Challenge : Prédiction de la volatilité en finance de marché Par CFM

Pirashanth RATNAMOGAN

pirashanth.ratnamogan@ens-paris-saclay.fr

Table des matières

1 Introduction

Dans le cadre du cours *Face à la Malédiction de la dimensionnalité* de Stéphane Mallat, nous avons l'occasion de mettre en pratique les divers concepts que l'on a pu voir au cours de cette année à travers un data challenge. J'ai décidé de travailler sur le challenge CFM : prédiction de la volatilité en finance de marché. Le but est de prédire la volatilité entre 14 et 16h étant donné l'information sur toutes les volatilités sur une période de 5mn entre 9h30 et 13h55, le signe du retour sur investissement à chaque pas de temps, la date et le produit considérés. Dans ce rapport, je présente la solution qui m'a permis d'atteindre à l'heure où j'écris ce rapport la 3ème (/42) place du public leaderboard avec un score de 21.75 avec la MAPE loss. Dans un premier temps, je fais une analyse des données disponibles dans le cadre du challenge avant d'expliciter les idées que j'ai explorées ainsi que ma solution finale.

2 Analyse des données

2.1 Première analyse

On peut rapidement analyser les données sur lesquelles on va travailler. Nous disposons de 636 313 exemples d'entrainement et 635 397 exemples de test pour évaluer notre algorithme. Chaque exemple comporte plusieurs informations : la date à laquelle les mesures ont été prises (qui permet uniquement de savoir quels évènements se déroulent en même temps), l'id du produit considéré, 54 données donnant l'évolution de la volatilité toutes les 5 mn entre 9h30 et 13h55 ainsi que 54 données donnant les signes des retours sur investissement associés aux mêmes pas de temps. Le but est de prédire la volatilité sur la période entre 14h00 et 16h00.

Dans le jeu d'entrainement il y a 2117 dates différentes et 318 produits différents. On peut associer ces 318 produits aux 318 produits présents dans le jeu d'entrainement. Il y a 2119 dates dans le jeu de test, celle-ci sont aléatoires et inassociables aux données du jeu d'entrainement.

2.2 Analyse de quelques courbes

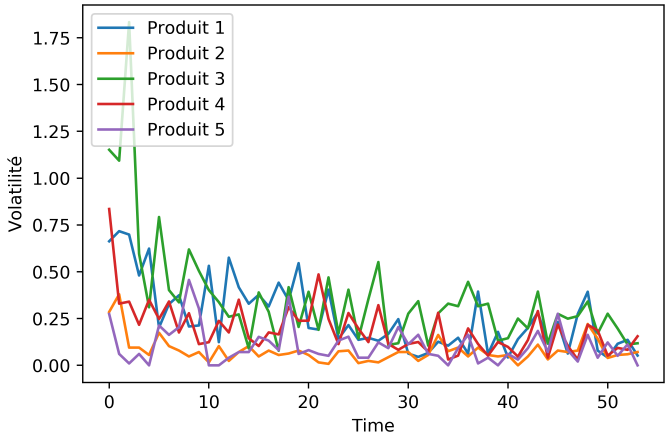


Figure 1: Evolution de la volatilité de différents produits le même jour (5 premières données du jeu d'entrainement)

Comme on pouvait s'en douter, on observe sur ce visuel que l'évolution de les volatilités de différents produits financiers sont fortement liées. En effet, le contexte économique dans lequel on se trouve subit les mêmes évolutions et perturbations et agissent sur les prix des produits financiers et ainsi sur leurs volatiltés. On observe que globalement sur tout le dataset on a des pics de volatilité au début et en fin (après 14h) de séance.

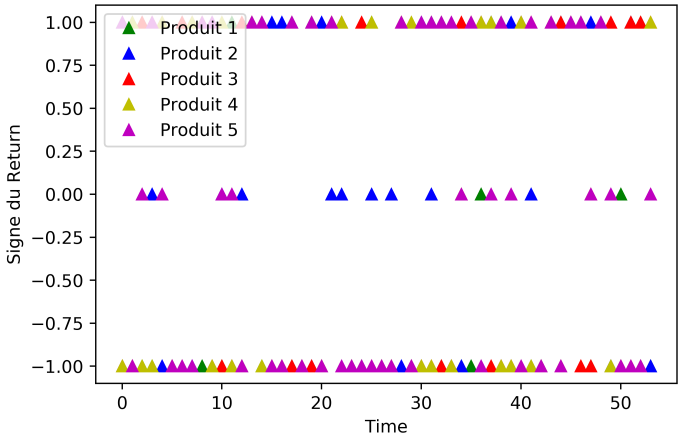
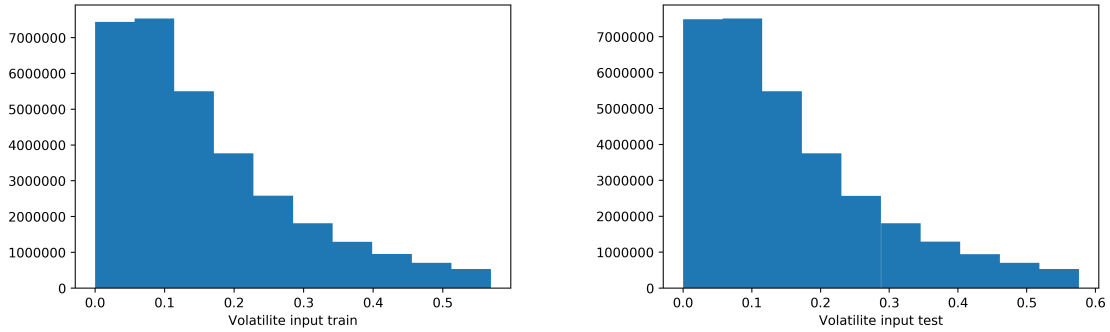


Figure 2: Evolution du signe du retour sur investissement de différents produits le même jour (5 premières données du jeu d'entrainement)

Il est difficile de tirer des informations de ce visuel, on peut voir que l'on a plusieurs valeurs manquantes (fixées à 0), certains produits sont sur des augmentations tandis que d'autres sur des descentes. Il n'y a pas de corrélations qui sautent aux yeux sur ce petit exemple.



(a) Train **(b)** Test

Figure 3: Histogramme des volatilités d'entrées (sans les outliers) dans les différents jeux de données

On voit que la distribution des volatilités sur une période de 5 mn entre 9h30 et 13h55 dans le jeu d'entrainement et de test sont assez proche. Les outliers (grand) ont été retirés (environ 30 000 données) mais l'on a parfois des volatilités de l'ordre de 40 etc. Ceux-ci ne sont pas particulièrement important puisqu'on utilise la MAPE loss qui donne plus d'importance aux faibles données.

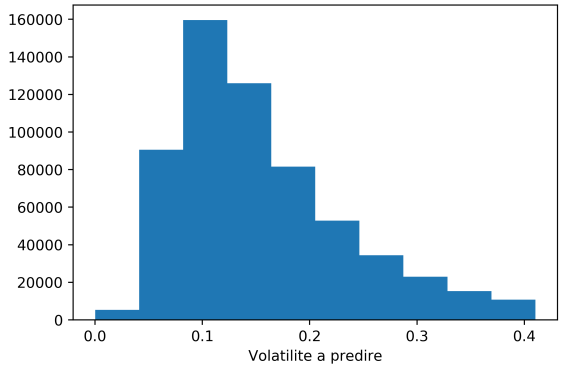
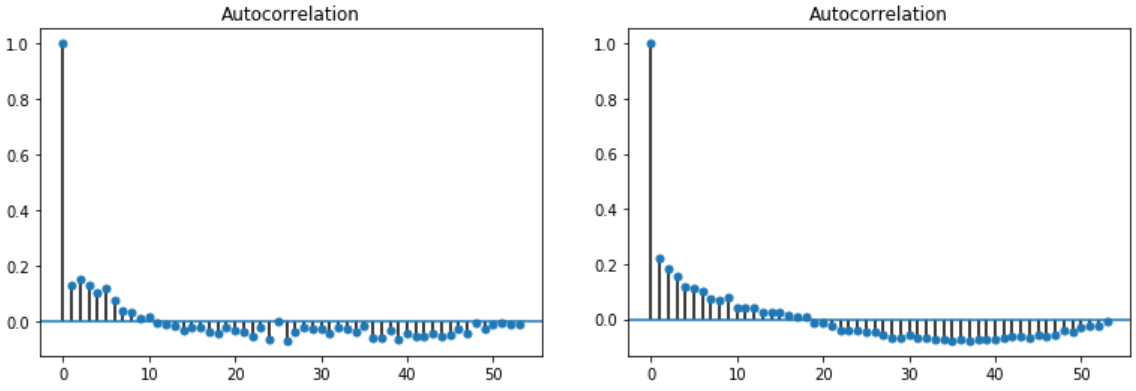


Figure 4: Histogramme des volatilités à prédire (sans les outliers) dans le jeux de données train

On observe les valeurs à prédire par notre algorithme pour le jeu d'entrainement. Etant donné que la distribution des données d'entrée dans le jeu d'entrainement est assez semblable à la distribution des données d'entrée dans le jeu de test, on peut s'attendre à obtenir des résultats avec une distribution assez proche.

2.3 Analyse statistique

2.3.1 Autocorrélation des volatilités



(a) Train **(b)** Test

Figure 5: Autocorrélation de la serie des volatilités (moyenne sur les 10000 premières données)

Ce simple tracé montre que l'on a une dépendance temporelle importante dans notre jeu de données. La dépendance à court terme est extrêmement importante comparativement aux dépendances à long terme.

2.3.2 Corrélations des différentes variables du problème initiale

On peut tracer une matrice de corrélation afin de savoir quelles variables de notre système de base sont corrélées. On rappelle que les caractéristiques sont données dans l'ordre suivant : date, id du produit, 54 features qui décrivent les volatilité toutes les 5 minutes entre 9h30 et 13h55, enfin 54 features qui décrivent le signe des retours sur investissement toutes les 5mn entre 9h30 et 13h55. La dernière caractéristique correspond à la volatilité entre 14 et 16h que l'on veut prédire.

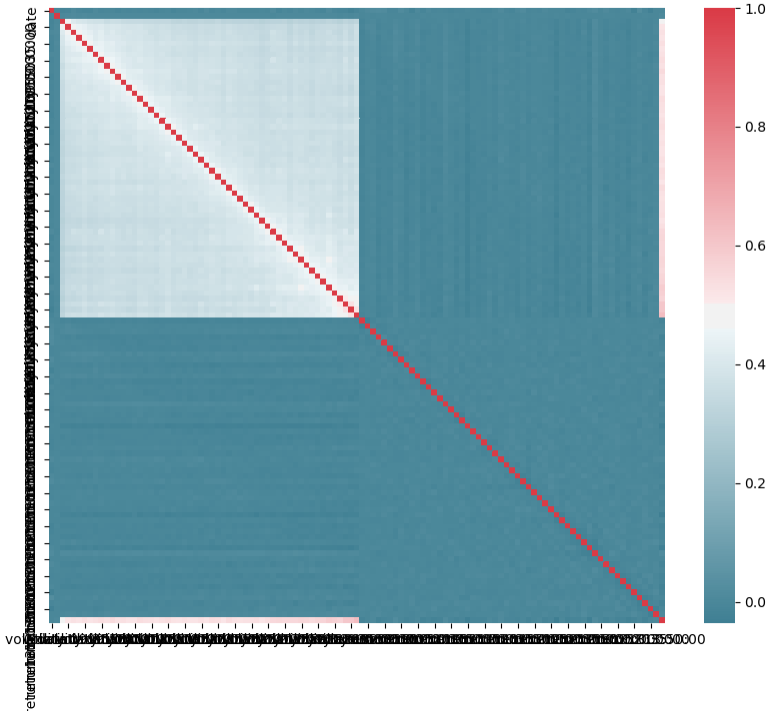


Figure 6: Corrélations des différentes données d'entrainement

Ce premier tracé nous permet d'observer qu'en fait on ne peut pas tirer directement des corrélations entre les variables catégorielles que l'on a et les autres variables (date, produit et les signes des retours sur investissement). Afin de les utiliser il faudra trouver une représentation pertinente de ces quantités. On voit par contre des corrélations non nulles dans les lignes qui correspondent aux 54 volatilités que l'on a dans nos caractéristiques d'entrées ainsi que dans la volatilité cible que l'on veut prédire. On peut analyser ces zones en zoomant sur les parties où l'on a des corrélations non nulles.

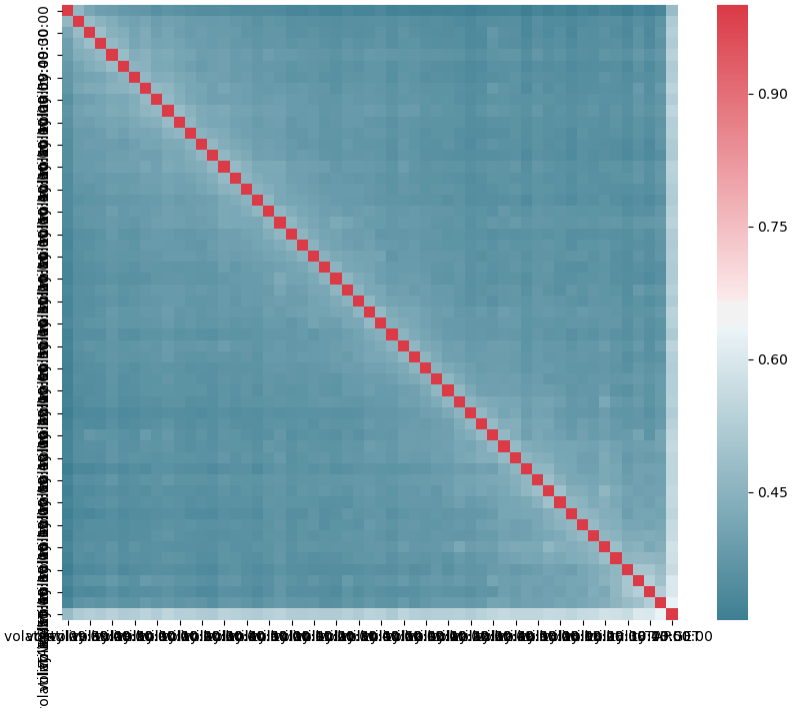


Figure 7: Corrélations des différentes données d'entrainement

On peut observer plusieurs choses grâce à cette representation. On voit que les volatilités pour un produit et une date donnés sont corrélées. On voit que la volatilités à un temps t (à produit et date fixés) est fortement corrélées avec les volatilités aux temps $t-1$ et $t+1$. De la même manière on voit que la volatilité entre 14 et 16 heures que l'on cherche à prédire est corrélée avec une intensité croissante dans le temps aux volatilités sur des périodes de 5mn entre 9h30 et 13h55. Ainsi les volatilités à 13h45,13h50,13h55 sont beaucoup plus informatives que les volatilités à 9h30,9h35,9h40 pour la prédiction que l'on veut réaliser...

2.4 Les valeurs manquantes

On remarque rapidement en parcourant les données que l'on a un nombre non négligeable de valeurs manquantes que l'on doit traiter d'une certaine manière. On a en effet pour les volatilités entre 9h30 et 13h55 sur nos 636 313 exemples de 54 volatilités **419 817 valeurs manquantes soit 1.2%**. Pour les données tests on a 635 397 exemples de 54 volatilités **437 696 valeurs manquantes soit 1.3%**. On a exactement les mêmes statistiques pour les signes des retours sur investissement.
Je pense qu'essayer d'approximer au mieux ces valeurs manquantes est un point déterminant afin d'obtenir des algorithmes performants. On étudiera différentes méthodes pour approximer les valeurs manquantes

2.5 Les produits particuliers

Enfin une dernière remarque importante sur l'analyse des données que l'on peut faire porte sur l'existence de produits particuliers. En effet, comme l'ont fait remarquer des intervenants sur le forum `datachallenge.cfm.fr`, on a **des produits illiquides et particuliers**. Il faudrait donc trouver une manière de distinguer les différents produits de manière pertinente.

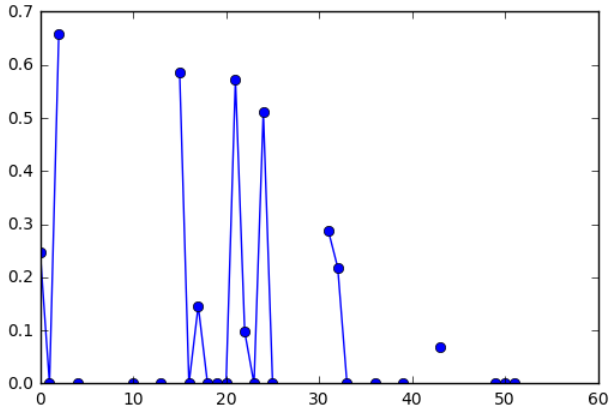


Figure 8: Représentation du produit 211 à une date donnée (image du forum `datachallenge.cfm.fr`

3 La littérature sur la prédiction de volatilités

Avant de commencer un challenge, il est important de voir quel est l'état de l'art dans la tâche que l'on doit réaliser. La littérature sur la prédiction de volatilité semble assez mince. Cela s'explique facilement par les enjeux économiques que peut avoir une prédiction précise de volatilité (les grands acteurs gardent donc leur meilleur solution). De plus, dans notre cas on est limité à l'utilisation de certaines quantités : volatilités sur une durée de 5 minutes, signe du retour sur investissemnt toutes les 5 minutes et des indications sur les produits et la "date' (qui n'est en fait qu'une indication sur quels évènements se passent en même temps).

3.1 La littérature scientifique

Lorsqu'on parle de prédiction de volatilités le modèle qui revient le plus suivant sont les modèles de type ARCH/GARCH. On peut difficilement utiliser des modèles de ce type car il nécessite l'information sur le rendement ou les prix des différents actifs. Ces modèles sont très utilisés en finance pour la prédiction de volatilités [3]. Les modèles GARCH sont réputés pour être capable de prédire efficacement les volatilités à court terme . Ils ont aussi pour avantage d'avoir une grande interprétabilité [4].
On ne peut donc pas utiliser ces modèles néanmoins est-ce réellement grave dans le cadre de ce challenge ? Je ne pense pas puisqu'on trouve nombre d'articles comparant les performances des modèles de statistiques classiques à d'autres techniques de machine learning et il s'avère que les modèles ARCH/GARCH sont souvent dépassés en termes de performances [1]. Leur grand intérêt repose sur l'interprétabilité des résultats et dans le cadre d'un challenge une interprétabilité du niveau de celle que l'on a avec les modèles de statistiques usuelles n'est pas nécessaire.

3.2 Les challenges traitant de sujets connexes

Le challenge est vraiment très similaire au challenge cfm d'il y a 2 ans **Prediction of transaction volumes in financial markets** qui contient des données sous le même format et est évalué de la même manière par la mean absolute percentage error. De plus tout le challenge est basé sur le marché boursier Américain. Dans ce challenge le but était de prédire à partir des informations passés sur le volume des échanges le volume des échanges futurs. Ce challenge est donc très semblable au challenge que l'on a à résoudre. Néanmoins puisque peut d'informations sur les meilleures solutions au cours des challenge data ens

sont disponible on peut difficilement comprendre ce qui a pu marcher dans ce challenge. On trouve également 2 challenges Kaggle portant sur des sujets de finance mais ils semblent assez éloignés de la tâche que l'on doit résoudre.

En conclusion, il y a peu de ressources permettant de gérer le problème que l'on essaye de résoudre dans le cadre de ce challenge. J'ai essayé au cours de ce challenge de faire des analogies avec les différents éléments que j'ai pu voir dans les cours du MVA cette année afin de fournir une solution au problème qui est posé.

4 Le prétraitement des données

4.1 Compléter les valeurs manquantes grâce à des heuristiques simples

Comme on l'a vu on a environ 1.2% de valeurs manquantes. Etant donné que l'essentiel de l'information est contenu dans les volatilités il est important d'essayer d'approximer au mieux les valeurs manquantes de ces volatilités. Etant donné que le signe du retour sur investissement à un temps donné peut être interprété comme une variable catégorielle, on peut en première approximation remplacer les valeurs manquantes dans ce cas par des 0 (comme une troisième catégorie qui signifierait que l'on ne connait pas le sens de l'évolution). Sur l'ensemble des données j'ai pu tester différentes manières simples de remplacer les données manquantes. Je présente ci-dessous les résultats que j'obtiens en retirant aléatoirement 10% des données et en tentant de les retrouver grâce aux variables toujours présentes. J'effectue ce test 10 fois et je présente ici la moyenne des résultats obtenues.
Les différentes méthodes simples testés sur toutes les dates sont :

(i) Interpolation linéaire : on estime la valeur comprise entre deux points déterminés grâce à la fonction affine passant par les deux points déterminés.
(ii) Remplacer par zéro : on remplace toutes les valeurs manquantes par des zéros
(iii) Remplacer par la moyenne : on remplace toutes les valeurs manquantes par la moyenne prises sur les valeurs déterminés que l'on à la date donnée pour le produit donné

Methode	RMSE	MAPE	MAE
Interpolation Lineaire	0.0612	52045.17	0.1366
Remplacer par Zero	0.11752	100	0.21156
Mean ligne	0.05249	70878.976	0.13210

Table 1: Resultat sur toutes les dates des différentes heuristiques pour compléter les valeurs manquantes (test sur 10% des données valides) moyenné sur 10 tests

4.2 Utiliser les techniques de completion de matrices

On remarque deux choses : les valeurs manquantes sont beaucoup moins nombreuses que les valeurs présentes, a une date donnée les valeurs manquantes pour les différents produits ne sont pas présentes en même temps (c-a-d que s'il manque la volatilité à 9h55 pour le produit 310 alor avec forte probabilité on aura les volatilités à ce temps pour une majorité des autres produits).
De plus, on peut estimer qu'au cours d'une même date les différents produits ont une certaine forme de régularité. Puisque les données manquantes n'apparaissent pas en même temps on peut utiliser des techniques classiques de complétion de matrices. On essaye de trouver une solution de bas rang qui permet de compléter la matrice grâce à un algorithme du gradient proximal. Cela revient donc pour résumer à créer 4 237 (nombre de dates) problèmes de complétion de matrices de taille 318x54 (nb_produit x nb_volatilites). J'ai utilisé la libraire `fancyimpute`. Voici le tableau des résultats pour la date 1 où l'on compare la nouvelle méthode étudiée avec les heuristiques précedemment décrites :

Methode	RMSE	MAPE	MAE
Interpolation Lineaire	0.02756319	36269.3220	0.099756
Remplacer par Zero	0.051486	100	0.15661
Mean ligne	0.0224266	40573.0094	0.09229767
Matrix Completion	0.014897	35261.0335	0.07486

Table 2: Resultat pour la date 1 (test sur 10% des données valides) moyenné sur 10 tests

On obtient effectivement le meilleur résultat avec la completion de matrices si on se réfère à la RMSE et la MAE. Néanmoins afin d'obtenir ce résultat sur une simple date il faut environ 20 mn (sur l'environnement de développement Google Colab) et 5000 itérations de l'algorithme (sans convergence). Le résultat pourrait encore être meilleur puisque l'on n'est pas allé jusqu'à convergence ($1e; -4$ comme critère). Cette méthode n'est donc pas utilisable sous cette forme puisque l'on a plus de 2000 dates juste dans le jeu d'entrainement.

4.3 D'autres idées de prétraitement auxquelles on peut penser

On peut également penser à utiliser des méthodes supervisées afin de reconstruire les valeurs manquantes. On peut par exemple utilser une architecture de réseaux de neurones de type auto-encoder que l'on alimenterait grâce à des exemples construit de la façon suivante : l'entrée serait les différentes séries de volatilités auxquelles on aurait retiré quelques entrées ; on associerait cette nouvelle entrée à une sortie qui serait la même entrée sans les masques sur les séries de volatilités que l'on aurait rajouté.
En pensant aux auto-encoders, on aurait aussi pu étudier différentes méthodes pour débruiter les données, ondelettes ou auto-encoders par exemple.

Solution utilisé au final : Au final j'ai estimé que l'interpolation linéaire fournissait le meilleur compromis parmi les méthodes que j'ai pu tester. Les méthodes de complétion de matrices sont trop couteuses. L'interpolation linéaire surpasse le remplacement par la moyenne et par des zéros lorsqu'on évalue la performance avec la mean squared error. Elle est très proche du remplacement par la moyenne lorsque l'évaluation est basée sur la mean absolute error mais est bien meilleure avec la mean absolute percentage error.

5 Mon algorithme de résolution

5.1 Le Benchmark

Le benchmark donné par CFM consiste à estimer la volatilité entre 14 et 16h comme la moyenne de la volatilité obtenue sur tous les pas de temps précédents. Ce procédé permet d'obtenir un score sur le public leaderboard de 28.207. Des petites variantes de cette méthode (prendre la médiane à la place de la moyenne, prendre la moyenne sur les 20 derniers pas de temps seulement ...) permet de légerement améliorer ce score et d'obenir des scores entre 26 et 28.

5.2 Le choix de l'algorithme de base

Dans le cadre du challenge on utilise la mean absolute percentage error alors que dans les régresseurs classiques (sur sklearn notamment) la loss par défaut est la mean squared error (la mean absolute percentage error n'est pratiquement jamais disponible puisqu'elle n'est pas défini lorsque la valeur à prédire est nulle ; de plus sa hessienne est nulle). Certains articles ont justifié l'utilisation de la loss sur laquelle le challenge sera évalué pour entrainer nos modèles [2]. Si l'on est évalué avec une mean absolute percentage error alors l'optimisation doit être fait sur cette loss. J'ai pu effectivement constater cela en utilisant des algorithmes donc la fonction coût n'est pas la mean absolute percentage error j'arrive à des résultats très proche du

Benchmark.
Ainsi, je n'ai trouvé que deux librairies pour lesquelles l'optimisation avec la mean absolute percantage error était un peu près bien gérée. `Light GBM` et les modèles de deep learning avec `Keras` (backend `tensorflow`). J'ai d'abord commencé par utiliser les modèles de deep learning pour fournir mes premières solutions mais j'ai atteint un palier avec cet algorithme à 22.3% sur le public leaderboard. J'ai donc finalement orienté mes études sur l'utilisation de `Light GBM` puisque je n'ai pas trouvé une architecture neuronale optimale pour résoudre le problème. Dans la suite je présente ma solution qui a fourni mon meilleur résultat sans présenter dans le détail tous les algorithmes et méthodes que j'ai pu tester.

5.3 Feature engineering

Comme a pu le décrire l'organisateur du challenge, on a un grand nombre de paramètres qui peuvent avoir de l'importance lorsque l'on veut prédire la volatilité entre 14 et 16h d'un produit p à la date d. En effet, on peut considérer que l'évolution de tous les autres produits à la date d donne une information sur l'évolution de la volatilité recherchée. De la même manière toutes les anciennes évolutions du produit dans notre base de données ont également une importance. Ainsi on a en principe environ 108*(318+2119)=263 196 caractéristiques à considérer pour prédire une volatilité cible. C'est évidemment beaucoup trop étant donné que l'on a 635 397 exemples. La première tache dans le cadre de la résolution du problème est donc de créer des caractéristiques/features afin d'alimenter l'algorithme final que l'on va utiliser. Dans ce qui suit je résume l'essentiel des caractéristiques que j'ai donné à mon algorithme final pour obtenir mon meilleur résultat.

5.3.1 Des features construit manuellement

Etant donné nos 110 caractéristiques directs par exemple d'entrainement on considère les caractéristiques suivantes :

- Les 5 dernières volatilités brutes (puisqu'on a remarqué que la corrélation étaient décroissante avec l'écart en temps)
- La moyenne des 54 volatilités entre 9h30 et 13h55
- La moyenne sur les 20 dernières volatilités que l'on a
- La moyenne mobile exponentiellement pondérée (EWMA) sur les 54 volatilités entre 9h30 et 13h55
- L'écart type des 54 volatilités entre 9h30 et 13h55
- La plus grande valeur de volatilité dans la journée
- La plus petite valeur de volatilité dans la journée
- L'aire sous la courbe défini par nos 54 volatilités
- Le kurtosis (moment d'ordre 4 sur le moment d'ordre 2) de la distribution des 54 volatilités
- Le coefficient d'asymétrie de la distribution des 54 volatilités

J'ai également pensé que la manière avec laquelle évolue les volatilités toutes les 5mn est une donnée assez importante. Ainsi j'ai également créé des features relatives au rapport entre les volatilités nun nulles aux temps t et aux temps $t+1$ pour la date et le produit donnés. J'ai donc considéré :

- La moyenne des rapports des volatilités entre les temps t et les temps $t+1$
- L'écart type des rapports des volatilités entre les temps t et les temps $t+1$
- La plus grande valeur des rapports des volatilités entre les temps t et les temps $t+1$
- La plus petite valeur des rapports des volatilités entre les temps t et les temps $t+1$
- Le rapport entre les volatilités à 13h55 et 13h50 (soit le dernier rapport de notre série)

J'ai utilisé peu de features sur le signe du retour sur investissement, j'ai eu du mal à exploiter cette information. J'ai considéré dans ce cadre :

— La moyenne des signes des retours sur investissement (pour savoir si le produit a plutôt tendance à monter ou non)

— L'écart type des signes des retours sur investissement

— Le dernier signe de retour sur investissement (soit celui à 13h55)

On a donc construit 21 features qui sont censées représenter les 108 caractéritisques initiales de chaque exemple (les 54 volatilités entre 9h30 et 13h55 et les 54 sens d'évolution entre 9h30 et 13h55). Il reste à concatener manuellement l'information contenu dans la donnée sur le produit et la date. J'ai donc construit manuellement les features suivantes pour représenter l'information sur la date :

— La moyenne des volatilités sur les 380 produits à 13h55 à la date considérée (pareil pour 13h50 puis pour 13h45)

— L'écart type des volatilités sur les 380 produits à 13h55 à la date considérée (pareil pour 13h50 puis pour 13h45)

— Le kurtosis sur les 380 produits à 13h55 à la date considérée (pareil pour 13h50 puis pour 13h45)

— Le coefficient d'asymétrie sur les 380 produits à 13h55 à la date considérée (pareil pour 13h50 puis pour 13h45)

— La plus grande valeur des volatilités sur les 380 produits à 13h55 à la date considérée (pareil pour 13h50 puis pour 13h45)

— La plus petite valeur des volatilités sur les 380 produits à 13h55 à la date considérée (pareil pour 13h50 puis pour 13h45)

— La moyenne de toutes les volatilités sur les 380 produits entre 9h30 et 13h55 à la date considérée

— La plus grande de toutes les volatilités sur les 380 produits entre 9h30 et 13h55 à la date considérée

— La plus petite de toutes les volatilités sur les 380 produits entre 9h30 et 13h55 à la date considérée

Pour synthétiser les informations portées par l'id du produit j'ai considéré :

— La moyenne des volatilités entre 14h et 16h pour le produit considéré (sur tout le training set)

— L'écart type des volatilités entre 14h et 16h pour le produit considéré (sur tout le training set)

— La plus petite valeur des volatilités entre 14h et 16h pour le produit considéré (sur tout le training set)

— La plus grande valeur des volatilités entre 14h et 16h pour le produit considéré (sur tout le training set)

— Le kurtosis des volatilités entre 14h et 16h pour le produit considéré (sur tout le training set)

— Le coefficient d'asymétrie des volatilités entre 14h et 16h pour le produit considéré (sur tout le training set)

Au final on a donc une **cinquantaine de features qui résument partiellement les 263 196 caractéristiques initiales**. Avec juste ces features et la libraire `Light GBM`, on arrive à un score d'environ 22.5 sur le public leaderboard (environ 6-10eme sur le public leaderboard).

Remarque importante : Puisqu'on a donné des informations sur les valeurs cibles à prédire et les dates, l'algorithme peut facilement overfitter les données d'entrainement. Afin de s'en rendre compte il est primordial lors de la validation de séparer les données en se basant sur l'information sur la date. Il faut que les dates qui apparaissent dans la validation n'apparaissent pas lors de la phase d'entrainement (pour être dans les conditions de la phase de test). Sinon, on peut facilement observe des scores de 18-19 lors de la validation alors que le score sera de l'ordre de 22-23 sur les leaderboards du challenge.

A ces features que j'ai créé manuellement j'ai voulu ajouter des features apprises grâce à différentes méthodes d'apprentissage et ensembliste afin d'améliorer mon résultat.

5.3.2 Créer un embedding pour les différents produits *Product2Vec*

`Word2Vec` [5] est probablement l'une des applications du Deep Learning appliqué au NLP les plus connues. Il s'agit de trouver une représentation vectorielle des mots dans un espace dans lequel deux mots à la sémantique proche seront proches. L'idée se base sur le fait que le sens d'un mot est défini par les mots qui entourent ce dernier. Ainsi, afin de trouver une représentation qui permet de fixer une relation entre les différents produits avec lesquelles on travaille on peut s'inspirer de cette idée pour la trouver.
L'idée : Pour implémenter cette idée en utilisant `Keras` j'associe à chaque produit une représentation vectorielle de dimension 50 entrainable. Etant donné un produit et une date (ce qui correspond à un exemple dans notre base). Je donne à l'algorithme les 50 volatilités entre 9h30 et 13h35 de **tous** les 318 produits concaténés avec la représentation vectorielle du produit que l'on a répété 318 fois (une fois pour chaque produit) et je lui demande de prédire les volatilités représentant l'évolution sur une durée de 5mn commençant à 13h35, 13h40, 13h45, 13h50, 13h55. On peut trouver ci-dessous l'architecture du réseau que j'ai utilisé :

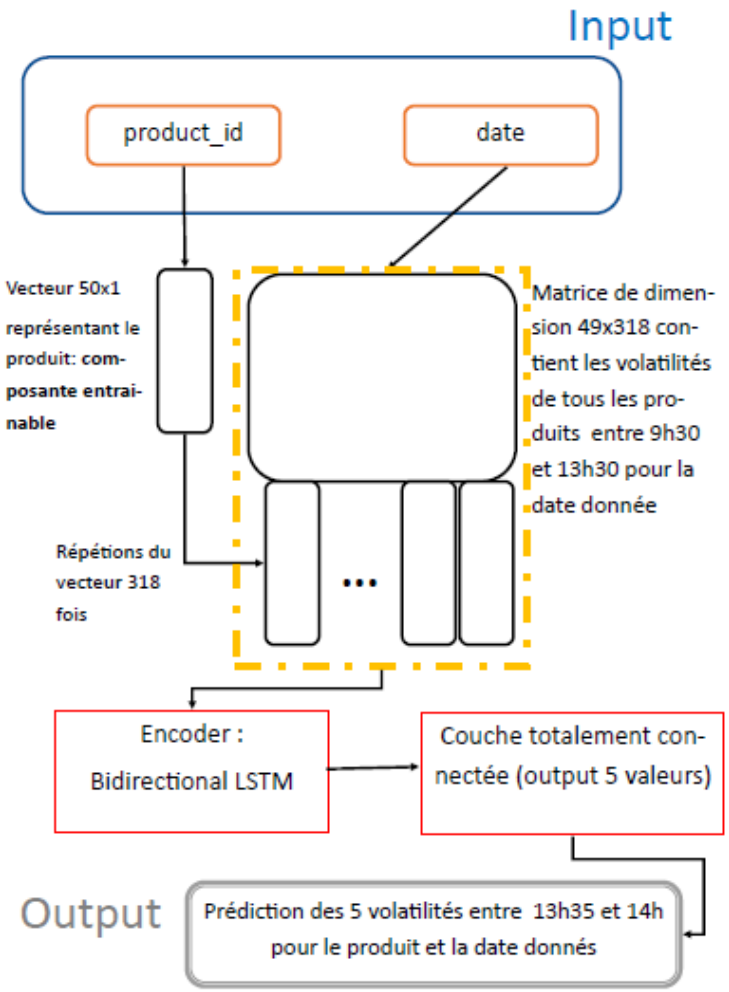


Figure 9: L'architecture neuronale utilisée : le but est de récupérer la représentation du produit que l'on a apprise

L'apprentissage est basé sur la MAPE loss et sur les exemples de test et d'entrainement de notre base

d'exemples. On peut projeter en dimension 2 avec t-SNE la représentation apprise. On observe des légers regroupements, il est néanmoins difficile d'interpréter la justesse du résultat de manière simple puisque l'on ne dispose pas d'informations sur les différents produits considérés.

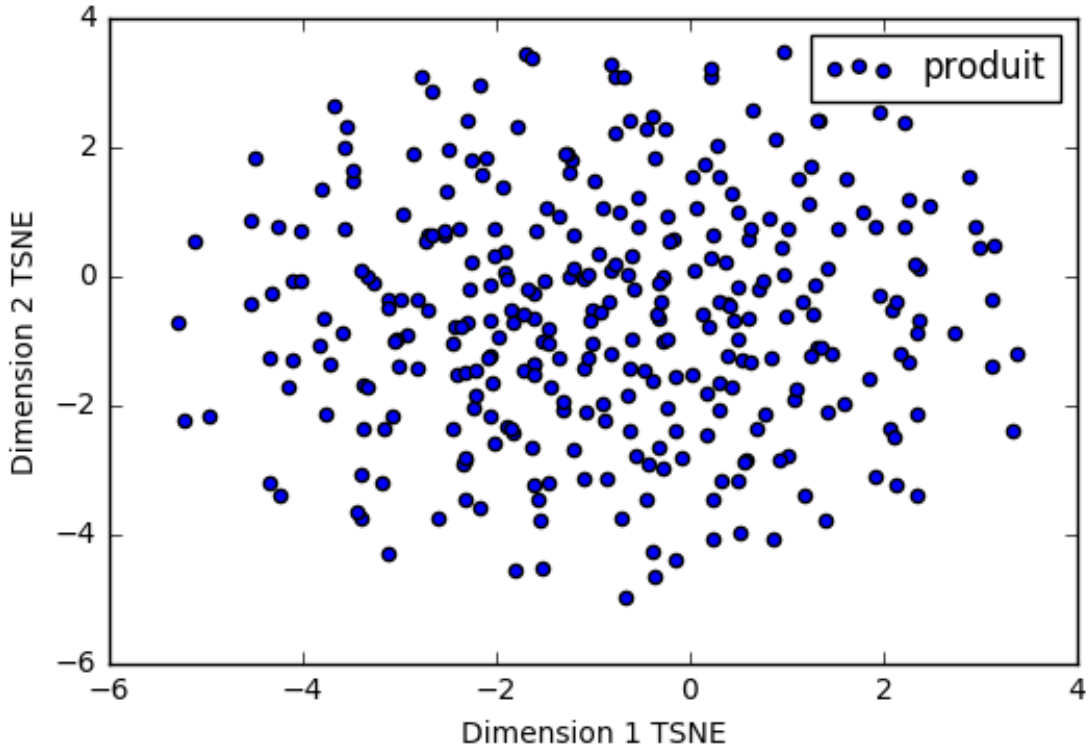


Figure 10: Visualisation avec TSNE de la représentation vectorielle des produits

On exploite cette représentation qui a été sérialisé avec `pickle` en la projetant sur ces 20 premières composantes principales. On ajoute donc à nos caractéristiques initialies décrites dans la partie précédente ces 20 caractéristiques correspondant à une représentation vectorielle du produit pour lequel on veut connaitre la volatilité entre 14h et 16h.

5.3.3 L'utilisation de méta-modèles : le Stacking

Une technique très répandue pour compléxifier le modèle et lui permettre d'apprendre des caractéristiques de haut niveau d'abstraction est le stacking ou mixture of experts. Concrètement, on utilise des modèles d'apprentissage qui doivent fournir des prédictions pour un algorithme qui va s'appuyer sur ces prédictions pour fournir une prédiction plus précise.
Il faut faire attention à ne pas biaiser son modèle lorsqu'on utilise cette méthode. La procédure que j'ai utilisé est la suivante : séparer le dataset d'entrainement en 2 datasets A et B, on entraine nos modèles avec le datasets A (resp B), ceux-ci doivent fournir une prédiction pour le dataset B (resp A) qui n'a pas été utilisé lors de l'entrainement, il fournit également une prédiction pour les données tests. On utilise ces prédictions comme des nouvelles features pour notre algorithme. Afin de rendre cette méthode efficace, il faut que les prédictions soit assez variées, il est inutile de ne mettre que des simple Light GBM. Ainsi j'ai utilisé plusieurs modèles qui constitueront les experts qui me fourniront des nouvelles features grâce à la méthode du stacking.

5.3.3.1 Les modèles "simples"

J'ai utilisé plusieurs modèles simples ou plutot simple à utiliser puisque leur utilisation est simplifiée grâce aux différentes librairies qui existent. J'ai ainsi eu des prédictions grâce aux modèles suivants :

— ElasticNet
— Une simple régression linéaire
— Un Light GBM optimisé avec la MAPE loss
— Catboost optimisé avec la RMSE (un nouveau algorithme de boosting réputé plus performant que XGBoost et Light GBM)

5.3.3.2 Transformer le problème de regression en problème de classification

Beaucoup de modèles de machine learning sont réputés être beaucoup plus performant pour la classification que pour la regression. Ainsi, une idée assez naturel que j'ai eu a été de transformer notre problème de regression en un problème de classification. Pour ce faire on peut utiliser un modèle type K-Means ou Gaussian Mixtures qui doit créer des clusters parmis les données de sorties que l'on cherche à prédire (c-a-d les volatilités entre 14 et 16h). Utiliser cela permet ensuite d'associer chaque exemple d'entrainement à un unique label correspondant au label de la cible à prédire dans notre modèle de clustering (K-Means ou Gaussian Mixtures). On voit que si l'on prédit à la perfection les données que l'on a en training obtient au mieux un score de 0.170 pour un Gaussian Mixtures avec 200 clusters (soft assignement) et entre 0.07 et 0.20 pour un K-Means entre 10 et 30 clusters. On utilise ensuite un classifieur (ici LGBM) qui prend en entrée les features qui ont été décrites dans les deux parties précédentes et est entrainé pour prédire les labels donnés par le modèle de clustering (GMM ou K-Means).
La valeur prédite sera alors la moyenne des centres de tous les clusters, chaque terme étant pondéré par la probabilité que notre exemple est une sortie associée à ce cluster (donnée par le classifieur LGBM). Cette méthode permet d'obtenir un score de validation d'environ $24 - 25$ s'il était utilisé directement.

5.3.3.3 Utilisation de modèles de Deep Learning

Enfin j'ai utilisés deux architecture de réseaux de neurones pour obtenir de nouvelles feature :

— Un réseau basé sur une seule couche de Bidirectional LSTM qui prend en entrée uniquement les 54 volatilités brutes et sort la volatilité entre 14 et 16h à prédire
— Un deuxième réseau qui a la même base que le réseau précédent on rajoute juste à la sortie du LSTM les features que l'on a calculé dans les 2 parties précédentes.

Ces réseaux sont optimisés avec la MAPE loss, des dropouts et recurrent dropout ont été utilisés pour éviter l'overfitting.

5.4 Eviter l'overfitting : L'utilisation du Bagging

Comme on l'a vu on a crée environ 80 features qui résument partiellement les informations que l'on a sur chaque exemples. J'ai utilisé les méthodes de régularisations direct et classique tel que le *Dropout* pour mes réseaux de neurones ou encore utiliser le paramètre *reg_lambda* dans `Light GBM` qui controle la complexité des arbres utilisés. J'ai également mis en place une stratégie basée sur le bagging qui est décrite ci-dessous :
Utiliser les méthode d'early stopping : Pour moi, une des meilleures manière d'arrêter un algorithme lorsqu'il surapprend sur la base d'entrainement est d'utiliser l'early stopping. Concrètement on utilise un certain pourcentage des données (dans mon cas 10%) que l'on sort de la base d'apprentissage et que l'on utilise comme données de validation pour surveiller l'évolution de la fonction coût sur des exemples qui ne sont pas dans la base d'apprentissage. On arrête l'algorithme lorsque l'on observe une dégradation du résultat sur les données de validation. Cette approche bien que très utile a deux défauts principaux : on se prive de

10% des données d'apprentissage, on va en quelques sortes surapprendre sur la base de validation puisque l'on arrête l'algorithme pile où il sera meilleur pour cette base. Ainsi afin d'améliorer la généralisation de mon modèle j'utilise le bagging.
Utiliser le bagging (Bootstrap aggregating) : Cette méthode utilisée notamment dans le random forest est très simple. On crée plusieurs régresseurs qui voient des sous-ensembles différents des données d'apprentissage puis on on agrége les résultats en faisant la moyenne de tous les régresseurs. Cette technique se combine parfaitement avec l'early stopping qui a été détaillé plus haut ! Ainsi dans le cadre de ma résolution, j'utilise **100** `Light GBM` qui utilisent chacun 90% des exemples d'entrainements comme training set et les 10% restants comme données de validations pour l'early stopping. Cette répartition entre données d'entrainement et données de validation est tirée aléatoirement pour chacun des 100 régresseurs.

5.5 Résumé de l'algorithme utilisé

Ci-dessous on trouve un schéma résumant l'algorithme qui a été utilisé :

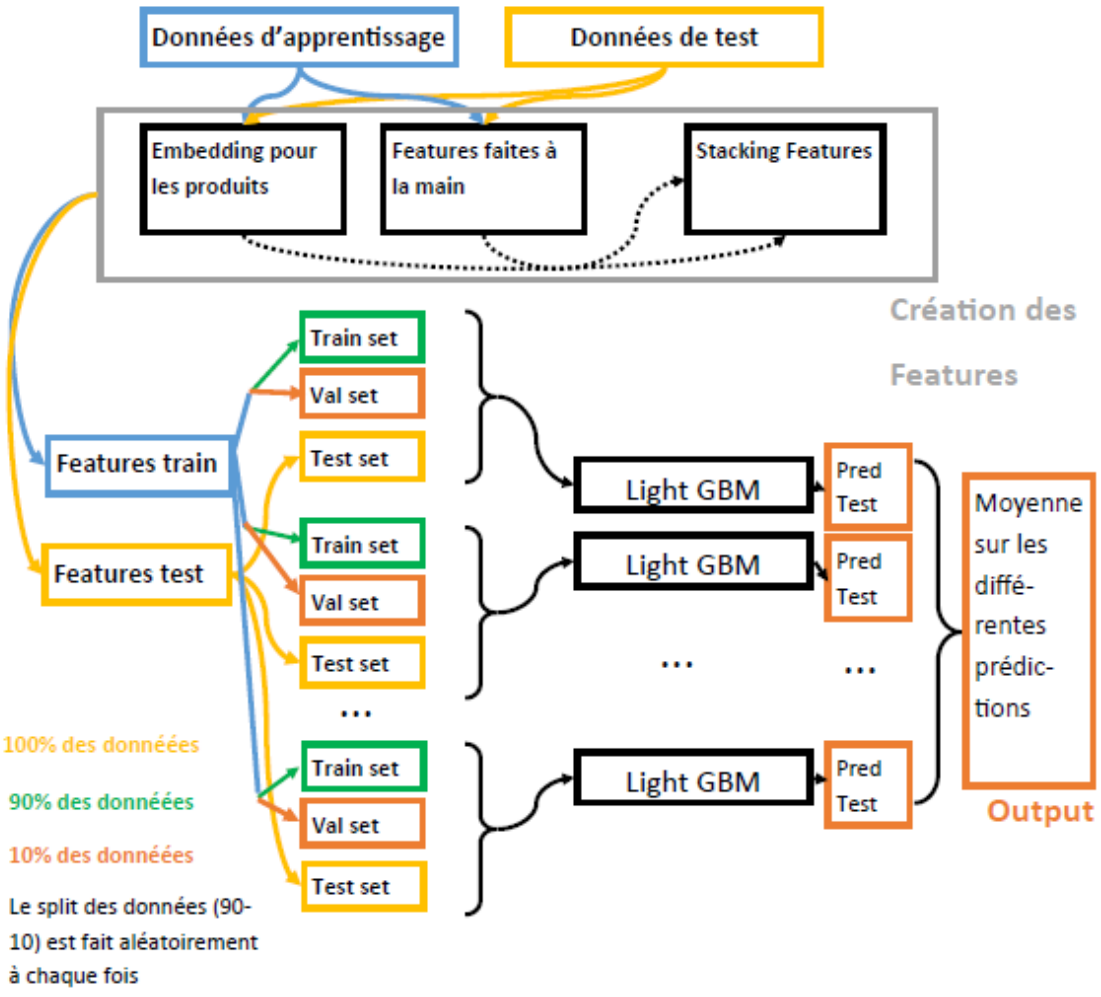


Figure 11: L'architecture globale de l'algorithme qui a été construit

5.6 Les features les plus importantes

Etant donné que l'algorithme principal est un algorithme à base d'arbres de décisions on peut mesurer l'importance de chaque features (si on utilisait pas le bagging). Sur un arbre de décision on mesure l'importance d'une caractéristique par l'amélioration dans le résultat qu'apporte une séparation par la donnée de la caractéristique (plus une caractéristique apparait haut dans l'arbre plus elle a de l'importance). Je liste ici certaines features des plus importantes (trié des plus importantes au moins importantes) :

— La prédiction que l'on obtient en transformant le problème en probème de classification (Stacking Features)
— La prédiction du LGBM (Stacking Features)
— Le résultat de la régression linéaire (Stacking Features)
— La moyenne des volatilités sur les 380 produits à la date considérée
— La prédiction du réseau de neurones avec les volatilités brutes et les features faites à la main en entrée (Stacking Features)
— Suivent ensuite les différentes features créé à la main concernant la date considérée de manière très proche

On voit que les features qui ont le plus d'importance sont globalement les features issues des autres regresseurs que l'on a utilisé, c'est à dire du Stacking. Ceci est tout à fait compréhensible puisque ces features contiennent en fait implicitement des informations sur les autres features puisque ces modèles ont été entrainés pour réagir aux autres features.

6 Data Augmentation

Durant ces derniers jours, j'ai voulu essayer d'agrandir ma base d'apprentissage en exploitant la propriété de reversibilité des séries temporelles de volatilités. Ceci n'est pas tout à fait observable sur nos séries où l'on a l'impression que l'essentiel de l'activité se fait en début et en fin de séance (volatilités plus élevés). Néanmoins puisque le début et la fin de l'activité semble être les moments de forte activité je pensais multiplier par deux la taille de mon jeu d'entrainement en retournant toutes les séquences pour créer de nouveaux exemples.
Néanmoins l'une des difficultés dans cette approche réside dans le fait que l'on veut prédire une volatilité qui s'étale sur une période de 2h alors que l'on a des volatilités qui s'étalent sur 5mn en entrée. Ainsi pour concrétiser mon idée, il faudrait une manière précise de transformer les 24 volatilités de 5mn en une volatilité sur 2h. En première approximation, j'avais pensé créer un estimateur de la forme :

$$\sigma_{2h} = \sigma_{\bar{5mn}}\sqrt{24}$$

Néanmoins ceci assume que les volatilités sont distribués de manière iid ce qui n'est pas le cas et cela fournit des données très différentes de celles que l'on trouve dans les vrais cibles.
Ainsi, rajouter les exemples crées ainsi vient dégrader le résultat que l'on a sans les utiliser.

7 Critique de l'algorithme en place et futurs améliorations

On peut formuler quelques critiques sur l'algorithme en place :

— On utilise pratiquement pas l'information sur le signe du retour sur investissement
— L'algorithme est assez complexe et met un temps non négligeable à être executé.
— On a également besoin de machines avec un GPU relativement puissant pour créer le product embedding (j'ai personnellement utilisé Google Colab).
— On utilise peu de features qui se basent sur le fait que l'on travaille avec des volatilités d'actifs financiers.

Dans le futur, je vais donc essayer de nouvelles méthodes pour améliorer mon résultat. J'ai notamment pensé à :

— Essayer un prétraitement/débruitage/compléter les données manquantes grâces à des auto-encoders

— Réfléchir à des nouveaux indicateurs métiers pertinents

— Peut être séparer l'apprentissage en fonction de groupe de produits (le produit 211 par exemple semble très particulier mais je le traite comme les autres).

— Essayer d'autres architectures pour le "product embedding"

— Essayer d'augmenter ma base d'apprentissage avec des techniques modernes (par exemple en utilisant des GAN ?).

8 Conclusion

Dans le cadre de ce challenge, qui avait pour but de prédire les volatilités futures entre 14h et 16h à partir des volatilités passées entre 9h30 et 13h55, j'ai eu l'occasion de mettre en pratique plusieurs concepts que j'ai pu voir pendant cette année au MVA. J'ai notamment essayé divers prétraitements pour remplacer les valeurs manquantes grâce notamment à des techniques de completion de matrices. J'ai également pu travailler sur ce problème qui pouvait potentiellement s'exprimer en très grande dimension (plus de 200 000 caractéristiques potentielles) en extrayant des features que j'ai jugé pertinentes et qui s'avèrent aider mes algorithmes. J'ai également chercher à trouver une représentation vectorielle des différents produits financiers. Enfin j'ai essayé d'appliquer différentes techniques de régularisation pour améliorer mon résultat.
En définitive, j'ai pu essayer de nombreuses choses afin de fournir une solution viable au problème qui était posé. Ce problème est très riche, il reste énormement de possibilités et d'idées à exploiter afin d'améliorer le résultat que j'ai pu atteindre (21.75 de MAPE loss sur le public leaderboard). Ce challenge fut très intéressant et m'a permis de beaucoup apprendre !

Références

[1] Fahima Charef, Fethi Ayachi, et al. A comparison between neural networks and garch models in exchange rate forecasting. 2016.

[2] Arnaud De Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. Using the mean absolute percentage error for regression models. In Proceedings, page 113. Presses universitaires de Louvain, 2015.

[3] Robert F Engle and Kevin Sheppard. Theoretical and empirical properties of dynamic conditional correlation multivariate garch. Technical report, National Bureau of Economic Research, 2001.

[4] Carole Fares. Estimation et prévision de la volatilité de l'indice s&p 500. 2008.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.