Prévision du cours d'une action avec une Régression linéaire

Nous avons conçu un modèle de Machine Learning pouvant prédire le cours d'une action en fonction des données passées. Pour ce faire nous devons commencer par définir les facteurs influant sur le cours d'une action. Ces facteurs seront alors les entrées X de notre système. La sortie Y de notre système est le cours de l'action. A noter que nous avons intégralement coder la régression linéaire (from scratch).

a) Bases du modèle

Facteurs influant sur le cours d'une action :

X1 – Bénéfice de la société

X2- Actifs de l'entreprise

X3- Dividendes

X4 – Prix le plus récent de l'action

Ces facteurs sont susceptibles de changer au cours du projet au profit de facteurs que nous pourrons trouver plus pertinents par la suite. A ce stade du projet ces facteurs nous semblent assez pertinent et représentatif de la réalité. Nous définissons ces facteurs comme étant les cordonnées de points X tel que X= (X1,…,X4). Chaque point est donc caractérisé par ces facteurs.

Estimation de Y noté Y~ :

Pour un point Xi (dans notre modèle pour chaque point X pris et caractérisé à la date i), on a :

$$Yi\tilde{} = F(X, \theta) = \theta_0 + \theta_1.X_{i1} + \theta_2.X_{i2} + \theta_3.X_{i3} + \theta_4.X_{i4}$$

Ti= {Xi1…Xi4} : La date i nous renseigne sur les facteurs du point Xi.

Avec Téta les poids de facteurs dans l'expression de Y. Plus le facteur Xn est significatif et déterminant pour la valeur de Y alors plus le poids téta associé sera grand.

Nous allons déterminer Téta qui minimise ½ (∑|Yi-Yi~|²) (l'erreur quadratique du système) dans la première phase de notre algorithme de Machine Learning : ***la phase d'entrainement.***

b) Première étape : la phase d'entrainement

Pour cette étape, l'algorithme doit déterminer les poids Téta les plus pertinents pour notre modèle. Pour ce faire, nous avons besoin d'un échantillon de N points Xn= (X1……X4) ainsi que leur Y (réel donc ! = Y~) associés. Il faut bien comprendre qu'à cette étape nous travaillons uniquement avec des ***données passées*** dont nous connaissons donc X et Y afin de trouver la corrélation entre X, Y et Téta.

Il faut prendre un N très grand car plus N est grand et donc plus nous avons de données réelles et plus le système est précis. Il nous faut donc recueillir une grande base de données. Chaque donnée appelée point sera donc recueilli dans le passé à un temps différent Ti pour avoir des cours d'action différents.

Soit X et Y les matrices suivantes :

$$X=\begin{bmatrix} 1 & X11 & \cdots & X14 \\ \vdots & & \ddots & \vdots \\ 1 & XN1 & \cdots & XN4 \end{bmatrix} \in \mathbb{R}^{(N,6)}$$ N : nombre de données.

(Exemple : XN4 correspond au paramètre X4 du point (ou donnée) N).

On ajoute une colonne de 1 dans X pour déterminer θ0

$$Y=\begin{bmatrix} Y1 \\ \vdots \\ \vdots \\ YN \end{bmatrix} \in \mathbb{R}^{(N,1)}$$ N : nombre de données.

Détermination de θ :

$$\theta = \begin{bmatrix} \theta 0 \\ \vdots \\ \vdots \\ \theta N \end{bmatrix} \in \mathbb{R}^{(N+1,1)}$$

$$\theta = (X^T X)^{-1} X^T Y$$ Minimise le plus ½ (∑|Yi-Yi~|²)

A la fin de l'entrainement, nous avons donc trouver des poids θ optimisés. Nous avons tous les éléments requis pour déterminer une prévision du cours de l'action futur Y~.

c) Phase de Test.

Nous devons maintenant tester la précision de notre algorithme. Pour ce faire nous comparons la sortie prédite d'une entrée jamais vu au préalable par notre algorithme avec sa sortie réelle afin de déterminer le pourcentage d'erreur entre la prédiction et la réalité. Nous trouvons une erreur généralement comprise entre 3 et 6% ce qui est très concluant.

d) Phase de Prédiction

Nous connaissons maintenant intégralement notre expression de Y~ =F(X, θ) pour calculer le cours de l'action futur en fonction d'un point X=(X1…..X4).

Pour prédire le cours de l'action futur, nous récupérons T0={X1,….,X4} qui correspond aux facteurs actuels nous donnant notre point X que l'on injecte dans Y~ =F(X, θ).Cela nous donnera donc Y~ le cours de l'action future en fonction des paramètres relevés actuels et permettra à l'investisseur de connaitre le rendement de son éventuel investissement par rapport à sa mise initiale (c'est-à-dire le prix de l'action récent X4).

Soit G~ le gain en pourcentage supposé, on a : G~= $\frac{Y\sim - X4}{X4}$. 100 .

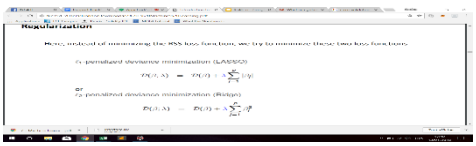
Selon le gain supposé, l'investisseur sera libre d'investir ou non en fonction de ses attentes en termes de gain.

Régularisation Ridge + cross-validation

Lorsque l'on utilise une régression linéaire avec la méthode des moindres de carrés, il y a un risque de générer un modèle trop complexe, car correspondant uniquement au jeu de données utilisé lors des tests de régression. Cela s'appelle l'overfitting.

Pour que notre modèle soit généralisable, et soit adapté aux données futures, il doit avoir une complexité modérée, par exemple en réduisant le nombre de variables explicatives.

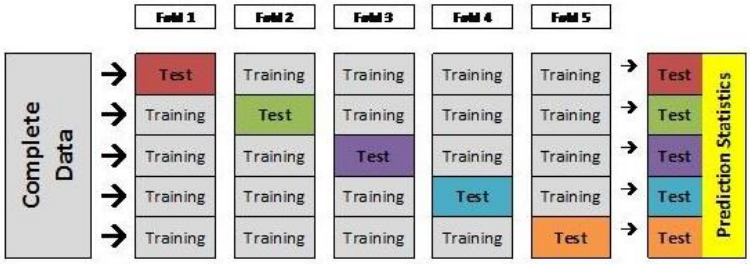
Ridge ajoute donc un terme pénalisant à l'erreur à minimiser, de façon à réduire l'importance des variables explicatives non pertinentes :



Ridge permet donc une régularisation du modèle en réduisant l'overfitting.

Lorsque nous entraînons un modèle en validation simple, nous sommes en outre contraints de séparer le jeu de données entre données d'entraînement (~70%) et données test (~30%). Certaines données sont donc “perdues” pour être testées et ne servent pas à entraîner le modèle.

C’est pourquoi nous avons utilisé la cross-validation, qui consiste à tester notre modèle de multiples fois en changeant de données d’entraînement de données test, puis de faire la moyenne des résultats obtenus.



Cela permet de tirer profit de l’ensemble de nos jeux de données.

Prévision du cours d’une action avec Random Forest

Nous avons décidé d’implémenter un arbre de décisions pour produire une autre prévision en complément de la régression linéaire. L’algorithme sélectionne automatiquement les variables explicatives discriminantes à partir de notre base de données lui permettant ainsi d’extraire des règles logiques de cause à effet qui n’apparaissaient pas initialement et ainsi prédire une sortie associée a l’entrée. Un arbre de régression se construit de manière itérative, en découpant à chaque étape la population en deux sous-ensembles. Le découpage (ou test) s’effectue suivant des règles simples portant sur les variables explicatives, en déterminant la règle optimale qui permet de construire deux populations les plus différenciées en termes de valeurs de la variable à expliquer.

De plus nous avons utilisé la méthode ensembliste qui consiste à coupler différents algorithmes de Machine Learning afin d’en tirer un modèle généralisant beaucoup mieux que les algorithmes initiaux. Ainsi nous avons implémenter un Random Forest c’est-à-dire plusieurs arbres de décisions dont nous faisons la moyenne des sorties. Nous avons eu recours à scikit learn pour implémenter notre algorithme.

Value at Risk

L’algorithme de value at risk a été codé en python. Il a été difficile de trouver de la documentation sur la value at risk avec une simulation de Monte-Carlo. La première étape a été de “créer une action” avec des valeurs test qui allaient être ensuite remplacées par les vraies valeurs de la base de données.

La Value at Risk a la formule suivante :

Prix simulé = Prix actuel * exponentiel (rendement-0.5 * volatilité 2) + volatilité * processus de Wiener * √ échéance.

En général les calculs de value at risk en simulation de Monte Carlo sont effectués sur excel. Il est facile sur excel de générer le processus de Wiener car une fonction intrinsèque à Excel le fait directement. La difficulté a été de générer le processus de Wiener. Pour cela un nombre aléatoire entre 0 et 1 est généré. L'étape suivante est de chercher l'image de ce nombre aléatoire dans la fonction normale inverse. Pour cela nous avons utilisé la fonction suivante : norm.ppf (nombre aléatoire).

Une fois cette le processus de Wiener généré, il suffit d'appliquer la formule ci-dessus en remplaçant les différents termes par les valeurs test mentionnées plus haut.

Enfin la dernière étape a été de relier cet algorithme avec la base de données afin qu'il calcule la value at risk de vraies actions. En effet la value at risk est calculée à partir de la volatilité (calculée par un autre algorithme de Risk-Less) mais aussi à partir des informations sur les actions présentes dans la base de données.

Volatilité

L'investisseur cherchant quel actif acquérir prête attention d'une part au rendement potentiel offert, et d'autre part au risque que cela implique.

Nous pouvons considérer que plus un actif offre un rendement stable au cours du temps, plus il est sûr. Inversement, plus son rendement à tendance à varier largement, plus le risque est élevé.

La volatilité (σ) indique la dispersion des rendements de l'action par rapport à la moyenne des rendements.

$$=1n(Ri - Rm)^2n$$

Avec
Ri le rendement annuel
Rm la moyenne des rendements annuels
n le nombre d'années

Outre le gain potentiel espéré, il est sage de faire attention à ce que la volatilité ne soit pas trop élevée.