Rapport de stage de M1

Apport du Machine Learning dans un fonds de gestion quantitative

Paul-Antoine Fruchtenreich,

encadré par Laurent Jaillet.

Table des matières

1 Contexte du stage

Lors de cette année, j'ai très rapidement été attiré par les mathématiques appliquées et plus particulièrement les mathématiques des données. C'est aussi dans ce domaine que je me sentais le plus à l'aise et naturellement, ma recherche de stage s'est tournée vers celui-ci. Après plusieurs entretiens dans différentes entreprises, j'ai été mis en contact avec Laurent Jaillet. Laurent m'a présenté son travail de gestionnaire de portefeuille - chez Syquant Capital - dans un fonds de gestion basé sur des stratégies quantitatives sur actions et m'a expliqué qu'il pensait à l'apprentissage automatique pour améliorer sa stratégie. J'ai tout de suite senti que ce sujet m'intéresserait et que je pourrais leur apporter mes connaissances dessus. Nous avons rapidement convenu des modalités du stage qui pu alors débuter début février. Celui-ci se déroulait à Paris, proche de la place de l'Etoile. Je fus installé à un bureau dans un large open space qui regroupait toute l'entreprise, environ une quarantaine de personnes. Les journées commençaient à 9h et finissaient à 19h. Je voyais Laurent tous les soirs pour présenter ce sur quoi j'avais avancé durant la journée. Naturellement, je n'expliciterai pas toute la stratégie du fonds dans ce rapport et je resterai parfois superficiel pour ne pas dévoiler d'informations confidentielles.

2 Principe du fonds Helium Alpha

Je parlerai souvent de positions Long et Short dans la suite. Une position Long correspond au fait d'acheter un titre et de le revendre plus tard avec une plus-value si le cours est en hausse. A l'inverse, une position Short traduit une vente d'un actif que l'on ne détient pas, il faut donc l'acheter ensuite : on mise sur la baisse du cours.

J'ai réalisé mon stage au sein de la société de gestion Syquant Capital qui regroupe 7 fonds d'investissement différents, ayant près de 3,5 milliards d'euros d'actifs sous gestion . Helium Alpha est un fonds avec 18 millions d'euros d'encours sous gestion. Il a été créé en septembre 2017 par Olivier Leymarie et Laurent Jaillet et repose sur une stratégie Long/Short qui combine des techniques de gestion quantitative et fondamentale. Le but est de collecter des données de différents types sur un éventail de plus de mille actifs et d'en tirer de l'information pour en déduire une stratégie d'investissement. Les questions qui se posent naturellement sont donc : comment détecter l'information, comment l'agréger, comment en déduire la stratégie d'investissment la plus performante pour un risque contrôlé ? Actuellement, le fonds Helium Alpha se base sur des scores, c'est-à-dire des fonctionnelles dont les variables appartiennent à l'ensemble des informations disponibles sur les titres, et qui doivent - entre autres - traduire des dynamiques de marchés et des caractéristiques spécifiques du titre. Je serai de manière volontaire flou à propos de la stratégie : le fonds utilise un modèle multi factoriel pour projeter les scores des actifs othogonalement aux facteurs de risque. On peut alors calculer les corrélations des titres entre eux. Grâce à cela, on trouve le portefeuille unitaire optimal pour un niveau de risque fixé à l'aide de la théorie moderne du portefeuille d'Harry Markowitz : c'est un problème d'optimisation quadratique classique entre moyenne et variance. C'est son application à la finance quantitative qui la rend originale. On cherche le meilleur compromis entre rendement et risque en imposant des contraintes linéaires. Cette optimisation est résoluble à l'aide de son Lagrangien et la composition du portefeuille s'en déduit facilement avec la matrice de

covariance des titres et le vecteur d'espérance des rendements.

On note $\mathbf{x}$ est le vecteur de poids du portefeuille, c'est-à-dire que la $i-ème$ coordonnée de $\mathbf{x}$ correspond au pourcentage du budget investi dans l'actif i. $\mathbf{\Omega}$ sera la matrice de covariance des titres et $\boldsymbol{\mu}$ le vecteur des espérances des rendements des titres. Sans contrainte, on cherche à minimiser la fonction suivante :

$$\mathcal{O} = \tfrac{1}{2}\ \mathbf{x}^T\mathbf{\Omega}\mathbf{x} - \tau\boldsymbol{\mu}^T\mathbf{x}$$

Le premier terme correspond à la variance du portefeuille, donc au risque pris, et le second au rendement de celui-ci avec un paramètre τ qui traduit l'aversion au risque pris par l'investisseur. Ainsi, pour chaque valeur de τ entre 0 et $+\infty$, on obtient facilement la composition du portefeuille $\mathbf{x}$ et on en déduit son rendement espéré et sa variance.

On rajoute ensuite une contrainte linéaire, par exemple sur le budget. Cette contrainte s'écrit $A\mathbf{x} = b$, où A et b sont une matrice et un vecteur constants. On forme alors le Lagrangien

$$\mathcal{L}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{\Omega}\mathbf{x} - \tau\boldsymbol{\mu}^T\mathbf{x} + \boldsymbol{\lambda}^T(A\mathbf{x} - b)$$

qu'on différentie par rapport à $\mathbf{x}$ et à $\boldsymbol{\lambda}$ et on impose le résultat à 0 en suivant les conditions de Karush-Kuhn-Tucker :

$$\begin{bmatrix} \mathbf{\Omega} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} + \begin{bmatrix} \boldsymbol{\mu} \\ 0 \end{bmatrix} \tau$$

Alors, comme $\mathbf{\Omega}$ est définie positive, la solution est

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1\tau$$

avec les expressions de $\mathbf{x}_0$ et $\mathbf{x}_1$ suivantes :

$$\mathbf{x}_0 = \mathbf{\Omega}^{-1}A^T\left[A\mathbf{\Omega}^{-1}A^T\right]^{-1}b,$$
$$\mathbf{x}_1 = \mathbf{\Omega}^{-1}\left[\mathbf{\Omega} - A^T\left[A\mathbf{\Omega}^{-1}A^T\right]^{-1}A\right]\mathbf{\Omega}^{-1}\boldsymbol{\mu}.$$

On obtient de cette manière la composition du portefeuille optimal en imposant un certain niveau de risque grâce à τ.

3 Déroulé du stage

Mon travail effectué lors de ce stage peut se décomposer en plusieurs parties : une première - d'environ 2 semaines - comme étant une revue de la littérature sur la finance quantitative et les différentes techniques d'apprentissage automatique. Dans un deuxième temps, j'avais pour objectif d'établir un classement des différents scores utilisés dans la stratégie du fonds Helium Alpha. Pour cela, j'ai construit plusieurs métriques grâce au bootstrap, une méthode utilisée en statistiques. J'ai ensuite rapidement travaillé sur le coefficient d'information utilisé pour pondérer les scores dans la construction du portefeuille final. Enfin, la dernière et plus longue partie se compose d'un projet de machine learning visant à intégrer cette technologie dans la stratégie, soit de manière purement prédictive soit de façon à localiser l'information (SHAP).

Dans le cadre de ce stage, j'ai dû répondre à certaines obligations réglementaires : des formations sur les abus de marché ou délits d'initié. J'ai aussi reçu une formation sur la lutte contre le blanchiment d'argent et le financement terroriste.

4 Bootstrap

Dans cette première partie, on souhaite évaluer les scores selon leurs performances passées. On caractérise la performance d'un score selon les rendements obtenus par la stratégie unitaire basée sur ce seul score. Le score donne des valeurs pour chaque titre ; suite à cela, ces valeurs sont centrées et réduites pour pouvoir les comparer. On en déduit ensuite un vecteur de poids sur les titres basé uniquement sur ce score et donc une série temporelle de rendements, qui correspond aux résultats du portefeuille construit à partir du vecteur de poids. Ainsi, pour chaque score, on dispose de 15 ans de rendements et on souhaite en déduire un classement de la qualité des scores. L'idée est de distinguer la chance de l'information contenue dans chaque score. Cela revient à dire que l'espérance de la loi sous-jacente de chaque score est significativement différente de 0. Or, chaque score nous donne une seule réalisation de sa loi sur les 15 ans. Pour remédier à cela, on applique la méthode statistique du bootstrap afin simuler "différentes réalités alternatives" pour chaque score : ainsi, avec un nombre suffisant d'itérations, l'espérance pourra être approchée avec plus de précision. Pour chaque ré-échantillonage, on calcule la moyenne des rendements : on peut ensuite en déduire la moyenne globale avec un intervalle de confiance.

Plus précisément, le boostrap consiste à estimer une quantité $\theta = T(F)$ où T est une fonctionnelle et F la fonction de répartition de la variable aléatoire dont on tire les observations, qu'on ne connaît pas. Classiquement, on estime θ grâce à $\hat{\theta} = T(\hat{F})$ où $\hat{F}$ est une estimation de la fonction de répartition F. Afin d'estimer F, on utilise la fonction de répartition empirique $\hat{F_n}$ obtenue à partir de l'échantillon initial en plaçant une probabilité de $\frac{1}{n}$ à chaque point. A partir de $\hat{F_n}$, on ré-échantillonne avec remise pour construire un nouvel ensemble de points $(X_1^*, \ldots, X_n^*)$ et on calcule la statistique d'intêret sur celui-ci, dans notre cas la moyenne :

$$\hat{\theta_b} = \frac{(X_1^*, \ldots, X_n^*)}{n}, \text{ où b est l'itération}$$

et on répète cela pour $1 \leq b \leq B$, où B est suffisamment grand (au moins 1000 pour l'obtention

d'intervalles de confiance). On obtient alors un estimateur de notre statistique d'intérêt en calculant la moyenne des nos estimateurs $\hat{\theta_b}$:

$$\hat{\theta} = \frac{1}{B}\sum_{i=1}^{B}\hat{\theta_b}$$

Dans notre cas, $\hat{\theta}$ représente la valeur intrinsèque du score, libérée de l'aspect chance sur une seule réalisation. On veut donc savoir si elle se situe significativement au-dessus de 0 ou non, cela grâce à un intervalle de confiance. Il existe plusieurs méthodes de construction d'intervalles de confiance pour un estimateur bootstrap : celui normal, celui basé sur les percentiles ou encore avec correction du biais et accélération BC_a. Pour le deuxième, l'intuition est simple : on classe les estimateurs $\hat{\theta_b}$ trouvés par ordre croissant et on chosit les bornes en fonction de la précision voulue. Par exemple, à 95%, on obtiendrait :

$$[\theta_{0.025}, \theta_{0.975}]$$

où $\theta_{0.025}$ et $\theta_{0.975}$ représentent les quantiles à 2.5% et 97.5% de la distribution des $\hat{\theta_b}$. L'inconvénient de cette méthode est que l'intervalle n'est seulement bon qu'au premier ordre, ie pour un niveau 2α avec un échantillon de taille n :

$$\mathbb{P}\left(\theta < \theta_{low}\right) \doteq \alpha + \frac{c_{low}}{\sqrt{n}} \quad \text{et} \quad \mathbb{P}\left(\theta > \theta_{up}\right) \doteq \alpha + \frac{c_{up}}{\sqrt{n}}$$

On peut être plus précis, notamment avec l'intervalle BC_a. Cet intervalle est obtenu à l'aide des percentiles mais de manière dynamique. Il faut calculer deux paramètres : l'accélération $\hat{a}$ et la correction de biais $\hat{z_0}$. La correction de biais a une construction naturelle : on compte le nombre d'estimateurs $\hat{\theta_b}$ plus petits que l'estimateur final $\hat{\theta}$ et on utilise ensuite la fonction inverse de la fonction de répartition de la normale :

$$\hat{z_0} = \Phi^{-1}\left(\frac{\#\{\hat{\theta_b} < \hat{\theta}\}}{B}\right)$$

Cela mesure l'écart entre la médiane des $\hat{\theta_b}$ et $\hat{\theta}$, tout cela en unité normale.

L'accélération mesure la vitesse du changement de l'erreur standard de $\hat{\theta}$ par rapport au vrai paramètre θ. L'approximation $\hat{\theta} \sim \mathcal{N}\left(\theta, se^2\right)$ traduit que l'erreur standart de $\hat{\theta}$ est la même quelque soit la valeur de θ, ce qui n'est pas vrai en pratique. Le paramètre $\hat{a}$ sert à corriger cela. Si $X_{(i)}$ est l'échantillon X original privé de la valeur x_i, on note $\hat{\theta}_{(i)} = T(X_{(i)})$, la statistique calculée sur ce sous-échantillon et on définit :

$$\hat{\theta}_{(\cdot)} = \sum_{i=1}^{n}\frac{\hat{\theta}_{(i)}}{n}$$

L'expression de la constante $\hat{a}$ est alors :

$$\hat{a} = \frac{\sum_{i=1}^{n}\left(\hat{\theta}_{(\cdot)} - \hat{\theta}_{(i)}\right)^3}{6\left[\sum_{i=1}^{n}\left(\hat{\theta}_{(\cdot)} - \hat{\theta}_{(i)}\right)^2\right]^{\frac{3}{2}}}$$

Une fois ces deux paramètres obtenus, nous pouvons calculer les nouveaux percentiles voulus pour un intervalle de confiance au niveau 2α :

$$\alpha_1 = \Phi\left(\hat{z_0} + \frac{\hat{z_0} + z^{(\alpha)}}{1 - \hat{a}(\hat{z_0} + z^{(\alpha)})}\right)$$
$$\alpha_2 = \Phi\left(\hat{z_0} + \frac{\hat{z_0} + z^{(1-\alpha)}}{1 - \hat{a}(\hat{z_0} + z^{(1-\alpha)})}\right)$$

où $z^{(\alpha)}$ correspond au $100\alpha - ème$ percentile de la loi normale. L'intervalle de confiance BC_a est alors :

$$\left[\hat{\theta}^{*(\alpha_1)}, \hat{\theta}^{*(\alpha_2)}\right]$$

avec $\hat{\theta}^{*(\alpha)}$ la valeur du $100\alpha - ème$ percentile sur les B estimations bootstrap de θ.

Cet intervalle possède deux avantages par rapport à celui classique des percentiles : il est précis au deuxième ordre

$$\mathbb{P}\left(\theta < \hat{\theta}^{*(\alpha_1)}\right) \doteq \alpha + \frac{c_{\alpha_1}}{n} \quad \text{et} \quad \mathbb{P}\left(\theta > \hat{\theta}^{*(\alpha_2)}\right) \doteq \alpha + \frac{c_{\alpha_2}}{n}$$

et il respecte les transformations. Cela signifie que les bornes du BC_a d'une statistique qui est fonction de θ sont obtenues en prenant les images des bornes du BC_a original par cette même fonction.

A partir de cela, il faut construire une métrique pour classer les scores. J'en ai imaginé plusieurs qui comparent différents aspects des stratégies unitaires. La première consiste à regarder la moyenne des bornes de l'intervalle de confiance trouvé grâce au bootstrap, la seconde à ne considérer que la borne basse, qui contient l'information quant à la différence par rapport à 0 pour chaque score. Ces deux métriques sont calculées sur la période entière de 15 ans. On peut aussi regarder en période glissante pour capturer de l'information localement et agréger ensuite : par exemple, j'ai étudié la moyenne des classements basés sur les rendements par période de 3 ans en fenêtre glissante. J'ai ensuite fait la même chose pour les bornes basses des intervalles de confiance et pour les médianes des rendements. J'ai aussi imaginé des métriques basées sur le drawdown (plus grosse chute du rendement) en rolling et en value at risk, c'est-à-dire en regardant des centiles extrêmes.

Grâce à ces métriques, nous obtenons des classements différents sur les scores. Pour évaluer la pertinence de ces métriques, nous construisons un modèle de stratégies emboîtées : à chaque itération, nous retirons le plus mauvais score encore en lice pour la métrique considérée et nous calculons les indicateurs de performance les plus classiques (rendement, variance et ratio de Sharpe = $\frac{rendement}{variance}$) de la stratégie "poids égaux" sur les scores gardés, c'est-à-dire qu'on investit de manière identique sur les scores conservés. On remarque que pour chaque métrique, la stratégie optimale (celle donnant le plus grand ratio de Sharpe) est obtenue en enlevant entre 2 et 10 scores.

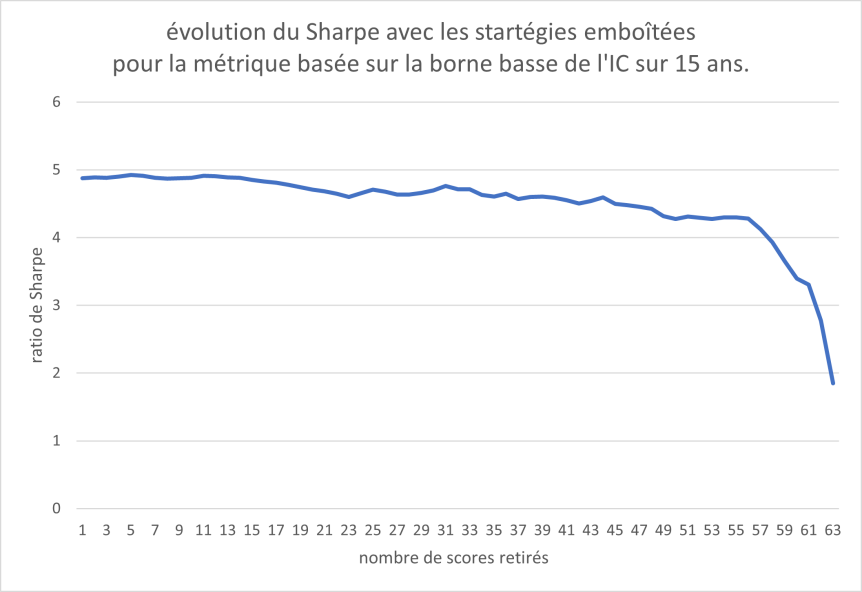

FIGURE 1 – Ratio de Sharpe en fonction du nombre de scores retirés pour la stratégie borne basse de l'IC sur 15 ans. Deux maximums locaux nous intéressent ici : pour 4 et 10 scores enlevés.

A partir de là, j'ai proposé, pour chaque métrique, un certain ensemble de scores à retirer de la stratégie et regardé les résultats des tests faits à partir des outils classiques d'optimisation utilisés par les gérants du fonds. La première chose à remarquer est que les scores retirés ne sont pas toujours exactement les mêmes en fonction de la métrique choisie : en effet, celles-ci capturent différentes informations sur les scores. On note ensuite que chaque test aboutit à une stratégie un peu meilleure que celle réalisée actuellement avec tous les scores étudiés. Bien évidemment, cela s'explique en partie car on retire les scores les moins bons sur toute cette période : le véritable essai doit être conduit à partir de la date de fin de la période de bootstrap en comparant les rendements futurs de la stratégie actuelle avec tous les scores et en ne considérant qu'un sous-ensemble de ceux-ci. Malheureusement, le temps du stage était limité et nous n'avions pas assez de nouvelles données pour confirmer nos résultats, c'est actuellement un de leurs sujets d'étude !

5 Machine Learning

La stratégie actuelle est basée sur un modèle multivarié avec des estimations de facteurs de risque grâce à de la régression multiple. Ce modèle est performant, les récents résultats du fonds en témoignent, mais avec l'évolution des méthodes d'analyse de données, il est légitime de penser qu'une amélioration est possible. En effet, le fonds Helium Alpha étant basé sur une stratégie quantitative et d'analyse de données, le développement de l'apprentissage automatique devrait permettre d'être plus précis dans l'estimation des facteurs de risque et donc par conséquent, de perfectionner les données prises en compte dans la stratégie. Parmi les évolutions récentes de l'analyse de données et donc de voies possibles d'amélioration pour le fonds, nous pouvons citer entre autres les arbres de décision et les forêts aléatoires, les k plus proches voisins ou encore les réseaux de neurones.

Le but était donc ensuite d'introduire de l'apprentissage automatique dans la stratégie afin de mieux capter l'information contenue dans les scores. Les données à notre disposition sont les valeurs des scores pour tous les titres et cela chaque jour sur une durée de 15 ans jusqu'à aujourd'hui et naturellement, les rendements associés aux titres. L'idéal serait de réussire à prédire l'évolution du rendement de chaque titre en ne connaissant que son vecteur de valeurs sur les scores 10 jours avant, qui correspond à la période de trading (les positions sur les titres sont prises tous les 10 jours). Ainsi, la première idée et probablement la plus naturelle est d'utiliser un algorithme de régression pour prédire le rendement cumulé des 10 jours suivants à partir des valeurs des p scores pour le titre i.

$$\sum_{k=1}^{10} rendement_i^{t+k} = f(X_i^t)$$

où i correspond à l'indice du titre, t à la date et le vecteur de features :

$$X_i^t = \left[score(1)_i^t,\ \dots\ , score(p)_i^t\right]$$

Ainsi, à une date donnée, en fonction des prédictions pour chaque titre, nous pourrions en déduire une stratégie d'investissement à partir des rendements espérés. En premier lieu, plutôt que de la régression pour prédire la valeurs des rendements, j'ai décidé d'utiliser de la classification. L'idée derrière est de repartir les titres dans des chapeaux en fonction de leurs rendements. A chaque date, on souhaite prédire dans quel quintile se placera un titre en fonction de ses valeurs de scores. Grâce à cela, les titres classés dans le meilleur quintile seraient placés en Long et ceux dans le moins bon en Short.

Après une revue de la littérature sur l'apprentissage supervisé dans le domaine de la finance quantitative et de manière plus générale, j'ai choisi l'algorithme XGBoost qui semble être le plus performant à l'heure actuelle pour de nombreuses applications. Celui-ci construit de manière séquentielle un ensemble d'arbres de décision afin d'effectuer sa prédiction. Il fait partie de la famille des algorithmes ensembliste utilisant le gradient boosting dont on détaille le principe ci-dessous.

Pour un ensemble de n données et p features, $D = \{(x_i, y_i),\ x_i \in \mathbb{R}^p,\ y_i \in \mathbb{R}\}$, un modèle ensembliste utilise K fonctions additives pour prédire :

$$\hat{y}_i = f(x_i) = \sum_{k=1}^{K} f_k(x_i),\ \text{où}\ f_k \in F$$

et $F = \{f : \mathbb{R}^p \longrightarrow \mathbb{R}^T,\ x \longmapsto f(x) = w_{q(x)}\}$, où T est le nombre de feuilles de l'abre, $q : \mathbb{R}^p \longrightarrow T,\ w \in \mathbb{R}^T$ représente la structure de l'abre et w les poids associés aux feuilles. Ainsi, F est l'espace des arbres de décision.

De manière classique, on voudrait minimiser la fonction objectif suivante :

$$L(f) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

où $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$. l est une fonction de perte convexe et différentiable qui quantifie l'erreur de prédiction et Ω est une fonction de pénalisation pour la complexité du modèle qui vise à éviter le sur-apprentissage. Malheuresement, étant donné que les fonctions sont des paramètres dans ce modèle, cette fonction de perte L ne peut être optimisée de manière classique. A la place, l'idée est d'entraîner ce modèle séquentiellement.

On note $\hat{y}_i^{(t)}$ la prédiction de la donnée i à la $t - ième$ itération. On va ajouter la nouvelle fonction f_k de sorte à minimiser

$$L^{(t)} = \sum_i l(\hat{y}_i^{(t-1)} + f_t(x_i), y_i) + \Omega(f_t)$$

La fonction f_t doit donc servir à prédire le résidu entre la valeur visée y_i et la prédiction à l'itération précédente. Au second ordre, avec $g_i = \partial_{\hat{y}_i^{(t-1)}} l(\hat{y}_i^{(t-1)}, y_i)$ et $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(\hat{y}_i^{(t-1)}, y_i)$, on a d'après J. Friedmann, T. Hastie et R. Tibshirani :

$$L^{(t)} \simeq \sum_i [l(\hat{y}_i^{(t-1)}, y_i) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

En enlevant les termes constants, avec l'expression de Ω et en notant I_j l'ensemble des données restantes à la feuille j, on a l'expression suivante :

$$\begin{aligned}
\tilde{L}^{(t)} &= \sum_i [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\
&= \sum_i [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{w(x_i)}^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2, \text{ car } f_t \in F \\
&= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T, \text{ en sommant sur les feuilles.}
\end{aligned}$$

Pour une structure d'arbre fixée, on calcule simplement le poids optimal w_j^* de la feuille j par

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

et on en déduit la valeur optimale de la fonction de coût :

$$\tilde{L}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda}$$

Cette formule permet de mesurer la qualité d'une structure d'arbre. On ne peut évidemment pas calculer cette quantité pour chaque arbre et prendre ensuite le meilleur. Au lieu de cela, on part d'un arbre à une feuille et on lui ajoute séquentiellement des branches ; il est alors facile de calculer la réduction de coût après le noeud et de choisir la séparation optimale pour cette technique. Si on note I_R et I_L les ensembles de données présentes dans les branches droite et gauche après la séparation du noeud, on pose $I = I_R \cup I_L$ et on a alors :

$$\Delta_{noeud} = \frac{1}{2}\left[\frac{\left(\sum_{i_L} g_i\right)^2}{\sum_{i_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

Dans le cadre de l'algorithme XGBoost, on utilise comme premier apprenant faible la moyenne des y_i de l'ensemble de données d'entraînement. On cherche alors à construire un nouvel arbre f_1 qui va prédire le premier résidu $h_1(x) = y - f_0(x)$. De manière plus générale, pour chaque itération $1 \leq m \leq M$, on calcule les pseudo-résidus pour chaque donnée i :

$$r_m^i = -\frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)}$$

On calibre ensuite un nouvel arbre sur les données (x_i, r_m^i) avec la méthode vue plus haute et on le pondère par γ_m tel que

$$\gamma_m = argmin_\gamma \sum_i L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i))$$

Le modèle ajusté est alors $f_m = f_{m-1} + \eta\gamma_m h_m$ où η est un paramètre nommé taux d'apprentissage. Cette boucle est répétée un certain nombre de fois pour s'approcher le plus possible de nos données.

Ceci pose les bases théoriques sur lesquelles s'appuient l'algorithme XGBoost ; il reste encore à l'implémenter et à le calibrer pour notre objectif. La première étape de ce projet consiste à construire la matrice de données : nous avons 15 ans de valeurs pour chaque titre et pour chaque score. Doit-on entraîner notre modèle sur cet ensemble ou bien considérer des sous-ensembles pour capter de l'information plus localement ? Tous les titres sont-ils tout le temps actifs ?

Je dispose d'une matrice de booléens qui donne à chaque date, pour chaque titre, si celui-ci est actif, c'est-à-dire si l'on pouvait investir dessus à cette date précise. En plus de cela, pour éviter un trop grand nombre de valeurs manquantes dans les rendements, on ajoute une condition : si dans l'année précédente, le titre a un nombre de rendements manquants supérieurs à 25%, on le considère automatiquement inactif. De plus, la périodicité de rebalancement (d'investissement) est de 10 jours de bourse (2 semaines), je cale donc mes dates de prédiction avec ce calendrier. Pour ce projet, j'utilise les données de 43 scores qui serviront donc de features. Pour la classification en 5 quintiles, je remplace les valeurs manquantes dans la matrice de rendements par la moyenne de la ligne pour coller à l'évolution du marché. A chaque date, je recupère le rendement cumulé sur les 10 prochains jours pour chaque titre et j'en déduis les quintiles.

$$\begin{array}{c} \\ \text{date t} \\ \text{date t+1} \\ \vdots \\ \text{date t+10} \end{array} \begin{array}{c} \begin{array}{ccc} \text{titre 1} & \dots & \text{titre n} \end{array} \\ \begin{pmatrix} 0 & \dots & 2 \\ 2 & \dots & 8 \\ \vdots & \ddots & \vdots \\ -2 & \dots & 0 \end{pmatrix} \end{array}$$

On somme les valeurs des rendements par colonne pour trouver les rendements cumulés sur 10 jours pour tous les titres et on répartit en quintile.

$$\begin{array}{c} \\ \text{rdt cumulé t+10} \end{array} \begin{array}{c} \begin{array}{ccc} \text{titre 1} & \dots & \text{titre n} \end{array} \\ \begin{pmatrix} 1 & \dots & -3 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \\ \text{quintile t+10} \end{array} \begin{array}{c} \begin{array}{ccc} \text{titre 1} & \dots & \text{titre n} \end{array} \\ \begin{pmatrix} 3 & \dots & 0 \end{pmatrix} \end{array}$$

Maitenant, l'objectif de l'algorithme est de prédire le quintile d'appartenance du titre à l'aide des scores de ce même titre mais qui datent de 10 jours avant.

$$\begin{array}{c} \\ \text{titre 1} \\ \vdots \\ \text{titre n} \end{array} \begin{array}{c} \begin{array}{ccc} score(1)(t) & \dots & score(43)(t) \end{array} \\ \begin{pmatrix} x_1^1 & \dots & x_1^{43} \\ \vdots & \ddots & \vdots \\ x_n^1 & \dots & x_n^{43} \end{pmatrix} \end{array} \overset{prédiction}{\longmapsto} \begin{array}{c} \\ \text{titre 1} \\ \vdots \\ \text{titre n} \end{array} \begin{array}{c} \begin{array}{c} \text{quintile t+10} \end{array} \\ \begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix} \end{array}$$

La conclusion de cette stratégie serait alors de se positionner en Long sur les scores qui sont prédits dans le meilleur quintile et en Short sur ceux dans le moins bon.

Ainsi, pour une date t, on doit récupérer les données sur une certaine période finissant nécéssairement à la date $t-11$ pour entraîner notre modèle et qu'en effectuant la prédiction à cette date, on n'utilise pas de rendements qui ne seraient pas encore disponibles dans un cadre réel. Il se pose alors la question de la longueur de cette période d'entraînement : doit-elle être courte pour capter de manière précise l'information locale ou bien plus longue et saisir les motifs plus globaux des titres ?

Il faut aussi fixer les hyperparamètres de notre modèle, de manière dynamique tous les 10 jours ou bien les fixer tout du long ? Les hyperparamètres de l'algorithme sont de plusieurs types : ceux liés au calcul numérique, ceux des arbres (profondeur maximale, nombre d'observartions minimales dans un noeud, échantillonage à chaque noeud) et enfin ceux propres à l'optimisation (taux d'apprentissage, réduction minimum de coût à un noeud). Nous essayons tout d'abord la méthode dynamique : tous les 10 jours, nous trouvons les hyperparamètres les plus proches de l'optimalité en effectuant une recherche aléatoire dans le produit cartésien défini par les ensembles de valeurs que nous prenons pour chaque hyperparamètre. A chaque date, nous estimons 20 modèles avec des hyperparamètres tirés au sort dans la grille et nous les notons à l'aide d'une métrique particulière : l'intuition est de pénaliser le modèle en fonction du nombre de classes d'écart entre la prédiction et la réalité, en surpondérant si le titre appartient, respectivement est, prédit dans un quintile extrême et donc doit, respectivement va, entrer dans la stratégie d'investissement.

Plus précisement, on définit cette métrique par :

$$\begin{array}{rccc} \phi : & \{0,1,2,3,4\}^m \times \{0,1,2,3,4\}^m & \longrightarrow & \mathbb{R}_+ \\ & (\hat{y}, y) & \longmapsto & \sqrt{\frac{1}{m}\sum_{i=1}^{m} \epsilon_i |y_i - \hat{y}_i|} \end{array}$$

où $\epsilon_i = 3$ si $(\hat{y}_i, y_i) \in \{0,4\}^2$, $\epsilon_i = 1$ sinon.

Après avoir trouvé nos meilleurs paramètres par recherche aléatoire - il est impossible de tester toutes les combinaisons à cause de la complexité algorithmique - nous calibrons notre modèle sur la période et nous effectuons nos prédictions pour chacun des titres. Nous en déduisons un vecteur qui donne les positions prises (ou non) sur chaque titre et cela sur toute la durée du test. Je confie ensuite la matrice qui récapitule ces résultats à Laurent qui utilise alors son code pour obtenir les résultats "officiels" qu'aurait eu cette stratégie. Les résultats sont décevants, largement moins bons que la stratégie actuelle et d'autres très simples.

En remarquant que les périodes d'entraînement choisies par l'algorithme étaient souvent petites, j'ai pensé que le modèle s'appuyait de manière beaucoup trop précise sur l'aspect local et avait donc toujours un temps de retard sur la réalité. Pour palier à cela, j'ai testé une stratégie où l'on choisit les hyperparamètres en les évaluant sur toute la période et on les fixe ensuite. Il y aura alors une utilisation des données futures pour effectuer les prédictions, il faudra faire attention avant de conclure à une stratégie viable si les résultats sont encourageants.

De manière pratique, pour trouver ces hyperparamètres globaux, je prends comme données d'entraînement 1 date tous les 10 jours, de manière croisée avec les dates de prédiction : si t et t+10 sont deux dates de prédiction, je prends t+5 pour l'inclure dans l'ensemble d'entraînement. Cela suffit pour construire un ensemble comprenant plus de 150 000 données (environ 700 titres actifs par jour $\times$ 250 dates). Nous effectuons cette fois-ci une recherche par grille pour être plus précis dans l'optimisation des hyperparamètres.

Il ne reste donc plus qu'à choisir comment construire notre ensemble d'entraînement à chaque date pour effecture nos prédictions : on essaie plusieurs méthodes. On prend d'abord les 15 dernières dates possibles puis les 30. Pour obtenir un aspect plus global, on réalise aussi des tests en prenant une date sur deux lors des 60 derniers jours, une sur trois lors des 90 derniers et de la même manière pour 120 et 180.

Les résultats pour 15 jours sont de nouveau moyens, l'aspect local est beaucoup trop fort alors que les hyperparamètres sont justement globaux. Les résultats semblent se stabiliser pour 60 et 90 jours mais restent toujours moins bons que la méthode actuelle.

La prochaine étape a donc été d'essayer de prédire plus précisement les rendements grâce à de la régression. Pour contrer la volatilité du marché, on décide de normaliser les rendements par date. De cette manière, les prédictions faites pour l'ensemble des titres actifs nous donneront un classement des performances des titres qu'on pourra ensuite utiliser pour construire notre portefeuille investissable. En classification, la prédiction pour un titre qui se trouvait à la fronière entre deux quintiles était binaire et pouvait grandement dévier de la réalité. La régression permet de lisser ce problème et de laisser l'algorithme être plus précis dans ces cas.

En régression, on prédit les rendements normalisés tous les 10 jours pour tous les titres grâce à des périodes d'entraînement de la même forme qu'en classification (30 jours puis 1 date sur 2 lors des 60 derniers et idem pour 90, 120 et 180). Je donne ensuite à Laurent les matrices des prédictions faites par l'algorithme pour les différentes méthodes. On teste les résultats d'abord en regroupant en quintiles les scores en fonction de leurs rendements. Les résultats s'améliorent légèrement par rapport à la classification. La vraie amélioration s'effectue lorsqu'on prend les prédictions de rendements normalisés à la place des valeurs des scores dans la stratégie actuelle. On se rapproche alors significativement des performances de la stratégie actuelle d'Helium Alpha. A ce moment-là, mon stage était sur le point de se terminer et je n'ai pas pu aller plus loin dans le projet mais je pense qu'il y a matière à encore améliorer les résultats, aussi bien dans la matière d'optimiser l'algorithme avec les hyperparamètres et la façon dont on lui fournit les données mais aussi dans l'utilisation de ses prédictions pour construire les portefeuilles.

6 SHAP

Le gros problème des algorithmes de Machine Learning actuellement concerne leur interprétabilité. Même si leurs performances s'améliorent de manière spectaculaire, cela se fait au profit de leur transparence. Une prédiction faite par un algorithme de régression linéaire est très simple à expliquer avec les coefficients mais celle-ci ne sera pas toujours très précise. A l'inverse, un modèle plus récent type boîte noire aura de meilleurs prédictions mais plus opaques. De nouveaux articles apparaissent à ce sujet dans le RGPD : par exemple, si une entreprise prend une décision sur la base d'un algorithme, le client a le droit de demander à l'entreprise la raison de cette décision. Suite à cela, la recherche sur l'interprétabilité des algorithmes s'est beaucoup développée et a abouti à différentes méthodes d'explicabilité.

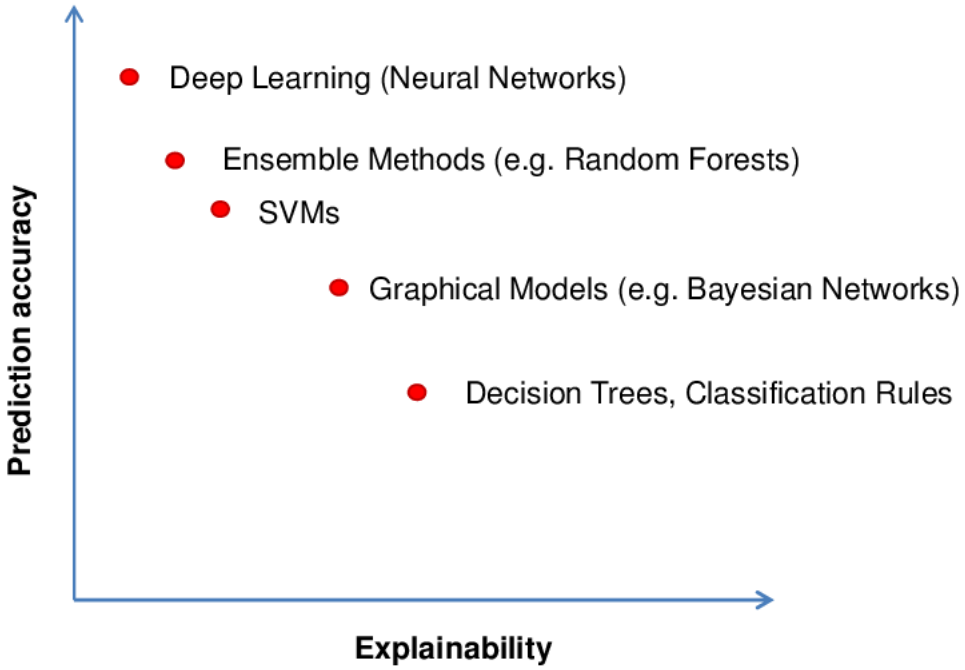

FIGURE 2 – Performance vs Interprétabilité

Le but du projet était de pouvoir localiser l'information contenue dans les scores et de l'utiliser dans la stratégie du fonds. Même si les résultats de la partie machine learning sont en dessous de ceux de la stratégie actuelle dans l'application directe, j'ai voulu savoir si on pouvait tout de même utiliser notre modèle pour s'informer sur les scores. En effet, pour effectuer une prédiction, l'algorithme ne s'appuie pas de la même façon sur tous les scores en fonction de l'ensemble d'entraînement. Ainsi, même si les prédictions ne sont pas excellentes, cela peut vouloir dire que le

modèle exploite de manière erronée les bons scores. On cherche donc à mesurer l'importance des scores sur les prédictions. Pour ce faire, il existe plusieurs méthodes : utilisation de la feature à un noeud, via permutation, LIME ou encore SHAP.

Les 2 premiers sont purements intuitifs et n'ont pas d'aspects mathématiques, nous éviterons donc de les utiliser. La méthode LIME consiste à approcher le modèle complexe par un modèle linéaire au voisinage de la prédiction qu'on cherche à expliquer. Ainsi, les coefficients de la régression donnent une approximation du poids qu'ont les features. Le plus gros inconvénient est l'aspect purement local de cette méthode qui ne permet pas d'agréger les résultats pour en déduire un résultat plus global que seulement sur une prédiction (1 titre à une date).

La méthode la plus intéressante est SHAP : elle est issue de la théorie des jeux et est donc la seule à avoir une fondation mathématique. Elle s'appuie sur les valeurs de Shapley qui donnent, dans le cadre d'un jeu coopératif, la répartition équitable des gains aux n joueurs en fonction de leurs apports. Les valeurs de Shapley sont définies par 4 axiomes. Si on note $\varphi_i(v)$ la valeur de Shapley du joueur i pour le jeu v, $N = \{1, \dots, n\}$, on pose :

Axiome 1 : $\sum_{i \in N} \varphi_i(v) = v(N)$, axiome d'efficacité ($v(N)$ est le gain maximal pouvant être obtenu par les joueurs.

Axiome 2 : Pour deux joueurs i et j, si pour tout $S \subseteq N \setminus \{i, j\}$ on a $v(S \cup \{i\}) = v(S \cup \{j\})$, alors $\varphi_i(v) = \varphi_j(v)$, axiome de symétrie.

Axiome 3 : Si $v(S \cup \{i\}) = v(S)$ pour tout $S \subseteq N \setminus \{i\}$, alors on a $\varphi_i(v) = 0$.

Axiome 4 : Pour toute paire de jeux (v, w), $\varphi(v + w) = \varphi(v) + \varphi(w)$ où $(v + w)(S) = v(S) + w(S)$.

Lloyd Shapley a alors montré le théorème suivant :

Théorème : Pour le jeu (N, v), il existe une unique fonction qui satisfasse ces 4 axiomes et elle est définie par :

$$\forall i \in N,\ \varphi_v(i) = \sum_{S \subseteq N \setminus \{i\}} \frac{(n - |S| - 1)!|S|!}{n!}(v(S \cup \{i\}) - v(S))$$

On calcule la contribution marginale de chaque joueur sur le gain du jeu : on moyenne la différence du gain avec le joueur en question et sans, cela sur toutes les coalitions possibles.

Dans notre cas, le jeu est remplacé par la prédiction et les joueurs sont les scores dont on veut évaluer l'impact. Les valeurs de Shapley sont calculées de manière locale, pour chaque prédiction, ce qui permet d'expliquer individuellement chaque prédiction. L'autre avantage de cette méthode est que ses résultats peuvent être agrégés et peuvent nous donner une estimation globale de l'importance des scores : on moyenne la valeur absolue des valeurs de Shapley pour chaque score sur toutes les prédictions. On prend la valeur absolue car sinon, un score qui contribue une fois positivement à la prédiction et une fois négativement de manière similaire aura une moyenne à 0 ce qui ne correspond

pas à la réalité : le score influe. On perd donc l'information sur l'orientation moyenne des prédictions propre à ce score : on quantifie seulement la manière dont le score est pris en compte dans le modèle.

Dans un premier temps, en regardant localement, on peut observer pour chaque prédiction l'impact des scores sur celle-ci, tout cela par rapport à la moyenne des valuers cibles de l'ensemble d'entraînement :

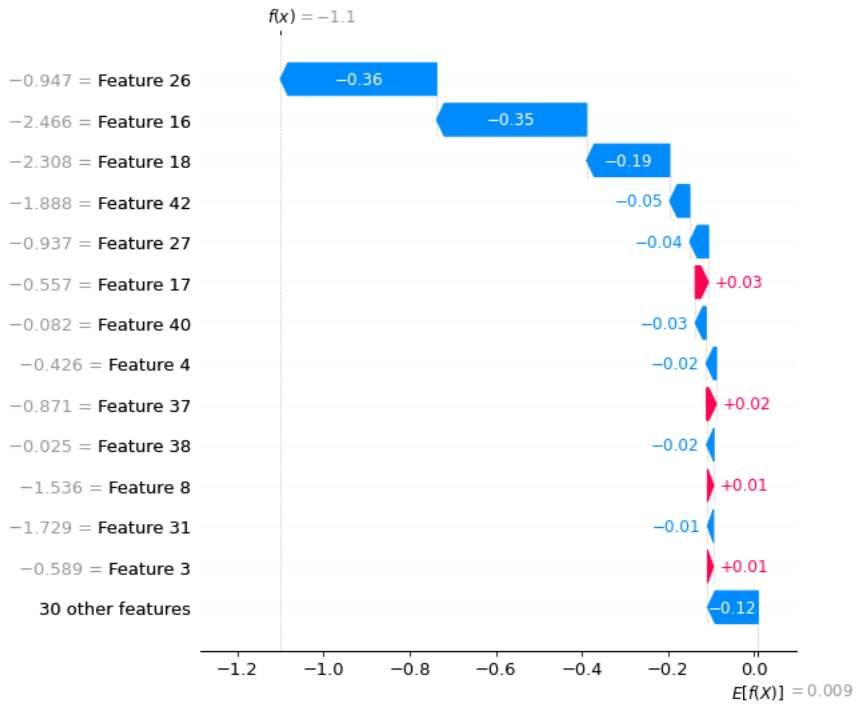

FIGURE 3 – Impact de chaque score pour une seule prédiction

Toutefois, ces informations sont très locales et ne sont pas utilisables directement : on souhaite les agréger comme expliqué plus haut. On obtient alors un diagramme où les barres horizontales traduisent l'importance du score sur l'ensemble d'entraînement tout entier du modèle :

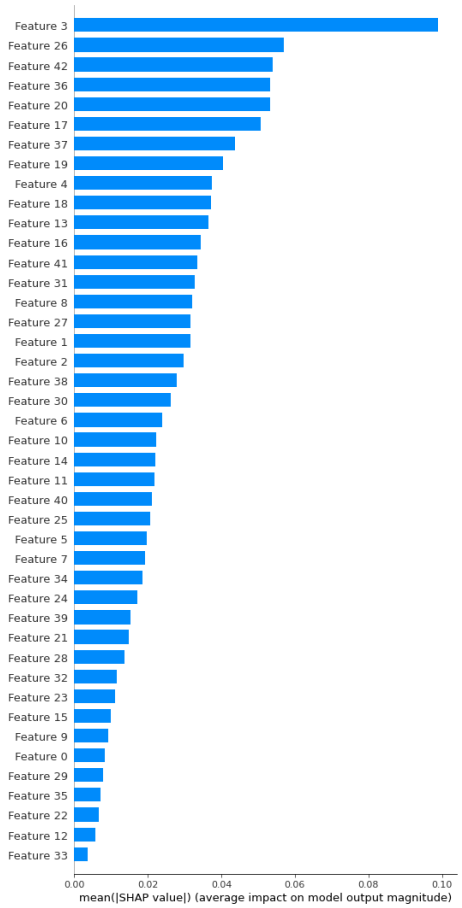

FIGURE 4 – Impact de chaque score pour un ensemble de prédictions

On voit ici que le score le plus utilisé par notre modèle est celui correspondant à la feature 3 donc au 4ème score.

Grâce aux valeurs de Shapley, le but était de trouver une localisation de l'information et d'en déduire quels scores sont fonctionnels et utilisables pour capter de l'information ; et qui se traduisent effectivement par des hausses ou baisses des cours. Par exemple, pour chaque score, en moyennant temporellement les dernières valeurs de Shapley obtenues pour tous les titres, on pourrait obtenir un vecteur qui quantifierait l'information contenue dans chacun des scores. Malheureusement, cette partie fut menée en parallèle du projet de Machine Learning et n'a pas non plus eu le temps d'aboutir complètement. Elle est en ce moment mise à l'essai par les gérants du fonds.

7 Conclusion

J'ai énormément apprécié ce stage, aussi bien sur l'aspect des mathématiques financières abordé durant ces 6 mois qu'humainement. L'équipe de Syquant Capital est très sympathique, je les remercie chaleureusement de m'avoir accueilli pendant cette période, en particulier Laurent qui fut très bienveillant. Mêmes si mes résultats semblent à première vue légèrement moins bons que ceux de la stratégie actuelle, je suis persuadé qu'avec plus de temps, ils pourraient être améliorés. Les données dont se sert l'algorithme sont de bonne qualité, en témoignent les performances actuelles du fonds. En calibrant plus précisement les hyperparamètres, en ajustant les ensembles d'entrainement et en testant de nouvelles variables cibles (output - régression/classification), l'algorithme pourrait être utilisé à son plein potentiel. C'est notamment une des pistes de réflexion explorées par les gérants du fonds, toujours en quête d'amélioration.

Ce stage m'a aussi permis de découvrir de manière un peu plus approfondie le monde de l'entreprise jusqu'ici pratiquement inconnu pour moi. A travers ces petites missions mathématiques confiées par Laurent, à savoir le classement des scores et le coefficient d'information, j'ai pu mettre en pratique mes qualités de recherche et d'adaptation aux problèmes posés. De plus, j'ai pu pour la première fois mener de bout en bout un projet de Machine Learning et ainsi me confronter à la réalité des ingénieurs en Data Science. Cela a permis de me conforter dans mon envie de faire partie du monde des mathématiques appliquées, plutôt orientées Machine Learning ; toutefois avec une forte envie de garder un important aspect de recherche dans mon activité future.

8 Références

[1] Bernd Scherer & Kenneth Winston. (2011). The Oxford Handbook of Quantitative Asset Management.

[2] Stefan Jansen. (2021). Machine Learning for Algorithmic Trading.

[3] Chloé-Agathe Azencott. (2022). Introduction au Machine Learning.

[4] Aurélien Géron. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow.

[5] Alan Julian Izenman. (2008). Modern Multivariate Statistical Techniques : Regression, Classification and Manifold Learning.

[6] Robin Genuer & Jean-Michel Poggi. Arbres CART et Forêts aléatoires, Importance et sélection de variables. 2017. ffhal-01387654v2.

[7] Wei-Yin Loh. (2014). Fifty years of classification and regression trees. International Statistical Review.

[8] Frederik Questier & Raf Put & Danny H. Coomans & Beata Walczak & Yvan Vander Heyden. (2005). The use of cart and multivariate regression trees for supervised and unsupervised feature selection. Chemometrics and Intelligent Laboratory Systems.

[9] Carolin Strobl & Anne-Laure Boulesteix & Achim Zeileis & Torsten Hothorn. (2007). Bias in random forest variable importance measures : Illustrations, sources and a solution. BMC bioinformatics.

[10] Zheng Tan & Ziqin Yan & Guangwei Zhu. (2019). Stock selection with random forest : an exploitation of excess return in the Chinese stock market. Helyion 5 (8), e02310.

[11] Anton Andriyashin & Wolfgang K. Härdle & Roman Timofeev. (2008). Recursive Portfolio Selection with Decision Trees.

[12] Simon Bernard & Laurent Heutte & Sébastien Adam. (2009). Influence of Hyperparameters on Random Forest Accuracy. International Workshop on Multiple Classifier Systems. ffhal-00436358f

[13] Tianqi Chen & Carlos Guestrin. (2016). Xgboost : A scalable tree boosting system. arXiv :1603.02754

[14] Didrik Nielsen. (2016). Tree Boosting with XGBoost.

[15] Robert Kosowski & Allan Timmermann & Russ Wermers & Hal White. (2005). Can Mutual Fund 'Stars' Really Pick Stocks ? New Evidence from a Boostrap Analysis.

[16] Eugene F. Fama & Kenneth R. French. (2010). Luck versus Skill in the Cross-Section of Mutual Fund Returns.

[17] Bradley Efron. (1993). An introduction to the Bootstrap.

[18] David Blake & Tristan Caulfield & Christos Ioannidis & Ian Tonks. (2017). New Evidence on Mutual Fund Performance : A Comparaison of Alternative Bootstrap Methods.

[19] Huazhu Zhang & Cheng Yan. (2017). A skeptical appraisal of the bootstrap approach in fund performance evaluation.

[20] Dimitri Delcaillau & Antoine Ly & Franck Vermet & Alizé Papp. (2020). Interprétabilité des modèles : état des lieux des méthodes et application à l'assurance. arXiv :2007.12919

[21] Scott Lundberg & Sun-In Lee. (2017). A unified approach to interpreting model predictions. arXiv :1705.07874

[22] Scott Lundberg & Gabriel G. Erion & Su-In Lee. (2019). Consistent Individualized Feature Attribution for Tree Ensembles. arXiv :1802.03888v3