Dauphine | PSL
UNIVERSITÉ PARIS

Mémoire universitaire de M1 Mathématiques appliquées

Sujet: Investir en prenant en compte les cycles économiques avec des algorithmes d'apprentissage automatique

Encadrant de mémoire:
M. Sylvain BENOIT

Étudiants :
Benslimane Mohamed

Hammadi Rayyan

Mahjoub Mazen

Année académique 2022/2023

Résumé

Les méthodes et algorithmes d'apprentissage automatique supervisés et non supervisés sont l'un des nombreux outils de prédiction des cycles économiques et d'anticipation en temps réel des points de retournement de l'économie aux États-Unis. La variable d'intérêt, appelée Y, est utilisée pour classer les régimes en accélération ou ralentissement. Nous mettons en exergue les nombreuses méthodes pour effectuer cette prédiction du présent en temps réel en simulant une fenêtre élargissante lors de nos estimations et ce en exploitant une base de données contenant plus de 70 indicateurs économiques ainsi que les points de retournements mensuels entre 1985 et 2020 de l'économie aux États-Unis.

Après avoir comparé la précision, la capacité de ces algorithmes à traiter un grand nombre de variables explicatives et d'estimer les données avec de nombreux outils économétriques ces résultats théoriques sont confrontés dans un second temps à l'implémentation de deux stratégies d'investissement hypothétiques et volontairement simples, les stratégies Equity et dynamique. La confrontation de ces deux stratégies avec la stratégie classique du benchmark dit 'Buy and Hold' permettra d'obtenir le meilleur choix d'investissement en fonction des cycles économiques aux États-Unis afin de savoir s'il est possible de battre le marché à l'aide de ces outils algorithmiques de prédiction.

Remerciement:
Un grand merci à monsieur Sylvain Benoit qui nous a guidés de par son temps et ses conseils tout au long de ce mémoire universitaire. :)

Plan

1 Introduction et objectifs

Les investisseurs, économistes et statisticiens sont confrontés à un problème récurrent qui est de déterminer avec précision les points de retournement de l'économie et les cycles économiques.

La prédiction de ces cycles économiques, c'est-à-dire l'anticipation des phases de récession, d'expansion et les moments clés de points de retournements de l'économie, ces moments de rupture entre deux phases consécutives du cycle économique est vital pour les décisions d'investissement ou encore les politiques gouvernementales.

Nos papiers de référence de Benoit S. & Raffinot T (2018) et Cooper & Priestley (2009) soulignent l'importance de la prédiction des cycles économiques et des rendements boursiers dans les décisions d'investissement. En effet, en théorie les stratégies d'investissement basées sur la prédiction de ces points de retournement devraient permettre d'obtenir des très hauts rendements et ce en fonction du risque.

L'objet de notre mémoire universitaire sera donc d'essayer de répondre à ce problème inhérent en finance et en économie : Peut-on battre le marché ?

Néanmoins, la prédiction et détection de ces points de retournement de l'économie n'est pas une tâche aisée. Ces méthodes de prédictions quasi instantanées dites de nowcasting font l'objet de nombreux ouvrages et études. En effet, le fait de produire des prédictions en temps quasi-réel de nombreuses variables économiques permettant presque instantanément de capturer les fluctuations économiques est devenu un outil majeur plébiscité par les décisions gouvernementales ou d'investissement là où les économistes peinent en général à détecter de manière précise si un nouveau cycle économique a commencé.

Tout l'objet de notre mémoire, qui reprend les travaux de Raffinot (2017) et Benoit (2018), se situe dans la continuité d'un long processus de développement de ces méthodes de prédictions économiques. Ainsi, depuis des décennies, les méthodes de nowcasting ont été utilisées même si leur utilisation s'est considérablement accrue cette dernière décennie avec l'essor de l'analyse de données à haute fréquence et du Machine Learning.

Nous appliquerons donc des méthodes et des algorithmes d'apprentissage automatique dans le but de fournir plusieurs modèles de prédictions visant à détecter avec précision les points de retournement du cycle économique aux États-Unis. Ces nombreux algorithmes de forêt aléatoire, Gradient Boosting ou encore de régression logistique seront une étape clé afin de traiter de nombreuses variables explicatives dans le but d'effectuer une prédiction sur notre variable d'intérêt, notre label Y défini par :

$$Y_t = \begin{cases} 1 & \text{en période d'accélération} \\ 0 & \text{en période de décélération} \end{cases}$$

Nous reviendrons plus en détail dans ce mémoire sur la présentation, la comparaison et les grandes capacités de ces algorithmes de prédiction. Pour finir, ces algorithmes d'apprentissage automatique se baseront sur notre ensemble de données fournies par l'OCDE, entre 1985 et 2020 aux États-Unis. Ce dataset à notre disposition contient 70 variables explicatives et la nature de nos variables explicatives est détaillée dans l'annexe **6.1** .

2 Approche et généralités sur les modèles

2.1 Méthodes, données et configuration empirique

Premièrement, dans ce mémoire, nous allons simuler la performance de nos modèles d'apprentissage statistique sur une certaine période donnée en utilisant la méthode dite de la fenêtre élargissante. Cette méthode consiste à faire tourner nos algorithmes en temps réel afin d'accumuler de l'information en temps réel. Voici le fonctionnement et principe cette méthode:

Algorithm 1 Algorithme de la fenêtre élargissante

Input: Données historiques $X_t, Y_{t\,t=1}^{T}$
Output: Prévisions pour la prochaine période
Initialiser le modèle avec les premières observations; **for** $t = 1$ *to* T **do**
  Estimer f telle que $Y_t = f(X_t)$;
  Prédire $\hat{Y_{t+1}} = \hat{f}(X_{t+1})$;
end

Plus précisément, nous allons prédire les régimes économiques en modélisant une variable qualitative qui prendra la valeur 0 en cas de ralentissement et 1 en cas d'accélération.

$$Y_t = \begin{cases} 0 & \text{si ralentissement} \\ 1 & \text{si accélération} \end{cases} \tag{1}$$

Cette variable sera expliquée par plus de 70 indicateurs économiques lors de l'entraînement des modèles. De plus, nous supposons que, à la date t, nous ne disposons pas des 12 derniers mois de cette variable Y. Par conséquent, nous allons prédire le présent avec des données datant d'il y a 12 mois, ce qui est l'un des objectifs de notre mémoire.

Algorithm 2 Algorithme de la fenêtre élargissante avec la contrainte

Input: Données historiques $X_t, Y_{t\,t=1}^{T-12}$
Output: Prévisions pour la prochaine période
Initialiser le modèle avec les premières observations; **for** $t = 1$ *to* $T - 12$ **do**
  Estimer f telle que $Y_{t-12} = f(X_{t-12})$;
  Prédire $\hat{Y_t} = \hat{f}(X_t)$;
end

En effet, cette hypothèse est conforme à la réalité, car l'OCDE (Organisation de coopération et de développement économiques) qui fournit cette variable Y n'a pas assez de recul pour classer correctement les 12 derniers Y. Cependant, cette contrainte peut avoir un impact négatif sur la performance de nos modèles, comme l'illustre le graphique ci-dessous de l'autocorrélation de Y, qui montre que la corrélation entre Y_t et Y_{t-12}, tend vers 0 et devient négative avec un lag plus grand.

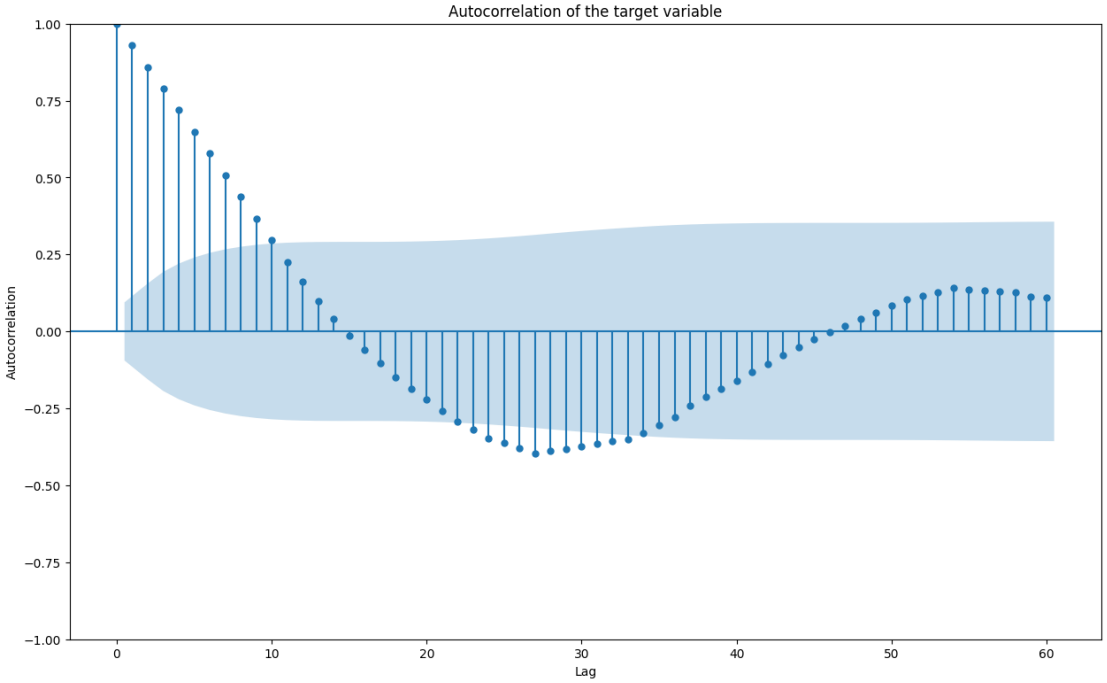


Figure 1: Graphique de l'autocorrélation de Y

Dans la suite de l'étude, nous présenterons différentes méthodes qui pourraient atténuer cette dégradation des performances due à cette contrainte temporelle et inhérente dans notre cas.

Notons aussi que notre variable d'intérêt Y n'est pas équilibrée au cours du temps. C'est à dire qu'il y a plus de phases d'accélération que de phase de ralentissement comme le montre l'histogramme de la figure **10**.

Les deux graphiques suivants mettent en évidence l'impact négatif de l'absence de notre variable cible des 12 derniers mois. Il est à noter que sans lag, les modèles performent considérablement bien. En effet on voit que nos prédictions sont conformes aux périodes d'expansion (en vert) et de récession (en rouge).

Figure 2: Graphe de probabilité des prédictions sans lag

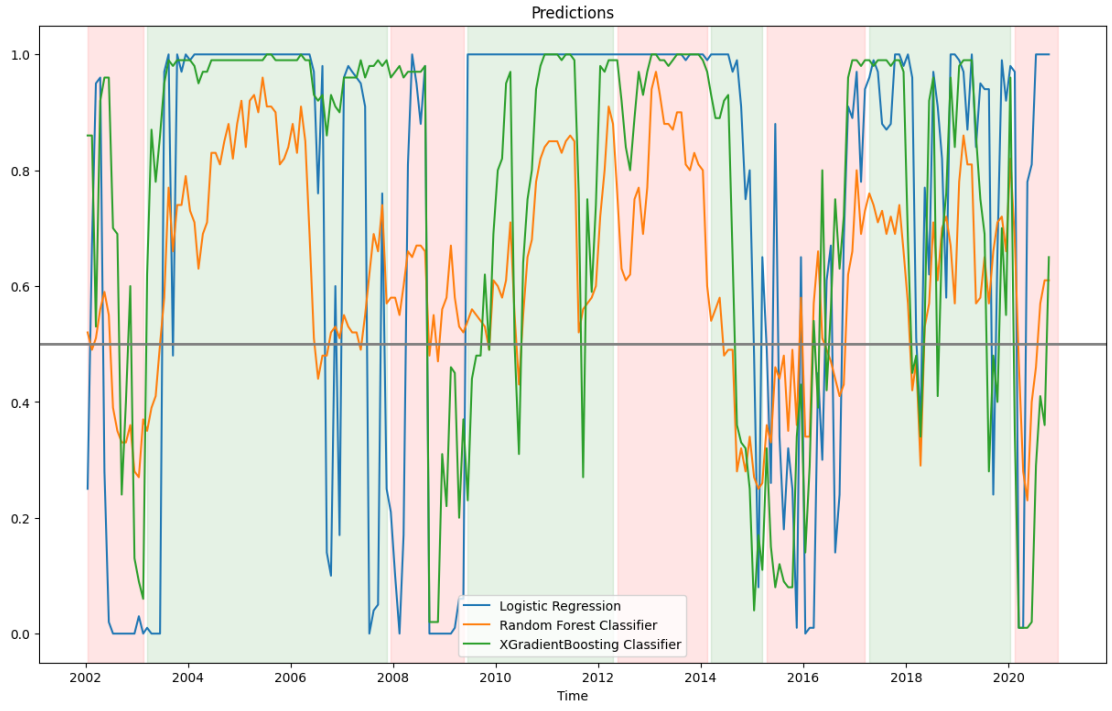


Figure 3: Graphe de probabilité des prédictions avec lag de 12 mois

Nous proposerons donc par la suite plusieurs méthodes d'amélioration pour faire en sorte que nos classifieurs fassent abstraction de cette absence de données.

2.2 Modèles

Les algorithmes dits d'apprentissage automatique sont particulièrement adaptés à la détection de points de retournement car ils peuvent s'adapter à des motifs changeants et apprendre à partir de données. En s'entraînant sur des données historiques, ces algorithmes d'apprentissage automatique peuvent identifier des motifs indicatifs de points de retournement, puis dans un second temps utiliser ces informations pour faire des prédictions sur des données futures. De plus, ces algorithmes d'apprentissage automatique peuvent utiliser une grande variété de variables d'entrée, telles que des indicateurs économiques, des données de marché financier et d'autres variables pertinentes. Dans notre projet, nous utilisons environ 70 variables explicatives pour entraîner nos modèles.

Dans cette section 2.2, nous allons explorer dans un premier temps l'utilisation des méthodes d'apprentissage automatique pour la détection de points de retournement dans l'économie. Nous discuterons des différents algorithmes utilisés en commençant par notre référence qui est la régression logistique, puis deux des techniques les plus populaires pour construire des modèles d'ensembles, qui sont la forêt aléatoire (Breiman, 2001) et le boosting (Schapire, 1990).

2.2.1 Régression logistique

Étant donné que notre variable cible est binaire, un point de départ cohérent pour notre analyse est d'employer une approche de régression logistique.
La régression logistique (ou Logit) est une méthode statistique paramétrique qui suppose une relation linéaire entre les variables prédictives et la probabilité de la variable de résultat. En effet, elle est considérée comme paramétrique car elle estime un nombre fixe de paramètres à partir des données d'entraînement, qui restent constants lors de la prédiction sur de nouvelles données. La fonction dite logistique est utilisée pour transformer l'équation linéaire en une fonction non linéaire garantissant que les probabilités prédites sont comprises entre 0 et 1. Cette méthode est interprétable et très efficace car elle peut traiter des ensembles de données volumineux avec un grand nombre de variables.

Au lieu d'ajuster une ligne droite ou un hyperplan, notre premier modèle de régression logistique utilise la fonction logistique pour réduire la sortie d'une équation linéaire entre 0 et 1. Cette fonction logistique est définie comme suit :

$$\text{logistic}(\eta) = \frac{1}{1+\exp(-\eta)}$$

En principe, le passage de la régression linéaire à la régression logistique est assez simple. Dans le modèle de régression linéaire, nous modélisons la relation entre la variable de résultat et les caractéristiques avec une équation linéaire :

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

Dans notre cas ici, c'est à dire pour un problème de classification, nous préférons des probabilités comprises entre 0 et 1, donc nous enveloppons le côté droit de l'équation dans la fonction logistique ce qui force la sortie à prendre uniquement des valeurs entre 0 et 1.

$$P\left(y^{(i)}=1\right)=\frac{1}{1+\exp\left(-\left(\beta_0+\beta_1 x_1^{(i)}+...+\beta_p x_p^{(i)}\right)\right)}$$

Quant à l'estimateur $(\beta_0, \beta_1, ..., \beta_p)$, que l'on note $\hat{\beta}$, c'est la valeur β maximisant la fonction de vraisemblance :

$$L=\prod_\omega P(Y(\omega)=1 \mid X(\omega))^{Y(\omega)} \times [1-P(Y(\omega)=1 \mid X(\omega))]^{1-Y(w)}$$

En pratique, de nombreuses procédures (solveurs) sont disponibles pour obtenir une solution satisfaisante au problème de maximisation en question. Ceci explique également pourquoi elles ne fournissent pas des solutions strictement identiques. Nous présentons maintenant l'une de ces procédures connue sous le nom de méthode de Newton-Raphson.

Méthode de Newton-Raphson

1. Fixer un point de départ $\boldsymbol{\beta}^{(0)}$.
2. A l'étape (k+1): calculer $\boldsymbol{\beta}^{(k+1)}=\boldsymbol{\beta}^{(k)}+A_k\nabla\mathcal{L}\left(\boldsymbol{\beta}^{(k)}\right)$,
avec $A_k=-\left[\mathcal{H}(\mathcal{L})\left(\beta^k\right)\right]^{-1}$ est la Hessienne de L
3. On s'arrête quand $\boldsymbol{\beta}^{(k+1)}\approx\boldsymbol{\beta}^{(k)}$ ou $\nabla\mathcal{L}\left(\beta^{k+1}\right)\approx\nabla\mathcal{L}\left(\boldsymbol{\beta}^{k}\right)$

Hypothèses de notre modèle de régression

Avant d'utiliser la régression pour prédire nos probabilités, les modèles de régression logistique doivent respecter différents critères pour que le modèle soit précis et que nos résultats soient valides.
Les observations utilisées dans le modèle de régression logistique doivent être toutes **indépendantes** et ne doivent pas être influencées par d'autres observations.
En effet, les variables prédictives utilisées dans le modèle de régression logistique ne doivent pas être fortement **corrélées** les unes avec les autres, car cela peut conduire à des estimations de coefficient instables et/ou peu fiables. Bien qu'une corrélation de 1 soit difficile à obtenir, nous observons néanmoins que certaines de nos variables explicatives sont fortement corrélées comme en témoigne le graphique **25** qui illustre la forte corrélation parmi les 12 premières variables.
Par conséquent, nous avons essayé et implémenté différentes techniques pour nous assurer que nos algorithmes convergent finalement vers une solution. La première consiste simplement à permettre aux algorithmes d'avoir un grand nombre d'itérations. La seconde est de réaliser une Analyse en Composantes Principales (ACP). Nous reviendrons plus en détail sur ce point-là dans notre partie consacrée au feature engineering.

Quant aux modèles de régression logistique, ils sont également relativement robustes aux valeurs aberrantes et ne nécessitent pas que les données soient distribuées selon une loi normale. Ils peuvent être régularisés pour éviter le surapprentissage et améliorer les performances de généralisation.

Les modèles de régression logistique utilisent généralement une technique de régularisation dite de "pénalisation L2" par défaut. Cette technique de pénalisation consiste à ajouter un terme de pénalité à la fonction de coût du modèle, ce qui permet de prévenir le surapprentissage dans les scénarios avec des prédicteurs de grande dimension ou corrélés en réduisant les coefficients de régression vers zéro.

Nous pouvons reformuler le problème d'optimisation que résout ridge comme suit:

Problème d'optimisation

$$\underset{\beta}{\text{Min}} \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2$$

avec $\|\cdot\|_2$ la norme euclienne ℓ_2.

Pour finir, lorsque nous avons comparé les performances de notre modèle de régression logistique en utilisant deux approches différentes, à savoir l'analyse en composantes principales (ACP) et la régularisation L2 par défaut, nous avons observé que cette dernière approche a donné des mesures de précision plus élevées.

Hyperparamètres du modèle de régression

Pénalité: Nous avons délibérément décidé d'utiliser le paramètre de pénalité par défaut sous forme de régularisation Ridge (l2) pour lutter contre le problème de surajustement dans nos modèles prédictifs.
Néanmoins, bien que la régularisation Lasso (dite l1) soit une technique alternative, elle a tendance à ne sélectionner qu'une seule variable parmi l'ensemble des prédicteurs hautement corrélés, comme c'est le cas dans notre étude, ce qui peut conduire à un fort biais dans le modèle où la prédiction devient trop dépendante d'une seule variable. En revanche, la régression Ridge quant à elle réduit la complexité du modèle sans réduire le nombre de variables. Cela est dû au fait qu'elle n'oblige pas les coefficients à être nuls, elle les rétrécit tout simplement. Par conséquent, la régularisation Ridge peut prévenir et éviter le surajustement et améliorer la performance de généralisation du modèle en réduisant la variance des coefficients estimés.
Étant donné que l'accent principal de notre analyse est mis sur la précision prédictive plutôt que sur l'interprétation, il est raisonnable d'utiliser la régularisation Ridge dans notre approche de modélisation. On opte pour cette régularisation car la régularisation Ridge est connue pour améliorer les performances prédictives en réduisant le surajustement et en augmentant la généralisation du modèle, sans nécessairement sacrifier le nombre de variables incluses dans le modèle.

2.2.2 Méthodes d'ensemble apprentissage automatique

Les méthodes d'ensemble, comme le boosting et la forêt aléatoire, sont des techniques populaires d'apprentissage automatique combinant les prédictions de plusieurs modèles. Elles sont très efficaces pour gérer un grand nombre de prédicteurs et effectuer de manière simultanée l'estimation et la sélection de variables. Bien que les deux méthodes utilisent une randomisation pour atténuer l'overfitting, elles diffèrent dans la manière dont elles forment concrètement des ensembles. En effet, la forêt aléatoire utilise une moyenne simple en sortie des modèles, tandis que la méthode de boosting englobe et combine des modèles qui fonctionnent individuellement mal en un seul modèle avec des propriétés améliorées. Le boosting ajoute de manière séquentielle de nouveaux modèles à l'ensemble et les entraîne par rapport à l'erreur de l'ensemble entier appris jusqu'à présent.

2.2.3 Random-Forest

Un autre algorithme de prédiction que nous implémentons dans le cadre de notre mémoire est la forêt aléatoire (RF). C'est une méthode statistique non paramétrique utilisée pour la classification et la régression, même dans des paramètres à haute dimensionnalité où la relation entre les prédicteurs et la variable de réponse peut être complexe et non linéaire comme dans notre cas. RF utilise un grand ensemble d'arbres de décision, chacun entraîné sur un sous-ensemble différent des données d'entraînement, et combine leurs sorties en moyennant. Cette technique permet de résoudre le problème récurrent de surapprentissage associé aux arbres de décision simples. De plus, RF est efficace en termes de temps d'exécution, résistant au bruit et performant sur des données non linéaires, ce qui en fait un choix populaire pour les applications pratiques et ce qui nous pousse à l'utiliser aussi dans le cadre de nos prédictions.

Arbres de régression et de classification (CART)

CART a été introduit par Breiman, Friedman, Olshen et Stone en 1984 et est largement utilisé dans divers domaines. Les arbres de classification et de régression (CART) sont des méthodes d'apprentissage automatique plébiscitées pour construire des modèles de prédictions à partir de données pour la classification et/ou la régression. Ces modèles sont créés en faisant la partition des données et en ajustant un modèle de prédiction volontairement simple dans chaque partition, ce qui nous donne un arbre de décision. En pratique, la procédure CART se décompose en deux étapes : tout d'abord la construction de l'arbre complet à l'aide d'une procédure de séparation dite binaire, puis dans un second temps le raffinement du modèle pour réduire la complexité et éviter le surajustement.

Détaillons cette partie de séparation dans CART avec le critère de Gini

Pour notre classification binaire, nous utilisons l'indice dit d'impureté de Gini pour trouver la meilleure séparation, définit en utilisant la formule suivante :

$$\text{GiniIndex} = 1 - \sum_j p_j^2$$

avec p_j est la probabilité de la classe j.
Cette mesure d'impureté de Gini mesure la fréquence à laquelle tout élément de l'ensemble de données sera mal étiqueté s'il est étiqueté de manière aléatoire.
De plus, remarquons que la valeur minimale de l'indice de Gini est 0. Cela se produit lorsque le "nœud est pur", ce qui signifie que tous les éléments contenus dans le nœud sont d'une seule classe unique. Par conséquent, ce nœud ne sera plus divisé. Ainsi, la division optimale est choisie en fonction des caractéristiques ayant le plus faible indice de Gini. De plus, l'indice de Gini atteint sa valeur maximale lorsque la probabilité des deux classes est la même.

$$\text{Gini}_{\text{min}} = 1 - \left(1^2\right) = 0$$
$$\text{Gini}_{\text{max}} = 1 - \left(0.5^2 + 0.5^2\right) = 0.5$$

Revenons à notre algorithme CART. Étant donné notre ensemble de données X, l'algorithme CART fonctionne selon le processus suivant:

Algorithm 3 Algorithme CART

1. Trouver pour chaque variable la meilleure séparation qui minimise le critère de Gini.

2. Trouver la meilleure séparation pour le nœud qui minimise le critère de Gini.

3. Si **a** est la meilleure séparation pour la variable choisie, alors toutes les valeurs inférieures à **a** sont envoyées à gauche et les autres à droite.

Algorithme de la foret aléatoire (Random Forest Algorithm)

En pratique, le random forest que nous utilisons dans notre code est construit en suivant les étapes suivantes :

Etape 1 : Étant donné un ensemble d'entraînement de N échantillons et P caractéristiques, nous fixons en tant qu'hyperparamètre p un nombre de caractéristiques tel que $p \leq P$ pour sélectionner aléatoirement pour chaque arbre et un nombre T (encore un hyperparamètre) qui représente le nombre d'arbres à cultiver.

Etape 2 : Prendre un échantillon bootstrap des N échantillons associés aux p caractéristiques.

Etape 3 : Nous utilisons l'échantillon bootstrap avec les caractéristiques associées pour faire pousser un CART.

Etape 4 : Répéter les étapes 2 et 3, T fois.

Etape 5 : Nous obtenons T arbres avec différentes sorties. Chaque arbre aura sa propre classification. La forêt aléatoire choisit alors la classe qui a le plus de votes.

Pour un problème de classification avec P caractéristiques, on voit que p $=\sqrt{P}$ caractéristiques sont utilisées. Ainsi, à chaque division, l'algorithme n'est pas autorisé à considérer une majorité des prédicteurs disponibles. L'idée sous-jacente est que s'il existe un prédicteur très fort dans l'ensemble de données, ainsi que d'autres prédicteurs modérément forts, alors dans la collection d'arbres bagués, ils utiliseront tous ce prédicteur fort dans la division principale. Par conséquent, tous les arbres bagués seront très similaires, et la moyenne de leurs prédictions ne réduira pas la variance, car les prédictions seraient fortement corrélées.

Les forêts aléatoires sont en capacité de dépasser cette difficulté en forçant chacune des divisions à ne considérer qu'un sous-ensemble de prédicteurs qui décorrélera efficacement les arbres.

Ainsi, l'algorithme RF suit deux étapes principales : le bootstrap et la moyenne. Tout d'abord, il crée une collection d'arbres de décision qui sont entraînés sur différents sous-ensembles des données d'entraînement grâce au bootstrap, puis il moyenne leurs prédictions pour faire la prédiction finale que nous analyserons dans les prochaines parties.
On peut donc résumer le fonctionnement de l'algorithme RF comme suit :

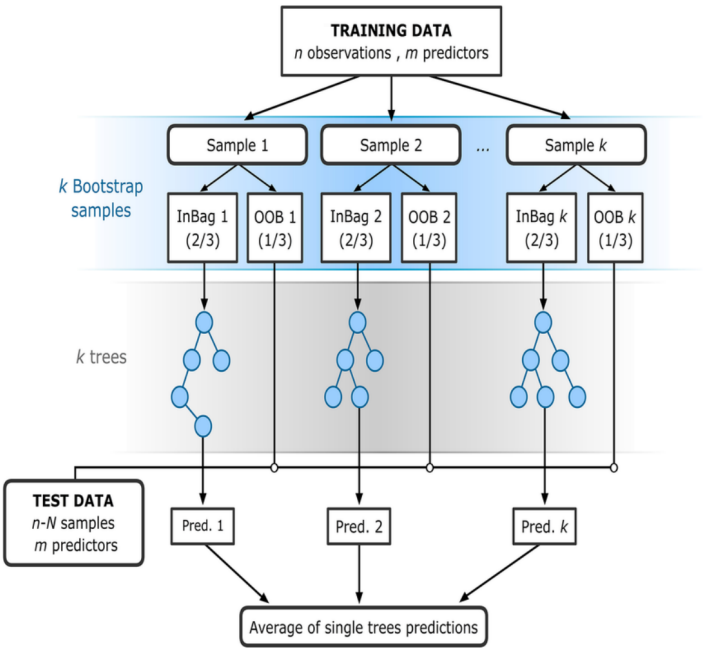


Figure 4: Fonctionnement de RF par Rodriguez-Galiano et al 2015

2.2.4 Gradient Boosting

Une autre méthode d'apprentissage automatique conseillée par le document de référence est le boosting. C'est une méthode d'apprentissage automatique qui se base sur l'agrégation "de nombreux apprenants faibles", ce sont des modèles caractérisés par un fort biais et une faible variance. Le raisonnement est le suivant: au lieu de former un seul modèle, le boosting ajoute séquentiellement de nouveaux apprenants faibles à l'ensemble, chaque nouveau modèle étant formé sur la base des erreurs des modèles précédents. Le modèle final devrait avoir des performances prédictives supérieures à celles de tout modèle individuel. Le boosting suit un processus itératif où les erreurs sont continuellement modélisées et corrigées (Schapire, 1990).

Gradient Descent Algorithm

La descente de gradient (GD) est un algorithme d'optimisation itératif largement utilisé pour déterminer un extremum local d'une fonction. Cette méthode est fréquemment mise en œuvre dans l'apprentissage automatique ou encore le Deep Learning pour minimiser une fonction de perte donnée.
Pour que l'algorithme GD fonctionne, la fonction doit être **différentiable** et **convexe**.
Intuitivement, le **gradient** est la pente d'une courbe en un point donné et ce dans une direction spécifique. Dans le cas d'une fonction qui est univariée, il s'agit simplement de la première dérivée au point donné. Dans le cas d'une fonction qui est multivariée, il s'agit du vecteur de dérivées dans chaque direction (variables).
Ainsi, l'algorithme de descente de gradient est une procédure itérative qui consiste à calculer le point suivant en utilisant le gradient de la position actuelle, en le mettant à l'échelle (par un taux d'apprentissage) et en soustrayant la valeur obtenue de la position actuelle, en prenant

ainsi une étape. La raison de la soustraction de la valeur est que l'algorithme vise en général à minimiser la fonction, plutôt que de la maximiser, ce qui nécessiterait une addition. Mathématiquement, le processus peut être exprimé comme suit:

$$p_{n+1} = p_n - \eta \nabla f\left(p_n\right)$$

Notre algorithme de descente de gradient utilise un paramètre essentiel, noté η, qui permet de mettre à l'échelle le gradient et régule ainsi la taille du pas. Ce paramètre η est appelé taux d'apprentissage dans le contexte de l'apprentissage automatique et est connu pour avoir un impact significatif sur les performances. Étudions cet impact:

1. Plus ce taux d'apprentissage est petit, plus l'algorithme de descente de gradient prendra beaucoup de temps à converger, ou dans le pire des cas il peut même atteindre le nombre maximum d'itérations avant d'atteindre le point optimal.

2. Si le taux d'apprentissage est trop grand, l'algorithme peut ne pas converger vers le point optimal (il peut sauter autour) ou même diverger complètement.

Pour finir, notons le fait que dans le cadre de notre mémoire, l'objectif est d'utiliser un ensemble d'apprentissage $\{y_i; \mathbf{x}_i\}_{(i=1,\ldots,n)}$ où y est notre variable binaire et $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$ est une variable d'entrée p-dimensionnelle, pour obtenir une estimation $\hat{f}(\mathrm{x})$ de la fonction $f(\mathbf{x})$ afin de minimiser l'espérance d'une fonction de perte (hyperparamètre du modèle) $L(y, f)$, c'est-à-dire :

$$\hat{f}(\mathbf{x}) = \arg\min_{f(\mathbf{x})} \mathbf{E}(L(y, f(\mathbf{x}))$$

En plus de fournir une fonction de perte particulière $L(y, f)$, nous devons également fournir ce que nous avons appelé un apprenant faible $h(\mathbf{x}, \theta)$ où θ est un ensemble de paramètres. Ainsi, le problème d'optimisation initial est maintenant un problème d'optimisation de paramètres. Ceci nous mène au schéma suivant:

Le Gradient Boosting est construit en suivant les étapes suivantes :

step 1: Initialiser le modèle avec une valeur constante:

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(y_i, \gamma\right)$$

step 2: Calculer les résidus: $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ pour $i = 1, \ldots, n$

step 3: Fit $h(\mathbf{x}, \theta)$ pour être le plus corrélé avec le vecteur gradient :

$$\theta_m = \arg\min_{\beta,\theta} \sum_{i=1}^{n} \left(r_{im} - \beta h\left(\mathbf{x}_i, \theta\right)\right)^2$$

.

step 4: Calculer la meilleure taille de pas de descente η_m

$$\eta_m = \arg\min_{\eta} \sum_{i=1}^{N} L\left(y_i, \hat{F}\left(\mathbf{x}_i\right)_{m-1} + \eta h\left(\mathbf{x}, \theta_m\right)\right).$$

step 5: Mettre à jour le modèle.:

$$\widehat{F}_m(\mathbf{x}) \leftarrow \hat{F}_{m-1}(\mathbf{x}) + \lambda \eta_m h\left(\mathbf{x}, \theta_m\right):$$

Ici, le paramètre $\lambda \in [0, 1]$ est ajouté pour contrôler la complexité du modèle, la réduction directement proportionnelle proposée par Friedman (2001) (la valeur par défaut est fixée à 0,1).

step 5: Réexécutez les étapes 2 à 5 jusqu'à ce que l'algorithme atteigne le nombre d'itérations autorisé.

XGBoost vs Gradient Boosting:

Pour finir sur cette partie de présentation de notre prédiction par Gradient Boosting, on cherche à comprendre les différences entre XGBoost et ce dernier. XGBoost est une forme plus régularisée du Gradient Boosting. XGBoost utilise une régularisation avancée (L1 L2), ce qui améliore les capacités de généralisation du modèle d'où le choix d'opter pour cette modélisation. XGBoost offre des performances élevées par rapport au Gradient Boosting. Son entraînement est très rapide et peut être parallélisé sur des clusters.

Ceci clôt cette partie de présentation théorique de nos modèles de prédiction.

3 Amélioration des modèles

L'interet du Feature Engineering

Cette section 3 aborde le concept de traitement des données dit feature engineering. Il consiste en l'ensemble des manipulations de données dans le but d'améliorer les performances prédictives de nos modèles. En effet, le processus de feature engineering est une composante essentielle de notre approche d'apprentissage supervisé et est donc intégré dans l'ensemble de notre processus de prédiction, qui est résumé par ce schéma (fortement inspiré de celui de Yating Liu, Introduction à l'apprentissage statistique) :

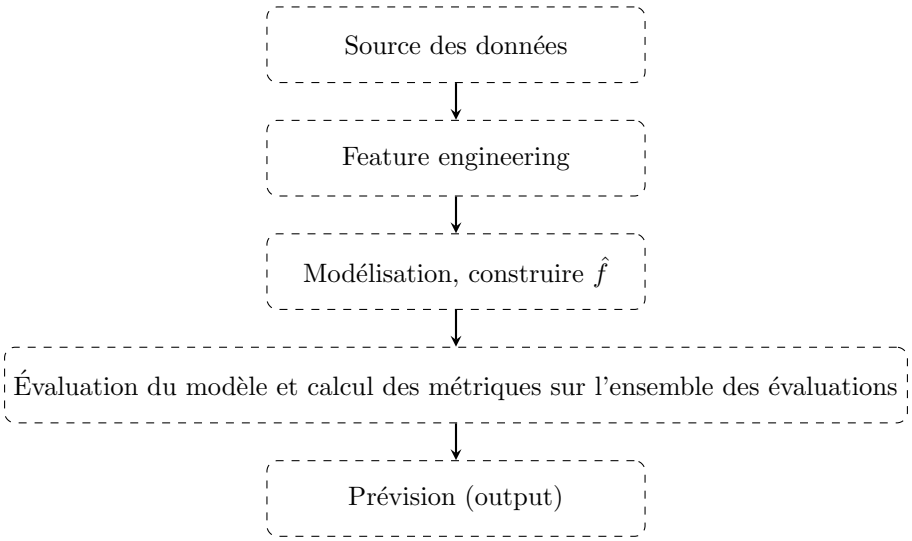


Fig 1. Représentation simplifiée de l'apprentissage supervisé

En effet, l'amélioration ou non de nos modèles de prédiction se traduira par une augmentation de leurs précisions, de leurs fiabilités ou encore de leurs capacités à généraliser les résultats obtenus à de nouveaux ensembles de données. Cette partie 'feature engineering' est essentielle car elle permet d'identifier les relations entre les variables, de mieux comprendre le dataset à notre disposition et enfin et de sélectionner les caractéristiques de nos variables explicatives les plus importantes pour la prédiction. Les algorithmes de machine Learning ont la capacité de traiter un grand nombre de prédicteurs, mais il est souvent nécessaire de sélectionner en amont les variables les plus pertinentes pour améliorer la performance du modèle et celle des métriques associées. Ainsi notre objectif à la fin de ce paragraphe est d'obtenir un ensemble de données que nous aurons modifié, normalisé ou standardisé de la forme: $D_N = \{(X_i, Y_i) \mid i = 1, \ldots, N\}$.

Remarque importante concernant nos données:

En ce qui concerne la matrice explicative X, il convient de noter que 70 variables peuvent être considérées comme un nombre important si l'on souhaite comprendre et interpréter les résultats de nos modèles. Un compromis doit être trouvé entre l'interprétation et la précision, car en théorie, plus nous avons de données, plus nous pouvons augmenter la précision de nos classifieurs à condition qu'ils apportent une nouvelle information. Dans ce mémoire, nous ne cherchons pas à trouver les meilleures variables explicatives (datamining), nous sommes contraints donc à augmenter la taille de notre ensemble de données en utilisant des transformations qui, nous l'espérons, amélioreront nos résultats. Ces transformations comprennent la variation des variables au cours du temps (les rendements), la différence entre la valeur actuelle d'une variable et celle d'il y a i périodes, ainsi que la tendance ou la saisonnalité de ces dernières. Ces nouvelles variables peuvent ajouter une dimension temporelle (saisonnalité) ou atténuer les effets indésirables (variation ou différence). Pour des raisons de clarté et de simplification, nous n'entrons pas ces nouvelles variables dans notre matrice explicative bien que nous ayons implémenté l'ajout de ces variables dans notre code source.

3.1 Traitement des données

3.1.1 Normalisation

Cette partie sera articulée autour de deux points. Tout d'abord, il sera question de normalisation des données par une normalisation MinMax, suivi de normalisation par standardisation.

Qu'est-ce que la mise à l'échelle ou scaling des données ?

Le scaling des données est une technique permettant de mettre les données à la même échelle afin d'améliorer en théorie les performances prédictives des algorithmes d'apprentissage automatique. Cette méthode de mise à l'échelle doit être appliquée sur l'ensemble d'entraînement pour éviter la surinterprétation des amplitudes relatives importantes. En pratique, on utilise le scaling pour accélérer la convergence de nos algorithmes comme la descente de gradient (utilisé par exemple dans la régression linéaire) et faciliter l'utilisation de techniques telles que l'analyse en composantes principales (ACP) qui se base sur la variance des données.

Scaling par normalisation MinMax

Tout d'abord, une approche courante pour améliorer les performances des modèles de machine Learning consiste à normaliser les données avec la méthode MinMax. La normalisation MinMax vise à mettre les valeurs des différentes colonnes des données à la même échelle. Pour effectuer cette normalisation, on peut utiliser des fonctions telles que MinMaxnorm du package Scikit-Learn. Cette fonction réalise une normalisation MinMax des données en réduisant les valeurs de chaque colonne dans l'intervalle [0, 1].
En pratique, le calcul de cette normalisation MinMax se fait de la manière suivante:

La normalisation MinMax peut être représentée par l'équation suivante :

$$x_t^{MinMax} = \frac{x_t - x_{Min}}{x_{Max} - x_{Min}} \tag{2}$$

avec:

- x_t représente la valeur de la variable à l'instant t
- x_{Min} représente la valeur minimale de la variable
- x_{Max} représente la valeur maximale de la variable.

Scaling par standardisation

Deuxièmement, une autre manière d'améliorer nos modèles est d'effectuer un travail de normalisation, cette fois-ci standardisation des données. La standardisation est une méthode de mise à l'échelle des données qui consiste à soustraire la moyenne de chaque variable (colonne) de nos données , puis à diviser par l'écart-type de chaque variable. Dans notre code Python, la fonction standardizationnorm utilise la classe StandardScaler de Scikit-Learn pour effectuer cette standardisation.

La standardisation est centrée sur la moyenne de la variable et utilise l'écart-type pour mettre à l'échelle les valeurs contrairement à la normalisation MinMax, qui ramène les valeurs d'une variable dans un intervalle spécifique. On utilise cette méthode de standardisation car elle est très utile lorsque nos variables ont des échelles différentes (ce qui est le cas) et permet de les rendre plus comparables pour faciliter leur traitement ultérieur, notamment dans nos modèles d'apprentissage automatique.

La formule de la standardisation est la suivante :

$$x_t^{Norm} = \frac{x_t - \mu}{\sigma} \tag{3}$$

avec:

- x_t représente la valeur de la variable x à l'observation t dans l'ensemble de données.
- μ est la moyenne de la variable x sur l'ensemble de données
- σ est l'écart-type de la variable x sur l'ensemble de données.

Pour des raisons de concisions, nous utiliserons cette dernière méthode de normalisation par standardisation et non la MinMax. En pratique, les deux méthodes donnent des performances similaires.

Visualisation de nos variables explicatives après standardisation

Voici les résultats obtenus avant et après la standardisation sur nos variables explicatives. La mise à l'échelle des variables explicatives est bien représentée.

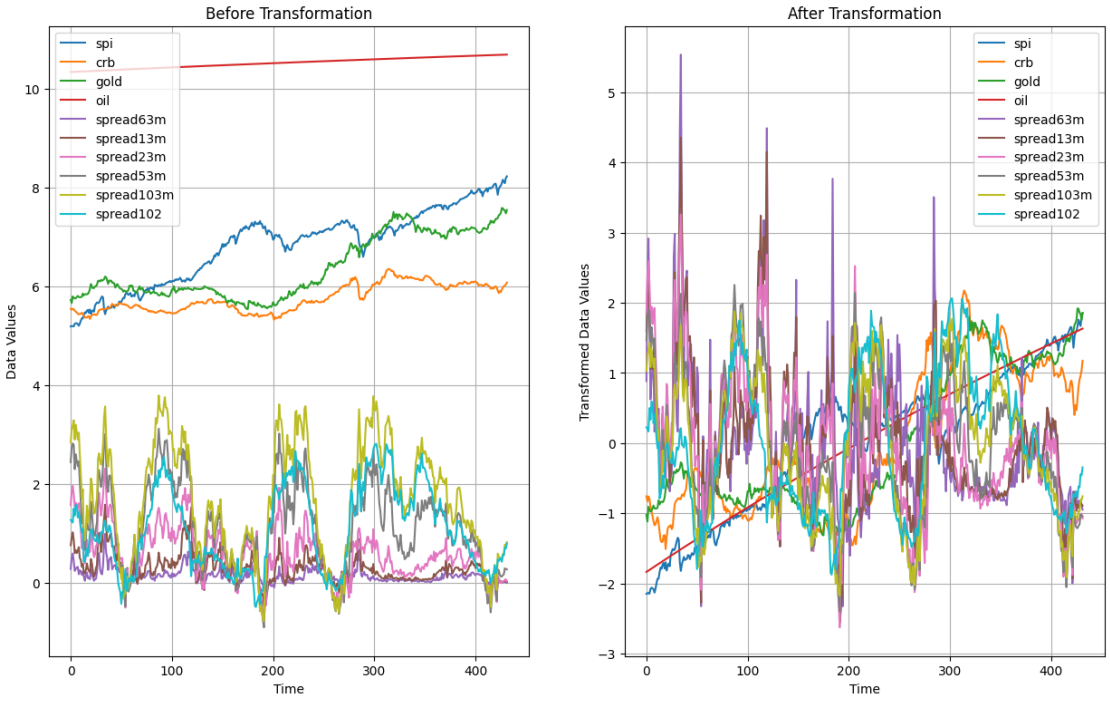


Figure 5: Effet de la standardisation sur les valeurs de nos variables explicatives

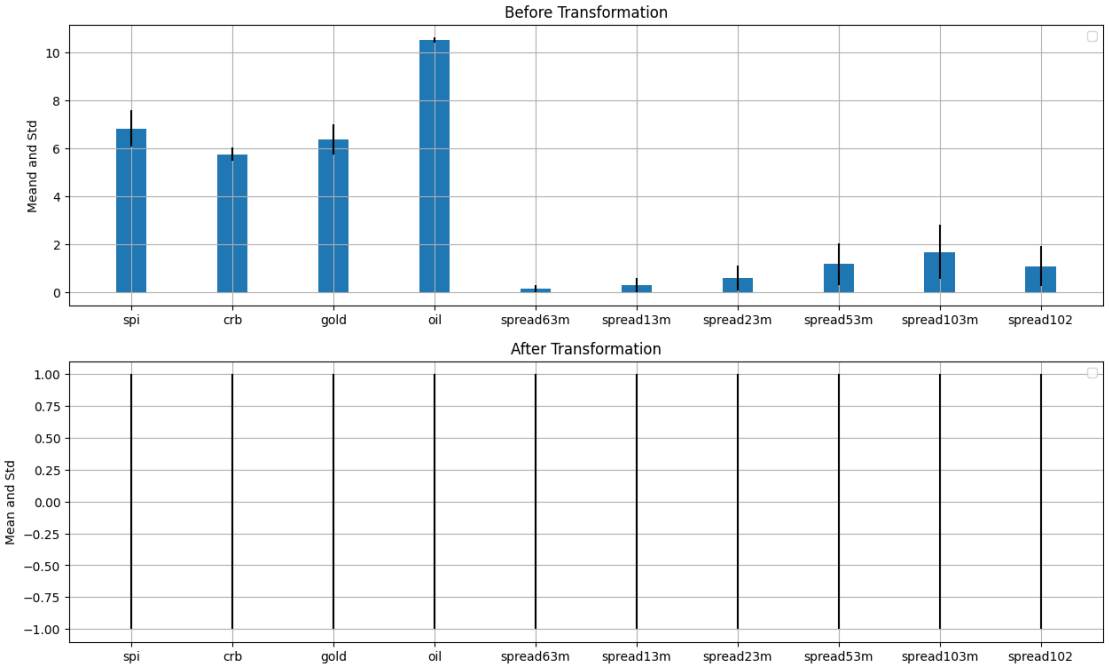


Figure 6: Effet sur la moyenne et l'écart type de nos variables

Résultats obtenus après normalisation par standardisation

Pour conclure sur ce paragraphe, présentons les prédictions après standardisation, sous forme de matrice de confusion pour les trois classifieurs.

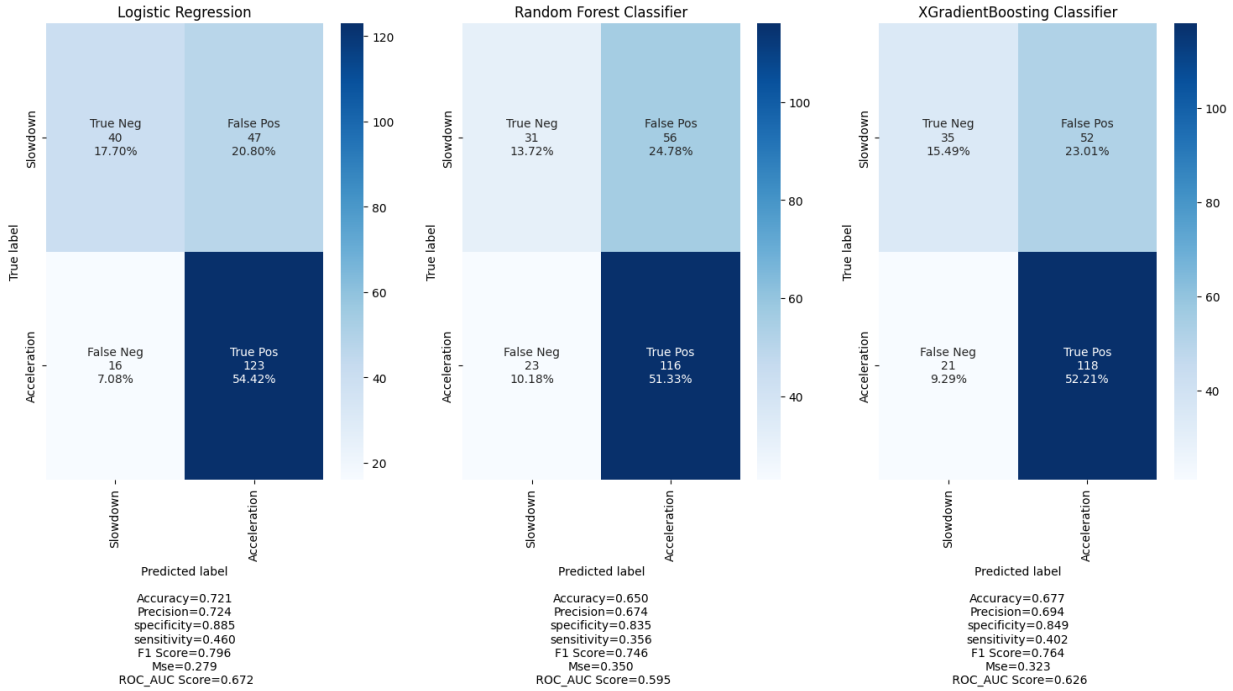


Figure 7: Matrices de confusion avec données brutes

En comparant la valeur des différentes métriques et coefficients de précision entre le graphique ci-dessus et ci-dessous, on constate dans le graphique ci-dessous, une nette amélioration pour le classifieur Logit. En ce qui concerne RF et XGB, ces derniers semblent insensibles à la normalisation par standardisation. Nous conservons cette méthode de standardisation des données.

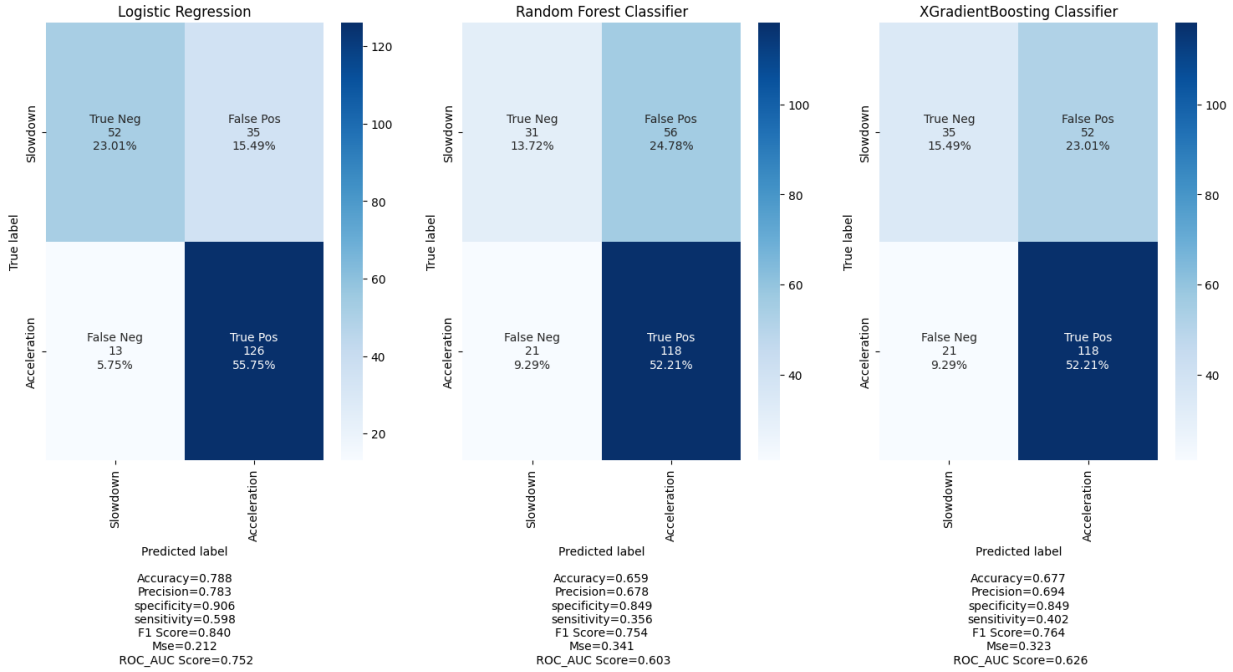


Figure 8: Matrices de confusion avec données normalisées par standardisation

3.1.2 Réduction de la dimension

Il est important de souligner que tout ce paragraphe tourne autour de l'utilité ou non d'une ACP dans nos modèles de prédiction. En effet on cherchera à, dans un premier temps présenter le but théorique de la réduction de la dimentionnalité. Dans un second temps, on adoptera une démarche critique sur nos résultats obtenus après ACP pour conclure que réduire la dimentionnalité alors que nous n'avons que 70 variables explicatives n'est pas un choix pertinent.

L'intérêt théorique de l'ACP

Une autre façon d'améliorer nos méthodes est de réduire la dimensionnalité en utilisant des méthodes d'analyse par composantes principales. Il est important de souligner que l'ACP est une méthode de réduction de dimension qui permet de réduire le nombre de variables tout en conservant le maximum de variabilité entre les vecteurs.

En quoi la présence de multicolinéarité entre les variables peut-elle poser un risque pour un modèle de prédiction en machine learning ?

La multicolinéarité dans le data train d'un modèle de prédiction en machine Learning peut être dangereuse car elle peut causer une instabilité et une incertitude dans les coefficients des variables. Donnons un exemple: lorsque deux ou plusieurs variables sont fortement corrélées, il devient difficile de déterminer la contribution individuelle de chaque variable à la prédiction finale, car les coefficients peuvent être très sensibles aux petites variations des données d'entrée. De plus, la présence de multicolinéarité peut poser un risque dans la mesure où elle peut rendre les coefficients de régression instables, ce qui signifie qu'ils peuvent avoir des valeurs extrêmes

et donc non représentatives du véritable effet d'une variable donnée sur la prédiction. Cela peut conduire à des prédictions erronées et à des performances mauvaises du modèle.

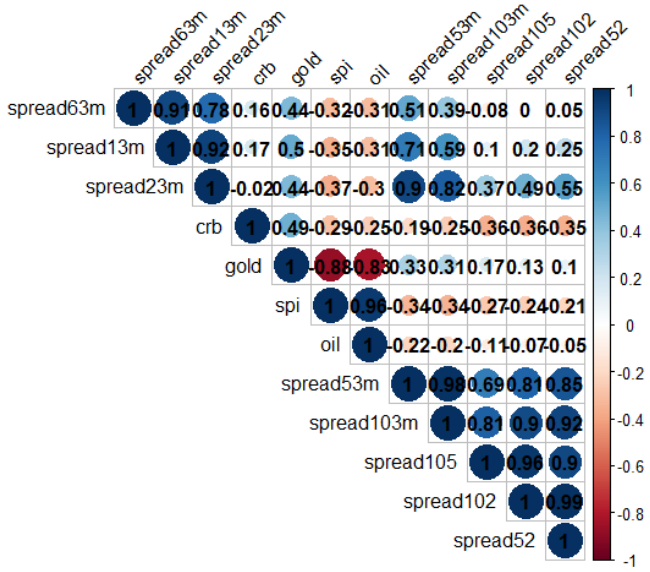


Figure 9: Corplot sur certaines variables explicatives

Ce corplot met en avant la présence de nombreuses variables fortement corrélées au sein de notre ensemble de données. Ainsi en pratique, dans notre code, nous utilisons la fonction PCA qui effectue une analyse en composantes principales (ACP) sur la matrice de design.

Paranthèse code : Fonctionnement de la fonction PCA (Sklearn)

L'idée sous-jacente de cette fonction est d'obtenir les variables qui ont la plus grande contribution (en valeur absolue) dans la construction des composantes principales. Elles sont considérées comme les plus importantes. Pour plus de details, dans notre code Python, cette fonction PCA renvoit donc une liste contenant les noms des variables les plus importantes selon la projection sur les axes principaux, qui sont stockés dans l'attribut components de l'objet PCA. Cette liste est triée par de manière décroissante en fonction de l'importance des variables (déterminée par la valeur absolue des coefficients de chaque variable dans chaque composante principale).

De très mauvais résultats obtenus après réduction de la dimension

Pour conclure sur la partie consacrée à la sélection de variables par ACP, mettons en avant le fait que les résultats obtenus après la sélection des variables sont très mauvais. **On conclut que dans le cadre de notre étude l'ACP n'est pas utile.**

En effet on remarque que nos modèles deviennent moins précis lorsque nous effectuons une ACP. Ceci nous permet d'écarter ces méthodes de réduction de la dimentionnalité en raison d'un très faible nombre de variables explicatives.

Jusqu'ici, nous avons présenté pour le moment trois méthodes qui en théorie devraient améliorer les prédictions : la normalisation MinMax, la standardisation des données et la réduction de la dimension. En combinant les effets positifs de ces méthodes, nous avons cherché à améliorer la précision de nos modèles de prédiction. En effet, la normalisation et la standardisation permettent de mettre les données à la même échelle et de les rendre plus comparables, tandis que la réduction de la dimension permet en théorie seulement de réduire le nombre de variables sans perdre d'informations importantes. En les combinant, nous obtenons des résultats plus fiables et plus précis dans nos prédictions.

3.1.3 Rééchantillonnage des données

Avant d'expliquer les méthodes de rééchantillonnage de données que nous avons utilisées dans notre mémoire, il est essentiel de comprendre les concepts théoriques sous-jacents. Le rééchantillonnage des données permet de créer de nouveaux échantillons de données et ce a partir des sous-ensembles aléatoires de l'ensemble de données brutes. En théorie, le rééchantilonage peut améliorer les performances des algorithmes de prédiction en permettant une utilisation plus efficace des données d'apprentissage.

Maintenant parlons du surapprentissage (ou overfitting en anglais) qui est un problème courant en apprentissage automatique: on dit qu'il y a overfitting quand un modèle est trop complexe et/ou s'adapte trop similairement aux données d'apprentissage, et ce, au point de perdre sa capacité à généraliser et à prédire de nouveaux exemples ce qui est très handicapant pour un algorithme de prédiction. Cela peut se produire lorsque le modèle est trop complexe par rapport à la quantité de données disponibles, ou lorsque les données d'apprentissage sont trop bruyantes ou comportent des biais.

Le rééchantillonnage des données peut aider à prévenir l'overfitting en permettant une utilisation plus efficace des données d'apprentissage à notre disposition. En pratique, la création de nouveaux échantillons à partir du dataset fournit au modèle plus de données d'apprentissage (et ce sans recourir à la collecte de nouvelles données). Cela peut aussi aider à équilibrer la distribution des classes (ce qui est notre cas dans notre mémoire) et à réduire le risque de surapprentissage en réduisant la variance du modèle comme dit précedemment. En outre, la réduction de la variance peut aider à améliorer la précision des prévisions pour les nouvelles données.

En pratique, la décision de rééchantillonner ou non les données dépend de plusieurs facteurs, tels que la qualité des données ou encore la taille de l'ensemble de données. Dans notre mémoire, puisque l'ensemble de données est relativement petit (moins de 70 variables explicatives) et que nos données ne sont pas équilibrées car il y a plus de 1 que de 0 dans notre variable Y (voir graphique ci-dessous), on va donc rééchantillonner les données pour équilibrer la répartition des classes et augmenter la taille de l'ensemble de données.

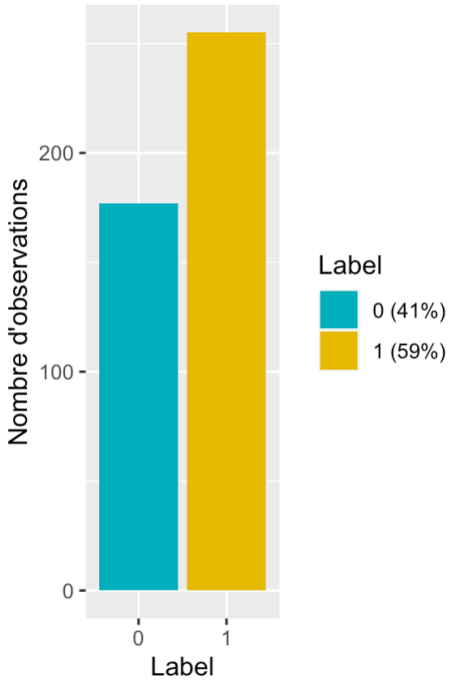


Figure 10: Une surreprésentation de la valeur 1 pour la variable d'interet Y

Paranthèse code : Méthode Resampledataframe() et son rapport avec l'oversampling

Dans notre code, on utilise la fonction Resampledataframe(). Cette dernière sélectionne un nombre égal d'échantillons de la classe minoritaire (ici 0) et échantillonne avec remplacement à partir de la classe majoritaire. Resampledataframe() permet donc de créer un ensemble de données équilibré en termes de classes en utilisant la technique d'oversampling.

En ce qui concerne l'oversampling, il consiste dans les grandes lignes à augmenter le nombre d'observations dans la classe minoritaire d'un ensemble de données déséquilibré. Notons que l'oversampling peut augmenter le risque de surapprentissage dans certains cas, c'est pourquoi on cherchera à valider les résultats obtenus à l'aide de techniques telles que la validation croisée en page 28.

D'après ces matrices de confusions ci-dessous, on remarque une amélioration pour le XGB. Nous maintenons le rééchantillonage uniquement pour ce dernier.

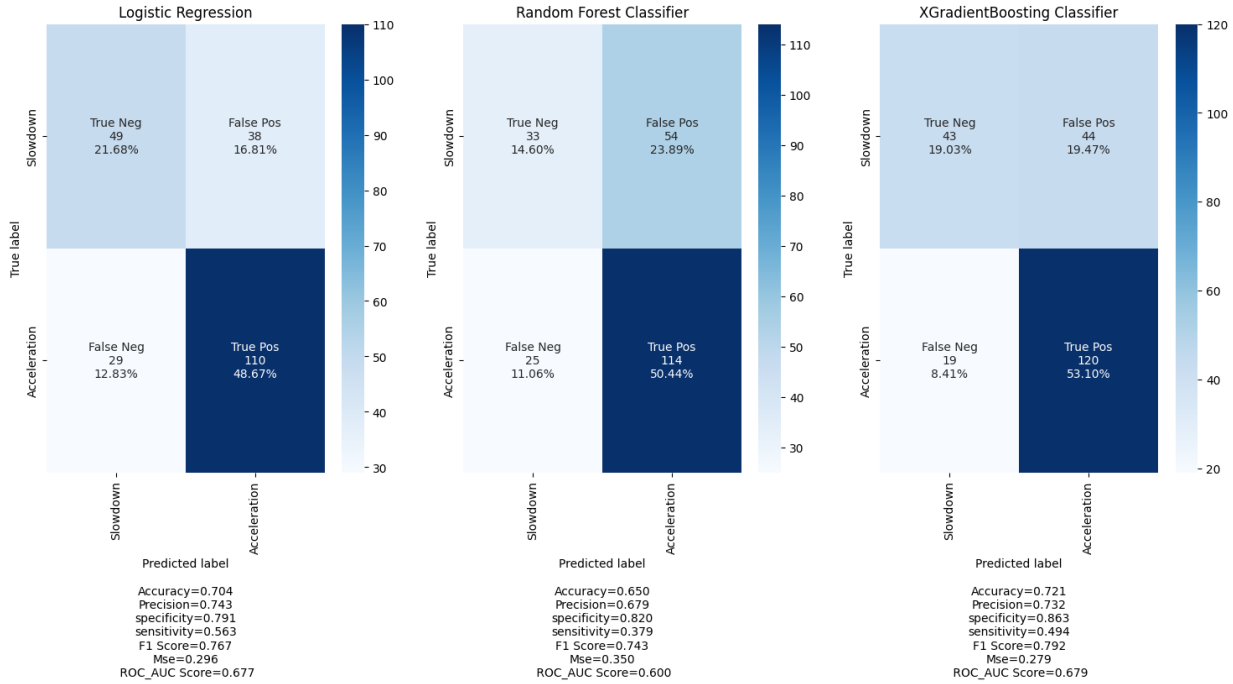


Figure 11: Matrices de confusion pour les classifieurs combinant standardisation des données et rééchantillonnage

Pour clore ce paragraphe, il convient de noter qu'il existe de nombreuses autres méthodes de rééchantillonnage de données, telles que le rééquilibrage avec undersampling. Cela consiste aussi à réduire la taille de l'échantillon de la classe majoritaire pour qu'elle soit plus proche de la taille de la classe minoritaire (idéalement dans notre cas un 50-50). L'undersampling est souvent utilisée en combinaison avec l'oversampling de la classe minoritaire pour obtenir ce nouvel ensemble de données qui est maintenant équilibré et éviter le surapprentissage. Cependant, notons que ces dernières méthodes ne seront pas abordées dans le cadre de notre mémoire.

3.2 Optimisation des paramétres

3.2.1 Optimisation du seuil de décision

L'optimisation du seuil de décision est une étape cruciale en classification binaire. Par défaut, le seuil est fixé à 0,5, ce qui signifie que toutes les prédictions avec une probabilité supérieure ou égale à 0,5 seront classifiés en 1. Cependant, il est souvent pertinent de modifier ce seuil par défaut en fonction des besoins du projet.

Dans le cadre de notre mémoire, il est important de modifier le seuil par défaut car notre variable cible est déséquilibrée, avec plus de 1 que de 0. En conséquence, nous devons ajuster le seuil pour éviter de biaiser nos prédictions.

En effet, si l'on prend comme métrique la précision (définie ci-dessous), notre modèle de prédiction peut donner plus de faux négatifs (c'est à dire prédire une classe 0 à la place d'une classe 1) que de faux positifs (c'est à dire prédire une classe 1 à la place d'une classe 0), car il est plus sûr de prédire la classe majoritaire (dans notre étude c'est la classe 1, c'est à dire être en période d'expansion). On obtiendrait donc une précision plus élevée pour la classe majoritaire, mais une précision plus faible pour la classe minoritaire, ce qui entraînera un fort biais dans nos prédictions. Heureusement, on peut améliorer l'équilibre entre les classes et éviter tout biais dans les prédictions en ajustant le seuil pour augmenter le nombre de vrais positifs et minimiser le nombre de faux positifs.

En pratique, pour optimiser le seuil de décision, nous devons choisir des métriques adaptées à notre problème de classification binaire.

Tout d'abord, la première métrique que nous cherchons à maximiser lors de l'optimisation du seuil est le F1 Score.

Qu'est ce que le F1 Score ?

C'est une métrique de performance comprise entre 0 et 1 utilisée pour évaluer la qualité d'un modèle de classification. Un F1-score de 1 indique une classification parfaite et un F1-score de 0 indique une classification totalement fausse.

$$F_1 Score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{recall} + \text{precision}}$$

où :

- precision : la proportion de vrais positifs parmi toutes les prédictions positives du modèle
$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
- recall : la proportion de vrais positifs qui ont été correctement identifiés parmi tous les vrais positifs
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Deux outils très utiles pour ce type de problème sont la courbe ROC et la courbe Pecision-Recall.

Une courbe ROC est une représentation graphique qui évalue un ensemble de prédictions de probabilités faites par un modèle sur un ensemble de test.

Différents seuils sont utilisés pour interpréter le taux de vrais positifs et le taux de faux positifs des prédictions de la classe minoritaire, et les scores sont tracés sur une ligne de seuils croissants pour créer une courbe.

Comme le montre la courbe ROC (Receiver Operating Characteristic) ci-dessous, le taux FP de faux positifs est tracé sur l'axe x et le taux de vrais positifs est tracé sur l'axe y. Une ligne diagonale sur le graphique, $\mathcal{D} : y = x$ indique la courbe d'un classificateur qui prédit la classe majoritaire dans tous les cas, et un point dans le coin supérieur gauche du graphique indique un modèle ayant une compétence parfaite.

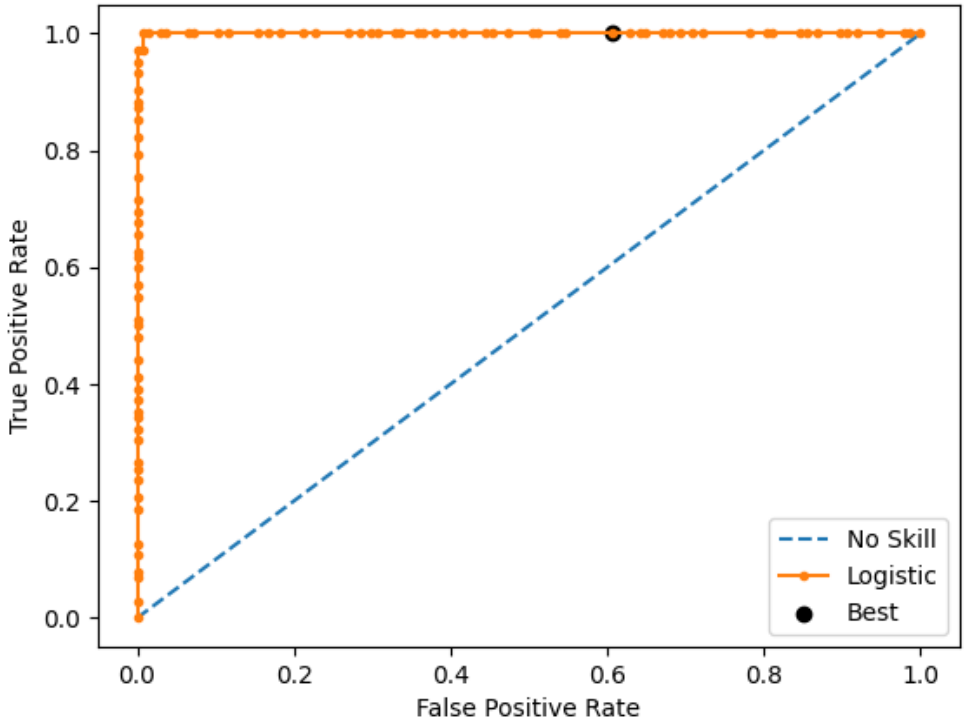


Figure 12: Courbe ROC sur le modèle Logit

Cette courbe est très utile pour comprendre le compromis entre le taux de vrais positifs et le taux de faux positifs pour différents seuils donnés. L'aire sous la courbe ROC, appelée AUROC, fournit un seul pour résumer la performance d'un modèle en terme de sa courbe ROC, avec une valeur entre 0,5 (aucune compétence/ No skill) et 1,0 (compétence parfaite).

Pour ce qui est de la courbe Precision-Recall, elle diffère de la courbe ROC par le fait qu'elle s'intéresse à la performance du classifieur sur la classe minoritaire seulement.

Une courbe de précision-Recall est calculée en créant des étiquettes de classe nettes pour les prédictions de probabilité sur un ensemble de seuils et en calculant la précision puis le rappel pour chacun des seuils. Un graphique en ligne est créé pour les seuils dans l'ordre croissant avec le rappel sur l'axe des x et la précision sur l'axe des y.

Un modèle dit sans compétence est représenté par une ligne horizontale avec une précision qui est le rapport d'exemples positifs dans l'ensemble de données (par exemple TP / (TP + TN)). Un classificateur avec une compétence parfaite a une précision et un rappel complets avec un point dans le coin supérieur droit.

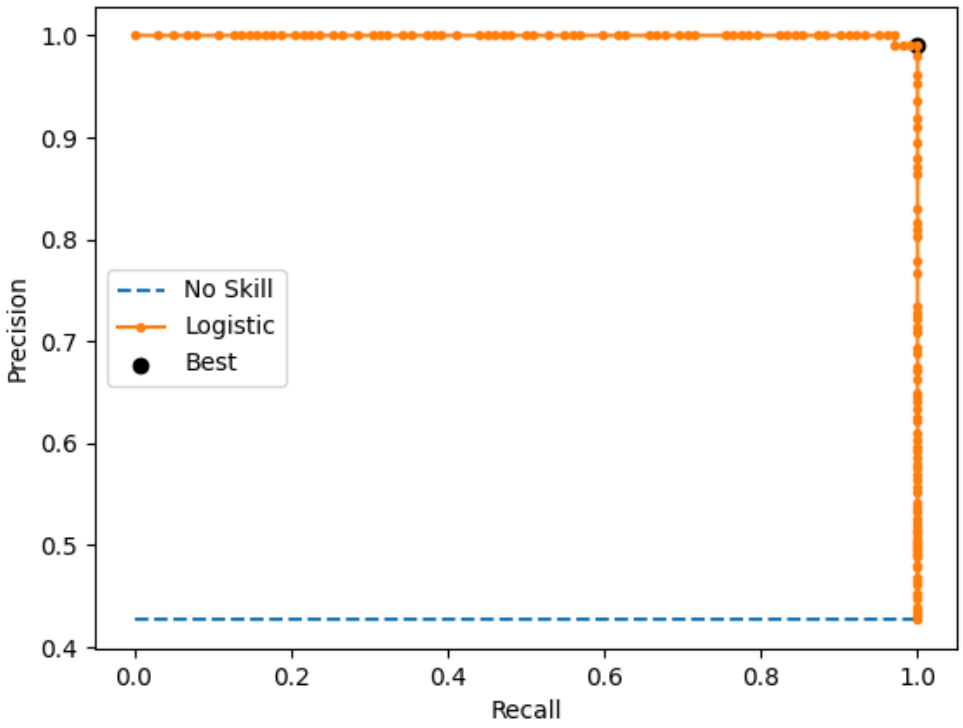


Figure 13: Courbe Précision-Rappel sur le modèle Logit

En pratique, nous définissons un ensemble discret de seuil allant de 0 à 1. Sachant toute l'information passée, nous évaluons les métriques choisies en tous ces points pour en déduire le seuil qui maximise (ou minimise) la métrique.

Toutefois, modifier ce seuil n'a été concluant pour aucun des classifieurs comme le montrent les courbes présentées ci-dessus. Nos algorithmes ont du mal à trouver le meilleur seuil sans nécessairement impliquer un surapprentissage ou sous-apprentissage. Il n'est donc pas pertinent pour notre problème en particulier d'intégrer cette méthode.

Voici le résultat pour le RF par exemple : On voit que cela améliore certaines métriques au détriment d'autres. Nous conservons donc le seuil de décision par défaut 0.5.

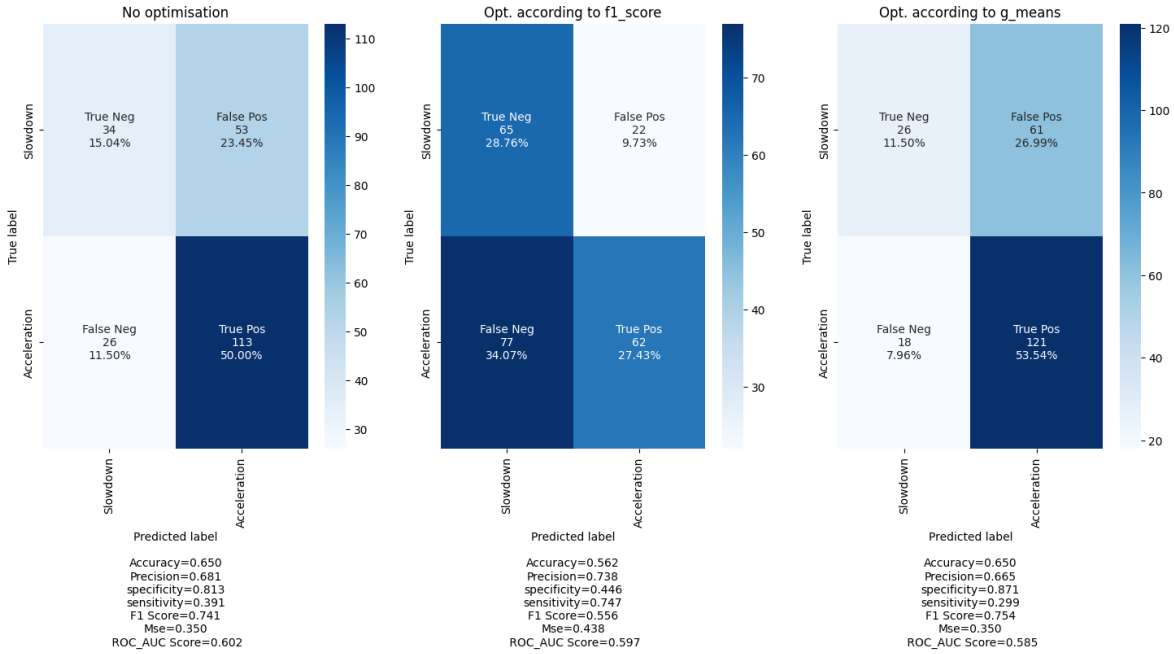


Figure 14: Matrices de confusion aprés optimisation du seuil pour le RF

3.2.2 Optimisation des hyperparamètres

Le paragraphe théorique sur la présentation de nos modèles en page 7 a introduit la notion d'hyperparamètres dans les modèles de prédiction et nous a permis de comprendre leur rôle. Nous allons maintenant nous intéresser à l'optimisation de ces hyperparamètres avant l'entraînement des modèles afin d'améliorer les performances des modèles de prédiction.

Tout d'abord qu'est-ce qu'on entend par optimisation des hyperparamètres en machine Learning ?

Premièrement, l'optimisation des hyperparamètres dans les algorithmes de machine Learning consiste à trouver les valeurs optimales des hyperparamètres d'un modèle qui permettent d'obtenir les meilleures performances de prédictions sur un ensemble de données donné. L'optimisation des hyperparamètres est importante car elle permet d'obtenir des modèles plus robustes et plus performants. (Hastie, Tibshirani Friedman 2007)

Qu'est ce que la validation croisée? Quel est son rapport avec l'optimisation des hyperparamètres ?

Ce même ouvrage Hastie, Tibshirani Friedman 2007 met en avant le fait que la validation croisée est une technique d'évaluation de la performance d'un modèle de prédiction. Concrètement, la validation croisée consiste à diviser l'ensemble de données en plusieurs sous-ensembles, à entraîner le modèle sur certains de ces sous-ensembles et à l'évaluer sur d'autres. Cette procédure est répétée plusieurs fois pour permettre une évaluation de la performance du modèle. La validation croisée permet aussi de sélectionner les meilleurs hyperparamètres et éviter le surapprentissage.

Quelles sont les différentes méthodes de validation croisée ?

La technique la plus répandue est la validation croisée k-fold (nous reviendrons quelques lignes plus bas sur le fonctionnement et le choix de cette technique). Néanmoins il existe plusieurs méthodes de validation croisée qui peuvent être utilisées pour évaluer la performance des modèles de prédiction.

Premièrement la validation croisée Leave-one-out est similaire à la validation croisée k-fold, si ce n'est qu'elle ne laisse qu'un seul échantillon de données pour l'ensemble de validation à chaque fois, et utilise les autres échantillons pour l'ensemble d'entraînement.

Quant à la validation croisée shuffle-split (Miika Nieminen, Juho Rousu et Samuel Kaski BMC Bioinformatics 2007), cette méthode consiste à diviser l'ensemble de données en un nombre spécifié de fois, mais à chaque fois de manière aléatoire. Cette méthode peut être utile lorsque l'ensemble de données est très grand.

Pour finir, on peut aussi évoquer la validation croisée dite stratifiée, elle est utilisée lorsque les classes de sortie ne sont pas équitablement représentées dans l'ensemble de données. Elle consiste à diviser l'ensemble de données de manière que chaque ensemble de validation contienne une proportion égale de chaque classe.

Nous ne nous attarderons pas sur les avantages et les inconvénients de chaque méthode de validation croisée, car chaque méthode présente ses propres forces et faiblesses.

Dans notre mémoire, on met l'accent sur la validation croisée par k-folds en utilisant GridSearchCV

Nous avons donc choisi d'utiliser la méthode k-folds de la validation croisée en utilisant la fonction GridSearchCV, car cette méthode est recommandée par Hastie et ses co-auteurs (Hastie, Tibshirani Friedman 2007) pour évaluer la performance des modèles. Ces derniers mettent en avant le fait que la validation croisée k-fold est méthode technique robuste et fiable pour estimer l'erreur de généralisation du modèle, et qu'elle est plus précise que la séparation simple des données en ensembles d'entraînement et de validation. Nous avons donc opté pour cette méthode pour obtenir des résultats plus fiables et précis.

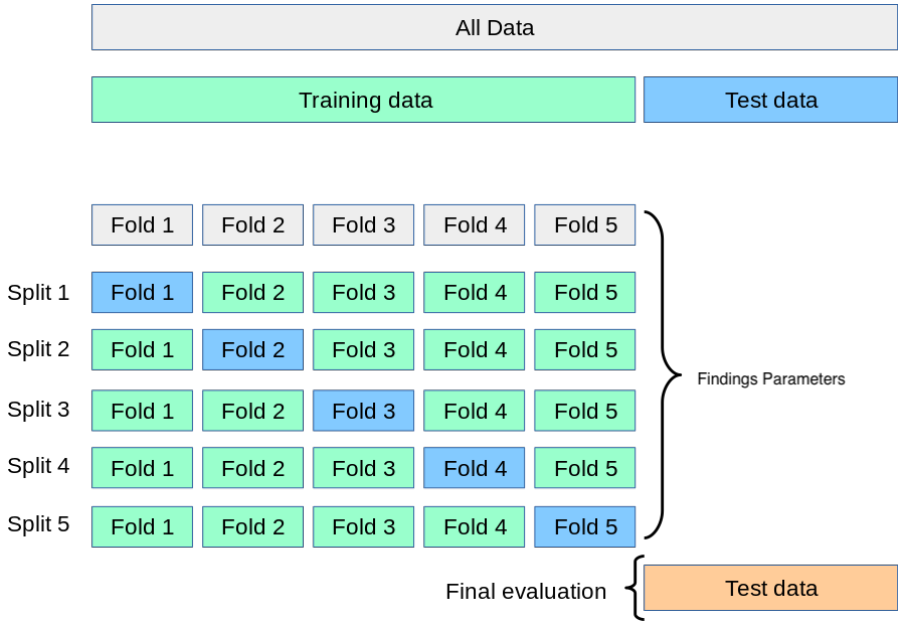


Figure 15: Fonctionnement de K-folds pour K=5 (par Sickitlearn.org)

Mais qu'est-ce que la méthode k-folds ?

Comme le montre le schéma ci-dessous, elle consiste à diviser l'ensemble de données en k sous-ensembles (dit folds en anglais) qui sont de taille égale. Le modèle de prédiction est ensuite entraîné sur k-1 sous-ensembles (c'est l'ensemble d'entraînement) et évalué sur ce sous-ensemble restant (c'est l'ensemble de validation). On répète cela k-fois, en alternant le choix de l'unique sous ensemble de validation. Pour finir on calcule la performance du modèle: c'est la moyenne des performances obtenues sur chaque différents ensembles de validation. Pour chaque évaluation, la performance du modèle est mesurée en utilisant une métrique de performance appropriée choisie au préalable (telles que l'accuracy, l'AUROC etc.).

Paranthèse code : Fonctionnement de GridSearchCV() et son rapport k-fold

En ce qui concerne le fonctionnement de la fonction GridSearchCV elle fonctionne en explorant un espace de recherche d'hyperparamètres pour trouver la combinaison optimale d'hyperparamètres pour un modèle de prédiction donné. Pour chaque combinaison d'hyperparamètres, la fonction GridSearchCV utilise la méthode de validation croisée k-fold pour évaluer la performance du modèle. Nous pouvons résumer le fonctionnement de GridSearchCV par le schéma suivant.

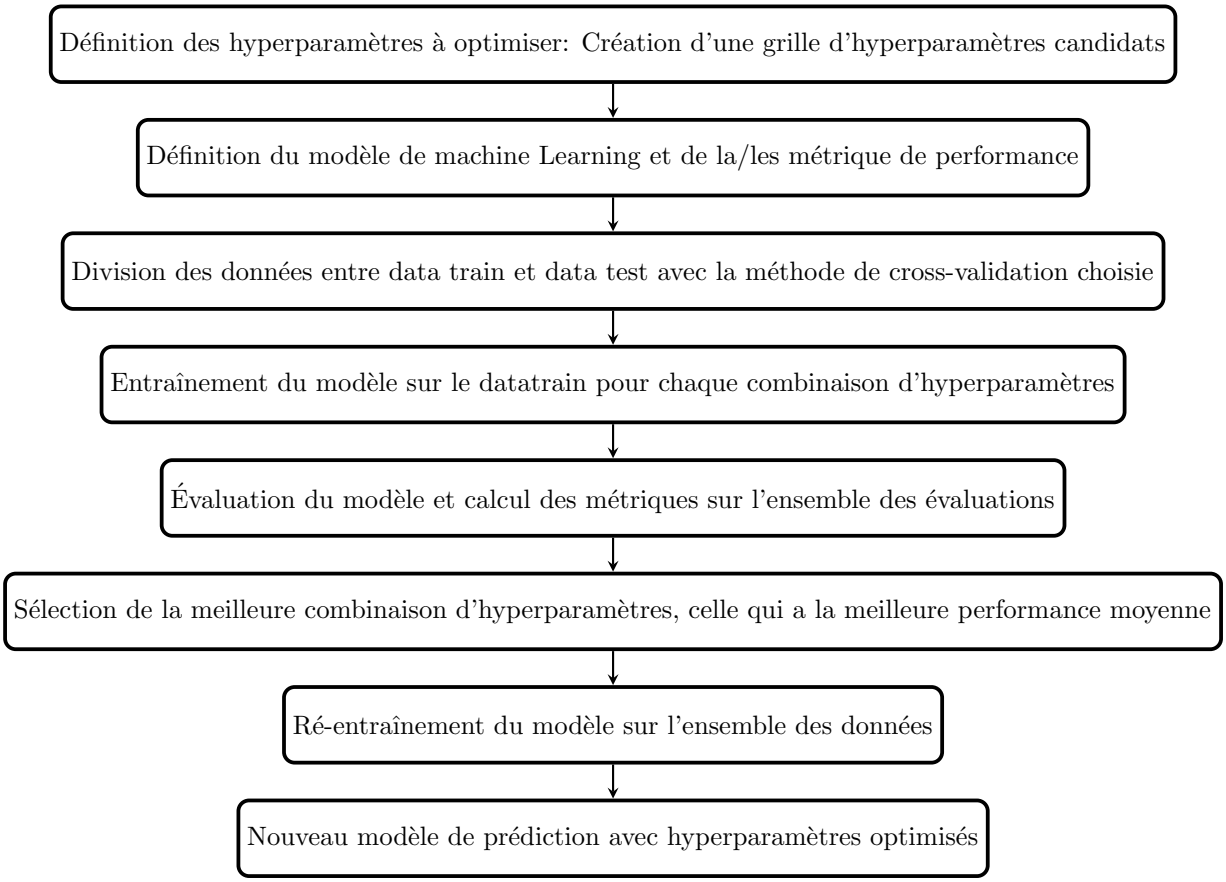


Figure 16: Schéma du fonctionnement de GridSearchCSV

Résultats obtenus avant et après utilisation de la cross-validation

Ainsi, en utilisant GridSearchCV sur nos différents algorithmes, nous obtenons des résultats qui soulignent l'importance de l'optimisation des hyperparamètres par validation croisée k-fold sur un faible échantillon. La complexité algorithmique de cette méthode dans le cadre de notre approche est chronophage. Bien qu'implémentée dans notre code, nous ne l'utiliserons pas.

3.3 Imputation des données manquantes

Comme nous l'avons déjà mentionné, nous supposons que nous ne connaissons pas les 12 dernières vraies valeurs de Y.

Le problème est que notre variable cible présente une autocorrélation qui décroît en fonction du temps. Cette décroissance atteint zéro au lag 12. Cela signifie que l'information la plus récente

ne donne que peu ou pas d'indication sur l'état actuel de l'économie. En raison de l'absence des vraies étiquettes Y, nous perdons également les variables explicatives associées. Cela peut considérablement dégrader la performance de nos modèles, car l'incertitude associée aux variables manquantes peut biaiser les prédictions et fausser les résultats.

Comment contourner ce problème ?

Pour contourner ce problème et faire usage de toutes les données à notre disposition au temps de la prédiction, nous avons recours à l'imputation des données manquantes en utilisant deux méthodes : une méthode naïve et déterministe et une autre statisticienne.

Cependant, en procédant ainsi, nous prenons le risque de donner de mauvaises informations au classifieur. De plus, si nous observons une dégradation de la performance, il sera difficile de savoir si elle est inhérente au classifieur ou si elle est due à l'estimation des variables manquantes Y. Toutefois, puisque nous mettons à jour le modèle à chaque prédiction, ce cumul d'erreurs potentiel s'effacera avec le temps.

Notez que nous allons donner quelques pistes sur la manière de quantifier le risque que nous prenons et l'incertitude que cette méthode engendre. En utilisant ces techniques, nous espérons mieux comprendre l'impact de l'imputation des données manquantes sur la performance de nos modèles et prendre des décisions plus éclairées quant à la manière de traiter ces données manquantes.

3.3.1 Méthode naive

La méthode naïve de l'imputation des données manquantes consiste simplement à remplacer les données manquantes par la dernière observation disponible de la variable cible. Cette méthode est peu convaincante dans notre cas car notre variable cible présente un problème d'autocorrélation qui diminue au fil du temps et passe par zéro à un certain lag, ce qui signifie que la dernière observation disponible ne reflète pas nécessairement l'état actuel de l'économie.

3.3.2 Estimation itérative

La méthode d'estimation itérative des données manquantes consiste à remplacer les valeurs manquantes par des prédictions itératives en utilisant le modèle initial à chaque étape. Cela signifie que l'on commence par prédire les valeurs manquantes à partir de la dernière observation disponible, puis on utilise ces prédictions pour prédire les valeurs manquantes suivantes jusqu'à ce qu'on atteigne le temps voulu.

Cependant, cette méthode comporte un risque, car elle peut entraîner une dégradation de la performance du modèle. Pour quantifier ce risque, on peut calculer la précision des prédictions des Y manquants lorsqu'on a connaissance des vrais Y, et tracer une courbe de précision des 11 derniers Y manquants et nos prédictions en fonction du temps.

La courbe de la figure 17 confirme notre intuition: moins bonne est la prédiction des données manquantes moins bonne est l'estimation du label Y associé au temps présent. Regardons par exemple la phase de ralentissement à la fin de l'année 2012. Le modèle ne prédit pas correctement les labels manquants (chute de la précision de la courbe orange), par conséquent les estimations de notre variable Y au temps présent (courbe en bleue) sont fausses, vraisemblablement à cause de la chute de la précision et non pas à cause de la capacité du modèle à prédire ce point en particulier.

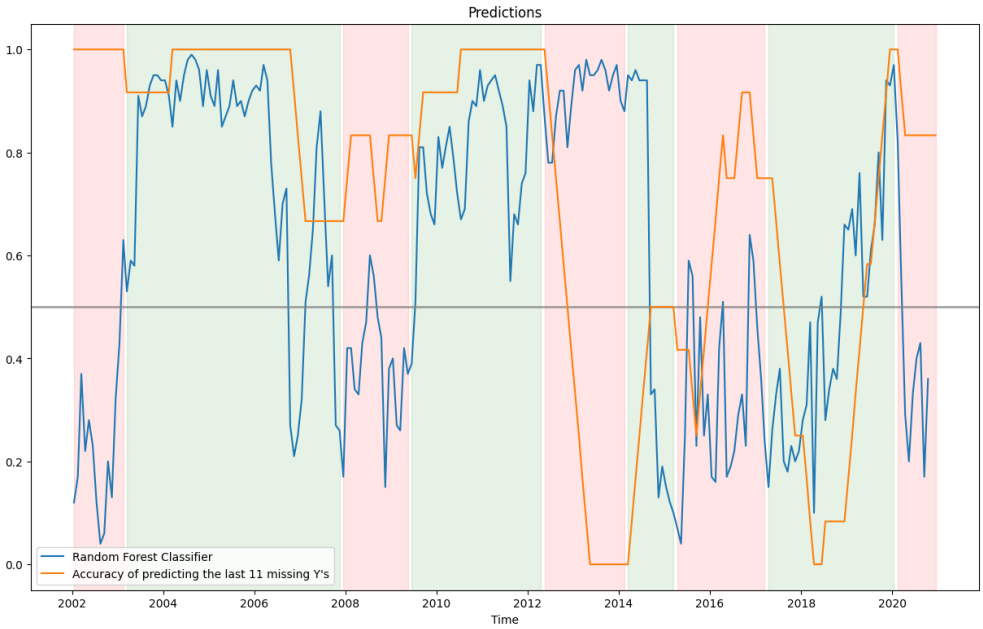


Figure 17: Lien entre précision de la prédiction des données manquantes et précision de la prédiction du temps présent

Analysons maintenant les résultats obtenus avec cette méthode d'estimation itérative.
L'explication de résultats obtenus a déjà été fournie un peu plus en haut.

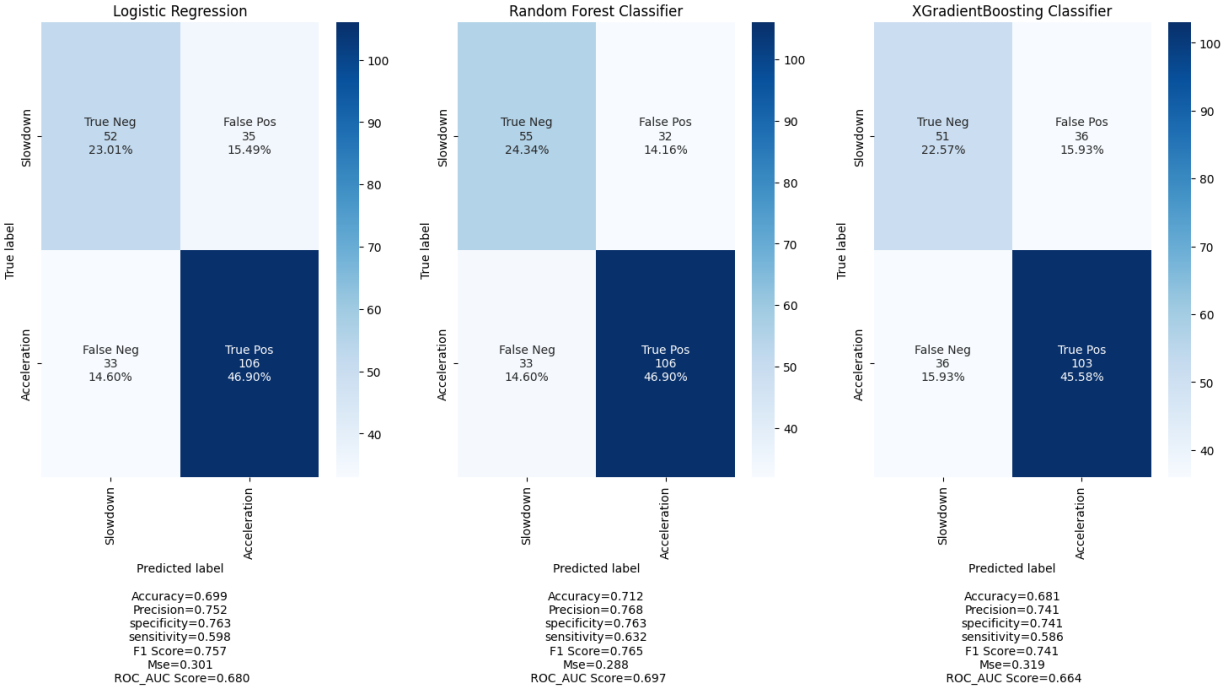


Figure 18: Matrices de confusion obtenues après imputation

D'après ces matrices de confusions et les métriques, nous conservons uniquement cette méthode pour le RF.

4 Implémentation des portefeuilles

La section actuelle consiste à confronter nos résultats et nos prédictions théoriques des points de retournement du cycle économique avec une application pratique de ces algorithmes.

Comme cela a été démontré dans des études antérieures telles que Cenesizoglu et Timmermann en 2012 ou Brown en 2008, il est possible qu'un écart plus ou moins grand se produise entre la prédictibilité économétrique et la rentabilité réelle. Pour évaluer l'importance économique de ces modèles, des stratégies de négociation hypothétiques très simplifiées ont été mises en place.

Les stratégies actives d'investissement qui s'appuient sur les régimes économiques identifiés par les modèles de prédiction ont le potentiel de générer des profits importants en théorie. En effet, ces stratégies dynamiques devraient être capables de réduire les retraits potentiels en tirant parti des régimes économiques positifs et en gérant les régimes économiques défavorables.

Nous allons aborder deux stratégies distinctes dans notre analyse : la stratégie Equity et la stratégie dynamique, que nous détaillerons ultérieurement. Ces deux stratégies d'investissement ont été simplifiées autant que possible pour nous permettre de concentrer notre attention sur la concurrence des modèles. Nous allons comparer les performances des deux stratégies actives (Equity et dynamique) à celles de la stratégie passive de dite du "buy-and-hold", qui est considérée comme le benchmark. Dans notre étude, l'actif risqué est l'indice SP 500 et l'actif non-risqué le bon du trésor (Treasury bill) américain d'une échéance de 13 semaines. C'est une obligation à court terme émise par le gouvernement américain.

4.1 Application des prédictions au portefeuille Equity

Présentons tout d'abord la nature du portefeuille Equity :

Nous étudions cette stratégie d'allocation d'actifs 120/80 en actions. Elle implique qu'un gestionnaire de portefeuille d'actions investisse 100 $ (ou euros) le 1er janvier 2002. Chaque mois, cet investisseur hypothétique décide de la fraction de sa richesse à investir en fonction de l'état actuel de l'économie prédit par le modèle de prédiction. Si ce dernier indique une période d'accélération, l'investisseur peut augmenter la valeur de son portefeuille (en investissant 120% de sa richesse en actifs et empruntant 20% en espèces), sinon, il n'investit que 80% de sa richesse et garde 20% en espèces. Cette stratégie est appelée allocation d'actifs statique ou stratégie Equity.

4.2 Application des prédictions au portefeuille dynamique

Présentons maintenant la nature du portefeuille dynamique :

Nous examinons dans un second temps également une allocation d'actifs dynamique, car les différentes classes d'actifs ont des comportements différents à différentes étapes des cycles économiques. Il peut être judicieux de rééquilibrer le portefeuille en modifiant les pondérations en fonction de l'étape du cycle de croissance (Raffinot, 2017). Cette stratégie dynamique vise à surpasser l'allocation d'actifs classique pour un portefeuille institutionnel allouant 60% du portefeuille aux actions et 40% aux titres à revenu fixe (obligations). Comme pour la stratégie Equity, l'investisseur décide chaque mois de rééquilibrer son portefeuille en fonction des

prévisions du modèle. Si le modèle indique une période d'accélération, alors 80% du portefeuille est alloué aux actions et 20% aux obligations, sinon, 40% du portefeuille est alloué aux actions et 60% aux obligations.

4.3 Résultats

Revenons aux méthodes que nous avons retenues pour chaque classifieurs. La régression logistique ne tolère pas beaucoup de complexification contrairement aux algorithmes d'apprentissage ensembliste. Les modèles finaux que nous conservons sont donc Logit avec normalisation, RF avec imputation, normalisation et rééchantillonage et XGB avec normalisation et rééchantillonage.

Indéniablement le Logit fait mieux sur tous les plans: Plus de vrais positifs, moins de faux négatifs et équivalent aux autres sur les autres métriques. On voit aussi que même malgré le lag, le Logit prédit souvent avec un temps d'avance les points de retournements.

Nous nous attendons donc à ce que le Logit donne de meilleurs résultats d'un point de vue financier.

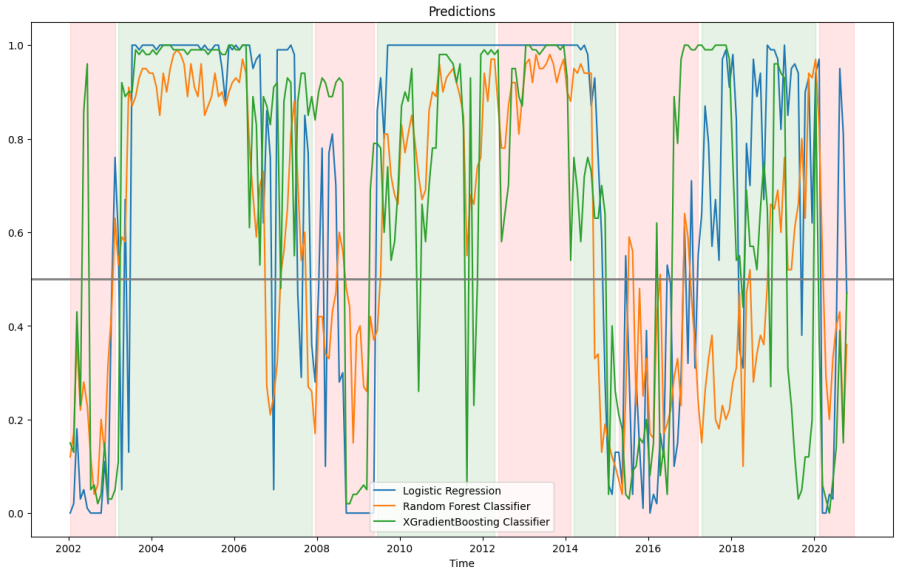


Figure 19: Probabilité des prédictions des meilleurs modèles

Nous choisissons comme outils de comparaison entre les portefeuilles le ratio de Sharpe. Bien qu'élémentaire, il nous donne une bonne indication sur le ratio risque/rendement.

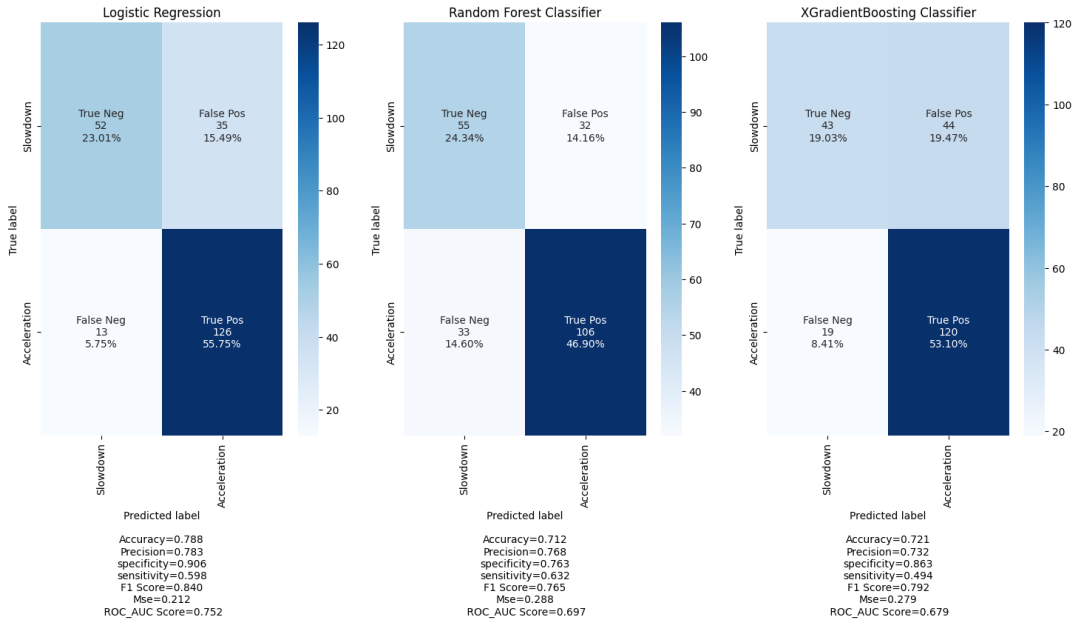


Figure 20: Matrice de confusion des meilleurs modèles

Qu'est-ce que le ratio de Sharpe ?

Le ratio de Sharpe est une mesure concrète de la performance ajustée au risque d'un portefeuille d'investissement.

On le calcule en utilisant la formule suivante :

$$\frac{\text{Rendement moyen investissement} - \text{Rendement moyen taux sans risque}}{\text{Écart type investissement}}$$

avec :

• Le Rendement moyen du taux sans risque (ici bon du trésor américain) correspond au rendement que l'on pourrait obtenir sur un placement sans risque, généralement représenté par un bon du Trésor. Ce rendement du sans risque sert de référence pour évaluer la performance de l'investissement en question.

• L'Écart type du portfeuille d'investissement est une mesure de la volatilité de l'investissement, c'est à dire l'ampleur des variations de rendement de l'investissement par rapport à sa moyenne. Plus cet écart type est élevé, plus l'investissement est risqué.

• Le Rendement moyen de l'investissement correspond au rendement moyen obtenu sur une période donnée pour l'investissement considéré. Il est calculé en prenant en compte les gains et/ou les pertes réalisés et la valeur actuelle de l'investissement.

En comparant le ratio de Sharpe des différentes stratégies d'investissement, on peut déterminer laquelle est la plus performante en termes de rendement ajusté au risque car en général, plus le ratio de Sharpe est élevé, meilleure est la performance ajustée au risque de l'investissement ou du portefeuille. Cela est particulièrement important pour les investisseurs car ils cherchent généralement à maximiser leurs rendements tout en minimisant leur exposition aux risques.

Commençons maintenant par comparer les résultats obtenus après toutes les méthodes d'amélioration des prédications dans la zone États-Unis pour la stratégie "Equity" et "Dynamique":

Résultats obtenus dans la zone États-Unis pour la stratégie Equity

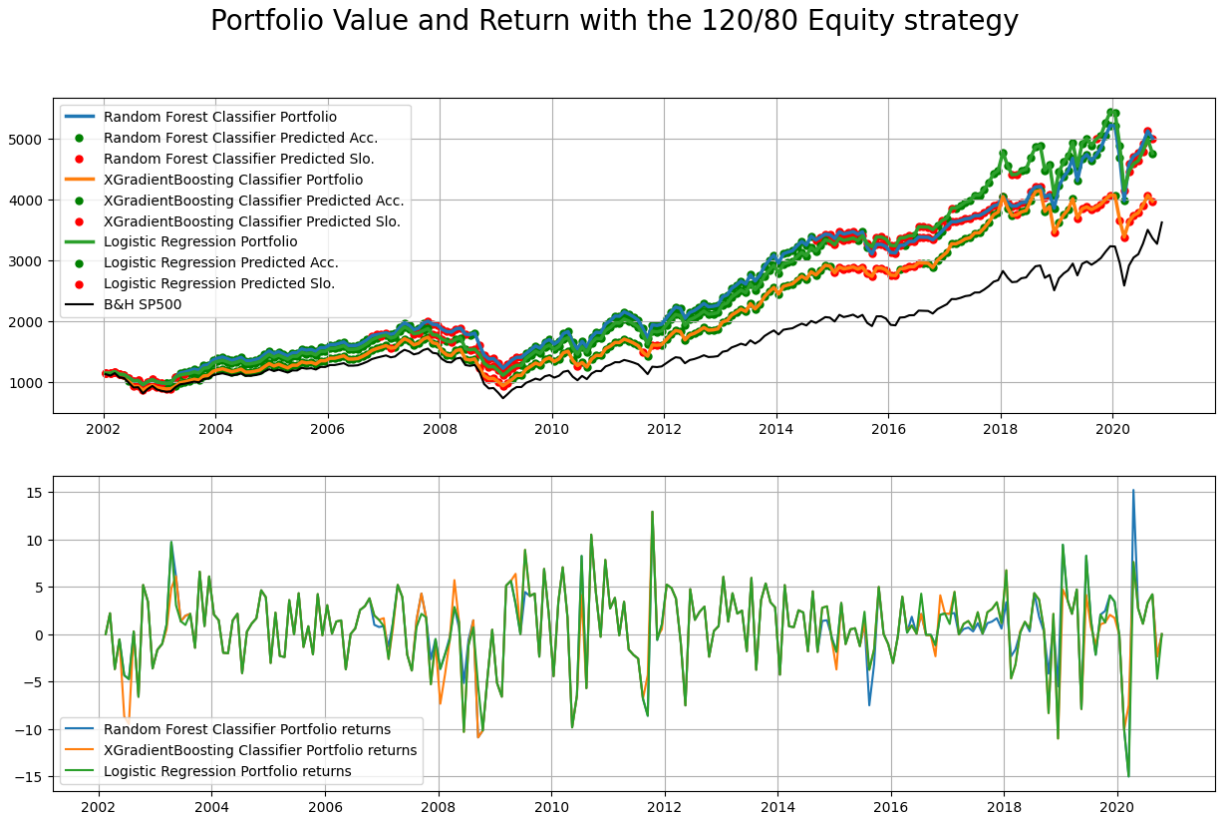


Figure 21: Portefeuilles obtenus avec les meilleurs modèles. Stratégie Equity

	Random Forest Classifier	XGradientBoosting Classifier	Logistic Regression
Max Monthly Drawdown in %	-15	-11	-15
Highest Monthly Return in %	15	13	13
Average Returns in %	0.737218	0.634653	0.717395
Volatility	4	4	4
Net Return in %	166	143	161
Sharpe ratio	0.493729	0.404608	0.469407

Figure 22: Récapitulatif de la stratégie Equity

Résultats obtenus dans la zone États-Unis pour la stratégie Dynamique

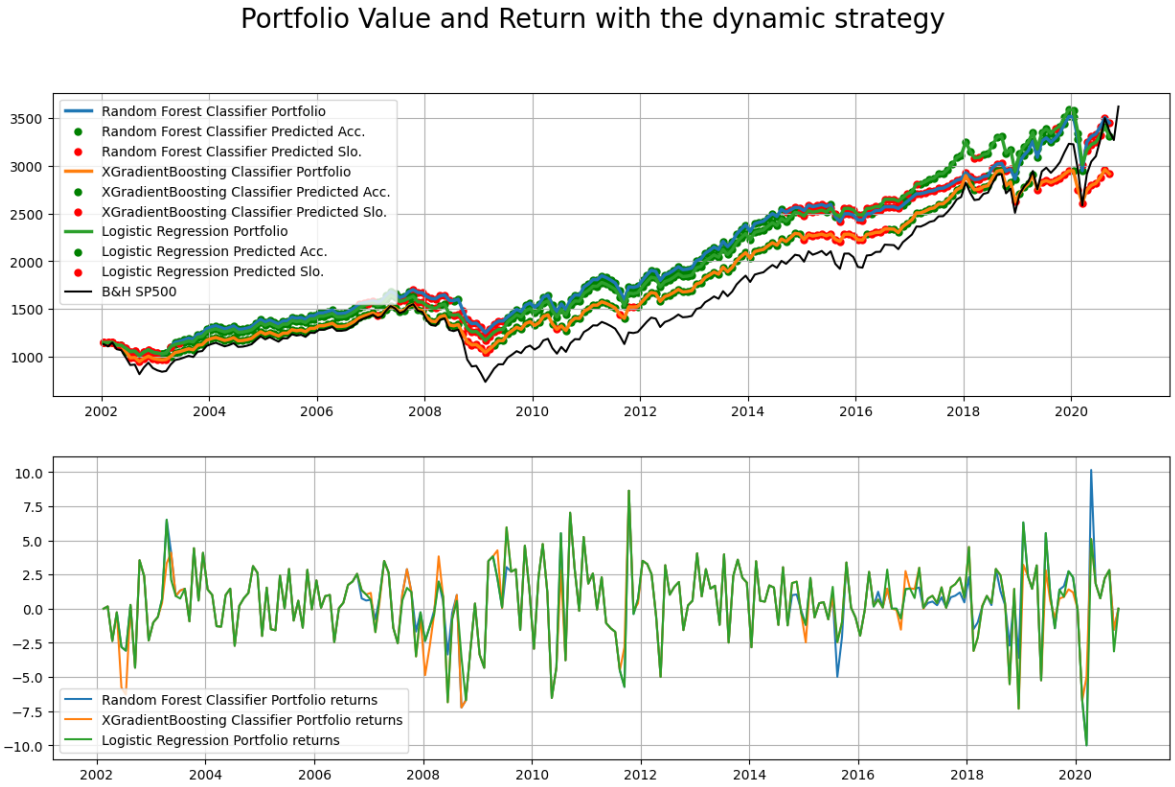


Figure 23: Portefeuilles obtenus avec les meilleurs modèles. Stratégie Equity

	Random Forest Classifier	XGradientBoosting Classifier	Logistic Regression
Max Monthly Drawdown in %	-10	-7	-10
Highest Monthly Return in %	10	9	9
Average Returns in %	0.526391	0.451425	0.508602
Volatility	3	3	3
Net Return in %	118	102	114
Sharpe ratio	0.467819	0.369628	0.437809

Figure 24: Récapitulatif de la stratégie Dynamique

5 Conclusion

En guise de conclusion, on peut affirmer que les modèles de classification d'apprentissage automatique ont une capacité convaincante même avec un lag pour prédire les points de retournements d'une économie. Ceci est d'autant plus convaincant si l'on regarde les résultats d'investissement selon la stratégie 120/80 éclairée par les prédictions de ces modèles. A la question que nous nous sommes posée, nous répondons que les modèles nous permettent d'anticiper les retournements et de battre le marché. Sauf bien évidemment dans des cas où d'évènements extrêmement rares d'un point de vue probabiliste (appelés Black Swans) surviennent: Le COVID par exemple.

Il est à noter que les modèles perçoivent très bien avec un coup d'avance les changements de régimes quand ceux-ci sont causés purement par une dégradation de certains indicateurs économiques. Le crash de 2008 en est un bon exemple. En effet, bien qu'étant en période de crise, économistes et politiciens ne la perçoivent pas encore alors que les modèles l'indiquent avec un temps d'avance.
Une autre remarque importante est que notre travail montre que l'ajout de complexité dans un modèle n'implique pas nécessairement une amélioration. La régression logistique en est l'exemple.

6 Annexe:

6.1 Présentation des données à notre disposition

L'ensemble de nos variables explicatives est fourni par l'OCDE aux États-Unis depuis 1985. Notre matrice design contient 70 variables explicatives séparées en data train et data test. On peut donner l'exemple du SP500, un indice boursier mesurant la performance d'entreprises cotées en bourse aux États Unis. Ces variables sont des obligations d'État, des spread entre les obligations d'entreprises d'investissement de qualité et à haut rendement, des actions de grandes capitalisations,des indices de , volatilité des actifs, indice VIX et indice VSTOXX, ou encore prix des matières premières (pétrole brut, gaz naturel, or, argent et indice CRB). Ces variables non revisées sont extraites de sondages d'entreprises et de séries financières disponibles en temps réel.

Notons que dans le cadre de ce mémoire, nos données sont non révisées, c'est-à-dire que certaines de ces variables explicatives seraient plus proches de la réalité économique si elles étaient actualisées.

Le graphique ci-dessous donne une première idée de la forte corrélation et la redondance de certaines variables explicatives. Nous écartons donc certaines variables dans le paragraphe 3.1 Normalisation des données et méthodes de réduction de la dimension.

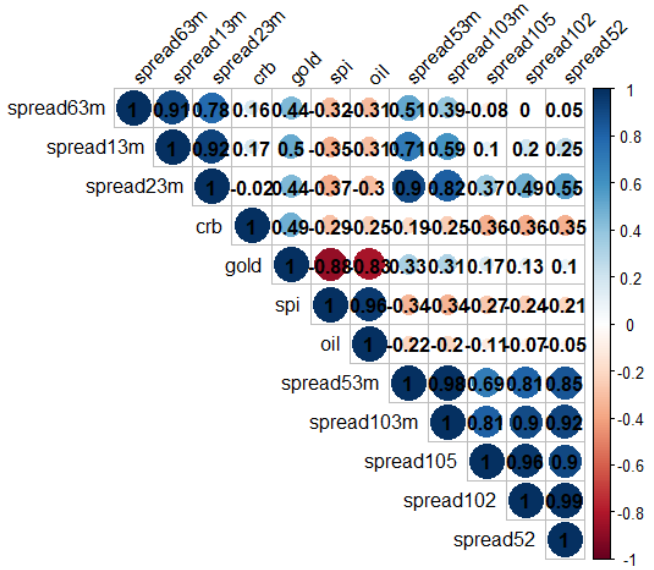
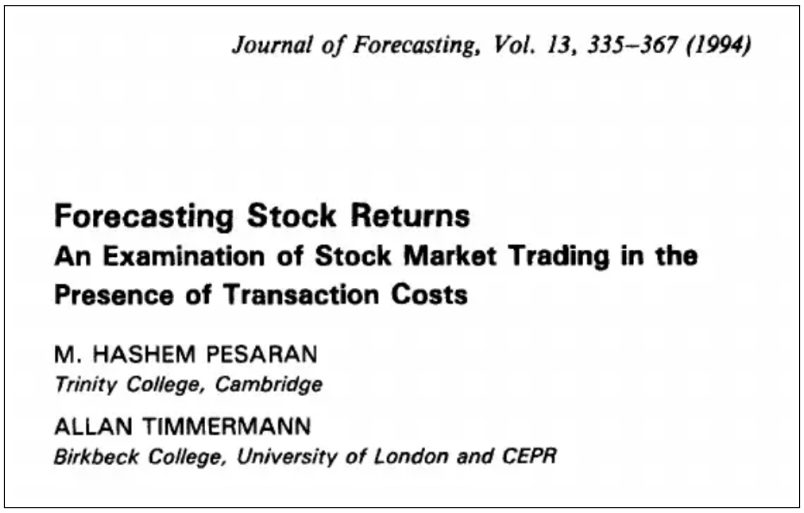


Figure 25: Corplot sur certaines variables explicatives

6.2 Pistes de réflexion



Journal of Forecasting, Vol. 13, 335–367 (1994)

Forecasting Stock Returns
An Examination of Stock Market Trading in the Presence of Transaction Costs

M. HASHEM PESARAN
Trinity College, Cambridge

ALLAN TIMMERMANN
Birkbeck College, University of London and CEPR

Figure 26: Pesaran, H., and A. Timmermann (1994)

Dans cette partie, notre objectif est de mener une analyse critique de notre processus de réflexion durant notre mémoire. En effet, rappelons que tout au long de ce mémoire, nous avons formulé plusieurs hypothèses de grande simplification des marchés financiers, notamment celle-ci que nous avons extraite de notre cours sur le mouvement brownien donné par Imen Ben Tahar, José Trashorras et Gabriel Turinici qui est:

Hypothèse: Marchés sans frictions

Les actifs de nos marchés financiers sont parfaitement divisibles et ils ne sont pas soumis à des coûts de transaction.

Cependant, nous avons découvert que cette hypothèse de simplification des marches avait un fort impact négatif sur la qualité de nos prédictions.

Nous proposons donc dans cette partie annexe d'explorer de manière partielle un article de référence. Le document de Pesaran et Timmermann intitulé "Prévision des rendements boursiers : une étude de la négociation sur le marché boursier en présence de coûts de transaction" met en avant le fait que la capacité des investisseurs à prédire les rendements boursiers depend aussi des coûts de transaction sur les marchés financiers: il y a un réel impact de ces coûts de transaction sur la rentabilité des stratégies de trading basées sur des prédictions de rendement. Pour palier à cela, les auteurs proposent un modèle d'estimation des rendements qui tient compte cette fois-ci de ces coûts et qui est basé sur une analyse de la variance de la prédiction des investisseurs.

Enfin, cet article de Pesaran et Timmermann offre aussi un panel de propositions pour améliorer la fiabilité des prédictions de rendement en utilisant par exemple des informations supplémentaires telles que les données sur les volumes de transactions et les positions de marché.

7 Bibliographie

Cette bibliographie contient les références du document de référence "Investing through Economic Cycles with Ensemble Machine Learning Algorithms" et nos références que nous avons le plus utilisé le plus pour lequelles nous donnons un lien hypertexte.

Sylvain Benoit, Thomas Raffinot "Investing Through Economic Cycles with Ensemble Machine Learning Algorithms"

Raffinot, T. (2017): "Time-varying risk premiums and economic cycles," Discussion paper .

Sophie Donnet 2021: Cours Modèle linéaire gaussien

Yating Liu: Introduction a l'apprentissage statistique

Imen Ben Tahar, José Trashorras et Gabriel Turinici: Éléments de calcul stochastique pour l'évaluation et la couverture des actifs dérivés

Cooper, I., and R. Priestley (2009): "Time-Varying Risk Premiums and the Output Gap, Review of Financial Studies, 22(7), 2801 – 2833. 3

Marco Peixeiro Decision Trees

Pablo Aznar Decision Trees & Gini definitions

Friedman, J. H., T. Hastie, and R. Tibshirani (2000): "Additive Logistic Regression: A Statistical View of Boosting (with Discussion)". The Annals of Statistics, 28, 337–407

Breiman, L. (2001): "Random Forests," Machine Learning", 45, 5–32.

Friedman, J. H. (2001): "Greedy Function Approximation: A Gradient Boosting Machine," The Annals of Statistics, 29, 1189–1232.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984): Classification and Regression Trees. Wadsworth and Brooks, Monterey,

Schapire, R. E. (1990):"The strength of weak learnability,"

Stock, J. H., and M. W. Watson (2014):"Estimating turning points using large data sets," Journal of Econometrics 178, 368 – 381.

Aruoba, S. B., F. X. Diebold, and C. Scotti (2009):"Real-Time Measurement of Business Conditions," Journal of Business & Economic Statistics, 27(4), 27(4), 417–427.

Schapire, R. E. (1990):"The strength of weak learnability,"

Bai, J., and S. Ng (2009): "Boosting diffusion indices," Journal of Applied Econometrics, 24(4), 607–629.

Bengoechea, P., M. Camacho, and G. Perez-Quiros (2006): "A useful tool for forecasting the Euro-area business cycle phases," International Journal of Forecasting, 22(4), 735 – 749.

Berge, T. (2015): "Predicting Recessions with Leading Indicators: Model Averaging and Selection over the Business Cycle," Journal of Forecasting, 34(6), 455–471.

Bergmeir, C., R. J. Hyndman, and B. Koo (2015): "A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction," Discussion paper.

Boivin, J., and S. Ng (2006): "Are more data always better for factor analysis?," Journal of Econometrics, 132(1), 169–194.

Brown, S. (2008): "Elusive return predictability: Discussion," International Journal of Forecasting, 24(1), 19–21.

Buhlmann, P., and B. Yu ¨ (2003): "Boosting with the L2 Loss: Regression and Classification," Journal of the American Statistical Association, 98, 324–338.

Camacho, M., G. Perez-Quiros, and P. Poncela (2015): "Extracting Nonlinear Signals from Several Economic Indicators," Journal of Applied Econometrics, 30(7), 1073–1089.

(2018): "Markov-switching dynamic factor models in real time," International Journal of Forecasting, 34(4), 598 – 611.

Caruana, R., and A. Niculescu-Mizil (2005): "An Empirical Comparison of Supervised Learning Algorithms Using Different Performance Metrics," in Proc. 23 rd Intl. Conf. Machine learning (ICML'06), pp. 161–168.

Cenesizoglu, T., and A. Timmermann (2012): "Do return prediction models add economic value?," Journal of Banking & Finance, 36(11), 2974 – 2987.

Chauvet, M., and J. Piger (2008): "A Comparison of the Real-Time Performance of Business Cycle Dating Methods," Journal of Business & Economic Statistics, 26(1), 42–49.

Doz, C., D. Giannone, and L. Reichlin (2011): "A two-step estimator for large approximate dynamic factor models based on Kalman filtering," Journal of Econometrics, 1(164), 188–205

Duarte, A., I. A. Venetis, and I. Paya (2005): "Predicting real growth and the probability of recession in the Euro area using the yield spread," International Journal of Forecasting, 21(2), 261–277

Efron, B., and R. Tibshirani (1994): An Introduction to the Bootstrap, Chapman Hall/CRC Monographs on Statistics Applied Probability. Taylor Francis.

Eilers, P. H. C., and B. D. Marx (1996): "Flexible smoothing with B-splines and penalties," Statist. Sci., 11(2), 89–121.

Florez-Lopez, R. (2007): "Modelling of insurers' rating determinants. An application of machine learning techniques and statistical models," European Journal of Operational Research, 183(3), 1488 – 1512

Freund, Y., and R. Schapire (1997): "A decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, 55, 119–139.

Giannone, D., L. Reichlin, and D. Small (2008): "Nowcasting: The real-time informational content of macroeconomic data," Journal of Monetary Economics, 55(4), 665 – 676

Giusto, A., and J. Piger (2017): "Identifying business cycle turning points in real time with vector quantization," International Journal of Forecasting, 33(1), 174–184.

Hamilton, J. (2011): "Calling recessions in real time," International Journal of Forecasting, 27(4), 1006–1026.

Han, Y., K. Yang, and G. Zhou (2013): "A New Anomaly: The Cross-Sectional Profitability of Technical Analysis," Journal of Financial and Quantitative Analysis, 48, 1433–1461.

Hanley, J. A., and B. J. McNeil (1982): "The meaning and use of the area under a receiver operating characteristic (ROC) curve.," Radiology, 143(1), 29–36.

Hansen, P., A. Lunde, and J. Nason (2011): "The Model Confidence Set," Econometrica, 79(2), 453–497.

Hastie, T. (2007): "Comment: Boosting Algorithms: Regularization, Prediction and Model Fitting," Statistical Science, 22, 513–515.

Hastie, T., R. Tibshirani, and J. Friedman (2009): The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, 2 edn.

Hsu, Y.-C., C.-M. Kuan, and M.-F. Yen (2014): "A Generalized Stepwise Procedure with Improved Power for Multiple Inequalities Testing," Journal of Financial Econometrics, 12(4), 730–755.

Ishwaran, H. (2007): "Variable importance in binary regression trees and forests," Electron. J. Statist., 1, 519–537.

Liu, W., and E. Moench (2016): "What predicts US recessions?," International Journal of Forecasting, 32(4), 1138 – 1150.

Mai, F., S. Tian, C. Lee, and L. Ma (2018): "Deep learning models for bankruptcy prediction using textual disclosures," European Journal of Operational Research, forthcoming.

Mintz, I. (1974): "Dating United States Growth Cycles," in Explorations in Economic Research, Volume 1, Number 1, pp. 1–113. National Bureau of Economic Research, Inc.

Monch, E., and H. Uhlig (2005): "Towards a Monthly Business Cycle Chronology for the Euro Area," Journal of Business Cycle Measurement and Analysis, 2005(1), 43–69.

Ng, S. (2014): "Viewpoint: Boosting Recessions," Canadian Journal of Economics, 47(1), 1–34.

Nilsson, R., and G. Gyomai (2011): "Cycle Extraction: A Comparison of the Phase-Average Trend Method, the Hodrick-Prescott and Christiano-Fitzgerald Filters," OECD Statistics Working Papers 2011/4, OECD Publishing.

Okun, A. (1962): "Potential GNP: Its Measurement and Significance," Proceedings of the business and Economic Statistics Section, American Statistical Association, pp. 98–104.

Pesaran, H., and A. Timmermann (1994): "Forecasting stock returns: an examination of stock market trading in the presence of transaction costs," Journal of forecasting, 13(4), 335–367.

Piger, J. (2011): Econometrics: Models of Regime Changes,pp. 190–202. Springer New York, New York, NY.

Romano, J. P., and M. Wolf (2005): "Stepwise Multiple Testing as Formalized Data Snooping," Econometrica, 73(4), 1237–1282. 24

Schapire, R. E. (1990): "The strength of weak learnability," Discussion paper

Nonlinear Estimation and Classification, chap. The Boosting Approach to Machine Learning: An Overview, pp. 149–171. Springer New York, New York, NY.