b) **feat_engi_abs_ret_sum_ln :** Additionne tous les rendements absolus d'une action donnée pour un jour donné et calcule le logarithme népérien. Cette variable n'a pas été utilisée.

c) **feat_engi_basic_stat_ret** : On ajoute trois colonnes nommées :

- « max_ret » : Représente pour chaque jour et stock la valeur maximale du retour absolue.
- « std_ret » : Représente pour chaque jour et stock la valeur de la déviation standard.
- « median_ret » : Représente pour chaque jour et stock la valeur médiane du retour absolue.

d) **feat_engi_percentile_stat_vol** : On additionne à la fin de la matrice deux colonnes nommées « 25%percentile_vol » et « 75%percentile_vol ». Dans ces deux colonnes, les valeurs qui appartient aux percentiles 25% et 75% sont représentés. Cela nous permet de filtrer quelques valeurs extrêmes.

e) **feat_engi_sum_ret_per_vol** : On approxime le prix moyen pondéré en fonction du volume VWAC d'une manière différente.

Premièrement on prend les positions des colonnes initiale et finale pour le retour absolue et volume relatif. Puis, le code rajoute à la fin une colonne nommée « sum_ret_per_vol » qui approxime le prix moyenne pondéré en fonction du volume VWAC en multipliant les deux valeurs : retour absolue et volume relatif.

f) **feat_engi_last_minutes_volume** : On mesure la volatilité du volume en fin de journée comme signal de couverture du risque.

Le code prend d'abord les cinq dernières colonnes pour le volume relatif et ensuite crée une colonne à la fin de la matrice nommée « last_minutes_volume » qui montre le résultat de la somme de ces 5 colonnes selon le jour et le stock. Cela représente la volatilité finale du stock et ont une certaine mesure c'est un indicateur du potentiel l'intérêt des acteurs lors de l'enchère.

g) **feat_engi_last_minutes_return** : Mesure la volatilité du prix en fin de journée comme signal de couverture du risque.

De la même manière que précédemment, le code prend d'abord les cinq dernières colonnes pour le retour absolue et ensuite crée une colonne à la fin de la matri nommée « last_minutes_return » qui montre le résultat de la somme de ces 5 colonnes selon le jour et le stock. Un nombre élevé de cette variable signifie un grand intérêt en fermer des positions à des prix hors norme.

h) **feat_engi_skewness_last_minutes** : Mesure l'asymétrie de la distribution

Premièrement on prend les positions des colonnes initiale et finale pour le retour absolue et volume relatif et puis on crée deux nouvelles colonnes nommées « skew_ret » et « skew_vol » qui nous permettent de visualiser l'asymétrie des deux intervalles de données.

i) **feat_engi_number_of_NaN_values** : On ajoute le nombre de valeurs NaN existants pour un PID et un jour « Count_NaN ». Cette variable a pour objectif d'identifier le nombre de potentiels occurrences qui ont pu causer le manque de données.

j) **feat_engi_number_of_zero_values** : Montre le nombre de zéros présents pour le volume relatif, pour mesurer la liquidité du stock « Count_zeros ».

5.4 Réduction des dimensions

5.4.1 Transformation d'espace *Principal Component Analysis* (PCA)

L'analyse en composantes principales (ACP) est une technique de transformation linéaire non supervisée qui est largement utilisée dans différents domaines, notamment pour l'extraction de caractéristiques et la réduction de la dimensionnalité [11].

L'ACP nous aide à identifier des relations dans les données, sur la base de la corrélation entre les caractéristiques. En bref, l'ACP vise à trouver les directions de variance maximale dans les données à haute dimension et à les projeter dans un nouveau sous-espace de dimensions égales ou inférieures à celles de l'original [16].

Pourtant, pour une base de données très vaste comme la nôtre, cette méthode peut optimiser le temps de traitement en réduisant les données d'entrée, et permet d'éliminer certaines variables qui réduisent la précision des algorithmes.

On peut résumer cette technique en quelques étapes simples [13] :

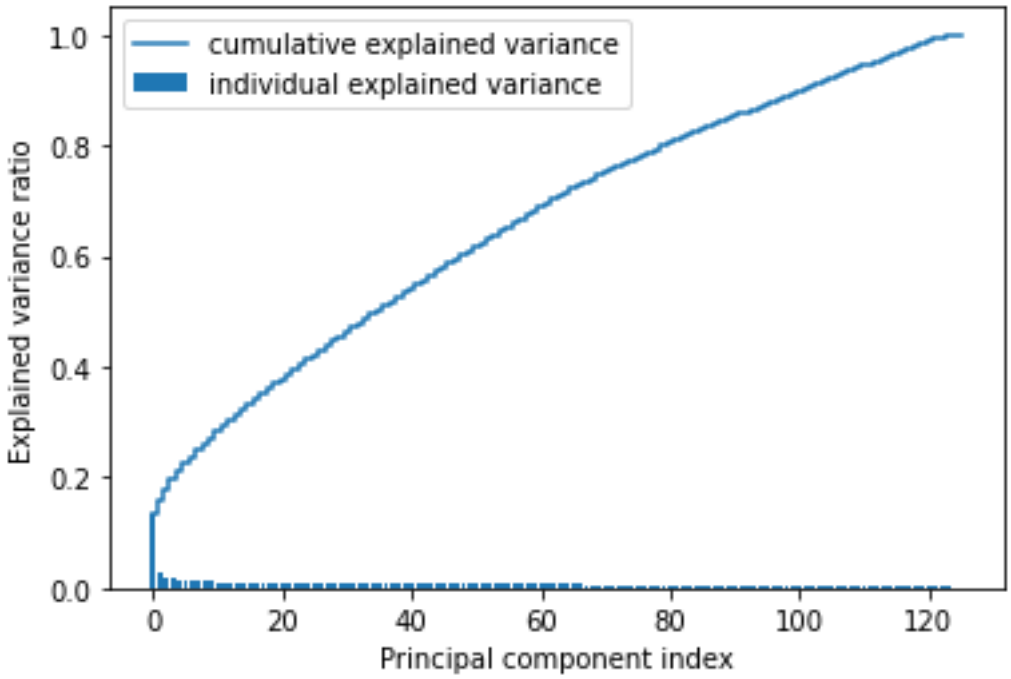
1. Normaliser l'ensemble de données à d dimensions.
2. Construire la matrice de covariance.
3. Décomposer la matrice de covariance en ses vecteurs propres et ses valeurs propres.
4. Trier les valeurs propres par ordre décroissant pour classer les vecteurs propres correspondants.

5. Sélectionner k vecteurs propres qui correspondent aux k plus grandes valeurs propres, où k est la dimensionnalité du nouveau sous-espace de caractéristiques (k ≤ d).
6. Construire une matrice de projection W à partir des k vecteurs propres " supérieurs ".
7. Transformer l'ensemble de données d'entrée X à d dimensions en utilisant la matrice de projection W pour obtenir le nouveau sous-espace de caractéristiques à k dimensions.

- Résultat de la méthode sur notre model.

Le problème détecté avec cette méthode est la non-apparition de composantes principales expliquent un pourcentage élevé de la variance totale, c'est-à-dire que la méthode n'est pas capable de réduire la taille de la matrice initiale car elle ne trouve pas de relation entre les variables.

Nous avons réalisé ce fait en traçant le graphique "variance expliquée" vs "composante principale" à partir duquel nous lisons la "variance expliquée cumulative" et la "variance expliquée individuelle".



c

Figure 19 : "Variance expliquée" vs "Composante principale"

Sur l'image, on aperçoit que la « *cumulative explained varience* » est très linéaire, ce qui nous ne permet pas de détecter les composantes principales. Avec ce résultat, on peut extraire donc que seulement le 17% de la variance totale est représentée avec la première composante principale, résultat non satisfaisant même si on prend plus de composantes principales.

Nous sommes donc obligés à refuser cette méthode linéaire.

5.4.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

La méthode t-SN est une technique de réduction dimensionnelle non linéaire dans laquelle des données inter dimensionnelles élevées (généralement des centaines ou des milliers de variables) sont converties en données de faible dimension (telles que 2 ou 3 variables) tout en conservant la structure significative (relation entre les points de données dans différentes variables) des données originales de haute dimension [16].

Cette implantation suit l'idée de réduire le nombre de variables dans notre base de données de façon à obtenir un résultat plus précis et optimal. Après le rejet de la méthode PCA, laquelle était linéaire, on a décidé d'aborder le problème avec une méthode non-linéaire telle que t-SNE. Parmi toute la liste de méthodes non-linéaires, t-SNE est la plus utilisée et recommandé pour des bases de données comme la nôtre, ainsi que la plus rapide.

Malheureusement, on a eu encore un problème de computation, et on n'a pas pu implémenter le code. On a traité d'optimiser les différents segments de l'algorithme et les tester indépendamment mais c'était impossible d'obtenir le résultat attendu.

Pourtant, on a refusé l'idée de réduire les dimensions de notre *Database* à cause de ne pas réussir à obtenir des bons résultats avec le modèle linéaire, et pour des raisons de computation avec le modèle non-linéaire.

5.5 Normalisation des données

Dans le cas des algorithmes d'arbres de décision la normalisation des données n'est pas nécessaire. En revanche, pour d'autres algorithmes, entre autres les réseaux neuronaux, on a besoin d'un ordre de d'échelle commun.

Rappel : c'est important de réaliser la normalisation du train et du test set avec la même « échelle »

6. Analyse et résultats

6.1. Diagnostique

Ce n'est pas parce qu'un algorithme d'apprentissage s'adapte bien à un ensemble d'apprentissage qu'il constitue une bonne hypothèse. Il pourrait être trop ajusté et, par conséquent, nos prédictions sur l'ensemble de test seraient mauvaises. L'erreur de notre hypothèse, mesurée sur l'ensemble de données avec lequel nous avons formé les paramètres, sera inférieure à l'erreur sur tout autre ensemble de données. Ce phénomène est appelé sur-apprentissage.

Afin d'évaluer les modèles et augmenter la précision des estimations en limitant le sur-apprentissage, on peut tester chaque modèle sur des différents données.

Voici une façon de répartir notre jeu de données en trois ensembles :

- Ensemble d'entraînement : 60%
- Ensemble de validation croisée : 20%
- Ensemble de test : 20%

Nous pouvons maintenant calculer trois valeurs d'erreur distinctes pour les trois ensembles différents.

6.1.1. Cross Validation

La validation croisée (*Cross Validation*) est une méthode statistique utilisée pour estimer la compétence des modèles d'apprentissage automatique [7].

Elle est couramment utilisée dans l'apprentissage automatique appliqué pour comparer et sélectionner un modèle pour un problème de modélisation prédictive donné, car elle est facile à comprendre, facile à mettre en œuvre et donne des estimations de compétence qui ont généralement un biais plus faible que les autres méthodes.

La procédure générale est la suivante [8] :

1. Mélanger l'ensemble des données de façon aléatoire.
2. Diviser l'ensemble de données en k groupes
3. Pour chaque groupe unique :
    a) Prendre le groupe comme un ensemble de données d'attente ou de test.
    b) Prendre les groupes restants comme un ensemble de données d'entraînement
    c) Ajuster un modèle sur l'ensemble d'apprentissage et évaluez-le sur l'ensemble de test.
    d) Conserver le score d'évaluation et éliminez le modèle.

4. Résumer la compétence du modèle en utilisant l'échantillon des scores d'évaluation du modèle.

La technique de validation croisée consiste donc à diviser la base de données d'entrainement en K-sous parties. La méthode K-fold cross validation, utilise K-1 sous parties pour entrainer le modèle et la restant pour l'évaluation. Ce processus est réalisé K fois, chaque fois avec une sous partie différente et le résultat finale s'obtient à travers une moyenne des K modèles [6].

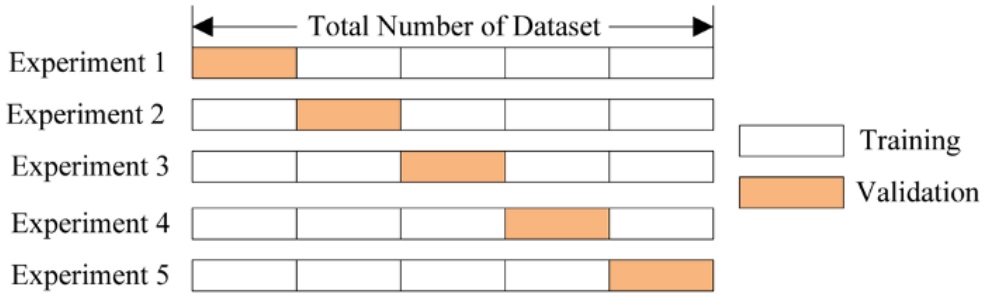


Figure 20 : Méthode K-fold cross validation

C'est important de noter que chaque observation de l'échantillon de données est affectée à un groupe individuel et reste dans ce groupe pendant toute la durée de la procédure. Cela signifie que chaque échantillon a la possibilité d'être utilisé 1 fois dans l'ensemble de maintien et utilisé pour entraîner le modèle k-1 fois.

- Avantages de la validation croisée :

A. Réduire le sur-apprentissage : Dans la validation croisée, nous divisons l'ensemble de données en plusieurs plis et entraînons l'algorithme sur différents plis. Cela empêche notre modèle de s'adapter excessivement à l'ensemble de données d'entraînement. Ainsi, de cette façon, le modèle atteint les capacités de généralisation qui sont un bon signe d'un algorithme robuste.

Note : Les chances de sur-adaptation sont moindres si l'ensemble de données est grand. La validation croisée peut donc ne pas être nécessaire dans les cas où nous disposons de suffisamment de données.

- Inconvénients de la validation croisée

A. Augmentation du temps d'entrainement : La validation croisée augmente drastiquement le temps d'entrainement. Auparavant, nous devions former notre modèle sur un seul ensemble d'entrainement, mais avec la validation croisée, nous devons former notre modèle sur plusieurs ensembles de formation.

   Par exemple, si on opte pour la validation croisée à 5 niveaux, on doit effectuer 5 cycles d'entrainement, chacun sur les 4/5 des données disponibles. Et ceci pour un seul choix d'hyperparamètres.
B. Nécessite des calculs coûteux : La validation croisée est très coûteuse en termes de puissance de calcul requise.

6.1.2. Random Search Cross Validation

La plupart des modèles implémentes dans les librairies (Sciket-learn, Tensorflow, Keras etc…) ont des hyperparamètres par défaut. En Machine Learning, un hyperparamètre est un paramètre dont la valeur est utilisée pour contrôler le processus d'apprentissage. En revanche, les valeurs des autres paramètres (ex : poids des nœuds en réseaux neuronaux) sont dérivées par l'entrainement.

Les hyperparamètres peuvent être classés en tant qu'hyperparamètres de modèle, qui ne peuvent pas être déduits lors de l'adaptation de la machine à l'ensemble d'apprentissage car ils font référence à la tâche de sélection du modèle, ou en tant qu'hyperparamètres d'algorithme, qui n'ont en principe aucune influence sur la performance du modèle mais affectent la vitesse et la qualité du processus d'apprentissage. Un exemple d'hyperparamètre de modèle est la topologie et la taille d'un réseau neuronal [9].

Quel que soit le modèle utilisé pour atteindre l'objectif, les valeurs par défauts des hyperparamètres ne sont pas adaptés (ou au moins les plus performants) à la résolution. La recherche des valeurs plus optimales au modèles qui aboutissent en une bonne capacité de généralisation et précision reste une des parties les plus compliqués à résoudre. Cette démarche pour trouver le *trade-off* entre sous-apprentissage et sur-apprentissage est souvent plutôt pratique que théorique.

Etant donné que, au début de la modélisation on a une vague idée de quelle plage de valeurs pour les hyperparamètres peut être effective. Créer une matrice d'avec les différents rangs de valeurs et évaluer postérieurement les différents combinaisons aléatoires (Important : pas toutes les combinaisons sont évaluées), est une approche pour réaliser un premier « filtrage ». Le module « sklearn.model_selection.RandomizedSearchCV » de la librarie scikit learn permet d'implémenter rapidement cette approche.

6.1.3. Grid Search Cross Validation

L'algorithme de recherche par grille (*Grid Search*) peut être très lent, en raison du nombre potentiellement énorme de combinaisons à tester. En outre, la validation croisée augmente encore le temps d'exécution et la complexité. Pourtant, une fois le rang de valeurs a été diminué grâce à « *Random search CV* », on peut réaliser une recherche beaucoup plus précise de toutes les combinaisons des hyperparamètres.

Nous le faisons avec *GridSearchCV*, une méthode qui, au lieu d'échantillonner au hasard à partir d'une distribution, évalue toutes les combinaisons que nous définissons. Pour utiliser *Grid Search*, nous créons une autre grille basée sur les meilleures valeurs fournies par la recherche aléatoire. Ainsi, à la fin, nous pouvons sélectionner les meilleurs paramètres parmi les hyperparamètres répertoriés.

Une fois les valeurs sont retrouvées, il est possible d'itérer pour trouver une meilleure performance.

Nous avons réussi à bien implémenter l'algorithme dans notre modèle, mais nous n'avons pas obtenu le résultat anticipé en raison de la puissance de calcul de nos ordinateurs qui n'étaient pas en mesure de gérer un ensemble de données aussi important. Nous avons

essayé d'effectuer l'opération à travers de serveurs en utilisant Google Collab, un outil informatique gratuit qui nous permet de traiter des fichiers plus volumineux, mais le résultat a été le même.

7. Modèles de machine learning

7.1 Modèle lineaire

La régression linéaire multiple (RLM), également appelée simplement régression multiple, est une technique statistique qui utilise plusieurs variables explicatives pour prédire le résultat d'une variable de réponse. L'objectif de la régression linéaire multiple est de modéliser la relation linéaire entre les variables explicatives (indépendantes) et les variables de réponse (dépendantes).

```
Train_sample Mean-Squared-error = 0.8254965975562495
Train_sample Mean-Absolute-%-error = 2.2105630668183247 %
Validation_sample Mean-Squared-error = 0.8283680126987758
Validation_sample Mean-Absolute-%-error = 2.1996859642919993 %
Test_sample Mean-Squared-error = 0.8260261155976408
Test_sample Mean-Absolute-%-error = 9.747952767302055 %
```

Figure 21 : Statistiques du modèle linéaire

7.2 Modèles d'arbres de decisión

7.2.1 Arbre simple

Cette méthode permet de créer un arbre de décision pour résoudre le problème de régression. Il existe des nombreux algorithmes de ce type pour faire l'estimation.

Les arbres de décision sont un type de *Machine Learning* supervisé (on précise l'entrée et la sortie correspondante dans les données de d'entrainement) où les données sont continuellement divisées en fonction d'un certain paramètre. L'arbre peut être expliqué par deux entités, à savoir les nœuds de décision et les feuilles. Les feuilles sont les décisions ou les résultats finaux et les nœuds de décision sont les endroits où les données sont divisées.

7.2.2 Random Forest

La forêt aléatoire, comme son nom l'indique, se compose d'un grand nombre d'arbres décisionnels individuels qui fonctionnent comme un ensemble. Chaque arbre individuel de la forêt aléatoire émet une prédiction de classe et la classe ayant reçu le plus de votes devient la prédiction de notre modèle.

En termes de science des données, la raison pour laquelle le modèle de forêt aléatoire fonctionne si bien est la suivante [10] :

- Un grand nombre de modèles (arbres) relativement non corrélés fonctionnant comme un comité sera plus performant que n'importe lequel des modèles individuels qui le composent.
- La faible corrélation entre les modèles est la clé.

Effectivement, les arbres se protègent mutuellement de leurs erreurs individuelles (tant qu'ils ne se trompent pas tous constamment dans la même direction).

```
Model Performance
Squewed error:  77.39208773674503
Mean Absolute Error:  0.6967561918475373
Average Error: 0.6968.
Accuracy (MAPE)= 0.36%.
MSE: 0.8146.
Time:  23554.039994532
```

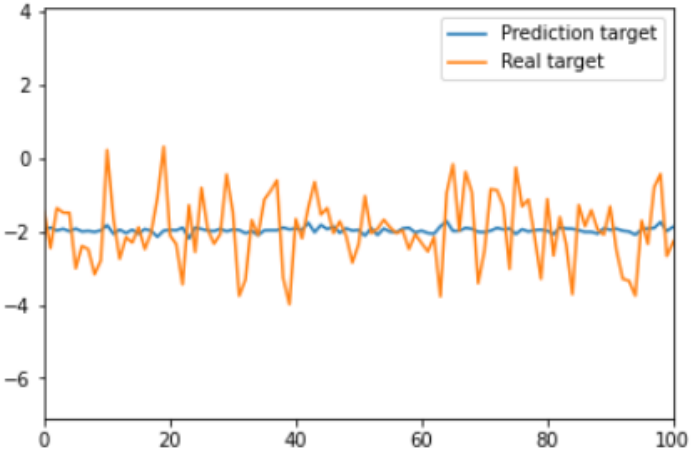


Figure 22 : Statistiques Random forest

7.2.3 ADA Boost

À différence de l'algorithme *Random Forest, ADA Boost* utilise des arbres avec un unique couche de profondeur et 2 feuilles. En effet, chaque arbre ne peut qu'avoir une seule variable. L'objectif de cet algorithme est de créer des arbres sous-entrainés et les combiner pour obtenir un résultat plus performant. Pour ne pas dépendre du facteur aléatoire du *Random Forest*, on utilise des algorithmes de *boosting* (dont *ADA Boost* fait partir). Le *Boosting*, consiste en donnée des poids différents aux données mal classifiés entre chaque arbre nouveau au lieu de créer aléatoirement les arbres. De ce fait, chaque arbre nouveau va être créer pour compenser la « faiblesse des arbres précédents ».

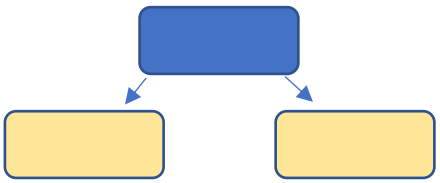


Figure 23 : Exemple d'un arbre à une couche de profondeur

ADA boost vs *Random Forest* :

- Les arbres sont créés séquentiellement pour minimiser l'erreur des arbres précédents (pas aléatoirement)
- Profondeur 1 avec 2 feuilles
- Chaque arbre à un poids différent lors de la prédiction (système de vote) selon son importance.
- Les coûts d'opérations

```
# Train the model
regr = AdaBoostRegressor(random_state=123, n_estimators=25, learning_rate=0.5)
# base_estimator = The base estimator from which the boosted ensemble is built (by
```

Figure 24: Hyperparamètres modèle ADA BOOST

```
---------------Evaluate test_in set: ----------------
Model Performance
Squewed error:  -557.2794743047624
Mean Absolute Error:  0.7000652680427725
Average Error: 0.7001.
Accuracy (MAPE)= 0.36%.
MSE: 0.8253.
---------------Evaluate val_in set: ----------------
Model Performance
Squewed error:  91.92693220161662
Mean Absolute Error:  0.7018779271378804
Average Error: 0.7019.
Accuracy (MAPE)= 0.36%.
MSE: 0.8276.
Time:  728.2579194819991
```

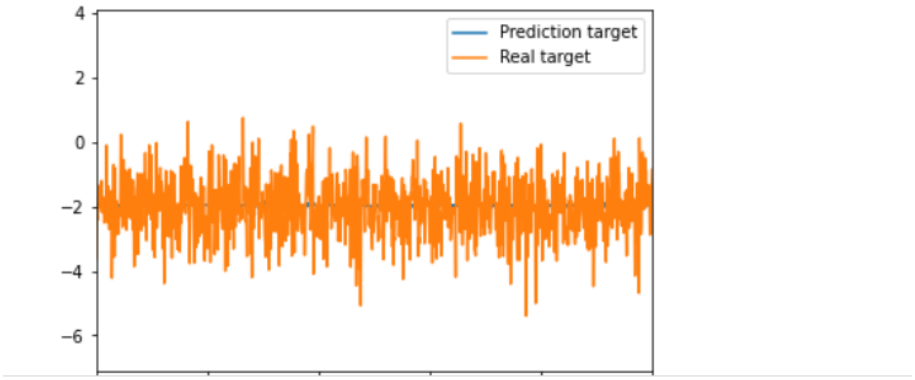


Figure 25 Résultats modèle ADA BOOST

7.2.4 Gradient Boosting Trees

Dans cet algorithme qui utilise le *Boosting*, la profondeur n'est pas limitée à un (cependant, le nombre de feuilles reste faible). Contrairement aux algorithmes antérieurs, celui la commence par créer une feuille qui a pour valeur la moyenne de la valeur à prédire sur tout le jeu d'entrainement. A chaque itération on n'essaye pas de prédire plus la variable objective, mais le résidu de la variable et la valeur moyenne à prédire calculé auparavant de chaque nouvel arbre. Ce modèle tend rapidement vers le sur-apprentissage.

```
#----------------------ADA BOOST Model-------------------------------

# Train the model
regr = GradientBoostingRegressor(random_state=123, n_estimators=50,\
                                 learning_rate=0.5, subsample = 1.0, verbose =1,\
                                 max_depth = 15)
regr.fit(train_in_set, train_out_set)
```

Figure 26: Hyperparmètres modèle GRADIENT BOOSTING

```
---------------Evaluate test_in set: ----------------
Model Performance
Squewed error:  -407.3015889343502
Mean Absolute Error:  0.7804258586065383
Average Error: 0.7804.
Accuracy (MAPE)= 0.56%.
MSE: 1.0126.
---------------Evaluate val_in set: ----------------
Model Performance
Squewed error:  277.1331761385296
Mean Absolute Error:  0.7818874649536626
Average Error: 0.7819.
Accuracy (MAPE)= 0.82%.
MSE: 1.0172.
```

Figure 27 : Résultats modèle GRADIENT BOOSTING

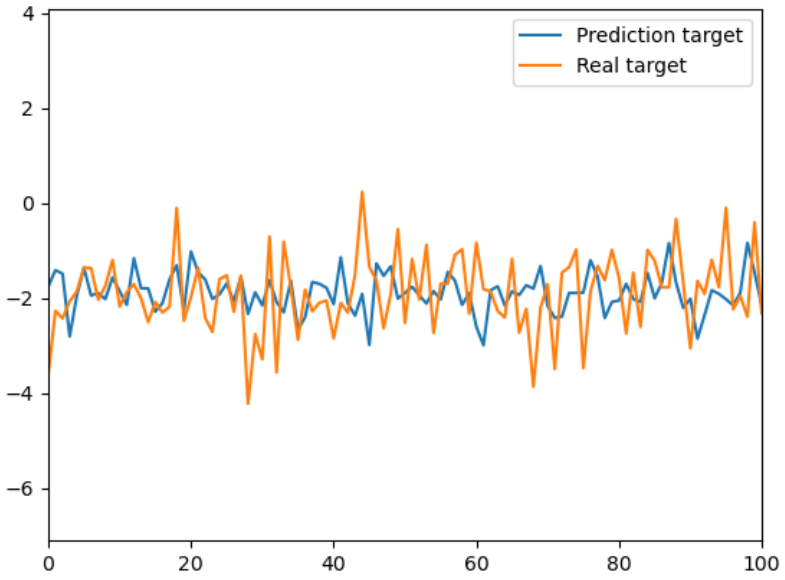


Figure 28 : Graphique des résultats

7.3 Réseaux neuronaux

Un réseau neuronal est une série d'algorithmes qui essaye de reconnaître les relations sous-jacentes dans un ensemble de données par un processus qui imite le fonctionnement du cerveau humain. En ce sens, les réseaux neuronaux font référence à des systèmes de neurones, bien qu'ils soient de nature organique ou de nature artificielle.

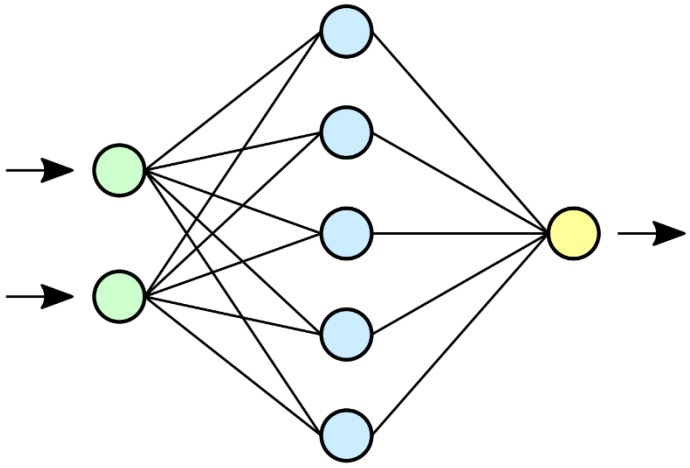


Figure 29: Modèle Réseaux neuronaux

```
def baseline_model():
  # create model
  model = Sequential()
  model.add(Dense(45, input_dim=45, kernel_initializer='normal', activation='relu'))
  model.add(Dense(1, kernel_initializer='normal'))
  # Compile model
  model.compile(loss='mean_squared_error', optimizer='adam')
  return model

# evaluate model
estimator = KerasRegressor(build_fn=baseline_model, epochs=2, batch_size=5, verbose=0)
kfold = KFold(n_splits=5)
results = cross_val_score(estimator, train_in_set, train_out_set, cv=kfold)
#The result reports the mean squared error including the average and standard deviation (average variance)
print("Baseline: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

Figure 30: Hyperparamètres RESEAU NEURONAL

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:82: DeprecationWarning: KerasRegressor is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras)
Baseline: -0.83 (0.04) MSE
---------------Evaluate train_in set: ----------------
Time:  1963.9808081890005
```

Figure 31: Résultats avec validation croisée

Finalement, voici les résultats obtenus des différents modèles évalués à travers de la plateforme *Data Challenge* :

Classement	Méthode	Paramètres	Score
1	Gradient Boosting	n_estimators = 50 learning_rate = 0,5 max_depth = 15	0,63
2	Régression linéaire	MSE	0,65
3	Arbre de décision simple	max_leaf_node= 50 max_depth = 25 max_features="auto" criterion = "mse"	0,71
4	Ada boost	n_estimators = 50 learning_rate = 0,5	0,94
5	Réseaux neuronaux	Couches = 3 Epochs = 100 Batch_size = 5 Optimizer = ‘adam’ Loss = MSE	0,95
6	Random forest	n_estimators = 100 learning_rate = 0,1 max_depth = 15	0,99

Tableau 1 : Résultats et soumissions concours

8. Conclusion

En conclusion, après avoir implémenté des différents algorithmes de *Machine Learning*, l'algorithme de *Gradient Boosting* nous donne la meilleure performance. Cependant, l'écart entre la performance de ce dernier et une simple régression linéaire reste assez faible. En outre, la complexité de la méthode, l'effort, le temps et les ressources computationnelles employés dans l'algorithme ne justifient pas l'écart de performance. En effet, ces méthodes de prédiction même si pointeurs et très en vogue, ne sont pas toujours les plus performantes et adaptés.

Le processus itératif et exploratoire des hyperparamètres a été fortement limité due à la taille des bases de données et la limitation de notre puissance computationnelle. Malgré cela, on a pu optimiser quelques hyperparamètres et évaluer l'influence positive ou négative de ces dernières.

Par rapport au processus en amont à la modélisation et optimisation des algorithmes de *Machine Learning* des nombreuses actions ont été menées, entre elles, le nettoyage, la validation des données, le remplissage des valeurs manquantes, la réduction d'espaces, la création des nouvelles variables. Cette étape a requis de plus du 70% du temps du travail puisque aucun modèle a d'utilité si les données employées sont imprécises et inutiles.

Pour la partie remplissage des valeurs manquantes, plusieurs méthodes ont été proposées. Cependant, les méthodes simples ont été priorisée, en sachant que la méthode la plus rapide n'est pas toujours la plus efficace.

Finalement, des potentiels axes d'amélioration restent à explorer :

- Utilisation de services de computations pour l'optimisation matricielle des hyperparamètres
- Aborder le prétraitement des données avec des différentes méthodes (par exemple : remplissage des valeurs nuls)
- Consulter un expert dans le domaine pour évaluer la pertinence des certaines variables et créer des nouvelles
- Combiner l'utilisation de plusieurs modèles de prédiction dans une même prédiction (exemple : en attribuant des différents poids dans la prédiction selon la performance pour éviter le sur-apprentissage)

9. Bibliographie

[1] https://www.bloomberg.com/professional/blog/intelligently-allocate-benchmark-closing-price/

[2] https://www.fixglobal.com/amp/the-12-rule-for-asias-closing-auctions/

[3] https://www.frontiersin.org/articles/10.3389/frai.2019.00021/full (deep learning models)

[4] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/

[5] https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=Cross%2Dvalidation%20is%20a%20resampling,k%2Dfold%20cross%2Dvalidation

[6] https://machinelearningmastery.com/training-validation-test-split-and-cross-validation-done-right/

[7] https://machinelearningmastery.com/k-fold-cross-validation/

[8] https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

[9] https://towardsdatascience.com/random-forest-in-python-24d0893d51c0

[10] https://machinelearningmastery.com/dimensionality-reduction-for-machine-learning/#:~:text=When%20dealing%20with%20high%20dimensional,This%20is%20called%20dimensionality%20reduction.

[11] https://towardsdatascience.com/getting-data-ready-for-modelling-feature-engineering-feature-selection-dimension-reduction-39dfa267b95a

[12] https://towardsdatascience.com/think-twice-before-you-use-principal-component-analysis-in-supervised-learning-tasks-70fbb68ebd0c

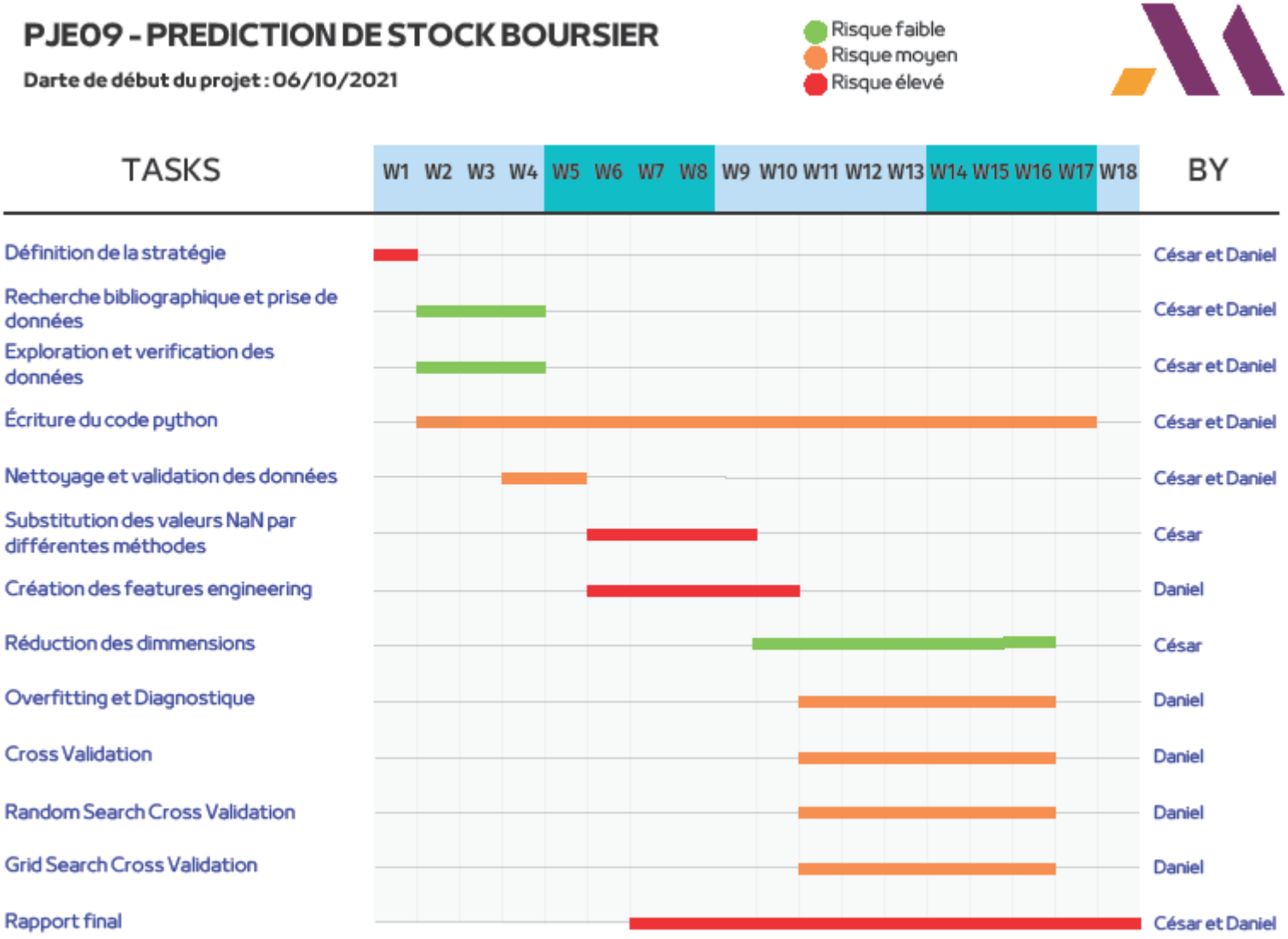
[13] https://towardsdatascience.com/11-dimensionality-reduction-techniques-you-should-know-in-2021-dcb9500d388b

[14] https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db

[15] https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad

[16] https://www.reneshbedre.com/blog/tsne.html

Annexe I


PJE09 - PREDICTION DE STOCK BOURSIER
Darte de début du projet : 06/10/2021
Risque faible
Risque moyen
Risque élevé
TASKS
W1 W2 W3 W4 W5 W6 W7 W8 W9 W10 W11 W12 W13 W14 W15 W16 W17 W18
BY
Définition de la stratégie
César et Daniel
Recherche bibliographique et prise de données
César et Daniel
Exploration et verification des données
César et Daniel
Écriture du code python
César et Daniel
Nettoyage et validation des données
César et Daniel
Substitution des valeurs NaN par différentes méthodes
César
Création des features engineering
Daniel
Réduction des dimmensions
César
Overfitting et Diagnostique
Daniel
Cross Validation
Daniel
Random Search Cross Validation
Daniel
Grid Search Cross Validation
Daniel
Rapport final
César et Daniel