École Centrale de Nantes

APRENTISSAGE STATISTIQUE : DATA CHALLENGE
ENS
OPTION INFOIA

Données de marché haute fréquence : classification des actions

---

Élèves :
Youssef SALHI
Karl SALIBA

Encadrants :
Bertrand MICHEL

24 novembre 2024

Table des matières

1 Introduction

1.1 Présentation du problème

Le défi consiste à identifier à quel stock appartient un ensemble de données d'échange recueillies au niveau de l'ordre par ordre (tick-by-tick). Ces données incluent des mises à jour détaillées du carnet d'ordres, comme les meilleurs prix d'achat et de vente, les volumes, les transactions effectuées, ainsi que les ordres ajoutés ou supprimés. Chaque séquence de données contient des indices subtils, tels que l'écart moyen entre les prix, les quantités typiques des actions, la fréquence des transactions ou la répartition des échanges sur plusieurs places boursières. L'objectif est d'exploiter ces caractéristiques pour classer correctement les données en fonction des 24 actions disponibles, en utilisant des observations issues de périodes de temps distinctes pour l'apprentissage et les tests.

Pour ce faire, nous avons principalement utilisé Jupyter Notebook pour la rédaction du code. Par ailleurs, étant donné la taille du jeu de données, nous avons entraîné nos modèles sur Kaggle. En effet, Kaggle est un site connu à l'échelle mondiale qui héberge de nombreuses compétitions de machine learning. Le site offre également des machines en cloud équipées de puissants CPU, GPU et même TPU. Enfin, la rédaction de ce rapport a été réalisée avec Overleaf.

Figure 1 – CFM Logo

2 Analyse des données

Il s'agit d'un problème de classification de séries temporelles multivariées. Chaque série est définie par 100 observations consécutives, décrites par des caractéristiques telles que *price*, *bid*, *ask*, etc. Ces observations sont regroupées par `obs_id`, qui correspond à une séquence associée à un stock spécifique. L'objectif est de prédire l'étiquette `eqt_code_cat`, représentant le stock (parmi 24 possibles), en fonction des dynamiques et des caractéristiques des séries temporelles.

2.1 Particularités du problème

— **Entrées multivariées :** Chaque séquence (`obs_id`) est composée de 100 observations, chacune décrite par plusieurs caractéristiques (*price*, *bid*, *ask*, etc.).
— **Classification :** La tâche consiste à associer chaque séquence (`obs_id`) à une catégorie spécifique (`eqt_code_cat`), correspondant au stock associé.

— **Non-linéarité et complexité temporelle :** Les variations au sein des séries temporelles (par exemple, fluctuations des prix, tailles des carnets d'ordres, actions sur les ordres) contiennent des indices utiles pour déterminer le stock associé.
— **Variabilité des données :** Les données incluent des mises à jour provenant de différentes bourses et de différents jours, ce qui ajoute une complexité supplémentaire à la classification.

3 Transformation des données et ingénierie des features

Pour rendre les données exploitables par des modèles d'apprentissage , une ingénierie des caractéristiques (*feature engineering*) a été réalisée. Les transformations suivantes ont été appliquées pour extraire des informations pertinentes et améliorer la performance des modèles. Nous présentons ici les transformations les plus importantes.

3.1 Statistiques globales sur les variables clés

Pour chaque séquence (`obs_id`), les statistiques suivantes ont été calculées pour les colonnes *price*, *bid*, *ask*, *bid_size*, *ask_size*, et *flux* :
— Moyenne, écart-type, minimum, maximum, amplitude (*range*),
Ces statistiques permettent de capturer la distribution globale et la variabilité des caractéristiques clés. Par exemple, l'écart-type des prix ou le volume échangé peuvent refléter des propriétés spécifiques de chaque stock, essentielles pour différencier les comportements dynamiques.

3.2 Caractéristiques liées au spread

Le spread, défini comme la différence entre le meilleur prix d'achat (*bid*) et le meilleur prix de vente (*ask*), est une métrique critique en finance. Les caractéristiques suivantes ont été extraites :
— Moyenne du spread,
— Écart-type du spread.
Ces caractéristiques permettent d'évaluer la liquidité et la stabilité d'un stock, deux indicateurs clés dans l'analyse des marchés financiers.

3.3 Caractéristiques liées aux bourses (*venues*)

Les données incluent des informations sur les bourses (*venues*) où les transactions ont lieu. Les caractéristiques suivantes ont été dérivées :
— Nombre de *venues* uniques par séquence (`venue_unique`),
— Fréquence des *venues* spécifiques (`venue_0_count`, etc.).
Ces métriques permettent de distinguer les actions en fonction de leur dispersion sur différents marchés et de leur dynamique d'échange.

3.4 Importance de l'autocorrélation dans la caractérisation des séries temporelles

L'autocorrélation est un outil clé pour capturer les dépendances temporelles dans les séries temporelles des flux d'ordres. Elle joue un rôle crucial dans l'analyse et la différenciation des stocks en fournissant des indices sur leurs comportements dynamiques. Voici les principales raisons pour lesquelles elle est utilisée :

3.4.1 Identifier des motifs temporels spécifiques aux actions

Certains stocks présentent des comportements répétitifs ou des tendances spécifiques, qui peuvent être révélés par l'autocorrélation. Par exemple :

- Une **autocorrélation positive** indique que les valeurs de flux tendent à se répéter (par exemple, des augmentations ou des diminutions consécutives).
- Une **autocorrélation négative** suggère des inversions fréquentes (hausse suivie de baisse, et vice versa).

Ces motifs temporels peuvent être caractéristiques d'un stock particulier et permettent de différencier les actions.

Exemple d'interprétation

Pour un stock dont le flux présente une **autocorrélation positive élevée** à un lag de 1 :

- Cela peut indiquer que le flux suit des motifs stables, ce qui peut être utile pour prédire son comportement futur.
- En revanche, une **autocorrélation proche de 0** pourrait signaler un comportement aléatoire ou imprévisible.

3.5 Conclusion

Ces transformations améliorent significativement la qualité des données pour la classification et permettent aux modèles d'apprentissage de mieux capturer les dynamiques complexes des séries temporelles.

[4]:

	obs_id	venue	order_id	action	side	price	bid	ask	bid_size	ask_size	trade	flux
0	0	4	0	A	A	0.30	0.00	0.01	100	1	False	100
1	0	4	1	A	B	-0.17	0.00	0.01	100	1	False	100
2	0	4	2	D	A	0.28	0.00	0.01	100	1	False	-100
3	0	4	3	A	A	0.30	0.00	0.01	100	1	False	100
4	0	4	4	D	A	0.37	0.00	0.01	100	1	False	-100
...	...	...	...	...	...	...	...	...	...	...	...	...
16079995	160799	4	61	D	A	1.32	0.01	0.06	735	261	False	-100
16079996	160799	0	70	A	A	0.06	0.01	0.06	735	361	False	100
16079997	160799	4	71	A	A	1.26	0.01	0.06	735	361	False	100
16079998	160799	4	72	A	A	1.26	0.01	0.06	735	361	False	100
16079999	160799	4	73	A	B	0.00	0.01	0.06	635	361	False	1

16080000 rows × 12 columns

Feature engineering

	price_mean	price_std	price_min	price_max	price_range	bid_mean	bid_std	bid_min	bid_max	bid_range	...
0	0.0535	0.172588	-0.29	0.44	0.73	0.0000	0.000000	0.00	0.00	0.00	...
1	-0.0257	0.182791	-0.79	0.25	1.04	0.0134	0.010466	0.00	0.04	0.04	...
2	0.0120	0.054198	-0.16	0.14	0.30	0.0000	0.000000	0.00	0.00	0.00	...
3	0.1797	0.233677	-0.81	0.83	1.64	0.0690	0.046482	0.00	0.10	0.10	...
4	0.0193	0.059768	-0.16	0.26	0.42	0.0097	0.001714	0.00	0.01	0.01	...
...	...	...	...	...	...	...	...	...	...	...	...
160795	-0.0302	0.351516	-0.81	0.41	1.22	0.0134	0.012245	0.00	0.03	0.03	...
160796	0.1773	0.553019	-0.39	1.89	2.28	0.0064	0.014738	0.00	0.04	0.04	...
160797	0.0116	0.164659	-0.50	0.52	1.02	0.0051	0.005024	0.00	0.01	0.01	...
160798	0.0147	0.149764	-0.77	0.54	1.31	0.0043	0.006073	-0.02	0.01	0.03	...
160799	0.0842	0.315858	-0.49	1.32	1.81	0.0028	0.004513	0.00	0.01	0.01	...

160800 rows × 38 columns

FIGURE 2 – Feature Engineering Process Visualization

Transformation des séries temporelles en feature (*Méthode 2*)

La démarche consiste à pivoter les données, en transformant chaque opération ou observation d'une sequence en colonne pour chaque caractéristique. Cela crée une structure large, où chaque séquence (`obs_id`) est représentée par un grand nombre de colonnes. Cependant, cette approche pose plusieurs problèmes :

- **Explosion de la dimensionnalité :** La transformation augmente significativement le nombre de colonnes, ce qui complique l'entraînement des modèles standards de machine learning et accroît le risque de surajustement.
- **Perte de la structure temporelle :** En pivotant les données, les relations séquentielles entre les opérations sont supprimées, rendant la structure temporelle invisible pour les modèles.
- **Efficacité computationnelle :** Cette transformation est inefficace sur le plan computationnel et difficile à généraliser pour des séquences de longueur variable ou des données manquantes.

Une approche basée sur des **agrégations globales** ou des modèles séquentiels (*e.g.*, GRU, LSTM, etc.) serait plus adaptée pour capturer la dynamique temporelle des séries tout en évitant les limitations mentionnées.

4 Méthodes d'apprentissage

Dans un premier temps, nous utilisons des modèles classiques de machine learning tels que Random Forest et XGBoost sur des données agrégées. Ces modèles sont efficaces pour capturer des relations globales et fournir une première base de comparaison. Cependant, ces approches ne tirent pas parti de la structure temporelle des données séquentielles.

C'est pourquoi, dans un second temps, nous utilisons un modèle LSTM (Long Short-Term Memory), spécialement conçu pour analyser les séries temporelles en capturant les dépendances temporelles. De plus, avec un grand volume de données, les réseaux de neurones, comme les LSTM, surpassent généralement les modèles classiques, offrant des performances supérieures grâce à leur capacité à apprendre des motifs complexes.

4.1 Grid SearchCV : 1ère approche

Pour optimiser les performances des modèles, nous avons utilisé une recherche par grille avec validation croisée (*GridSearchCV*) afin de trouver les meilleurs hyperparamètres. Cette méthode explore systématiquement différentes combinaisons d'hyperparamètres pour identifier celles qui offrent les meilleurs résultats sur les données d'entraînement. On rajoute aussi qu'un échantillonnage a été effectuée afin de réduire le temps de calcul.

Après plusieurs itérations, nous avons obtenu les paramètres optimaux suivants :

- **Random Forest** :
  - `max_depth` : 20,
  - `n_estimators` : 400.
- **XGBoost** :
  - `gamma` : 0.1,
  - `learning_rate` : 0.1,
  - `n_estimators` : 300.

Ces configurations permettent aux modèles d'atteindre une performance optimale tout en équilibrant précision et efficacité.

4.2 Random Forest-Xgboost : 1ère approche

Les résultats montrent une diminution significative des performances des modèles lorsqu'ils sont évalués sur les données de test du challenge.

- Sur les données d'entraînement (`X_train X_test`), les modèles atteignent des précisions de :
  - **Random Forest** : 46.92%,
  - **XGBoost** : 49.73%.

  Ces résultats reflètent une performance correcte en phase d'entraînement.
- Cependant, sur les données de test du challenge, les précisions chutent respectivement à :
  - **Random Forest** : 28.14%,
  - **XGBoost** : 27.48%.

  Ces résultats indiquent un problème probable de généralisation des modèles.

4.3 Analyse des resultats

L'agrégation des données temporelles permet de capturer une signature globale, mais elle reste insuffisante pour représenter la complexité intrinsèque des séries temporelles. En effet, des métriques comme la moyenne ou l'écart-type ne différencient pas les formes spécifiques des séries, rendant difficile la classification précise de certaines classes.

Par exemple, le stock numéro 2, avec une précision et un rappel de 0.2, illustre ce manque de distinction. Pour renforcer les modèles comme **XGBoost** et **Random Forest**,

des techniques comme l'analyse de Fourier et l'étude des autocorrélations pourraient être envisagées pour extraire des motifs cachés dans les séries.

Toutefois, il est essentiel d'ajouter ces paramètres avec prudence pour éviter un sur-apprentissage, tout en maximisant la capacité des modèles à généraliser.

4.4 XGBoost - 2ème approche

Après un pivotage des données, comme expliqué précédemment, nous avons opté uniquement pour un modèle XGBoost. En effet, compte tenu du nombre important de colonnes dans ce cas, RandomForest nécessitait un temps d'entraînement très long. Par ailleurs, la librairie XGBoost prend en charge l'utilisation des GPU pour accélérer l'entraînement.

L'utilisation des GPU avec XGBoost repose sur des optimisations spécifiques, notamment l'algorithme Histogram-based. Ce dernier regroupe les données en histogrammes, ce qui permet de réduire le coût des calculs nécessaires pour identifier les meilleurs points de séparation (splits). Grâce à leur capacité à exécuter de nombreuses opérations en parallèle, les GPU permettent de traiter rapidement les grandes quantités de données et les nombreuses colonnes.

Après des testes "à la main" des valeurs de n-estimators, nous avons choisi d'entraîner le modèle avec 400 estimateurs. Le score obtenu est le suivant :

$$\boxed{Score = 0.2510}$$

Une carte graphique nvidia Tesla T4 16 gb vram a été utilisée.

4.5 LSTM

Nous avons opté pour une architecture récurrente afin de prendre en compte la nature séquentielle de notre problème. Pour cela, nous avons choisi d'utiliser deux Bi-LSTM, qui traitent la séquence dans les deux sens. Le principe des LSTM est détaillé dans les annexes. Nous avons également ajouté, en sortie, deux couches denses pour la classification :

Niveau	**Type de couche**	**Nombre de neurones**
1	Bi-LSTM	128
2	Bi-LSTM	64
3	Dense	16
4	Dense	24

TABLE 1 – Architecture du modèle

Afin d'accélérer l'entraînement, nous avons utilisé des TPU (Tensor Processing Units). Ces accélérateurs matériels, développés par Google, sont spécialement conçus pour exécuter efficacement des calculs liés à l'apprentissage automatique. Ils sont optimisés pour les opérations matricielles intensives utilisées dans les modèles de deep learning.

L'optimisation a duré trois heures : environ 40 minutes pour chaque configuration testée. Nous avons exploré plusieurs valeurs pour le nombre de neurones, le nombre d'époques

(epochs) et la taille des batchs (batch size). La meilleure configuration est présentée dans le tableau ci-dessus.

Bien que les RNN ne soient pas conçus pour être compatibles avec le calcul parallèle en raison de leur nature récurrente, nous avons constaté que, sans l'utilisation des TPU, la durée minimale d'entraînement s'élèverait à 12 heures. Cela illustre la complexité de notre problème ainsi que l'importance des GPU/TPU pour l'entraînement des modèles de deep learning.

Après soumission, nous avons obtenu le score suivant :

$$\boxed{Score = 0.4026}$$

On trouve ainsi un score meilleur que les autres approches testées.

4.6 Tableau comparatif entre les différents approches

Model	Paramètres et Descriptions	Score
Random forest méthode 1	max-depth : 20, n-estimators : 400	0.2814
XGBoost méthode 1	gamma : 0.1, learning-rate : 0.1, n-estimators : 300.	0.2748
XGBoost méthode 2	n-estimators = 400	0.2510
bi-LSTM	nbr de neurones par couche : 128 - 64 - 16 - 24	0.4026
Benchmark	GRU + embeddings des variables categoriques	0.3593

TABLE 2 – Tableau de résultats des modèles

5 Conclusion

En conclusion, le meilleur score a été obtenu grâce au modèle bi-LSTM, qui a atteint une précision de 0.4026. Cette approche, basée sur les réseaux neuronaux récurrents, s'est révélée particulièrement efficace pour capturer la dynamique temporelle des séries, notamment dans un contexte où les données séquentielles sont complexes et variées.

Cependant, le principal défi de ce projet résidait dans la phase de prétraitement des données, particulièrement pour les modèles Random Forest et XGBoost, qui ne sont pas conçus pour traiter directement des observations dépendantes dans le temps. L'application des modèles dépendait fortement de la qualité des données et des transformations effectuées. La création de caractéristiques adaptées, l'utilisation d'agrégations pertinentes et l'analyse approfondie des motifs temporels (comme l'autocorrélation et le spread) ont joué un rôle clé.

Enfin, en comparant la méthode d'apprentissage profond appliquée et l'approche ML "classique", on observe clairement la capacité des réseaux de neurones à apprendre une représentation pertinente pour résoudre le problème, contrairement aux autres méthodes qui nécessitaient un prétraitement important pour obtenir des résultats satisfaisants.

6 Annexe

6.1 LSTM (Long Short-Term Memory)

Les réseaux LSTM (Long Short-Term Memory) sont une architecture de réseaux de neurones récurrents spécifiquement conçues pour le traitement des séquences temporelles. Les LSTM incorporent des mécanismes de portes, tels que la porte d'oubli, la porte d'entrée et la porte de sortie, régulant le flux d'informations à travers la cellule de mémoire. Ces portes permettent au modèle de mémoriser des informations importantes sur des périodes plus longues tout en filtrant les informations moins pertinentes. Chaque cellules LSTM utilise des fonctions d'activation, notamment la fonction sigmoïde pour les portes et la fonction tangente hyperbolique pour les mises à jour de l'état. Ces fonctions d'activation contribuent à la gestion précise de l'information dans la cellule, facilitant la modélisation efficace de dépendances séquentielles complexes à travers le temps. Les LSTM sont ainsi particulièrement adaptés à la capture des structures temporelles dans les données.

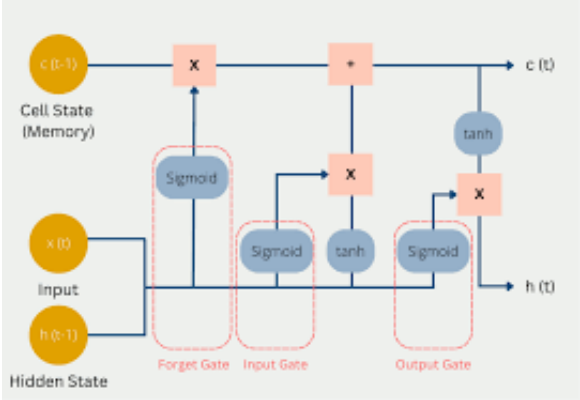

FIGURE 3 – Architechture LSTM

Où x_t désigne les données d'entrée du neurone, c_t l'état du neurone et h_t état caché ou sortie du neurone. Les équations qui régissent ces quantités sont :

$$
\begin{aligned}
f_t &= \sigma_g \left(W_f x_t + U_f h_{t-1} + b_f \right) \\
i_t &= \sigma_g \left(W_i x_t + U_i h_{t-1} + b_i \right) \\
o_t &= \sigma_g \left(W_o x_t + U_o h_{t-1} + b_o \right) \\
\tilde{c}_t &= \sigma_c \left(W_c x_t + U_c h_{t-1} + b_c \right) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \sigma_h \left(c_t \right)
\end{aligned}
$$

avec σ_g la fonction sigmoïde ; σ_g et σ_h la fonction tangente hyperbolique, f_t, i_t et o_t les activations de la "forget gate", "l'input gate" et "l'output gate" respectivement ; $\tilde{c}_t$ l'activation de l'entrée du neurone ; et W, U et b les matrices des poids et le vecteur de biais à ajuster durant l'entraînement.