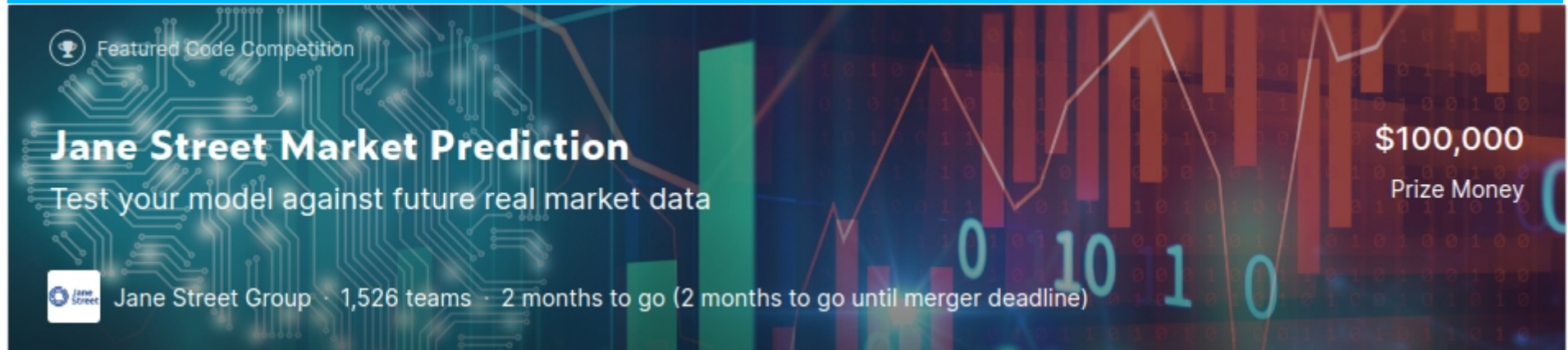


Projet de Data Science n°9


Participez à une compétition Kaggle



Featured Code Competition

Jane Street Market Prediction

Test your model against future real market data

 Jane Street Group · 1,526 teams · 2 months to go (2 months to go until merger deadline)

\$100,000
Prize Money

The banner features a dark background with a circuit-like pattern on the left and a bar chart with a line graph on the right. Binary digits (0s and 1s) are scattered throughout the design.



kaggle

L'objectif



Chaque ligne du dataset est une opportunité de trading

Pour chaque opportunité : le modèle doit **accepter** ou **refuser**

En cas de refus :

Le trade rapportera 0

En cas d'acceptation :

Le trade rapportera un retour positif ou négatif

L'objectif est de maximiser un utility score calculé à partir des montants de retour de chaque trade

Les contraintes



Des features anonymisées difficiles à décrypter

Un rapport signal / bruit très faible

⇒ Difficile de distinguer les prédictions fiables du facteur chance

Vu la nature des données (trading), la distribution de la variable à prédire pourra être très différente sur le test set final

Le système d'inférence, pour le livrable de la compétition, est fortement contraint d'un point de vue performance et doit se faire ligne par ligne

Description du dataset

| | |
|--------------------|--------------|
| Nombre de lignes | 2.4 millions |
| Nombre de jours | 500 |
| Nombre de features | 131 |

Chaque ligne est une opportunité de trading. Pour chaque ligne, on a :

- 130 features anonymisées (129 flottants et 1 booléen)
- Une variable **weight** (poids) ≥ 0
- Une variable **resp** (response : ce que rapporte le trade) ≥ 0 ou < 0
- Des variables **resp1**, **resp2**, **resp3**, **resp4** (différents horizons de retour)
- Le gain (ou la perte) effectif d'un trade est : **weight * resp * action**
action étant la décision du modèle : 1 (prendre le trade) ou 0

Fonction d'utilité

$$u = \min(\max(t, 0), 6) \sum p_i.$$

Régularisation pénalisant
la volatilité (écart type)
des gains

$$t = \frac{\sum p_i}{\sqrt{\sum p_i^2}} * \sqrt{\frac{250}{|i|}},$$

Gain du trade

Pour chaque trade j et chaque date i :
(Une date pouvant contenir plusieurs trades)

$$p_i = \sum_j (\text{weight}_{ij} * \text{resp}_{ij} * \text{action}_{ij}),$$

⇒ Nous avons implémenté cette fonction d'utilité pour évaluer
la performance des résultats

1ère approche du projet

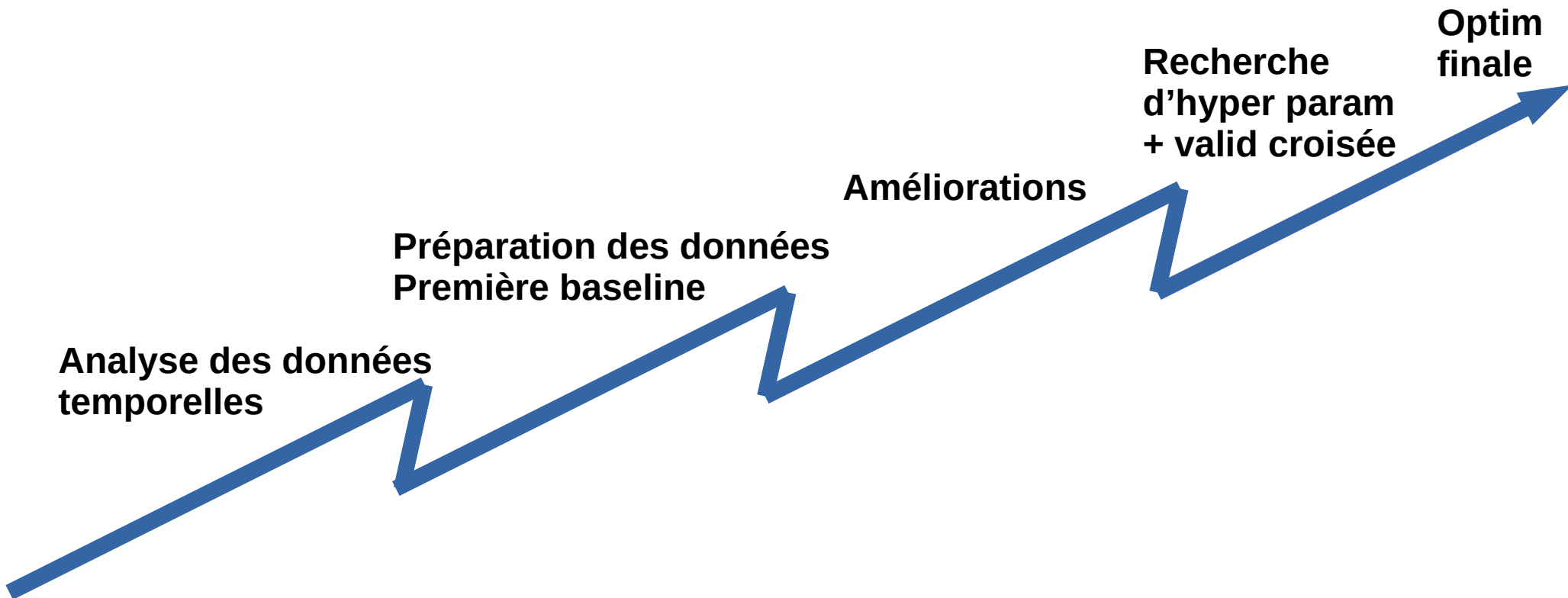
Machine learning (sans réseau de neurones)

Choix d'algorithme : **xgboost**

Pourquoi xgboost ?

- ⇒ C'est l'algorithme « star » des compétitions kaggle
- ⇒ Pour progresser dans la connaissance de cet algorithme et de l'optimisation de ses hyper paramètres

Les étapes de la démarche machine learning





Analyse des données temporelles

Analyse de notebooks de la communauté

Des notebooks publics nous ont permis de nous familiariser avec le jeu de données

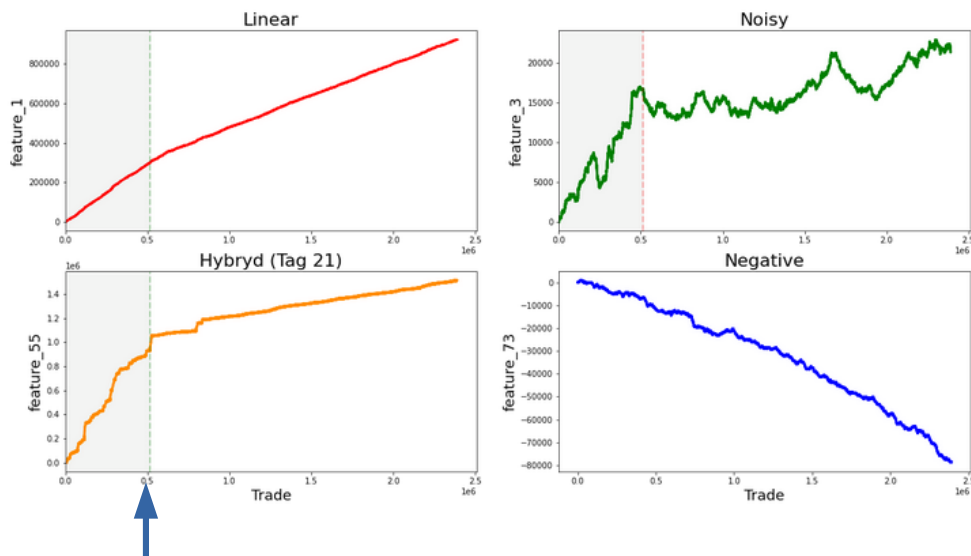
Exemple :

<https://www.kaggle.com/carlmcbrideellis/jane-street-eda-of-day-0-and-feature-importance?scriptVersionId=50152944>

<https://www.kaggle.com/mlconsult/feature-visualization>

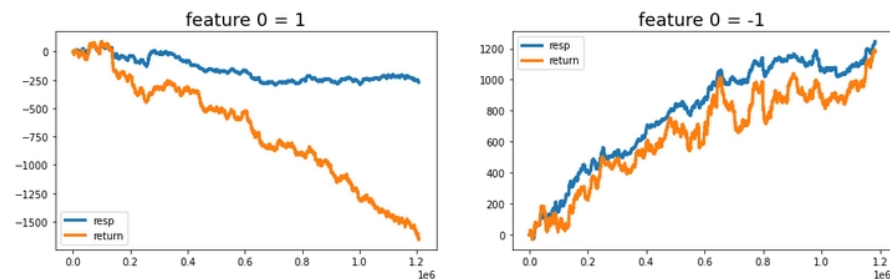
4 types de features principaux

Plot des montants cumulés :



+ la feature 0, binaire :

Plot des montants cumulés :



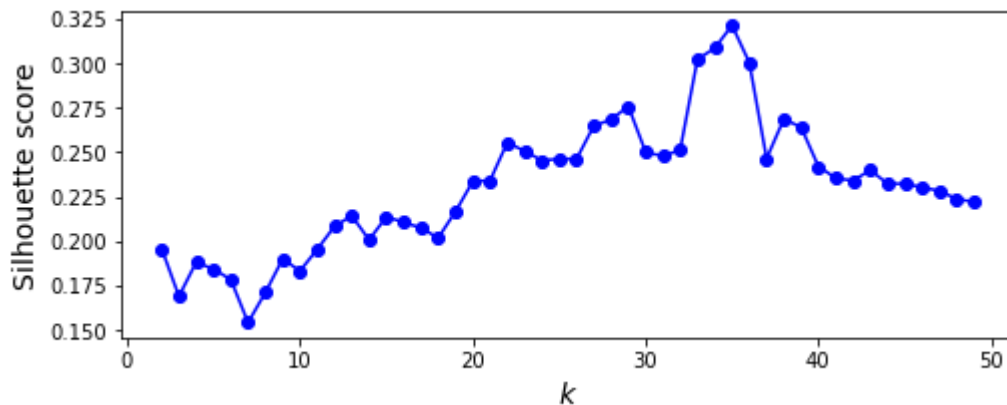
Constat d'un changement vers le jour 85 (barre horizontale)

Notebook réalisé : clustering des méta données

Des méta données sont fournies avec les features.

Nous avons réalisé un notebook de clustering des features à partir de ces méta données.

<https://www.kaggle.com/franoisboyer/jane-street-features-clustering>

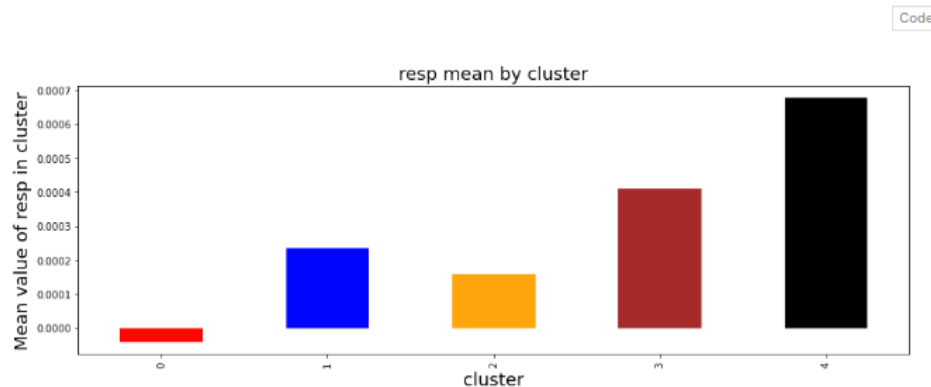
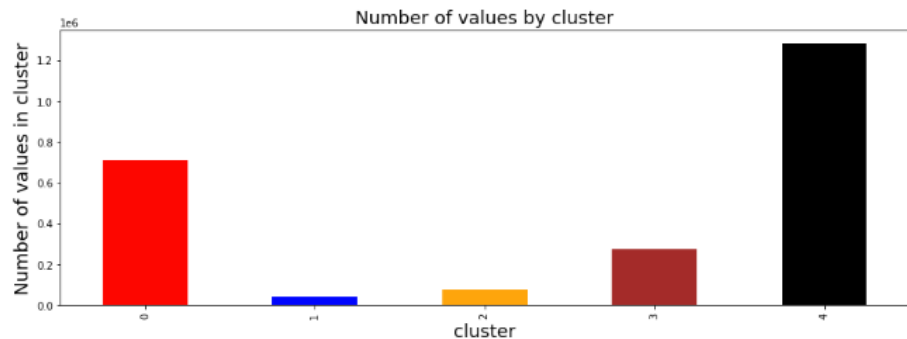


Avec l'algorithme Kmeans il apparaît un nombre de clusters k optimal
k = 35

Notebook réalisé : clustering des features

Nous avons réalisé un notebook de clustering des valeurs des features avec KMeans
Vu la volumétrie (2M de lignes) nous avons utilisé la librairie FAISS

<https://www.kaggle.com/franoisboyer/janestreet-feature-values-clustering>



On constate une démarcation des clusters par rapport à la valeur moyenne du retour (variable resp) alors que cette variable n'a pas été fournie lors de l'entraînement

⇒ L'approche clustering ou KNN porte donc probablement une valeur prédictive

Notebook : analyse de séries temporelles

Nous avons partagé avec la communauté un notebook d'analyse orienté sur les séries temporelles :

<https://www.kaggle.com/franoisboyer/jane-street-time-series-exploration>

Sachant que les différents trades concernent probablement plusieurs instruments financiers et que les features sont anonymes, l'objectif était de vérifier plusieurs éléments :

- La corrélation entre les différentes features
- Cela a-t-il du sens de considérer les features comme des séries temporelles espacées d'un step de 1 entre l'instance t et l'instance $t-1$? Ou bien faut-il appliquer un autre step pour l'analyse ?
- Les features sont-elles stationnaires ?
- Faire une première visualisation de feature engineering avec 2 méthodes issues de la finance : MACD et FFD

Livable :

[P9_01_CODE_jane-street-time-series-exploration.ipynb](#)

[P9_02_lienkaggle.txt](#)

Corrélation entre les différentes features

Coefficient de spearman :



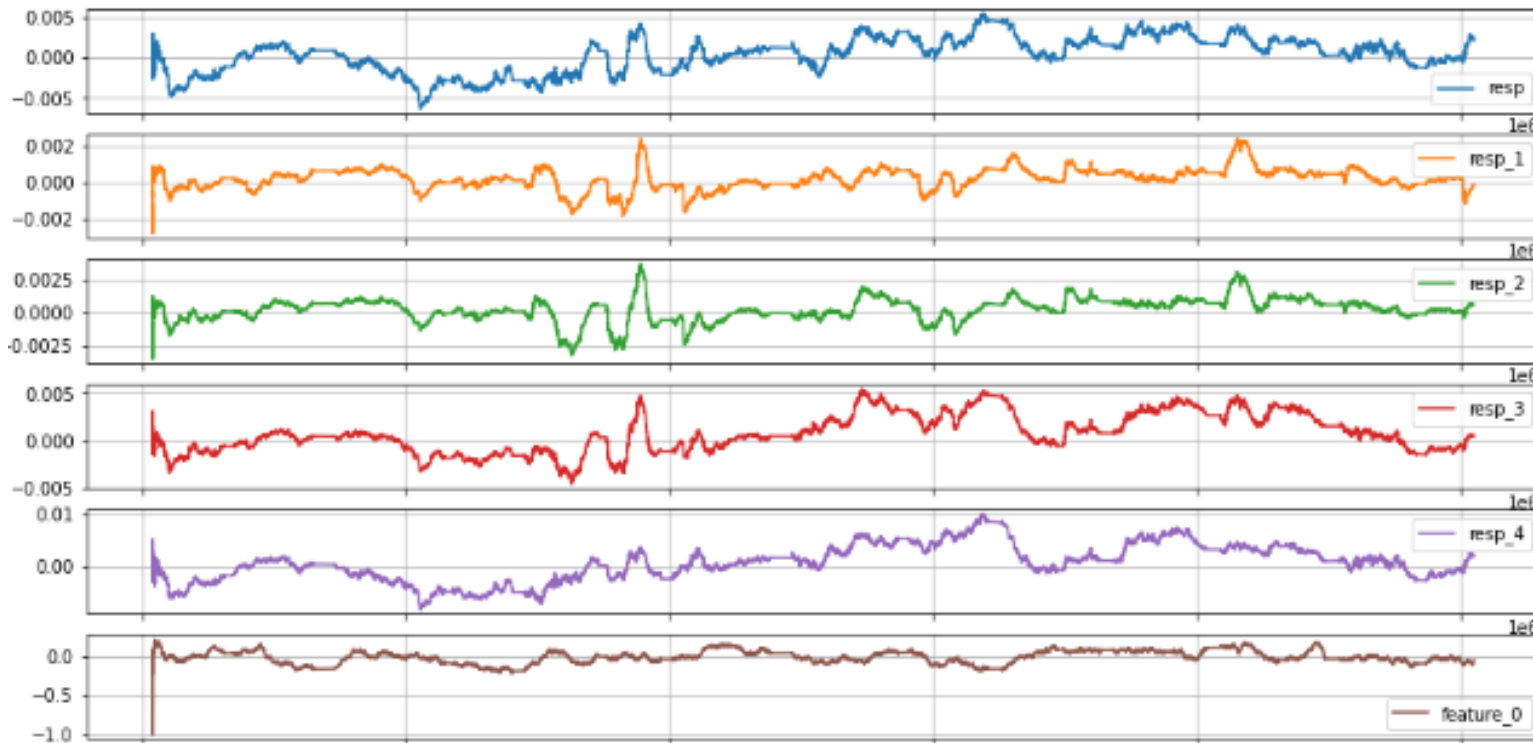
On constate :

- plusieurs groupes de features corrélées
- et proches les uns des autres (ce point pourrait avoir de l'importance pour une approche CNN)

Les motifs de features corrélées apparaissent en « double » car nous avons aussi représenté sur ce schéma les features MACD

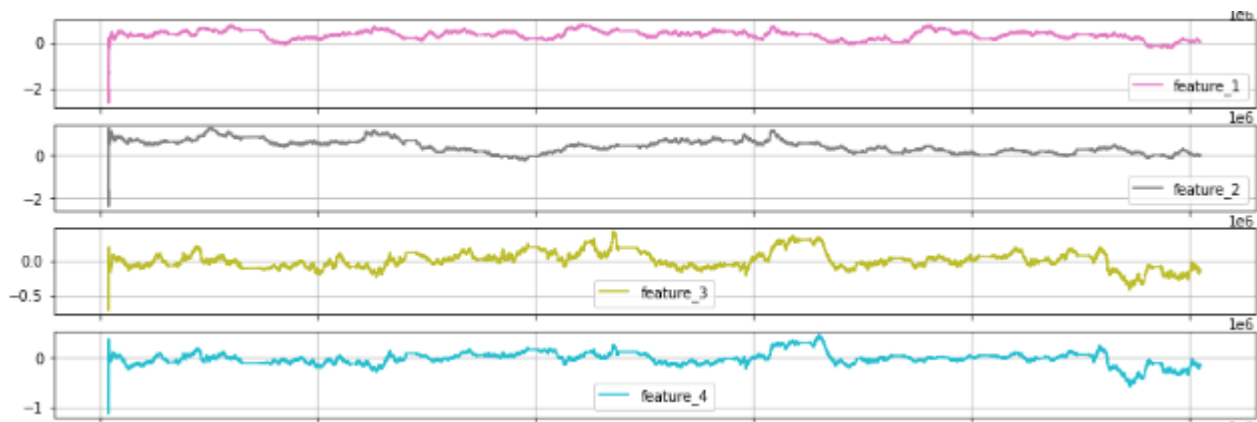
Représentation temporelle avec smoothing

Nous avons représenté les données sur 9 jours avec un smoothing (exponential weighted mean)



Représentation temporelle avec smoothing

Nous avons représenté les données sur 9 jours avec un smoothing (exponential weighted mean)



L'inspection visuelle montre des données globalement stationnaires, avec des tendances locales sans phénomène de saisonnalité clairement visible

Nous avons confirmé ces points par des analyses ci-après

Test de stationarité

Nous avons réalisé le test de Dickey Fuller

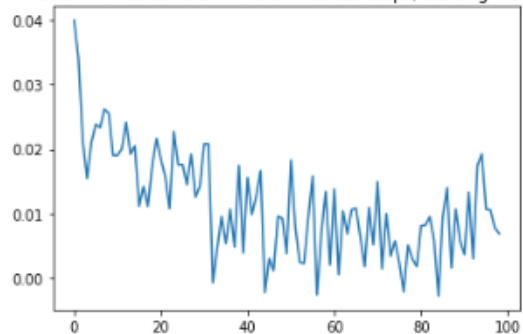
Ce test montre des données stationnaires sur l'ensemble des features

```
Results of Dickey-Fuller Test for column: feature_1
Test Statistic                -25.556036
p-value                       0.000000
No Lags Used                  44.000000
Number of Observations Used  42424.000000
Critical Value (1%)           -3.430504
Critical Value (5%)           -2.861608
Critical Value (10%)          -2.566806
dtype: float64
Conclusion:====>
Reject the null hypothesis
Data is stationary
```

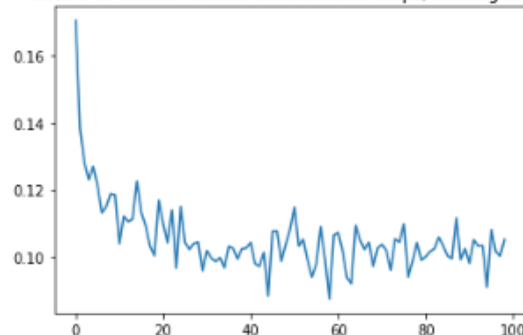

Auto corrélation

Nous avons représenté l'auto corrélation de pearson (en ordonnée) pour différents steps (en abscisse) entre les données temporelles

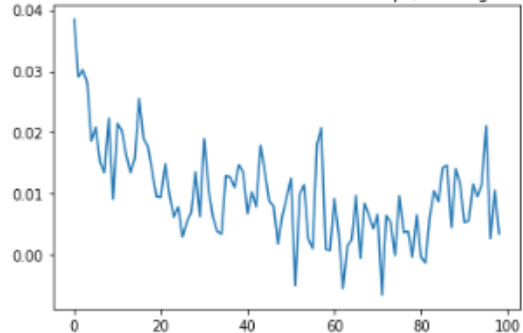
Feature 0 autocorrelation value accross steps, starting with 1



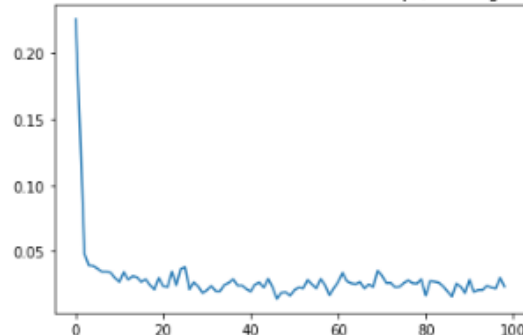
Feature 45 autocorrelation value accross steps, starting with 1



Feature 1 autocorrelation value accross steps, starting with 1



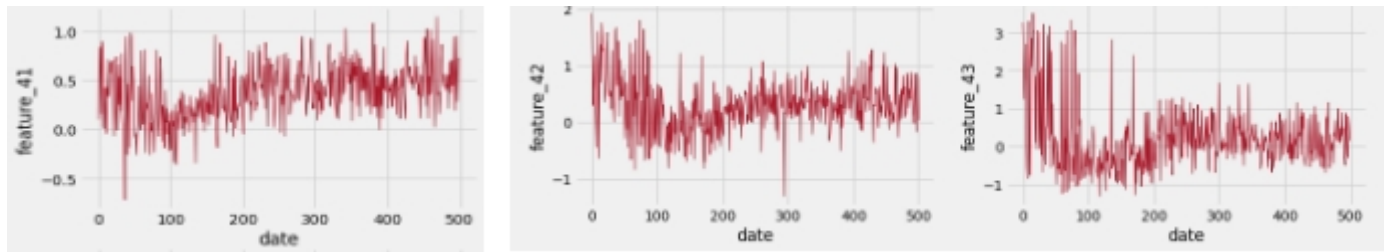
Feature 46 autocorrelation value accross steps, starting with 1



On constate :

- Une tendance décroissante de l'auto corrélation ce qui tend à montrer un step de 1 entre les données temporelles
- Pour la plupart des features, l'autocorrélation au step 1 dépasse rarement 0.03 : l'aspect linéaire de la corrélation est donc faible (mais décroît néanmoins au cours du temps ce qui ne contredit pas l'hypothèse du step de 1 entre les features)

Features « stock price » ?



<https://www.kaggle.com/mlconsult/feature-visualization>

A noter les features 41, 42 et 43 qui d'après l'un des notebooks publics
(<https://www.kaggle.com/carlmcbrideellis/jane-street-eda-of-day-0-and-feature-importance>)

sont peut-être liées à la notion de **stock price**

Cela montre une période plus « chaotique » entre les jours 1 et 200 environ

⇒ Le test set public (et caché) comporte-t-il ce type de période ?

⇒ Le test set final d'évaluation en comportera-t-il ?

Technique MA CD

Convergence
Divergence

Convergence
Divergence

Cette technique de trading consiste à faire la différence entre :

- Une moyenne glissante court terme de la feature financière liée au stock price
- Et une moyenne glissante de long terme de cette même feature

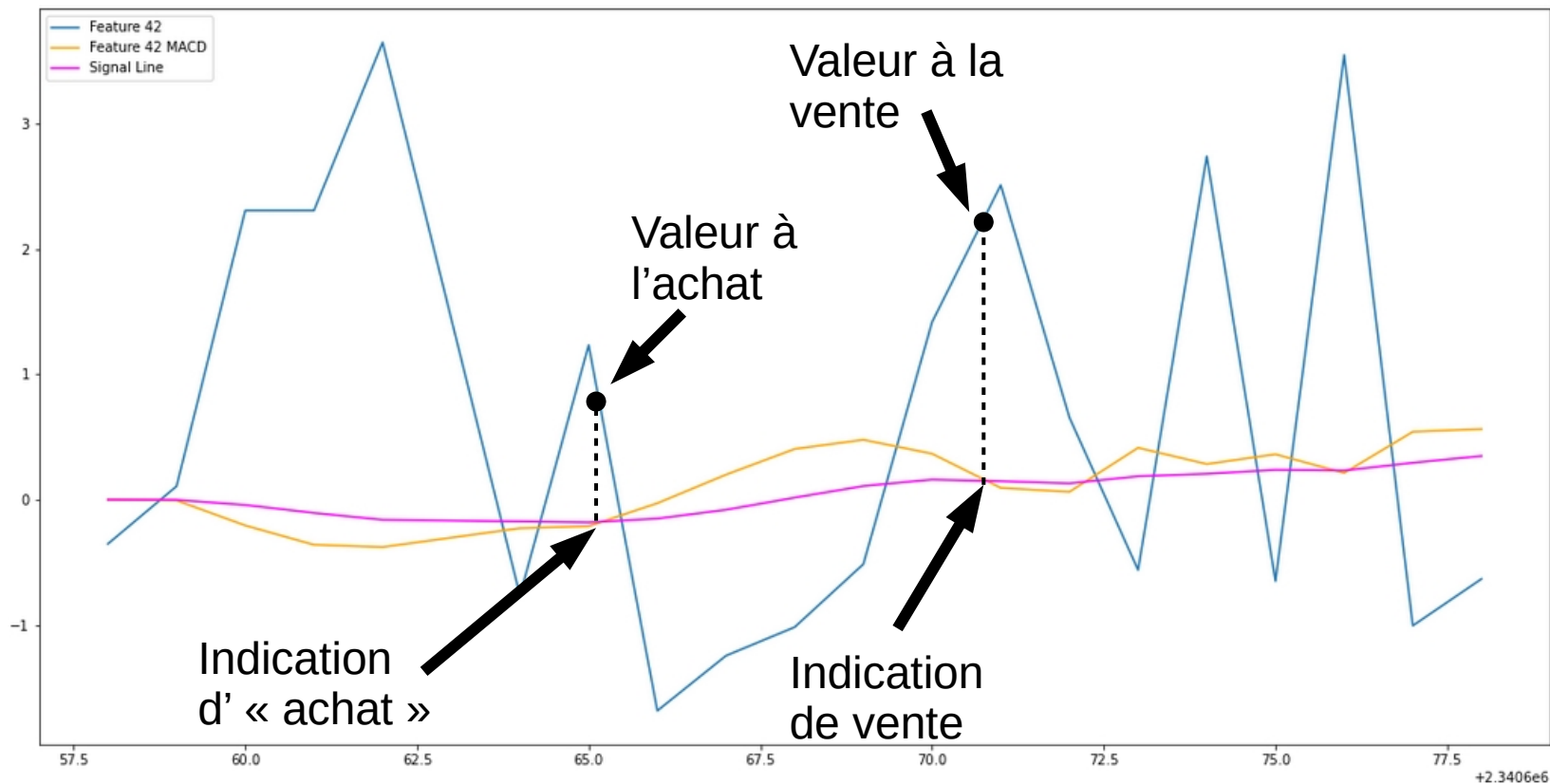
=> On obtient en résultat une série temporelle MACD

=> On fait à nouveau une moyenne glissante de cette série MACD (appelée signal)

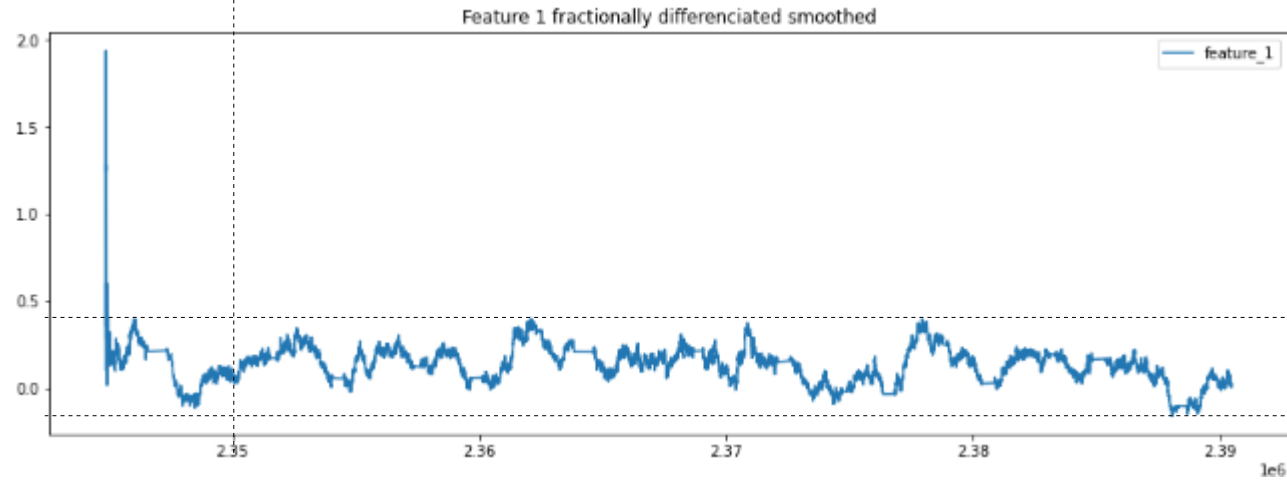
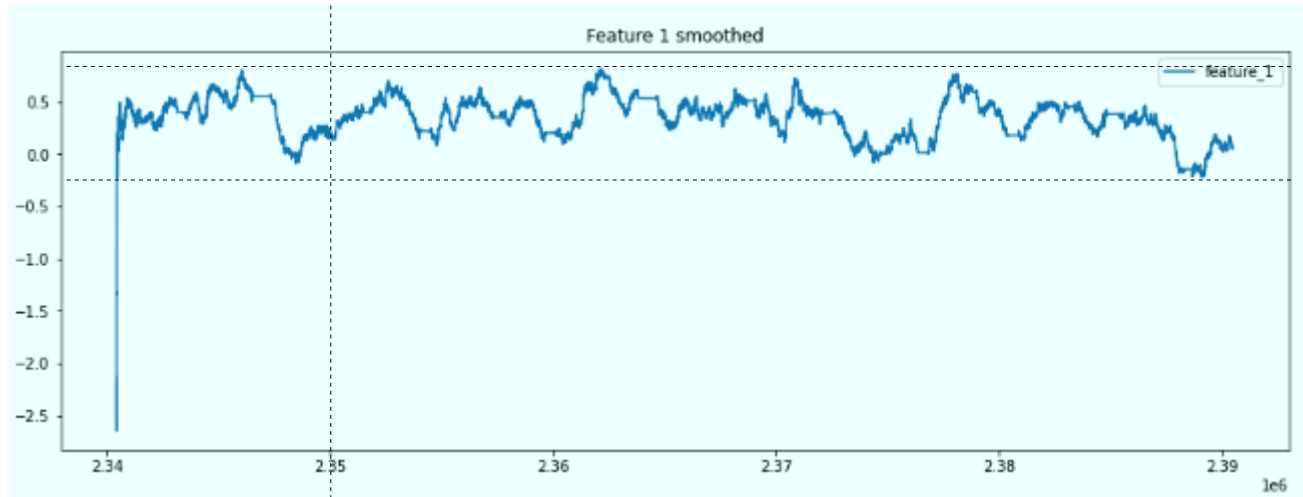
=> Lorsque la courbe MACD passe au dessus du signal, c'est que le prix va monter de façon relativement durable

MACD : illustration sur le dataset

Voici la technique MACD(12, 26, 9) illustrée sur 20 échantillons consécutifs de la variable «41» du dataset de la compétition



FFD: illustration sur le dataset



Conclusion sur l'analyse des données temporelles

- Sachant qu'il y a un step de 1 entre les échantillons, il est possible d'envisager des technique de feature engineering liées aux séries temporelles avec ce step
- Néanmoins, sachant :
 - Que les données sont globalement stationnaires
 - Qu'il y a un step de 1 entre les échantillons alors que pourtant les données concernent plusieurs instruments financiers
 - Qu'il y a déjà 130 features dans le dataset

Alors il y a un problemement déjà eu un feature engineering important déjà réalisé en amont

⇒ Pour la suite, nous avons essayé les features MACD dans un modèle mais nous n'avons pas poussé plus loin le feature engineering. Nous avons privilégié la mise en place d'une validation croisée et recherche d'hyperparamètres



Préparation des données et première baseline

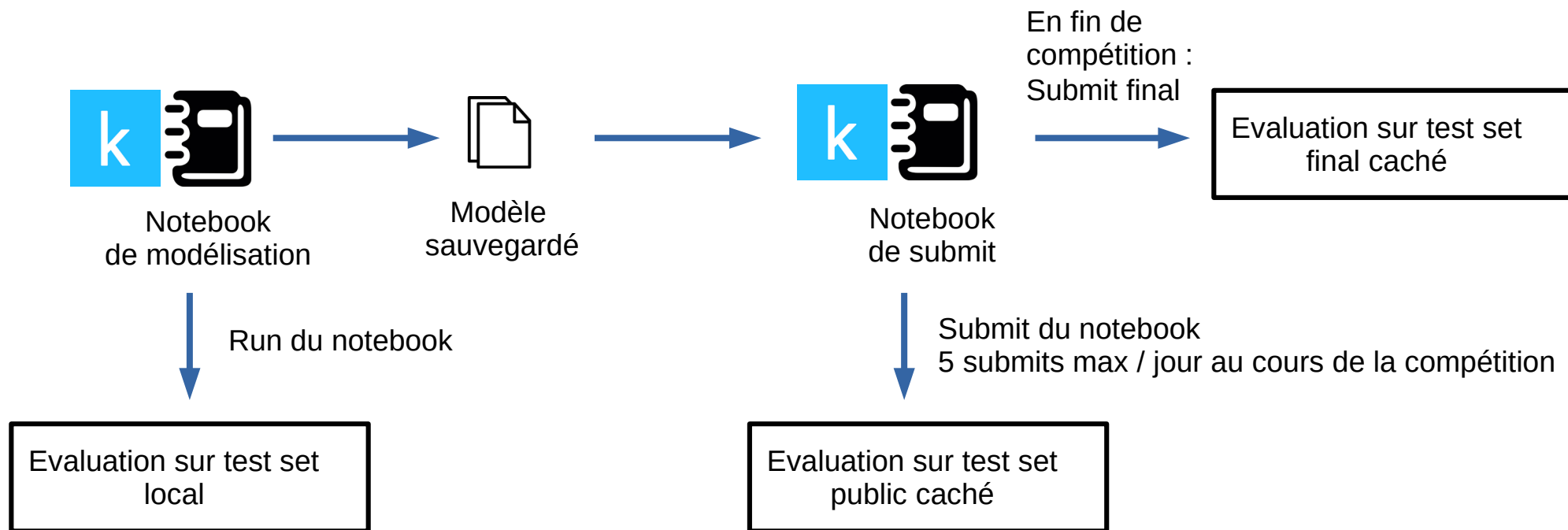
Préparation des données

- Très peu de data cleaning car un feature engineering important a déjà été réalisé en amont
- Calcul de la variable à prédire: un booléen : $\text{resp} > 0$ ou $\text{resp} \leq 0$
- Test de 3 stratégies d'imputation des données manquantes :
 - Imputation par la moyenne
 - Imputation à -999
 - Laisser les données vide (xgboost est capable de les gérer)
⇒ **La 3ème solution a été retenue car elle donne les meilleures performances**
- Test de 3 stratégies pour les outliers :
 - Suppression des z-score supérieurs à un certain seuil (4)
(<https://www.kaggle.com/blurredmachine/jane-street-market-eda-viz-prediction>)
 - Suppression des quantiles inférieurs à un certain seuil (0.001) ou supérieurs à un certain seuil (0.999)
 - Laisser les outliers
⇒ **La 3ème solution a été retenue car elle donne les meilleures performances**

Première baseline naïve : modèle random

| | |
|--------------------|---|
| Stratégie de split | Aucune |
| Algorithme | Random 50 % |
| Score local | Utility : non implémenté Accuracy : 0.499 |
| Score après submit | Utility : 2.5 |

Stratégie pour les premiers modèles



Aspects optimisation mémoire, inférence

- Optimisations mises en œuvre pour la mémoire :
 - Utilisation du package datatable pour charger les données
 - Conversion de float64 en float32
 - Diverses optimisations comme ne pas afficher de tableaux volumineux

⇒ La perte de précision liée à la conversion en float32 a toutefois eu un impact négatif sur les performances. Par la suite nous sommes repassés en float64
- Optimisations mises en œuvre pour l'inférence :
 - Un code le plus concis possible (le minimum d'affectations et d'opérations)
 - Utilisation de la fonction predict_proba qui est plus rapide que predict

Première baseline XGBoost

- Implémentation de la fonction d'utilité pour évaluer les résultats
- Premier modèle avec paramètres simples

| | |
|--------------------|--|
| Stratégie de split | 2 splits non chronologique avec shuffle |
| Algorithme | <u>XGBoost</u> max_depth=50, n_estimators=50 tree_method = 'gpu_hist' (pour utiliser le GPU) |
| Score local | Utility : 9868 Accuracy : 0.66 |
| Score après submit | Utility : 1006 |



Améliorations

Améliorations (1)

- Mise en place de 2 splits chronologiques
- Mise en place d'hyper paramètres plus sophistiqués

| | |
|---------------------------|--|
| Stratégie de split | 2 splits Chronologique sans shuffle |
| Algorithme | <u>XGBoost</u> max_depth= 12 , n_estimators= 500 learning_rate = 0.05 , Subsample = 0.9 colsample_bytree = 0.7 |
| Score local | Utility : 1883 Accuracy : 0.5241 |
| Score après submit | Utility : 3041 |

Améliorations (2)

- Meilleurs hyper paramètres
- Utilisation de TimeSeriesSplit() de scikit learn

| | |
|---------------------------|---|
| Stratégie de split | 3 splits chronologiques |
| Algorithme | <u>XGBoost</u> max_depth=12, n_estimators=500 learning_rate = 0.01, Subsample = 0.9 colsample_bytree = 0.2 |
| Score local | Utility : 2469 Accuracy : 0.5295 |
| Score après submit | Utility : 4037 |

Implémentation des features MACD



Test de modèles avec

- Chaque feature + sa version MACD moins signal
- Un mix des features MACD et MACD moins signal
- Avec seulement certaines features MACD moins signal (41, 42, 43, 44, 45)

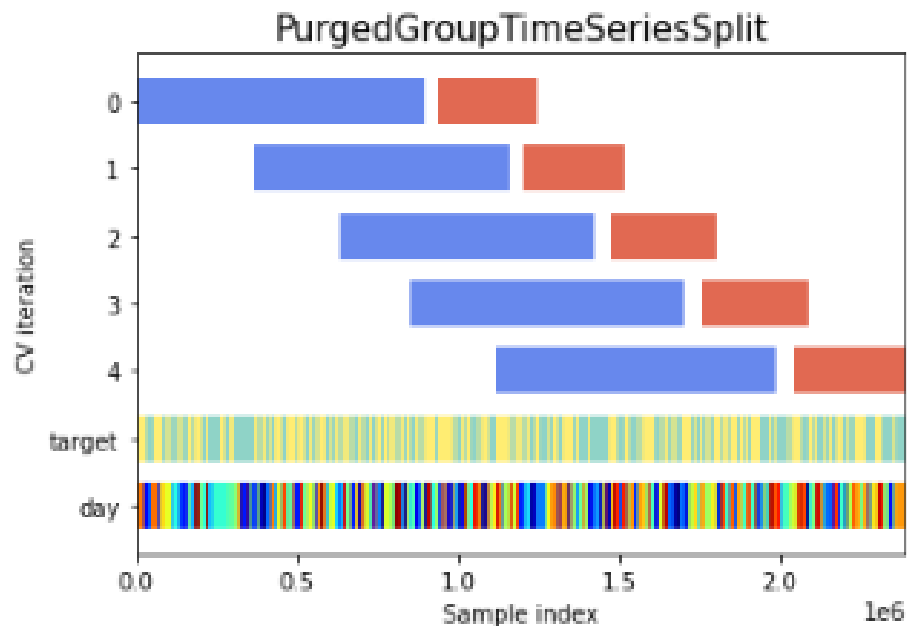
⇒ Meilleur score sur test local : **2295**

⇒ Les features MACD dégradent un peu les performances

Features MACD non retenues en l'état

Pour poursuivre dans cette voie il faudrait essayer d'autres valeurs pour les moyennes glissantes et d'autres combinaisons de features

Mise en place d'un « purged grouped time series split »



- 5 folds de 1 training set, 1 validation set
- Les training sets restent chronologiquement avant les test sets
- Coupure « nette » entre les sets selon le groupe (= la journée) pour éviter le data leak
- Un écart est laissé entre les training sets et les test sets

Utilisation de la fonction `PurgedGroupTimeSeriesSplit()` mise à disposition dans le notebook :

<https://www.kaggle.com/marketneutral/purged-time-series-cv-xgboost-optuna>

Recherche
d'hyper
paramètres

Validation
croisée



Stratégie pour les modèles optimisés



Notebooks
de prototypage
(float32)



Script d'hyper
optimisation
(float64)



Résultats



Notebook
d'analyse
des résultats



Script
d'entraînement
du modèle
retenu



Modèle



Notebook
de submit

Recherche d'hyper paramètres



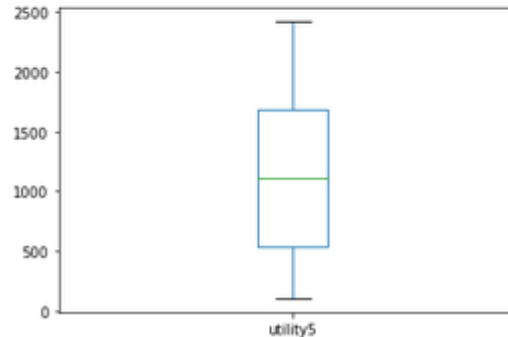
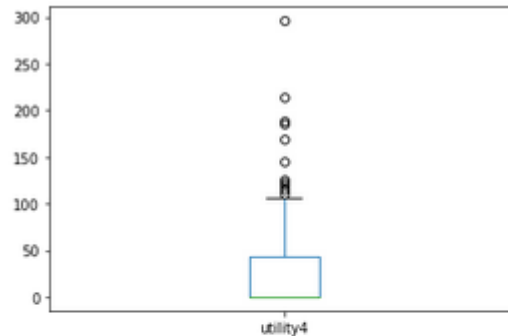
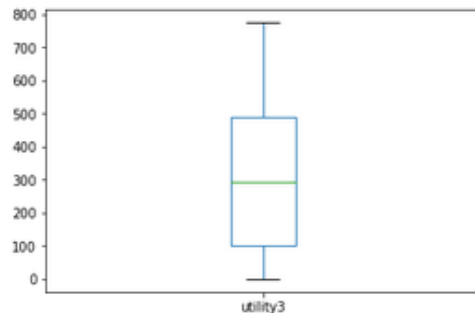
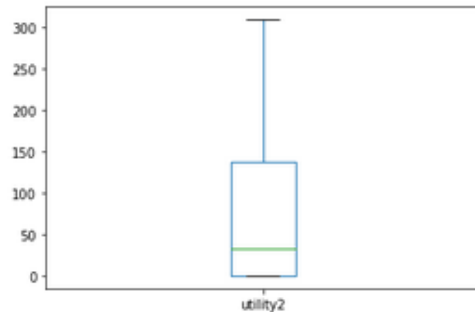
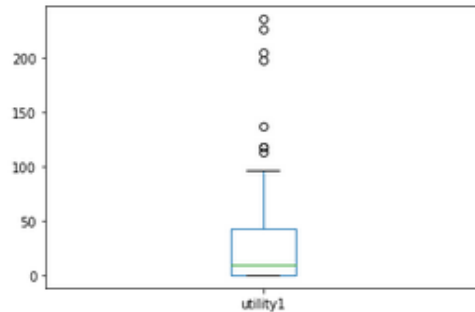
Utilisation de l'algorithme de recherche séquentielle **hyperopt**
Score optimisé = l'utility score total sur les 5 splits



Lancement sur une instance **AWS EC2 P3** avec GPU

| Features | Avec ou sans weight |
|------------------|---|
| max_depth | [8, 10, 12, 15, 20] |
| n_estimators | [50, 250, 500, 1000] |
| learning_rate | [0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5] |
| subsample | [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9] |
| colsample_bytree | [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1] |

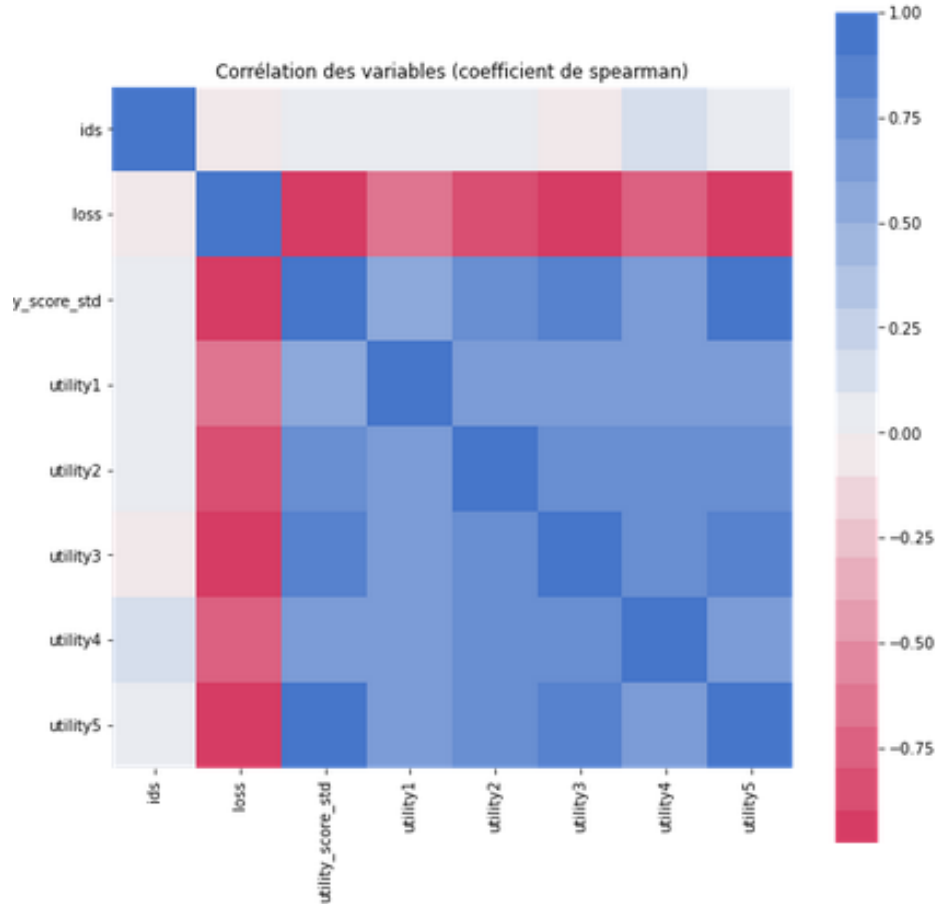
Résultat de l'optimisation : observations statistiques



On constate des disparités importantes sur les scores entre les splits

⇒ Comme on ne connaît pas la distribution du test set final, l'enjeu pour la modélisation sera d'obtenir le modèle le plus polyvalent sur l'ensemble des splits

Résultat de l'optimisation : observations statistiques



On constate :

- Le split 1 est le moins corrélé aux autres (maximum 0.65 de corrélation)
- Les splits 3 et 5 sont les plus corrélés (0.81)

Résultat de l'optimisation

Nous avons shortlisté les 5 meilleurs modèles de chaque split
Le meilleur modèle global étant aussi le meilleur modèle du split 5
A noter que le split 5 obtient des scores largement supérieurs aux autres

| loss | utility_scores | utility_score_std | accuracy | vals | book_time | utility1 | utility2 | utility3 | utility4 | utility5 |
|--------------|---|-------------------|--|---|-------------------------------|------------|------------|------------|------------|-------------|
| -3265.602358 | [235.21497907016132, 226.84574867761827, 595.8539048955168, 11.199105022790173, 2196.4886205363105] | 794.189258 | [0.5165684121296189, 0.5156591110734784, 0.5164959227685328, 0.512529325056574, 0.5172206728551928] | {'colsample_bytree': [5], 'features': [0], 'learning_rate': [1], 'max_depth': [1], 'n_estimators': [1], 'random_state': [0], 'subsample': [6], 'tree_method': [0]} | 2020-12-20 08:05:47.764000 | 235.214979 | 226.845749 | 595.853905 | 11.199105 | 2196.488621 |
| -2768.765124 | [74.29833817630187, 309.25601126096655, 706.3137076089242, 105.56135009856858, 1573.335717075042] | 557.378685 | [0.5161599815050283, 0.512132267671952, 0.5163700930790176, 0.5123943778935786, 0.515985650213111] | {'colsample_bytree': [2], 'features': [0], 'learning_rate': [5], 'max_depth': [0], 'n_estimators': [1], 'random_state': [0], 'subsample': [6], 'tree_method': [0]} | 2020-12-20 01:49:38.601000 | 74.298338 | 309.256011 | 706.313708 | 105.561350 | 1573.335717 |
| -3175.499804 | [225.59143670415475, 248.38493912832203, 774.5085428708145, 56.324169850579885, 1870.6907153451414] | 663.075625 | [0.5191153238546603, 0.5160493463559019, 0.5175794562060252, 0.5139826021965246, 0.5195390486920831] | {'colsample_bytree': [1], 'features': [0], 'learning_rate': [0], 'max_depth': [1], 'n_estimators': [2], 'random_state': [0], 'subsample': [3], 'tree_method': [0]} | 2020-12-20 09:35:54.915000 | 225.591437 | 248.384939 | 774.508543 | 56.324170 | 1870.690715 |
| -2781.075189 | [15.809447163022076, 166.76569856467142, 627.4190264586754, 295.92548122850434, 1675.155535552794] | 594.736494 | [0.5158517319770354, 0.5148160556048463, 0.5160170708945442, 0.5129030248925613, 0.5174528199683661] | {'colsample_bytree': [8], 'features': [0], 'learning_rate': [1], 'max_depth': [1], 'n_estimators': [2], 'random_state': [0], 'subsample': [3], 'tree_method': [0]} | 2020-12-20 11:39:25.524000 | 15.809447 | 166.765699 | 627.419026 | 295.925481 | 1675.155536 |
| -3449.952551 | [8.21122821970743, 249.99630629505893, 687.2746632257905, 85.67024740143403, 2418.80010572738] | 895.813693 | [0.515350826494047, 0.5141828436371401, 0.5172928441354626, 0.5125846879952388, 0.5186847473156055] | {'colsample_bytree': [5], 'features': [0], 'learning_rate': [1], 'max_depth': [1], 'n_estimators': [2], 'random_state': [0], 'subsample': [3], 'tree_method': [0]} | 2020-12-20 08:17:45.267000 | 8.211228 | 249.996306 | 687.274663 | 85.670247 | 2418.800106 |

Scores des modèles shortlistés après submit

Meilleur modèle split 5 4921

| | |
|-------------------------|-------------|
| Meilleur modèle split 4 | 5359 |
|-------------------------|-------------|

Meilleur modèle split 3 3734

| | |
|-------------------------|------|
| Meilleur modèle split 2 | 4121 |
|-------------------------|------|

Meilleur modèle split 1 4657

→ **Modèle final retenu**

```
'features': ['feature_'+str(i) for i in range(130)],  
'random_state': 42,  
'max_depth': 10,  
'n_estimators': 500,  
'learning_rate': 0.02,  
'subsample': 0.5,  
'colsample_bytree': 0.9,  
'tree_method': 'gpu_hist'
```

Livable

P9_01_CODE_janestreet_train_model_V2 (ENTRAINEMENT MODELE FINAL RETENU APRES RECHERCHE).py

P9_01_CODE_convert_params_hyperopt_to_xgboost (ENTRAINEMENT MODELE APRES RECHERCHE)

Recherche d'hyper paramètres avec gamma



Nouvelle recherche afin d'optimiser l'hyper paramètre gamma (qu'il est recommandé d'optimiser à la fin, après avoir optimisé les autres hyper paramètres)

⇒ **Sélection des meilleurs hyper params du run précédent et ajout de gamma**

| Features | Sans weight |
|------------------|-------------------------|
| max_depth | [8, 9, 10] |
| n_estimators | [250, 500] |
| learning_rate | [0.01, 0.02, 0.1] |
| subsample | [0.5, 0.8] |
| colsample_bytree | [0.2, 0.3, 0.6, 0.9] |
| gamma | [0.01, 0.1, 0.5, 1, 10] |

Résultat de l'optimisation

Cette fois le meilleur modèle ne coïncide plus avec le meilleur modèle du test set 5
Ce meilleur modèle est aussi le meilleur parmi tous les autres, sur notre test set local
⇒ Nous allons voir slide suivante que ce n'est toujours pas lui qui performe le mieux sur le test set kaggle

| loss | utility_scores | utility_score_std | accuracy | vals | book_time | utility1 | utility2 | utility3 | utility4 | utility5 |
|--------------|---|-------------------|--|--|----------------------------|------------|------------|------------|------------|-------------|
| -3641.386462 | [143.1717637986452, 216.75405151348141, 932.7505815612767, 104.51025663653327, 2244.199808479232] | 816.445984 | [0.5179593881246869, 0.5165242553316816, 0.5165833044973628, 0.5136296634625366, 0.5191180885935289] | {'colsample_bytree': [1], 'features': [0], 'gamma': [2], 'learning_rate': [1], 'max_depth': [2], 'n_estimators': [0], 'random_state': [0], 'subsample': [1], 'tree_method': [0]} | 2020-12-21 04:22:46.061000 | 143.171764 | 216.754052 | 932.750582 | 104.510257 | 2244.199808 |
| -2970.193311 | [320.5617282413911, 212.94263363963776, 575.748869634276, 62.97257568435906, 1797.9675042522535] | 624.793434 | [0.5177166416213925, 0.5171832375538874, 0.5169607935659085, 0.5135050968505408, 0.5190592779915251] | {'colsample_bytree': [0], 'features': [0], 'gamma': [0], 'learning_rate': [1], 'max_depth': [1], 'n_estimators': [0], 'random_state': [0], 'subsample': [1], 'tree_method': [0]} | 2020-12-21 10:35:21.362000 | 320.561728 | 212.942634 | 575.748870 | 62.972576 | 1797.967504 |
| -3188.682910 | [204.93372753500478, 415.04590001516743, 645.5437629718652, 52.1578582539327, 1871.00166164696] | 648.199908 | [0.5167109775363157, 0.5173857181249563, 0.517558484591106, 0.51353622385035398, 0.5190221344534174] | {'colsample_bytree': [1], 'features': [0], 'gamma': [4], 'learning_rate': [1], 'max_depth': [1], 'n_estimators': [0], 'random_state': [0], 'subsample': [0], 'tree_method': [0]} | 2020-12-21 11:15:37.454000 | 204.933728 | 415.045900 | 645.543763 | 52.157858 | 1871.001662 |
| -3229.080033 | [261.48180976101315, 207.15113000064008, 944.2887240020398, 7.119925819097419, 1809.038443570358] | 661.852662 | [0.5178476476707895, 0.5148712775787742, 0.5157758973229734, 0.5114739690382766, 0.516084699648065] | {'colsample_bytree': [0], 'features': [0], 'gamma': [4], 'learning_rate': [2], 'max_depth': [1], 'n_estimators': [0], 'random_state': [0], 'subsample': [1], 'tree_method': [0]} | 2020-12-21 01:21:23.360000 | 261.481810 | 207.151130 | 944.288724 | 7.119926 | 1809.038444 |
| -2911.632657 | [31.91080308378941, 161.0089306545413, 591.1637882656056, 291.015533883735, 1836.5336006250188] | 653.978832 | [0.5169074866104111, 0.5149817215266299, 0.5172648819822371, 0.5145293112158393, 0.5166882821423154] | {'colsample_bytree': [3], 'features': [0], 'gamma': [2], 'learning_rate': [1], 'max_depth': [1], 'n_estimators': [1], 'random_state': [0], 'subsample': [0], 'tree_method': [0]} | 2020-12-21 04:19:03.484000 | 31.910803 | 161.008931 | 591.163788 | 291.015534 | 1836.533601 |
| -3083.802130 | [133.9792458175221, 101.27508237934157, 357.05623443101365, 58.28318483638094, 2433.208382378865] | 914.071013 | [0.5155743074018417, 0.5168334983856776, 0.516516894383452, 0.5125846879952388, 0.5182483107428398] | {'colsample_bytree': [3], 'features': [0], 'gamma': [4], 'learning_rate': [1], 'max_depth': [0], 'n_estimators': [0], 'random_state': [0], 'subsample': [1], 'tree_method': [0]} | 2020-12-21 07:21:02.239000 | 133.979246 | 101.275082 | 357.056234 | 58.283185 | 2433.208382 |

Scores des modèles avec gamma après submit

| | |
|-------------------------|---------------|
| Meilleur modèle | 4161 |
| Meilleur modèle split 5 | 4186 |
| Meilleur modèle split 4 | 4336 |
| Meilleur modèle split 3 | 3732 |
| Meilleur modèle split 2 | 3702 |
| Meilleur modèle split 1 | Pas de submit |

→ **Meilleur modèle avec gamma**

'features': ['feature_'+str(i) for i in range(130)],
'gamma': 0.5,
'learning_rate': 0.02,
'max_depth': 9,
'n_estimators': 500,
'random_state': 42,
'subsample': 0.5,
'colsample_bytree': 0.9,
'tree_method': 'gpu_hist'

Conclusion des hyper optimisations

Les hyper optimisations ont permis de déterminer

- Le meilleur modèle sur le test set caché de kaggle ⇒ **1^{er} modèle retenu**
- Le meilleur modèle sur notre propre test set ⇒ **2ème modèle retenu**

Pour ce projet nous retenons le 1^{er} modèle



Recherche d'améliorations complémentaires

Livrables :

P9_01_CODE_janestreet_train_model_V3 (AMELIORATIONS COMPLEMENTAIRES).py
P9_01_CODE_janestreet_train_model_V3_different_data (AMELIORATIONS COMPLEMENTAIRES).py
P9_01_CODE_janestreet_train_model_V5 (AMELIORATIONS COMPLEMENTAIRES).py


Tentatives d'amélioration complémentaires

Nous avons fait d'autres tentatives d'amélioration du modèle qui, elles, n'ont pas amené de meilleures performances

| | Score après submit |
|---|--------------------|
| Suppression des dates ≤ 85 | 5156 |
| Conservation des données des split 4 et 5 uniquement | 4791 |
| Suppression des données weight = 0 | 4867 |
| Prédiction du montant de « resp » (régression) au lieu de prédire « resp > 0 » (classification) | 4477 |

➔ **Ce dernier modèle obtient toutefois le meilleur score de tous sur le split 3 de notre test set (1128, contre 295 pour le meilleur modèle final retenu)**

Il pourrait donc être complémentaire dans une approche ensembliste

A close-up photograph of a person's right hand holding a white marker, writing the word "more" in cursive on a whiteboard. The hand is wearing a black wristband. The whiteboard is mounted on a wall, and a window with light-colored curtains is visible in the background.

more

Perspectives

Pour aller plus loin, il est possible ...

- **Approches d'ensemble learning**

Mettre en place un meta model qui prend en entrée les prédictions d'autres types de modèle :

- Une régression \Rightarrow On a vu qu'elle est intéressante
- Un clustering \Rightarrow On a vu qu'il est intéressant
- Le meilleur modèle sur notre test set interne
- Une prédiction de la valeur resp à $t-1$ à partir des features à t

- **Feature engineering**

- Mettre en œuvre des techniques de feature selection
- Ajouter des lag features liées aux séries temporelles
- Ajouter des combinaisons entre feature 0 (booléen) et les autres
- Pousser plus loin le MACD en essayant d'autres valeurs de moyennes glissantes

- **Approche réseau de neurones**

- Mettre en œuvre une approche réseau de neurones avec par exemple un réseau de convolution (qui semble moins utilisé que les autres pour ce type de problématiques)



Questions

Annexe : algorithme TPE

- L'algorithme **T**ree-structured **P**arzen **E**stimation est expliqué ici :
https://asim-fachtagung-spl.de/asim2019/papers/47_Proof_164.pdf

