M2 2024/2025

Université Gustave Eiffel

Rapport Challenge data Compétition de Science des Données

Christian KY
Maxime Sun
Axel Robert
Adlane Gil

M2 SSIO et M2 SIA

Groupe : Apprentis

Enseignant :

M.Jawad Alaoui

Données de marché haute fréquence : saurez-vous identifier l’action?

Table des matières

Introduction & compréhension du problème

Introduction :

Le défi auquel nous sommes confrontés consiste à identifier une action boursière à partir de séquences de données du carnet d'ordres. Ces données, qui comprennent des mises à jour atomiques sur les prix d'achat et de vente, les volumes d'ordres, et les transactions, sont essentielles pour comprendre le comportement du marché. L'objectif principal de ce projet est de développer un modèle de classification qui peut prédire avec précision l'action correspondante à partir de ces données.

Compréhension du problème

Les objectifs spécifiques de ce projet sont les suivants :

1 - Explorer les données et effectuer un traitement adéquat au problème et aux caractéristiques du jeu de données.
2 - Développer un modèle de classification qui prédit l'action boursière à partir des mises à jour du carnet d'ordres.
3 - Évaluer la performance du modèle sur un ensemble de tests distincts pour garantir sa capacité à généraliser.
4 - Analyser l'importance des caractéristiques pour comprendre les facteurs influençant les prédictions.

Défis Spécifiques aux Données

Le traitement des données pour ce projet présente plusieurs défis :

- **Données manquantes :** Les mises à jour du carnet d'ordres pourraient contenir des valeurs manquantes. La gestion de ces données est cruciale, et des stratégies telles que l'imputation ou la suppression de lignes peuvent être appliquées en fonction de l'impact sur le modèle. Après avoir étudié le jeu de données, il n'y a ici pas de données manquantes. Cependant, de manière générale, si le nombre de données manquantes est faible, le plus simple est de supprimer les lignes avec des données manquantes, préservant ainsi l'intégrité du jeu de données. Si une majorité des lignes possèdent une ou des données manquantes, ce qui est souvent le cas pour des données financières, la suppression des lignes entraînerait une perte trop importante d'informations. Des stratégies d'imputation peuvent alors être mises en place. Une imputation par la moyenne ou la médiane des valeurs

des données d'entraînement peut être une solution, mais cela peut entraîner une modification de la distribution du jeu de données, avec une concentration autour d'un seul point de données. Des techniques telles que l'imputation KNN (k-nearest neighbors) permettent de chercher les voisins les plus proches d'une observation avec valeurs manquantes, et alors lui assigner la valeur moyenne de ses plus proches voisins.

- **Complexité des Données :** Les données brutes peuvent être peu descriptives et nécessitent une ingénierie des caractéristiques pour extraire des informations pertinentes. Cela inclut la création de nouvelles variables qui capturent des tendances ou des comportements spécifiques dans les données. Nous discuterons de cette partie plus en détail après l'exploration des données.

Exploration des données & prétraitement

Analyse exploratoire des données (EDA)

L'analyse exploratoire des données (EDA) a permis de mieux comprendre la structure et la qualité des données. Ainsi on a pu remarquer certains points:

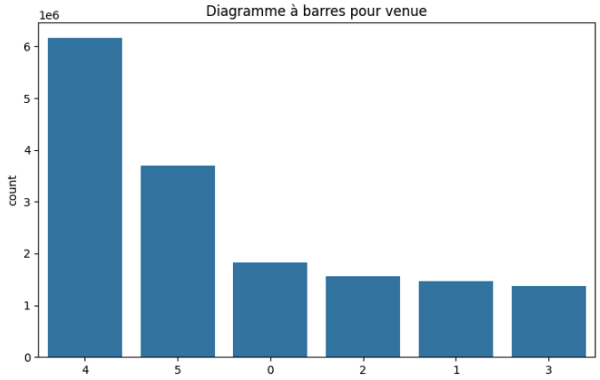
- Les données contiennent des colonnes variées, comme des caractéristiques catégorielles et des valeurs numériques que l'on va pouvoir exploiter.

- En parcourant les valeurs nous avons pu apercevoir des anomalies notamment des valeurs négatives qui ont été détectées dans les colonnes bid_size et ask_size, qui pourraient affecter les analyses ultérieures. Cela concerne très peu d'observations, et une valeur négative pour une taille d'ordre ne fait pas de sens financièrement. Cela nous amène donc à dire que c'est une erreur, donc nous pouvons enlever ces observations du jeu d'entraînement afin de ne pas amener de bruits supplémentaires.

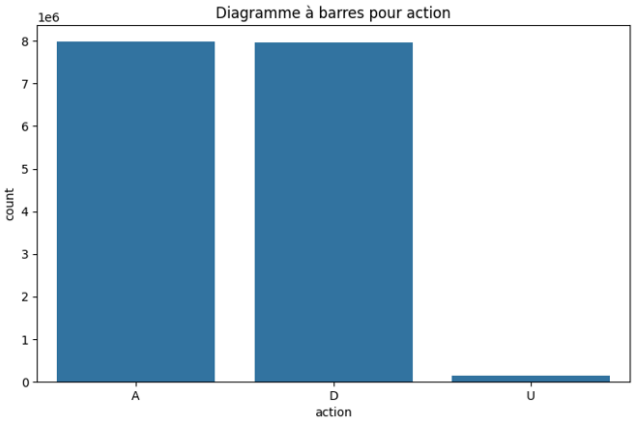
- Lors de l'analyse des classes cibles, nous avons constaté une distribution uniforme pour toutes les classes. Si cela reflète la réalité des données, alors les conditions pour le modèle sont idéales. En effet, lorsqu'il y a de forts déséquilibres dans la distribution des classes cibles, cela peut conduire le modèle à ne concentrer sa prédiction que sur la classe prépondérante afin de maximiser l'accuracy, en délaissant les autres classes, ce qui pourrait avoir un impact négatif sur les performances du modèle dans les données réels. Par exemple, si 95% des donnés sont de la catégorie 0, un « modèle » factice ne prédisant que des 0 va avoir une accuracy de 95%, qui est donc une référence très difficile à battre. Hors ce n'est pas un modèle performant en réalité.

```
eqt_code_cat 0: 6700 fois
eqt_code_cat 1: 6700 fois
eqt_code_cat 2: 6700 fois
eqt_code_cat 3: 6700 fois
eqt_code_cat 4: 6700 fois
eqt_code_cat 5: 6700 fois
eqt_code_cat 6: 6700 fois
eqt_code_cat 7: 6700 fois
eqt_code_cat 8: 6700 fois
eqt_code_cat 9: 6700 fois
eqt_code_cat 10: 6700 fois
eqt_code_cat 11: 6700 fois
eqt_code_cat 12: 6700 fois
eqt_code_cat 13: 6700 fois
eqt_code_cat 14: 6700 fois
eqt_code_cat 15: 6700 fois
eqt_code_cat 16: 6700 fois
eqt_code_cat 17: 6700 fois
eqt_code_cat 18: 6700 fois
eqt_code_cat 19: 6700 fois
eqt_code_cat 20: 6700 fois
eqt_code_cat 21: 6700 fois
eqt_code_cat 22: 6700 fois
eqt_code_cat 23: 6700 fois
```

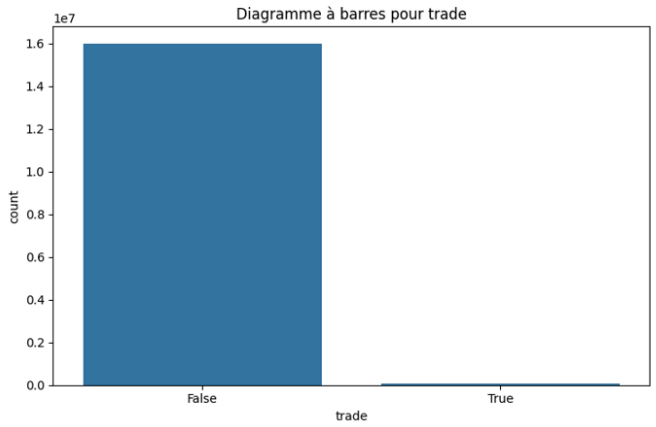
- Nous avons ensuite effectué une analyse des caractéristiques, en commençant par les colonnes catégoriques (venue, action et trade) qui nous ont données des informations importantes:



Pour la colonne venue, il existe une concentration significative des observations dans certaines catégories (ici la valeurs 4 et 5). Cela pourrait indiquer que certaines places boursières sont plus actives que d’autres et pourraient influencer les stratégies de trading.

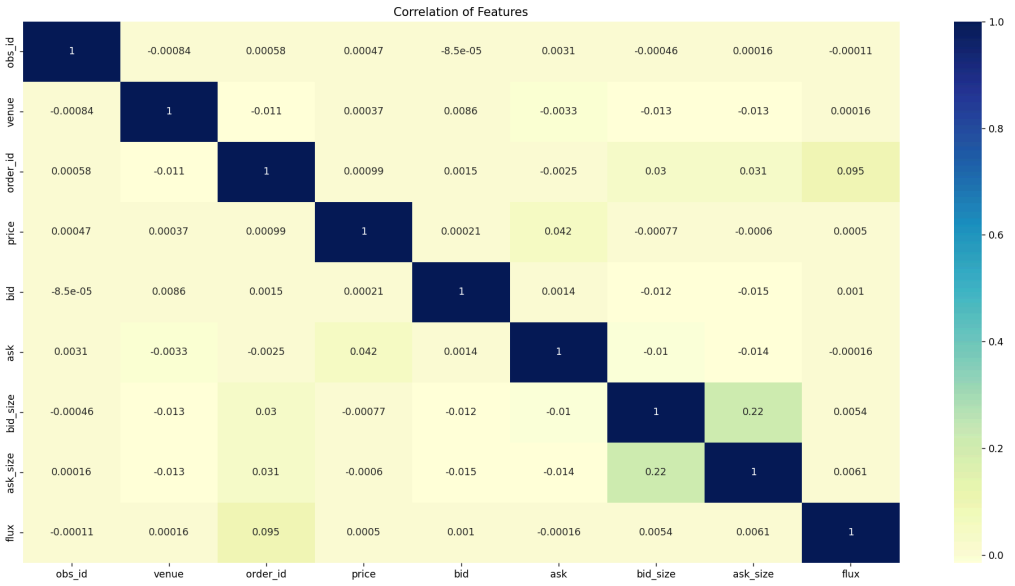


La colonne action présente une distribution inégale, avec des valeurs A (ajout d'un nouvel ordre) et D (suppression d'un ordre) ayant une fréquence élevée, tandis que la valeur U (mise à jour d'un ordre) est très peu représentée. Cela suggère que les événements d'ajout et de suppression d'ordres sont plus fréquents que les mises à jour, ce qui pourrait influencer la dynamique du marché.



Concernant la colonne trade, il existe un déséquilibre marqué entre les valeurs True et False, avec une majorité d'observations classées comme False. Cela indique que la plupart des enregistrements ne correspondent pas à des transactions effectives. Dans ce contexte, trade est un booléen qui indique si un événement de suppression ou de mise à jour est dû à une vente (True) ou à une annulation (False).

L'analyse de la matrice de corrélation a révélé des relations intéressantes entre les différentes caractéristiques numériques du jeu de données. La plupart des corrélations observées sont faibles, ce qui indique que les variables sont relativement indépendantes les unes des autres. Cependant, une corrélation positive notable a été identifiée entre les colonnes bid_size et ask_size, suggérant que lorsque la taille des ordres d'achat augmente, la taille des ordres de vente a également tendance à augmenter. Cette relation pourrait être indicative d'une dynamique de marché où une augmentation de l'intérêt des acheteurs entraîne une réponse similaire de la part des vendeurs. En revanche, les autres variables, telles que price, bid, et ask, montrent des corrélations faibles avec les tailles d'ordre, ce qui pourrait indiquer que les prix ne sont pas directement influencés par le volume des ordres à un moment donné. Cette matrice de corrélation nous aide à mieux comprendre les interactions entre les différentes caractéristiques et peut orienter les choix de modélisation en identifiant les variables les plus pertinentes à inclure dans les modèles prédictifs.



Discuter des étapes de nettoyage des données, de l'ingénierie des caractéristiques et des transformations.

Nettoyage des Données

Le nettoyage des données est une étape cruciale pour garantir la qualité et la fiabilité des analyses. Dans ce cas, les valeurs négatives dans les colonnes bid_size et ask_size ont été supprimées. Ces valeurs peuvent indiquer des erreurs de saisie ou des anomalies dans les données, et leur présence pourrait fausser les résultats des analyses ultérieures. Cela préserve l'intégrité des données et permet au modèle de fonctionner sans erreurs, garantissant ainsi une préparation adéquate pour l'apprentissage automatique.

Ingénierie des Caractéristiques

L'ingénierie des caractéristiques joue un rôle clé dans l'amélioration des performances des modèles. Les colonnes catégorielles venue et action ont été transformées en variables indicatrices (one-hot encoding), ce qui permet au modèle de mieux interpréter ces informations. La catégorie trade étant déjà une colonne binaire, aucune transformation n'est nécessaire. Cette transformation est particulièrement importante pour les algorithmes qui ne peuvent pas traiter directement les variables catégorielles. De plus, des transformations logarithmiques ont été appliquées à certaines colonnes pour normaliser les distributions. Cela aide à atténuer l'impact des valeurs extrêmes, rendant les données plus adaptées à l'apprentissage automatique. En effet, les catégories de flux, ask_size et bid_size apportent certainement beaucoup d'information concernant la liquidité d'une action (et donc à catégoriser l'action en elle même), mais la valeur brute est difficile à gérer pour un modèle de Machine Learning, puisque la distance entre 10 et 100 est aussi importante qu'entre 10010 et 10100. Hors dans le marché financier, les flux et taille d'ordre sont plutôt comparés à la taille des entreprises, et donc un pourcentage par rapport au nombre initial plutôt qu'une distance. Une transformation logarithmique est donc plus adaptée.
Le problème de la transformation logarithmique est qu'elle ne supporte pas de valeurs négatives. Cela ne pose pas de problème pour ask_size et bid_size. Cependant, il existe beaucoup de flux négatifs (ce qui fait sens). Nous avons donc choisi d'utiliser le logarithme de la valeur absolue du flux. En règle générale, cela peut induire une perte d'information et nous pourrions rajouter une variable catégorique afin d'indiquer si le flux initial est positif ou négatif. Cependant, ici tous les flux positifs ont pour colonne Actions « A » et les flux négatifs correspondent aux Actions « D » et « U ». Il n'y a donc ici aucune perte d'information. Enfin, lorsque le flux est nul, la transformation logarithmique assigne alors une valeur - infinie, pouvant entraîner des résultats indésirables. Pour y remédier, toutes les occurrences de valeurs infinies négatives (-inf) ont été remplacées par 0 pour préserver le flux nul initial.

Transformations des Données

L'utilisation de ColumnTransformer a permis d'appliquer différentes transformations de manière systématique et efficace. Ce préprocesseur a facilité le passage direct pour certaines colonnes, l'encodage pour les variables catégorielles, et les transformations logarithmiques pour les colonnes numériques. Cela a permis de maintenir une structure de données propre et organisée. Enfin, la préparation des tenseurs, en regroupant les données par obs_id, a assuré que les données étaient dans un format adéquat pour l'entraînement du modèle. Cette étape est essentielle pour garantir que le modèle puisse traiter les données de manière efficace et performante.
Nous avons effectué 2 types de transformations différentes selon l'utilisation du modèle. Pour les réseaux de neurones, l'utilisation d'un tenseur de taille (100, 16) en regroupant les obs_id permet d'utiliser le GRU et exploiter au mieux les relations temporelles entre ces mêmes observations. Pour la régression linéaire et le Random Forest, nous avons dû concaténer les caractéristiques d'un même obs_if afin d'obtenir une observation de taille 100x16. Cela est moins optimale que la méthode précédente, car nous perdons toute relation temporelle, mais la plus grande simplicité des modèles requiert une traitement différent puisqu'on ne peut pas utiliser de GRU ou autres modèles de RNN (Recurrent Neural Network)

Toutes ces étapes ont permis de préparer les données pour l'apprentissage du modèle pour augmenter les chances d'obtenir des résultats précis et fiables**.**

Approches méthodologiques avec avantages & inconvénients

1. Régression Linéaire

Nous avons décidé d'utiliser la régression linéaire pour la première approche car nous avions pu la voir en cours et c'est également un moyen pour nous de découvrir un peu le modèle sur un projet plus concret et difficile. Cette approche nous permet de mettre en pratique les concepts théoriques que nous avons appris, tout en nous familiarisant avec les outils et les techniques nécessaires pour traiter des données réelles.

En appliquant la régression linéaire, nous avons pu bénéficier de plusieurs avantages, notamment la rapidité d'entraînement et la facilité d'interprétation des résultats. Cela nous a permis d'analyser rapidement les impacts des différentes caractéristiques sur notre variable cible, tout en identifiant les relations linéaires potentielles.

Cependant, nous restons conscients des limites de cette approche. Nous devons veiller à respecter les hypothèses de la régression linéaire. De plus, nous avons constaté après soumission des résultats dans ChallengeData un score assez moyen ainsi ce modèle n'était pas le plus adapté pour ce challenge.

4	1 janvier 2025 18:21	Modele Lineaire	rien	0,14745098039215687

Cela était attendu, car les relations dans les données financières sont rarement linéaires. Les plus grosses limites de la régression linéaire est qu'elle ne peut pas prendre en compte et donc modéliser les relations complexes non linéaires entre variables indépendantes et dépendantes, mais également les potentielles interactions entre variables dépendantes.

En conclusion, bien que la régression linéaire soit une approche utile pour commencer, son utilisation dans ce projet représente également une étape importante dans notre apprentissage. Elle nous permet d'appliquer nos connaissances théoriques à un cas pratique tout en nous préparant à explorer des méthodes plus avancées si nécessaire. Il est essentiel d'explorer d'autres modèles et techniques pour améliorer la performance et mieux répondre aux exigences des données, notamment en tenant compte des transformations nécessaires et des relations non linéaires qui pourraient exister. Cela nous conduit donc à explorer le modèle suivant de Random Forest.

2. Random Forest

Pour la 2e approche nous avons décidé d'utiliser la méthode Random Forest car c'est un algorithme largement reconnu en machine learning. Cet algorithme repose sur l'assemblage d'arbres de décision, en effet elle construit une multitude d'arbres de décision lors de l'entraînement et produit la classe qui est la mode des classes ou la moyenne des prédictions des arbres individuels. Chaque arbre est construit à partir d'un sous-ensemble aléatoire des données d'entraînement et des colonnes caractéristiques, ce qui contribue à la diversité des modèles et améliore la robustesse des prédictions en diminuant le risque de surapprentissage. En effet, un des risques majeurs des modèles de Machine Learning est le surapprentissage, c'est-à-dire que le modèle obtient une très bonne performance sur les données d'entraînement, mais ne se généralise pas car il a modélisé des bruits qui proviennent seulement du jeu d'entraînement, mais qui ne se reproduisent pas en réalité.

Random Forest est un algorithme intuitif et facile à comprendre, ce qui nous à permis de prendre en main assez facilement. Il est également rapide à entraîner notamment sur de grands ensembles de données. Il produit des résultats généralisables en réduisant le risque de surapprentissage par la combinaison des prédictions de plusieurs arbres.

De plus, en soumettant notre résultat dans ChallengeData nous avons obtenu un score très similaire à celui obtenu avec la méthode Régression Linéaire. Ce qui nous à permis de nous dire que ce n'était pas la meilleure approche pour ce challenge.

5	1 janvier 2025 23:35	La foret aleatoires	n_estimators: 1000, criterion:entropy, random_state:42, n_jobs:-1, verbose:1	0,14588235294117646

Ici, bien que le modèle de Random Forest peut exploiter des relations complexes non linéaires entre variables indépendantes et dépendantes ainsi que des interactions entre variables dépendantes, nous ne pouvons ici pas exploiter la relation temporelle des ordres au sein de chacun obs_id. En effet, lorsqu'un arbre est construit, les colonnes possibles servant à différencier les groupes afin de constituer les branches de l'arbre sont traités de manière indépendantes, puisque l'algorithme choisit à chaque profondeur la variable qui permet de séparer les observations en 2 groupes avec des groupes les plus différents entre elles mais des observations les plus similaires au sein de chaque groupe.

En résumé, l'utilisation de Random Forest dans ce projet nous permet de tirer parti de ses nombreux avantages tout en nous offrant une approche robuste pour traiter des données complexes. Cela nous aide à améliorer la performance de notre modèle et à mieux répondre aux exigences des données, mais les résultats de celui-ci n'était pas assez satisfaisant pour autant.

3. Réseau de neurones

Pour notre troisième approche, nous avons décidé d'utiliser un réseau de neurones basé sur des unités GRU. Cette méthode est particulièrement adaptée en raison de la nature séquentielle des données, où l'ordre des mises à jour du carnet d'ordres joue un rôle crucial dans l'identification des actions, ce qui en fait un choix pertinent pour notre problème.

Pour ce challenge, cette approche présente plusieurs avantages significatifs. Les GRU sont particulièrement adaptés aux données séquentielles, ce qui leur permet de prendre en compte l'ordre des mises à jour du carnet d'ordres, essentiel pour une classification précise. Leur capacité à capturer des dépendances temporelles facilite la mémorisation d'informations pertinentes sur de longues périodes. De plus, les GRU sont rapides à entraîner et comportent moins de paramètres, ce qui réduit le risque de surajustement et améliore la généralisation. Leur flexibilité permet d'intégrer facilement des couches denses et d'autres mécanismes d'attention. Enfin, l'utilisation de la fonction de perte cross-entropy avec cette architecture est efficace pour les tâches de classification multi-classes, rendant cette approche particulièrement pertinente pour identifier les 24 actions dans ce challenge. L'utilisation d'une couche de dropout (oubli) permet également de diminuer le risque de surapprentissage, qui est très prépondérant lors d'utilisation de réseaux de neurones (récurrents).

Enfin, les réseaux de neurones donnent la possibilité d'effectuer un entraînement par batch, ce qui est optimal pour la descente du gradient stochastique afin d'éviter d'être coincé dans des minimum locaux de la fonction de perte. Cela permet également de contrôler l'évolution de la performance à la fois sur le jeu d'entraînement et le jeu de validation, afin de contrôler qu'il n'y a pas un surentraînement lorsque la fonction de perte du jeu d'entraînement continue à diminuer sans contribuer à la diminution de la fonction de perte du jeu de validation.

Mais ce modèle présente certains inconvénients, lorsque l'on utilise avec un ensemble de données volumineux il nécessite un temps d'entraînement considérable dans notre cas plus de 5h. L'utilisation de GRU peut entraîner une surcharge de calcul, en particulier si le modèle est complexe ou si des hyperparamètres doivent être ajustés, ainsi nous avions dû faire attention à celà en ajoutant des dropouts pour éviter la surcharge de calcul.

En conclusion, l'utilisation d'un réseau de neurones GRU dans ce projet nous permet d'exploiter la puissance des modèles séquentiels pour améliorer la précision des prédictions. Cette approche représente une avancée par rapport aux méthodes précédentes car le résultat obtenu par rapport aux autres modèles précédents est bien meilleur.

Sélection, Réglage et Validation du modèle

1. Stratégie de réglage d'hyperparametres

La recherche d'hyperparamètres pour des modèles comme le random forest et les réseaux de neurones est essentielle pour optimiser leurs performances, car ces hyperparamètres influencent directement le comportement, la complexité, et la capacité prédictive du modèle.

La performance du Random Forest dépend beaucoup des caractéristiques de la « forêt », notamment le nombre d'arbres (un nombre insuffisant d'arbre va produire un modèle sous-entraîné, tandis qu'un nombre trop important va conduire à un entraînement long sans augmenter la performance du modèle), la profondeur des arbres (une trop grande profondeur peut conduire à un surapprentissage), nombre de caractéristiques maximale pour la division afin de contrôler la diversité des arbres (si tous les arbres se ressemblent, on perd l'avantage du Random Forest à être un modèle d'ensemble permettant de diminuant le risque de surapprentissage), taille minimale des feuilles ou split pour ne pas rendre les arbres trop complexes et moins significatifs.

De même, les réseaux de neurones sont également sensibles aux différents hyperparamètres. Le taux d'apprentissage est un des paramètres le plus important, puisqu'il contrôle la convergence du modèle (un taux trop élevé peut empêcher la

convergence car l'impossibilité de se retrouver dans un minimum, un taux trop faible réduit considérablement l'apprentissage). Le nombre de couches et de neurones par couche jouent également un rôle important, avec une optimal à trouver entre capter des relations complexes et risques de surapprentissage. Les fonctions d'activation peuvent changer grandement les performances, ainsi que la taille des mini-lots (batch size), le nombre d'époque et le taux d'oubli (dropout).

Différentes techniques de recherche d'hyperparametres existent, notamment les suivantes : la recherche par grille, la recherche aléatoire et l'optimisation bayésienne.

Chacune des techniques possèdent avantages et inconvénients :

1. La recherche par grille a pour avantage la simplicité d'utilisation et de compréhension, l'exhaustivité des recherches (sur l'espace des hyperparamètres de recherche) et la reproductibilité. Cependant, le coût computationnel extrêmement élevé rend en réalité cette technique inefficace, puisque la grille de recherche sera alors réduite pour réussir à finir la recherche, ce qui va probablement conduire à passer à côté des hyperparamètres optimaux.
2. La recherche aléatoire est une légère modification de la recherche par grille. Au lieu d'effectuer la recherche sur toute la grille, des hyperparamètres de la grille vont être tirés aléatoirement un certain nombre de fois. La caractéristique probabiliste de cette technique permet de trouver une combinaison presque optimale en réduisant drastiquement le coût computationnel
3. Enfin, l'optimisation bayésienne permet d'orienter les recherches suivantes en fonction des performances des hyperparamètres précédents, permettant alors de trouver un optimal en moins d'essai que la recherche aléatoire. Cependant, elle possède des inconvénients tels que la possibilité que la recherche soit « coincée » dans un optimum local (bien que des techniques de semi-aléatoires peuvent être mises en place pour éviter ce défaut). Cette fonction bayésienne peut cependant être coûteuse lorsqu'un grand ensemble de données est utilisé. C'est une technique plus complexe à mettre en place.

Dans ce projet, nous avons choisi d'utiliser la recherche aléatoire pour le Random Forest. En effet, l'importance du jeu de données et la complexité à bien utiliser l'optimisation bayésienne, ainsi que le coût computationnel de la recherche par grille nous a dirigé vers ce choix.
Pour le réseau de neurones, bien qu'il aurait été mieux d'effectuer une recherche d'hyperparametres, le temps considérable pour l'entraînement (plus de 5h pour les paramètres de bases) ne nous a pas permis d'effectuer une recherche complète. Nous avons donc choisi d'utiliser les hyperparamètres par défaut ou conseillés par le projet. Nous avons pu tester quelques variantes de ces hyperparamètres, et n'avons pas constaté de grandes différences avec celles utilisées, d'où le fait de ne pas avoir effectué une recherche d'hyperparametres.

2. Cadre de validation

Le cadre de validation est essentiel pour évaluer les performances d'un modèle de Machine Learning. Il est primordial d'évaluer les performances sur un jeu de donnée « nouveau », qui n'a pas été utilisé pour l'entraînement, afin de mesurer la capacité du modèle à généraliser sur des données nouvelles, et donc d'obtenir un aperçu des performances réelles du modèle.

Lors de la recherche d'hyperparametres avec le Random Forest, nous avons utilisé une validation croisée (cross-validation). Cela implique la division du jeu de données en k-sous ensemble, la moyenne des performances sur les k-itérations est donc utilisée. Cela permet de réduire le risque de biais lié à un sous-ensemble de données, et fournit une estimation plus robuste de la performance. Cependant, cela augmente le temps de calcul drastiquement (k-fois le nombre de recherche).

Pour le réseau de neurones, étant donné le temps d'entraînement et la volumétrie du jeu de données, une validation simple (train/validation set de taille 80%/20%) a été utilisée. Bien que ce cadre est moins robuste, il nous permet tout de même de bien contrôler l'évolution de performances du modèle lors des différentes mini-lots utilisées, et donc de minimiser le risque de surentraînement. L'early stopping mis en place pour l'entraînement du réseau de neurone est donc utilisé afin d'arrêter l'entraînement si la fonction de perte dans le jeu de validation ne diminue plus au bout d'un certain nombre de batchs.

Solution finale choisie & analyse approfondie

1. Approche la plus performante

Comme détectée par l'accuracy sur la validation set et le score obtenue lors de la soumission pour le test set, l'approche la plus performante est le réseau de neurones avec GRU.

Rang	Date	Méthode	Paramètres	Score public	Sélection
1	2 janvier 2025 18:41	ReseauNeuronnal	moi	0,3679411764705882	Sélectionné

C'est donc ce modèle que nous avons retenu. La possibilité d'exploiter les dépendances temporelles et séquentielles du carnet d'ordres expliquent certainement la différence de performance avec la régression linéaire et le Random Forest.
Les analyses de résultats nous montrent que le choix d'un modèle dépend fortement de la problématique et du jeu de données. Le Random Forest, bien qu'en majorité beaucoup plus performant et complexe que la régression linéaire, ne donne ici qu'une amélioration faible du score. Cela est dû au fait qu'on perd toute information sur la temporalité du carnet d'ordres.
Bien que le temps d'entraînement du réseau de neurones est très important, ce qui nous a empêché de mettre en place toutes les stratégies de recherche d'hyperparametres, la différence de performance nous amène à choisir ce modèle.

2. Interprétation des prédictions modèle avec SHAP

L'analyse de SHAP (SHapley Additive exPlanations) est une méthode d'interprétabilité pour expliquer les prédictions d'un modèle. SHAP quantifie l'impact de chaque caractéristique (feature) sur la prédiction d'un modèle, en attribuant une valeur de contribution à chacune d'elles.

Il permet de répondre à des questions comme :

- Pourquoi le modèle a-t-il fait cette prédiction spécifique? SHAP va identifier pourquoi une prédiction spécifique a été faite en décomposant la contribution des caractéristiques. Par exemple, SHAP peut nous dire qu'une obs_id a été catégorisée comme de la classe 1 car le bid_size élevé a augmenté la probabilité de 30% que l'observation soit de cette catégorie, mais le fait qu'elle soit tradé dans la venue 2 a baissé la probabilité de 5%
- Quelle est l'importance globale et locale des caractéristiques ? SHAP va classer l'importance moyenne des caractéristiques pour comprendre quels facteurs influencent le plus le modèle. Par exemple, SHAP peut nous indiquer que 30% de la probabilité est influencée par la caractéristique de flux.
- Quelles caractéristiques influencent le plus (positivement ou négativement) les prédictions. Par exemple, SHAP peut nous indiquer qu'un flux important

augmente la probabilité que l'action soit de classe 1, mais un flux faible baisse cette probabilité
- Quelles sont les interactions entre les différentes caractéristiques ?

SHAP est une librairie simple d'utilisation très visuelle, et propose les notamment visualisations suivantes :

- Summary plot : Montre l'importance globale des caractéristiques et leur impact directionnel (positif ou négatif).
- Force plot : Visualisation intuitive des contributions des caractéristiques à une prédiction spécifique.
- Dependence plot : Illustre la relation entre une caractéristique et la sortie du modèle, tout en mettant en évidence les interactions avec une autre caractéristique.
- Bar plot : Classement des caractéristiques par importance moyenne.

En plus de donner une interprétabilité au modèle, cela peut également être utile afin d'améliorer le modèle. En effet, on peut alors supprimer des caractéristiques peu utiles ou redondantes, ou bien raffiner la transformation des caractéristiques importantes afin d'améliorer les performances du modèle.

Cependant, malgré notre volonté d'explorer les prédictions des modèles avec SHAP, nous n'avons malheureusement pas réussi du fait de l'incompatibilité de SHAP avec notre modèle de réseau de neurones GRU. En effet, SHAP ne s'apparente pas bien aux données temporelles, puisqu'il est alors difficile d'expliquer à la fois la contribution séquentielle et l'interaction entre les caractéristiques en même temps.

Puis nous avons essayé de faire un modèle SHAP sur notre modèle Random_Forest mais malheureusement nous avons eu un problème de mémoire. En effet l'un des inconvénients les plus importants de SHAP est la complexité de calcul, notamment lorsque le nombre de caractéristiques est important puisque SHAP doit calculer les interactions entre elles (complexité quadratique). Le fait de devoir agréger les 100 ordres atomiques pour chaque obs_id nous fait exploser le nombre de caractéristiques, donc la complexité de calcul et l'explosion de la mémoire.

```
Traceback (most recent call last):
  File "e:/ChallengeData/ChallengeData/RandomF.py", line 116, in <module>
    shap_values = explainer.shap_values(x_test_flat)
  File "C:\Users\drago\AppData\Local\Programs\Python\Python38\lib\site-packages\shap\explainers\_tree.py", line 470, in shap_values
    phi = np.zeros((X.shape[0], X.shape[1]+1, self.model.num_outputs))
numpy.core._exceptions.MemoryError: Unable to allocate 23.4 GiB for an array with shape (81600, 1601, 24) and data type float64
```

3. Pistes d’amélioration

Il est certainement possible d’améliorer les performances du modèle. Une première piste d'amélioration réside dans le traitement des données et l'ingénierie des caractéristiques. En approfondissant l'analyse des données, il serait bénéfique d'explorer davantage les relations entre les différentes variables et de créer de nouvelles caractéristiques pertinentes. De plus, une normalisation ou une standardisation plus rigoureuse des données pourrait faciliter la convergence lors de l'entraînement des réseaux de neurones.

La recherche d’hyperparamètres étant clé pour les réseaux de neurones, il est également possible de trouver des hyperparamètres qui donneraient de meilleurs résultats. L'utilisation de cartes graphiques (GPU) ou d'ordinateurs plus puissants pourrait considérablement accélérer l'entraînement, permettant ainsi de tester un plus grand nombre de configurations d'hyperparamètres dans un laps de temps réduit. Une approche intéressante serait de réaliser une première estimation des hyperparamètres optimaux en effectuant des tests sur un entraînement partiel. Cela permettrait d'affiner les réglages avant de procéder à un entraînement complet avec ces hyperparamètres, maximisant ainsi les chances d'améliorer les performances du modèle.

Enfin, il serait intéressant également d’explorer les améliorations de SHAP afin de pouvoir traiter les séquences temporelles et donc pouvoir interpréter les résultats du réseau de neurones avec GRU.

Avec ces idées d’amélioration, il serait possible d’améliorer les performances du modèle et d’obtenir un meilleur score.

Conclusion & leçons apprises

Ce projet a permis d'explorer l'utilisation de modèles d'apprentissage automatique pour prédire des actions boursières à partir de données de carnet d'ordres. Nous avons constaté que les réseaux de neurones, en particulier ceux utilisant des unités GRU, ont surpassé les autres approches utilisées comme la régression linéaire et Random Forest, soulignant l'importance des dépendances temporelles dans les données financières.

Cependant, plusieurs défis ont été rencontrés tout au long du projet. La gestion des données, notamment le traitement des valeurs aberrantes et l'ingénierie des caractéristiques, s'est révélée cruciale pour garantir la qualité des entrées du modèle. De plus, la recherche d'hyperparamètres pour les réseaux de neurones a été limitée par le temps d'entraînement élevé, ce qui a restreint notre capacité à optimiser pleinement le modèle. L'incompatibilité de certaines techniques d'interprétabilité, comme SHAP, avec notre modèle GRU a également constitué un

obstacle, soulignant l'importance de choisir des outils compatibles dès le début du projet.

Les leçons tirées incluent l'importance d'une préparation rigoureuse des données et d'une ingénierie des caractéristiques bien pensée, qui peuvent considérablement améliorer les performances du modèle. De plus, il est essentiel de planifier les ressources nécessaires pour l'entraînement des modèles afin d'explorer pleinement les différentes configurations d'hyperparamètres.

En conclusion, ce challenge nous a permis de comprendre les différents modèles utilisés dans le Machine Learning, en particulier dans le domaine financier.

72	**2 janvier 2025 18:41**	**shootoku**	**0,3679**
73	20 septembre 2024 14:42	momo	0,3668
74	6 décembre 2024 23:39	coucou	0,3658
75	17 février 2024 12:30	emiled	0,3648
76	9 février 2024 14:55	dheurtel	0,3640
77	13 décembre 2024 12:01	MercedesHacker	0,3629
78	20 novembre 2024 14:39	karlsaliba	0,3617
79	25 juin 2024 21:08	1+1_3	0,3615
80	11 mars 2024 23:26	MSU	0,3611
81	20 novembre 2024 10:56	zakil-22	0,3606
82	20 juin 2024 21:10	Le_Navil	0,3599
83	3 décembre 2024 21:52	mrochk	0,3594
84	**-**	**benchmark**	**0,3593**

Le compte utiliser dans le Challenge Data est : shootoku