**PROJET 2**

**But : créer un indicateur « alerte tel mot est apparu sur tel ticker, il faut acheter/vendre tel actif… »**

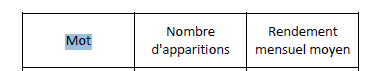
Idéal : construire un modèle par ticker mais on fait le modèle sur tous les tickers car sinon on n’aurait pas assez de lignes

Etape 1 : mots les plus fréquents   
rendements moyens mensuels présagent de rendements mensuels > rendements moyens sans le mot   
Apparition mot => rendement plus élevé : dans le tableau, les mots sont > 1% while en moyenne rendement <1%  
=> on constate que ces mots sont associés à des rendements plus élevés

Etape 2 : construire un modèle pour prévoir les rendements RDMT (Horizon de la prédiction = à la carte : on choisit si on veut travailler sur l’hebdo, le mensuel ou le journalier : on prévoit soit le mensuel, soit le journalier, soit l’hebdo)  
Priorité : non pas les mots car on a déjà filtré les lignes avec une actualité   
on se concentre sur la variable de rendements => construction variable binaire « ACHETER » ou « VENDRE » ou scoring d’aide à la décision

Output : lorsque ce mot est arrivé ça signifie qu’il fallait mieux acheter/ vendre   
Ce mot était arrivé sur telle ligne => est ce qu’on peut faire confiance à telle ligne ?

**Q2.**  statistiques : volatilité des rendements, changements de volumes   
à orienter par rapport à la compréhension du sujet   
statistiques à faire qq soit le ticker

**Q3.**  
rendement moyen = RDMT\_M

Suppression des lignes sans les mots donne une data base de 11 à 15000 lignes (mais à vérifier)  
On doit avoir une condition en OU : on a tel mot OU tel mot

**Q4**.

* Si RDMT\_M = variable expliquée => toutes les autres variables hormis les trop corrélées (RDMT\_J, RDMT\_S) sont explicatives=> BIEN SUPPRIMER LES AUTRES RENDEMENTS POUR EVITER L’OVERFITTING
* Corrélation trop élevée = corrélation de Pearson > 75%
* Les variables HISTO sont candidates en tant que variables explicatives mais à voir si on les garde selon si elles sont pas trop corrélées

**Q5.** Choix de l’algo : choisir celui où est le plus à l’aise   
XGBOOST : résultats plus rapides et plus probants (pas forcément les meilleurs) mais plus dur à utiliser   
Toutes les questions à partir de la question 4 sont sur la base filtrée

HISTO = entre t-5 et t

**Q6.**

Dans un premier temps : dvlper et débuguer sur une faible partie de la base (5% = base train) pour ne pas perdre du temps (car si on fait sur toute la database ça va mettre du temps à tourner)

**Analyse des hyperparamètres** :   
hyperparamètres = paramètres de l’algorithme étant fixés avant la phase d’apprentissage

**Cross validation** : permet de comparer plsrs modèles et de choisir le meilleur  
Cross validation de 4 folds = consiste à diviser l’ensemble des données d’apprentissage en 4 partitions distinctes de taille égales pour utiliser 3 partitions pour l’apprentissage et la dernière pour évaluer la performance du modèle. Ce processus est répété 𝑘 fois pour chacun des 𝑘 segments disponibles pour le test

**Grid computing** : 100 N-uplets  
A chaque test de validation, les paramètres du modèle changent suivant une grille de recherche prédéfinie, appelée Grid search method.La technique traditionnelle pour optimiser les hyper-paramètres est la méthode de Grid Search, qui consiste **à tester toutes les combinaisons possibles des hyper-paramètres que l’on a fournis**. L’algorithme de Grid Search doit être guidé par l’évolution d’une métrique d’erreur, mesurée généralement par validation croisée sur l’algorithme d’apprentissage

Dans le cas du xg-boost, les hyper paramètres les plus influents sont :

* eta : paramètre de pénalisation, qui selon ses valeurs module le surapprentissage. Compris entre 0 et 1, il pénalise l’ajout d’un nouvel arbre dans l’agrégation et ralentit la convergence. Si sa valeur est faible, cela conduit à accroître le nombre d’arbres mais entraîne généralement une amélioration de la qualité de prévision. Le boosting est cependant un algorithme qui peut converger exactement donc éventuellement vers une situation de sur-apprentissage.
* max\_depth : profondeur maximale des arbres construits à chaque itération. Plus cette profondeur est grande, plus complexe est le modèle. Généralement compris entre 2 et 30, selon le nbr de variables explicatives
* subsample : pourcentage des observations prises aléatoirement à chaque instance de l’algorithme pour construire un arbre. Compris entre 0,1 et 1

Autres hyperparamètres :

* In\_childweight : 1, 5, 100
* N\_estimator : nbr d’estimateurs : 100 par défaut. Hausse du nbr d’estimateur => modèle plus long à tourner => le garder à 100
* Learning\_rate : garder par défaut
* Lamda, alpha : test de l’augmeter
* ColSample\_bylevel : compris entre 0,1 et 1 : nbr de variables explicatives prises en compte à chaque level de l’arbre
* & ColSample\_bytree : compris entre 0,1 et 1

**Q7.** Modèle doit prévoir achat/vente sur un ticket quand un mot parmi le plus fréquent sort

**Q8.** AUC min attendu : 75%

Remarque : on peut allez récupérer d’autres data et les ajouter à la database

Paul, Julien, Salima font le projet 2

Prochaine date : jeudi 7 19h/20h