

Module 8

Participez à une compétition Kaggle

Problématique

La compétition choisie consiste à effectuer des prévisions sur le volume de vente d'une chaîne de magasins, en fonction de l'identifiant du magasin, de la nature du produit, et de la date.

La compétition choisie pour les débutants sur Kaggle : elle n'est pas limitée dans le temps, et n'a pas de prix à gagner.

Data

Les données mises à disposition contiennent aussi des données exogènes : événements exceptionnels, cours du pétrole, calendrier des jours fériés.

Les données courent du 01/01/2013 au 15/08/2017, et les prévisions à effectuer sont du 16/08/2017 au 31/08/2017.

Sachant que les données sont réparties en fonction de l'id du magasin (55 au total) et de la famille de produits (33 au total), il faut faire une prévision sur $16j * 54 \text{ stores} * 33 \text{ familles} = \mathbf{28512}$ points au total.

Les données sélectionnées sont donc :

- Date
- ID Store
- Family
- Sales

Solution proposée

GluonTS

Nous avons orienté notre choix vers la librairie GluonTS (<https://ts.gluon.ai/stable>).

Cette librairie est dédiée pour traiter du Time Series Forecasting, elle met à disposition des modèles probabilistes, à base de réseaux de neurones, prés entraînés.

Le choix de modèle est assez vaste, proposant plus de 20 modèles (https://ts.gluon.ai/stable/getting_started/models.html).

Pour notre sujet, nous utilisons le modèle recommandé par défaut : DeepAR. C'est un modèle qui est basé sur un RNN (recurrent Neural Network).

Cette librairie, et ce modèle en particulier, est utilisé par AWS pour leur offre 'Forecast' (<https://aws.amazon.com/fr/forecast/>).

Etat de l'art

Aujourd'hui, les RNN sont des modèles très utilisés pour le time series forecasting, ils ont démontré leur efficacité.

Une variante existe, les modèles LSTM qui sont une évolution des modèles récurrents RNN, et permettent de garder en "mémoire" les données, ceci afin de répondre au problème du 'vanishing gradient' inhérent aux RNN.

Autres approches

Il existe d'autres approches pour traiter du TSF : modèles statistiques ou gradient boosting.

Modèle statistique : nous avons commencé à tester la librairie Prophet (<https://facebook.github.io/prophet/>), qui propose des modèles à base statistique (ARIMA, ...), mais nous n'avons pas obtenus de bons résultats de prime abord.

Gradient Boosting / XGBoost : ce type de modèle est souvent choisi par les membres de la communauté Kaggle pour cette compétition.

Préparation des Données

Données manquantes

GluonTs a une contrainte concernant le format et le contenu des données pour le training : il faut que toutes les dates soient contiguës et suivre un certain format.

Après analyse des données, on s'aperçoit qu'il manque pour l'ensemble des catégories, une valeur de vente pour le 25 décembre de chaque année. Nous avons donc comblé ce manque en indiquant un volume de vente à 0

Données Multivariées

Le format GluonTS peut être généré à partir d'un DataFrame pandas, et peut être de type multivarié, avec une seule variable.

En pratique, dans notre dataset nous avons deux variables : l'id du store et la famille de produit. Donc, la solution utilisée est de splitter le dataset par id de store, ensuite effectuer un training et des prévisions multivariées en fonction de la famille de produits.

Nous avons donc N modèles, selon N stores. Nous stockons les résultats des prévisions de ces modèles dans une structure qui sera ensuite utilisée pour produire le fichier de prévisions.

Par ailleurs, l'avantage de cette approche, permet de réduire le nombre de points à prévoir pour chaque modèle (pour rappel au total > 28000 points). Par exemple chaque modèle ne devra prévoir que 16 points (16 jours)

Scaling et Outliers

Scaling

Il est normalement nécessaire d'effectuer un scaling (MinMax 0, 1) des données afin de permettre un meilleur entraînement des modèles. Or dans notre cas, la performance était moins bonne.

Outliers

Pour enlever les outliers, on utilise la méthode IQR / mediane : mettre la valeur médiane pour les données comprise entre ± 1.5 IQR (IQR étant la différence entre le dernier et le premier quartile).

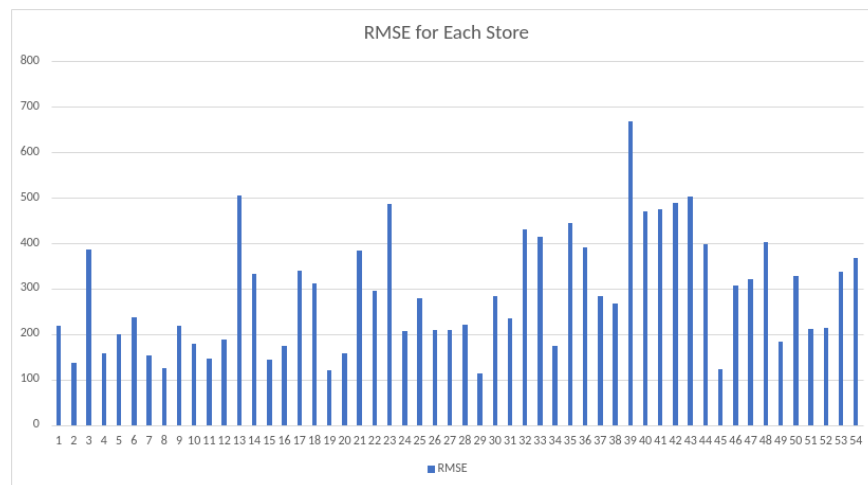
Or, les performances sont moins bonnes après traitement.

Mesure de la performance du modèle

GluonTS propose plusieurs indicateurs pour mesurer la performance du modèle. Nous choisissons de nous focaliser sur la RMSE.

Le meilleur score obtenu est une **RMSE** de **285** (en moyenne, car dépendant de chaque famille de produit).

Cf ci-dessous un graphe qui représente les RMSE obtenus pour chaque store. On voit qu'on obtient un rapport de 1 à 7, suivant chaque modèle.



Caractéristiques

Le meilleur compromis performance / temps de calcul des hyper paramètres du modèle sont :

- Nombre d'épochs : 5
- Learning rate : 10^{-3}
- Nombre de batches par epoch: 10
- Patience: 2 (early stopping)

Perspectives

Nous avons des marges de manœuvre pour tenter d'améliorer la performance du modèle.

Données

Nous n'avons pas encore exploité toutes les données mises à disposition, on pourrait donc enrichir la donnée ou avoir une approche plus précise des outliers (événements exceptionnels).

De même, il faudrait creuser pour comprendre pourquoi le scaling donne de moins bons résultats (ce qui est contre intuitif de prime abord).

Modèle

On pourrait faire un grid search afin d'affiner les hyper paramètres.

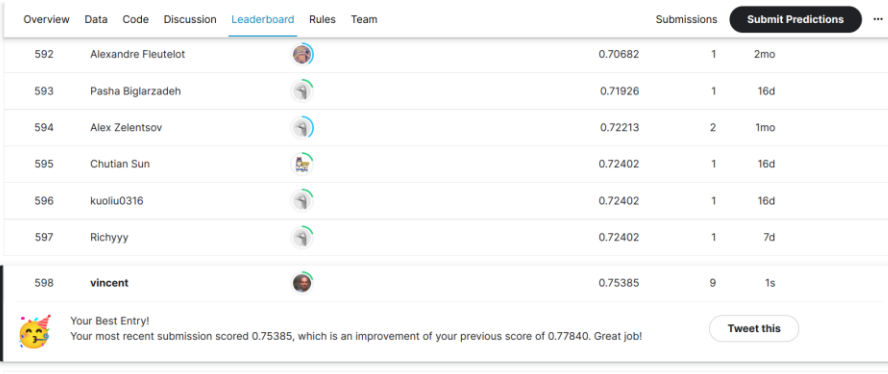
On pourrait aussi utiliser un autre modèle proposé par GluonTS, mais dans ce cas cela aura vraisemblablement un impact sur le format des données (certains modèles nécessitent un format particulier : ajout de variables, etc.)

On pourrait avoir une approche complètement différente à base de gradient boosting, en utilisant XGBoost / LightGBM par exemple.

Conclusion et Résultats obtenus

En fonction des données utilisées mentionnées ci-dessus, le meilleur rang obtenu est de 588ème sur +700 (au moment de la soumission). Ce qui est un score honorable à ce stade (avant applications des optimisations évoquées ci-dessus).

Erratum : après avoir introduit la donnée sur le cours du pétrole, on obtient un score amélioré à 0.75 (5% de gain).



The screenshot shows a competition leaderboard with the following data:

Rank	Participant	Score	Submissions	Time
592	Alexandre Fleutelot	0.70682	1	2mo
593	Pasha Biglarzadeh	0.71926	1	16d
594	Alex Zelentsov	0.72213	2	1mo
595	Chutian Sun	0.72402	1	16d
596	kuollu0316	0.72402	1	16d
597	Richyyy	0.72402	1	7d
598	vincent	0.75385	9	1s

Below the table, a message states: "Your Best Entry! Your most recent submission scored 0.75385, which is an improvement of your previous score of 0.77840. Great job!" with a "Tweet this" button.

Participer à cette compétition a été un challenge intéressant, me permettant de mettre en œuvre du TSF, ce qui est assez rare (peu de compétitions de ce type sont disponibles).

La communauté est active, et permet de trouver des exemples de notebook intéressants et instructifs.

En revanche, pour la compétition choisie, on ne peut pas voir le notebook qui a gagné, et donc comprendre comment le problème a été traité, ce qui est dommage pour l'apprentissage mais compréhensible pour la compétition.