Compte-Rendu Data Challenge APST1

Bilel MEZRANI - William SCHTRAOUSS - BITCOIN

Décembre 2020

Table des Matières

1 Introduction

Dans le cadre des Data Challenges en Apprentissage Statistique 1, nous avons choisi le sujet *"Prediction of direction of Bitcoin based on sentiment data"*, présenté par le *Napoleon Group*. Cette entreprise spécialisée dans les investissements financiers (dans notre cas - dans le secteur des cryptomonnaies) via l'utilisation d'algorithmes de traitement de Data, cherche dans ce problème à prédire le comportement du Bitcoin à l'aide de données non-financières, dites de "sentiments". Le marché du Bitcoin étant assez récent et sans beaucoup de liquidités, les comportements sont bien plus mimétiques, influençables et volatiles que pour les actifs classiques, d'où la nécessité de prendre en compte l'activité des (jeunes) intervenants sur les forums et réseaux sociaux.

2 Présentation du Problème et des Données

Nous avons dans cette étude accès à des fichiers $X_{train}.csv$, $Y_{train}.csv$ et $X_{test}.csv$ de dimensions respectives [14 000, 486], [14 000, 4] et [5 000, 486]. L'objectif étant de retourner un fichier $Y_{test}.csv$ de taille [5 000, 4]. Interessons nous plus précisément à ces données.

• **Echantillons :** Les 19 000 échantillons (apprentissage et test confondus), représentent toutes les périodes de 49 heures consécutives entre Juillet 2017 et fin Septembre 2019 (à priori, pas dans leur ordre chronologique) et sont indexés par un numéro d'identification.

• **Données de variation du Bitcoin :** Pour chaque échantillon (donc chaque période indépendante de 49h), on dispose de 5 indicateurs normalisés X_1 à X_5, qui donnent l'évolution du prix du Bitcoin. En se plaçant à 48 heures sur 49, ils représentent la variation passée sur respectivement 1 heure, 6 heures, 12 heures, 24 heures et 48 heures, normalisée par la volatilité sur les 48 heures, qui est un écart-type.

• **Données de sentiments :** De même, pour chaque échantillon, on dispose d'indicateurs de sentiments. Ces indicateurs sont au nombre de 10, autrement dit, on utilisera dans notre étude 10 "thèmes" différents (abordés sur Internet) ayant lien avec le Bitcoin et influençant son cours. Pour chacun de ces 10 indicateurs, on aura 48 valeurs normalisées - une valeur de chaque indicateur, par heure, sur 48 heures (de l'instant "0" à 48 heures sur 49). Soit un total de 480 valeurs normalisées, notées $I_{i,lag\ k}\ \forall i \in [\![1,10]\!], \forall k \in [\![0,47]\!]$.

Comment ces données de sentiment ont-elles été obtenues précisément ? Pour chacun des 10 thèmes et pour chaque tranche d'une heure, ont été comptées les récurrences de certains mots sur des forums spécialisés ou sur les réseaux sociaux. Pour un thème T, sur l'échantillon i et pendant la k-ième heure, on notera $T_{i,k}$ le nombre comptabilisé de ces mots. La moyenne empirique des comptages sur un seul échantillon est $\overline{T_i} = \frac{1}{48}\sum_{k=0}^{47} T_{i,k}$, et celle sur tous les échantillons est : $\overline{T} = \frac{1}{N}\sum_{i=1}^{N} \overline{T_i}$. Alors pour obtenir la valeur $F_{i,k}$ de la feature $I_{T,lag\ k}$, on effectue un Z-score (division de l'écart à la moyenne par l'écart-type) multiplié par le rapport $\overline{T_i}$ sur $\overline{T}$:

$$\boxed{F_{i,k} = \frac{T_{i,k} - \overline{T_i}}{\sqrt{\frac{1}{47}\sum_{j=0}^{47}(T_{i,j} - \overline{T_i})^2}} \cdot \frac{\overline{T_i}}{\overline{T}}}$$

• **Classes :** A partir de ces 485 indicateurs par échantillon, on va chercher à déterminer l'augmentation en pourcentage, appelée *"return"*, de la valeur du Bitcoin au cours de la 49-ième heure (c'est à dire $\frac{P(48h+1h)}{P(48h)} - 1$, avec P le prix du Bitcoin). Ce return sera divisé en 3 classes - la première étant pour les variations inférieures à $-0,2\%$, la deuxième pour celles entre $-0,2\%$ et $+0,2\%$, et la troisième et dernière pour celles supérieures à $+0,2\%$. Le choix de ces valeurs frontières étant motivé par le fait que $+0,2\%$ soit le centile à $66,7\%$ de la distribution.

• **Métrique :** La métrique utilisée dans le problème est la perte logistique, qui va calculer la log-vraisemblance négative des probabilités de classification calculées sur $X_{test}.csv$ par rapport à la classification réelle Y_{test}. Si l'on note Y, la matrice des encodages one-hot $y_{i,k}$ des classes k pour les échantillons i, et P la matrice des probabilités estimées $p_{i,k}$ pour les échantillons i d'appartenir aux classes k, alors la perte logistique pour un échantillon de taille N s'écrit :

$$\boxed{L_{log}(Y,P) = -\ log(Proba(Y|P)) = -\ \frac{1}{N}\sum_{i=1}^{N}\sum_{K=1}^{3} y_{i,k} \cdot log(p_{i,k})}$$

• **Objectif & Benchmark :** On cherche bien entendu à minimiser cette perte logistique grâce au traitement des données ainsi qu'à l'application d'algorithmes sur celles-ci. La perte logistique de référence, de valeur **1.1005**, a été obtenue par une régression logistique sur les seuls paramètres X_1 à X_5.

3 Traitement des Données

3.1 Questions à se poser

• **Données manquantes:** Après avoir regardé à quoi ressemblaient les données, il faut vérifier qu'il n'y a pas de données manquantes (souvent notées N/A ou NaN). On utilise pour cela une librairie appelée 'missingno' qui permet de déterminer visuellement s'il y en a. Dans notre cas, toutes les données sont présentes.

• **Outliers:** Il faut aussi se poser la question de la présence d'outliers. Il s'agit des observations qui diffèrent grandement des autres valeurs de la même variable. Elles peuvent être de deux natures : une erreur liée à la prise d'information (comptage des thèmes qui est mauvais par exemple) ou bien juste des caractéristiques de l'environnement dans lequel on évolue (un marché financier qui subit soudainement une crise pendant 48h). Nous avons estimé que nous nous trouvions dans ce deuxième cas, ayant conscience de la volatilité du Bitcoin. Ainsi, nous n'avons pas supprimé les outliers car il faut les prendre en compte dans notre prédiction.

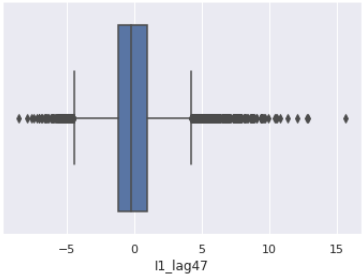

Figure 1: Box Plot

3.2 Feature Engineering

3.2.1 Choix des indicateurs à garder

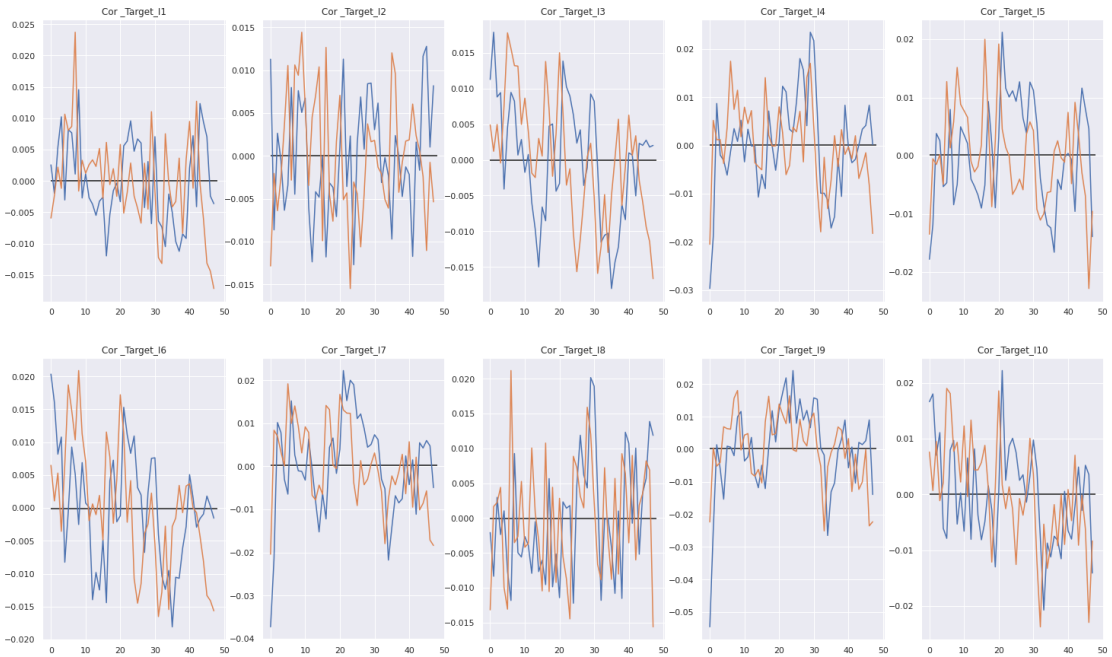

Figure 2: Corrélation des features

Pour ce type de problème, il est assez courant de tracer l'évolution de la corrélation au cours du temps entre les indicateurs et la Target pour essayer de mieux visualiser ce qu'il se passe.

En effet, il se peut que certains thèmes n'aient aucune influence sur le cours du Bitcoin ou alors une influece négligeable. On a par ailleurs rajouté l'évolution de la corrélation entre les indicateurs et la variation du Bitcoin sur la dernière heure.

On a alors testé plusieurs choix d'ensembles d'indicateurs (pas tous par contrainte de temps) et toutes les modifications de features sont basées sur ces ensembles.

3.2.2 Modification des features

On a décidé d'utiliser une moyenne mobile afin de 'lisser' les données et pour enlever le bruit lié aux petites variations. On s'est inspirés de modèles en finance qui utilisent souvent ce principe. Nous n'avons pas eu le temps d'essayer beaucoup de fenêtres (des fenêtres de taille 6 et 4 ont été testées). Les petites fenêtres ont tendance à garder plus de bruit, tandis que les grandes fenêtres font perdre de l'information.

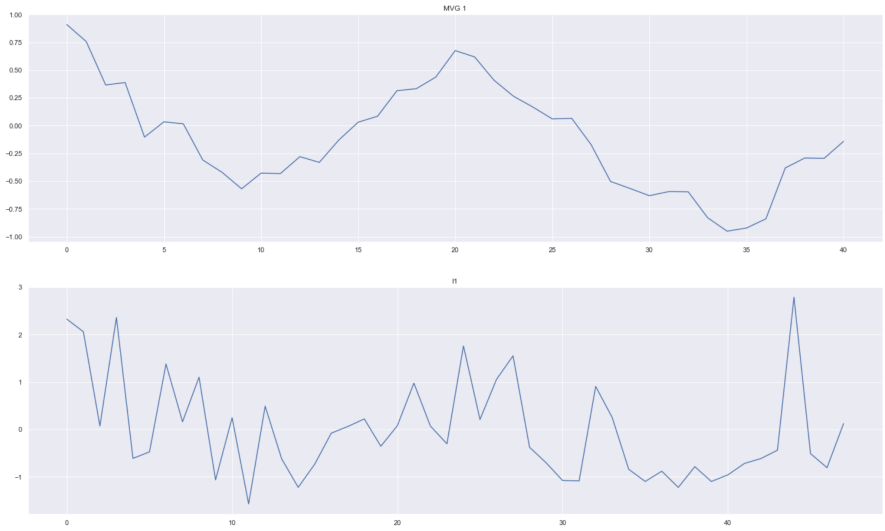

Figure 3: Evolution de la moyenne mobile et des données non modifiées pour le 1er échantillon de l'indicateur 1

On a par ailleurs voulu testé d'autres features basées sur ce document, qui montre des features utilisées en finance comme le momentum. Cependant, les contraintes de temps (temps de calculs longs) ne nous ont pas permis de les utiliser. Il serait cependant intéressant de les tester car nos données temporelles de sentiments peuvent être plus ou moins similaires à des indicateurs financiers (retours, volumes...).

Ainsi, nos nouvelles features contiendront ces moyennes mobiles, ainsi que les variations du Bitcoin sur les dernières 1h, 6h, 12h, 24h et 48h.

3.2.3 Sélection des features

La sélection des features a été importante car si l'on gardait toutes les features, le temps computationnel serait très élevé. Nous avons essayé plusieurs méthodes :

• **Corrélation de Pearson:** le coefficient de Pearson est un indice reflétant une relation linéaire entre deux variables continues. Le coefficient de corrélation varie entre -1 et +1. 0 indique une relation nulle entre les deux variables, une valeur négative (corrélation négative) signifie que lorsqu'une des variables augmente, l'autre diminue, tandis qu'une valeur positive (corrélation positive) montre que les deux variables varient ensemble dans le même sens. On avait alors à l'issue de l'algorithme des features ordonnées selon la corrélation. Il s'agissait alors de trouver le nombre optimal de features à choisir pour obtenir la meilleure prédiction. 200 features était un choix qui permettait d'avor une bonne prédiction. Cependant, dans notre cas, il y a un léger problème car notre Target est discontinue, mais nous avons ignoré ce fait car c'est avec cette méthode que nous avons eu les meilleurs résultats.

Voici la formule :

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2} \cdot \sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}$$

• **Recursive feature elimination:** Cette technique débute par la construction d'un modèle sur l'ensemble des features et associe un score d'importance à chaque feature. La(les) feature(s) la(les) moins importante(s) est(sont) enlevée(s) et le modèle est reconstruit sur le nouvel ensemble de features. On répète ce principe jusqu'à obtenir le nombre de features désiré (il s'agit donc d'un hyperparamètre à régler).

• **Analyse en composantes principales:** Ce n'est pas réellement une méthode de sélection de features mais plutôt une méthode pour réduire la dimension. L'ACP est une analyse factorielle, en ce sens qu'elle produit des facteurs (ou axes principaux) qui sont des combinaisons linéaires des variables initiales, hiérarchisées et indépendantes les unes des autres. La recherche de ces axes porte sur des axes indépendants expliquant au mieux la variabilité — la variance — des données. Ici, la réduction de dimensions via l'ACP n'apporte pas d'améliorations au niveau du modèle après différents tests.

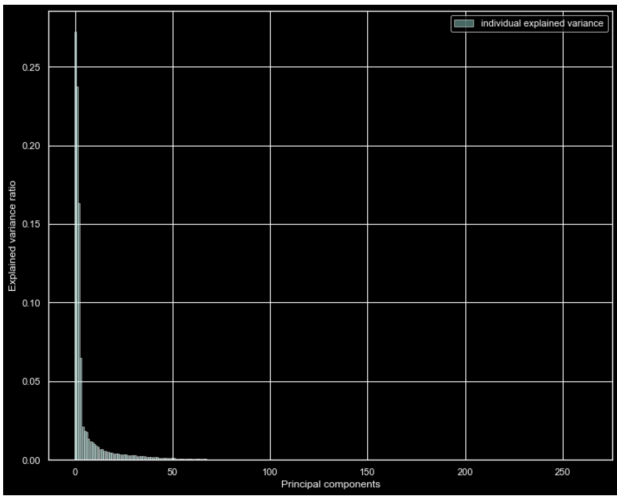

Figure 4: Explained Variance pour l'ACP

4 Manipulation des Données

Une fois les données traitées, nous avons appliqué différentes méthodes pour retourner les probabilités d'appartenance aux classes.

• **Régression Logistique :** la régression logistique est une méthode linéaire généralisée qui utilise une fonction logistique (souvent de type sigmoïde), dont on va déterminer les paramètres optimaux pour distinguer les différentes classes en sortie. Dans notre cas, nous avons effectué plusieurs régressions logistiques en trouvant le meilleur coefficient C possible (en utilisant la fonction GridSearchCV). Nos résultats se sont rapprochés du benchmark de l'énoncé avec un score de **1.0926**.

• **K plus-proches voisins :** la méthode KNN appliquée aux problèmes de classification, consiste à affecter à une donnée la classe majoritaire parmi ses k plus proches voisins. Si la méthode est facilement "calculable" puisqu'il suffit juste construire et stocker des vecteurs caractéristiques associés à des etiquettes de classes, une asymétrie du problème peut rendre la méthode peu fiable. Dans notre problème, cette méthode a montré ses limites avec un score de **1.0700** pour $k = 11$.

• **Réseaux de neurones :** la méthode RNN, à partir de couches de neurones données par l'utilisateur, va tenter d'optimiser les poids associés aux synapses pour résoudre le problème de classification. Après plusieurs tests sur la taille des couches et leurs fonctions d'activation, nous avons constaté que cette méthode n'était pas particulièrement adaptée à notre problème. En effet, l'algorithme fait preuve de sur-apprentissage et n'arrive pas à minimiser l'erreur de validation, mais a tout de même été obtenu un score final de **1.0538**.

• **Forêts aléatoires :** la méthode des Random Forests se base sur l'élaboration puis la fusion d'arbres de classication, entraînés avec la méthode du bagging. Cet alorgithme est bien adapté aux problèmes de classification, et aux problèmes contenant bon nombre de features, puisqu'il arrive à tenir compte de leur importance. En utilisant la fonction GridSearchCV sur différents paramètres comme la profondeur des forêts, le nombre de noeuds et le nombre d'estimateurs, nous avons obtenu des résultats assez probants avec un score de **1.0520**, ce qui en fait la troisième méthode la plus efficace.

• **Gradient boosting :** L'idée générale de la méthode consiste à calculer une série d'arbres de décision simple, où chaque arbre de décision consécutif est construit pour prévoir les résidus de la prévision de l'arbre précédent. On a alors une fonction de coût à minimiser.

Exemple: Considérons un algorithme de gradient boosting (XGboost ou bien lightgbm que l'on a tous deux utilisés pour ce challenge) avec M étapes. A chaque étape m ($1 \leqslant m \leqslant M$), on suppose qu'on a un modèle imparfait F_m (pour des petits m, le modèle retourne simplement $\hat{y}_i = \bar{y}$ où $\bar{y}$ est la moyenne empirique de y). Pour améliorer F_m, notre algorithme doit rajouter un nouvel estimateur, $h_m(x)$. Par conséquent : $F_{m+1} = F_m + h_m = y$. (avec y le vecteur des observations). Ainsi, le gradient boosting va ajuster h_m sur le résidu $y - F_m$.

Cette méthode nous a donné notre deuxième meilleur résultat avec un score de **1.0451** avec un learning rate de 0.01, une profondeur de 4 et 700 estimateurs.

• **Machine à vecteurs de support :** la méthode des SVM, utilisée pour des problèmes de classification et de régression, consiste à trouver les meilleurs hyperplans possibles pour séparer le jeu de données en plusieurs classes, c'est à dire les hyperplans qui permettent la plus grande marge possible avec les points d'entraînement. La méthode a pour avantage d'être précise (même si coûteuse en calculs pour de grands datasets) et est utilisée pour des problèmes impliquant l'analyse de sentiments. Après avoir appliqué la fonction GridSearchCV sur les paramètres C et γ, nous avons obtenu pour un noyau gaussien et les valeurs $C = 0.7$ et $\gamma = 0.025$, notre meilleur score qui est de **1.0331**. Confirmant ainsi que cette méthode est bien adaptée à ce type de problèmes.

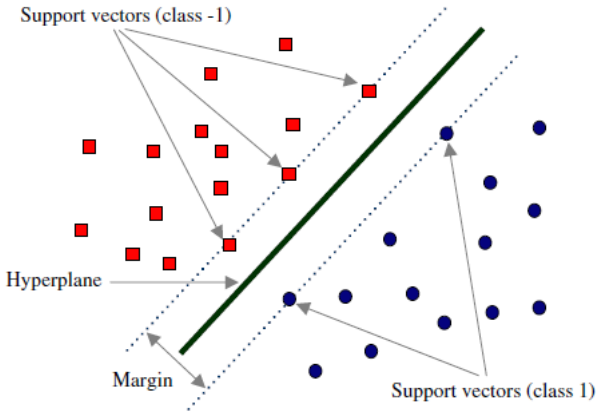

Figure 5: Schéma simplifié de la méthode SVM

5 Outils Utilisés

Pour l'élaboration du rapport, nous avons utilisé l'outil collaboratif *Overleaf*, codé en LaTeX, tandis que pour le partage et l'exécution des codes, nous avons opté pour *Google Colab*.

6 Conclusion

Les meilleurs résultats pour chaque méthode sont résumés dans le tableau ci dessous :

Méthode	**Régression Logistique**	**K-plus proches voisins**	**Réseau de Neurones**
Score	1.0926	1.0700	1.0538
Méthode	**Random Forest**	**Gradient Boosting (XGboost/LightGBM)**	**SVM**
Score	1.0520	1.0451	1.0331

La méthode des machines à vecteurs de support est donc celle qui a apporté les résultats les plus probants avec un score de **1.0331**, mais l'influence du feature engineering a été prépondérante dans la résolution.

Par ailleurs, lors de la sélection, nous avons remarqué que les variations du bitcoin sur les dernières 1h, 6h, 12h, 24h et 48h sont les features qui semblent les plus importantes. Si elles ne sont pas prises en compte, le score diminue drastiquement, ce qui est cohérent.

Cependant, la sélection des features opérée par la corrélation de Pearson ne semble pas privilégier les indicateurs de sentiments sur les dernières heures, comme on aurait pu s'y attendre. En effet, il semble logique que les sentiments sur la dernière heure aient une plus grande influence sur la variation du Bitcoin dans l'heure qui suit. Il aurait peut-être fallu prendre compte l'évolution temporelle avec une moyenne mobile exponentielle (qui met plus de poids sur les features les plus récentes) ou encore choisir des modèles tels que les réseaux de neurones LSTM.