



Séries temporelles

Méthodes ML et DL

Machine learning pour la prévision

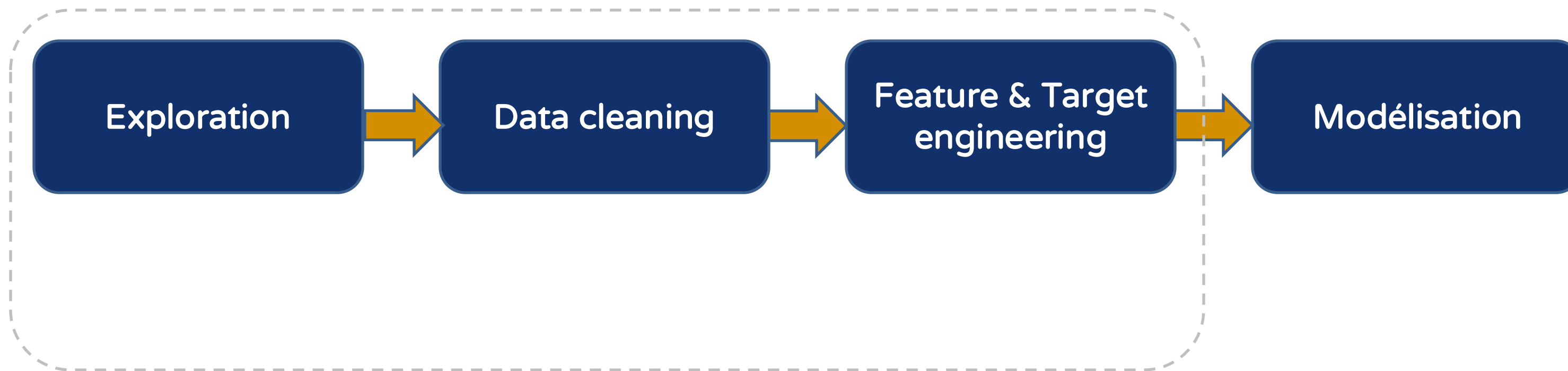
Intervenant

Guillaume Hochard

3.1. Introduction

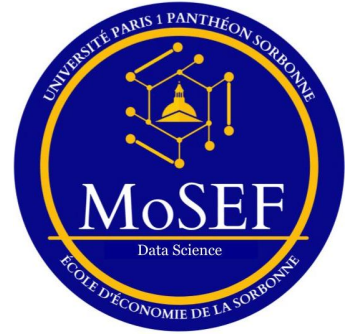


De la donnée à la modélisation : les grandes étapes d'un projet de machine learning pour un cas d'usage de prévision



Objectif de ce chapitre : découvrir et comprendre ces étapes clés dans le cadre d'un projet de prévision

3.2. EDA: Exploratory Data Analysis 1/15

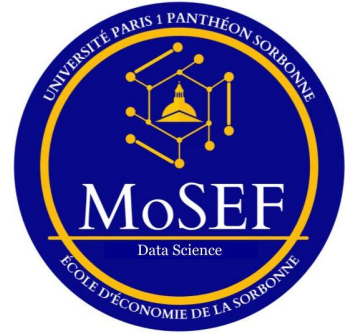


L'analyse exploratoire des données (EDA) est une **étape essentielle et trop souvent négligée** dans un projet de prévision de séries temporelles.

Cette étape permet de mieux apprécier les données, de mieux comprendre le problème que vous devez résoudre.

- **L'EDA doit être orientée** : il ne s'agit pas de tracer des centaines de graphs qui n'ont pas de sens avec le problème posé.
- L'EDA doit être **conduite sous forme d'une série de questions**, que vous vous posez au sujet du jeu de données.
- Tout graph, figure, produit doit pouvoir répondre à une question.

3.2. EDA: Exploratory Data Analysis 2/15



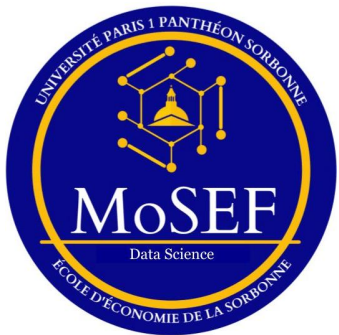
Analyse exploratoire des données (EDA) : illustration sur un jeu de données Walmart

Définition du problème : Prédire les ventes des produits pendant 28 jours.

Données : Données hiérarchiques pour des magasins Walmart pour différentes catégories provenant de trois états, Californie, Wisconsin et Texas. Ventes individuelles pour chaque produit pendant 1914 jours.

- calendar.csv – Données calendaires
- sales_train_validation.csv - Données historiques sur les ventes unitaires quotidiennes par produit et par magasin [d_1 - d_1913].
- sell_prices.csv - Contient des informations sur les prix des produits vendus par magasin et par date.

3.2. EDA: Exploratory Data Analysis 3/15



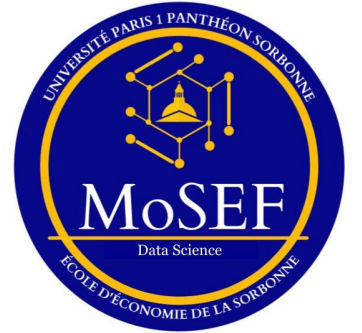
Analyse exploratoire des données (EDA) : illustration sur un jeu de données Walmart

- calendar.csv – Données calendaires

df.head() : Quelles sont les variables du jeu de données ?

	date	wm_yr_wk	weekday	wday	month	year	d	event_name_1	event_type_1	event_name_2	event_type_2	snap_CA	snap_TX	snap_WI
0	2011-01-29	11101	Saturday	1	1	2011	d_1	NaN	NaN	NaN	NaN	0	0	0
1	2011-01-30	11101	Sunday	2	1	2011	d_2	NaN	NaN	NaN	NaN	0	0	0
2	2011-01-31	11101	Monday	3	1	2011	d_3	NaN	NaN	NaN	NaN	0	0	0
3	2011-02-01	11101	Tuesday	4	2	2011	d_4	NaN	NaN	NaN	NaN	1	1	0
4	2011-02-02	11101	Wednesday	5	2	2011	d_5	NaN	NaN	NaN	NaN	1	0	1

3.2. EDA: Exploratory Data Analysis 4/15

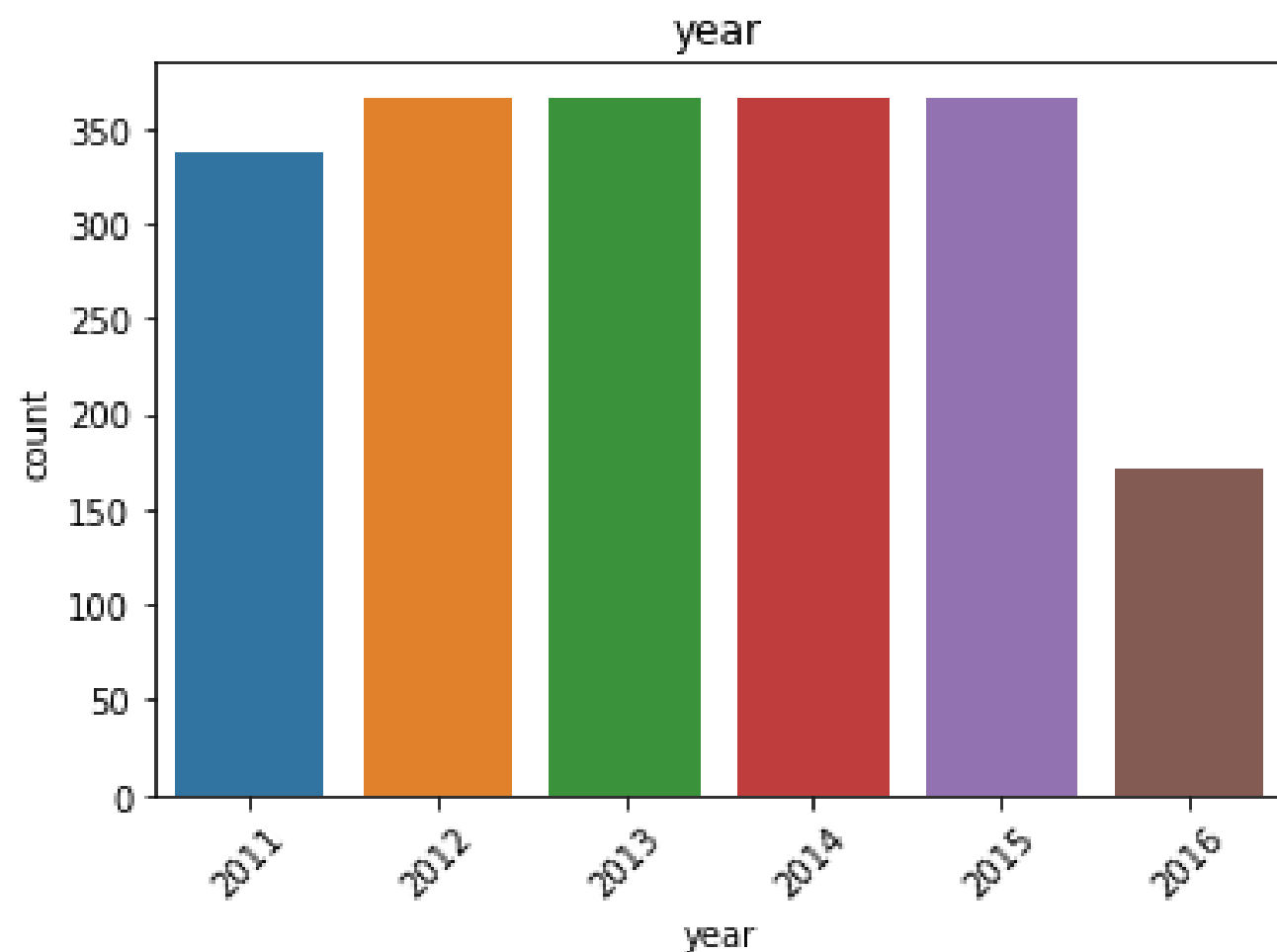


Analyse exploratoire des données (EDA) : illustration sur un jeu de données Walmart

- calendar.csv – Données calendaires

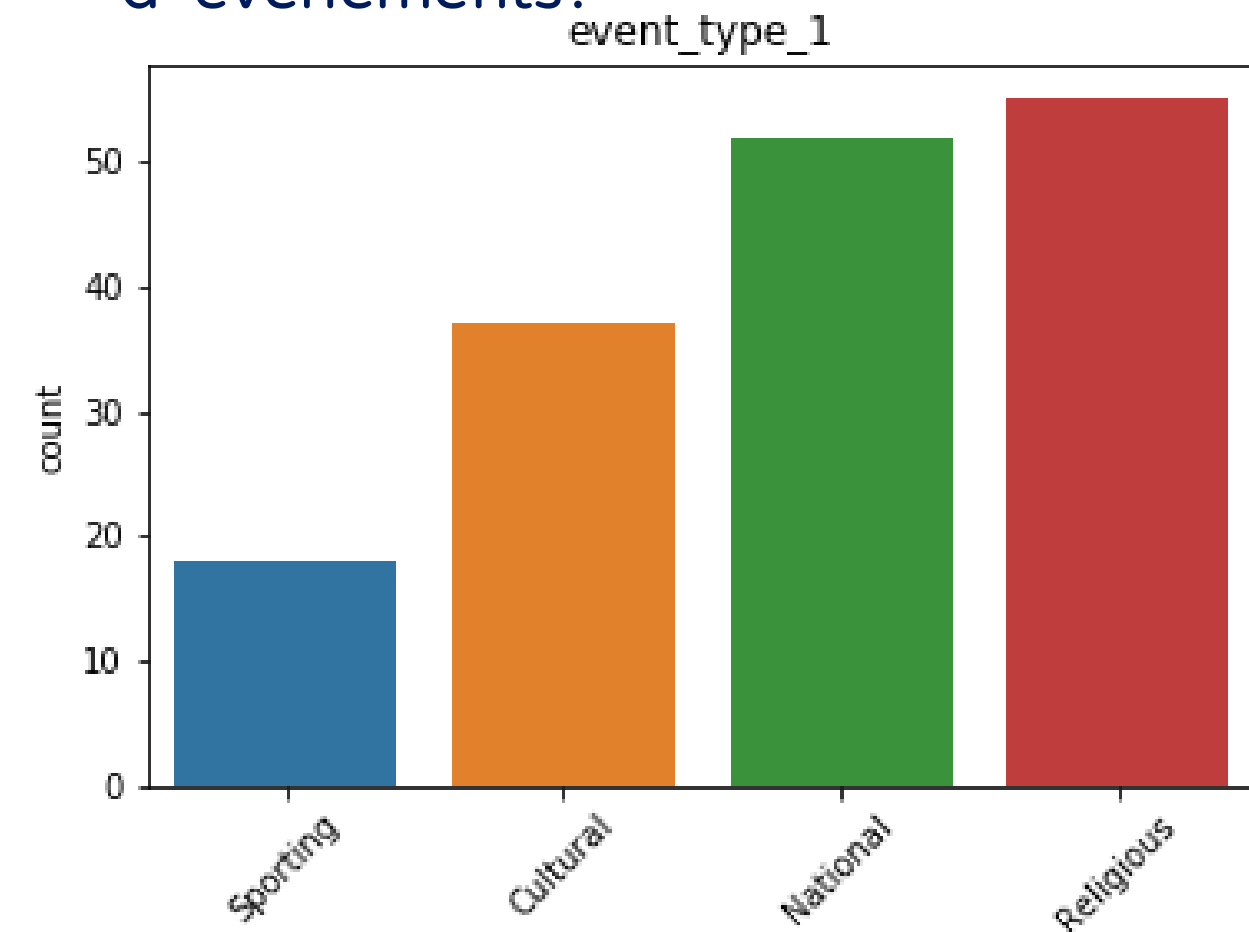
```
sns.countplot(df[year])
```

Possède-t-on des années complètes ?

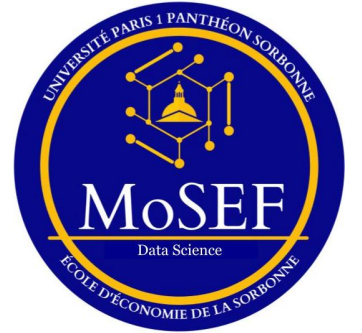


```
sns.countplot(df[event_type_1])
```

Quelle est la fréquence des types d'événements?



3.2. EDA: Exploratory Data Analysis 5/15

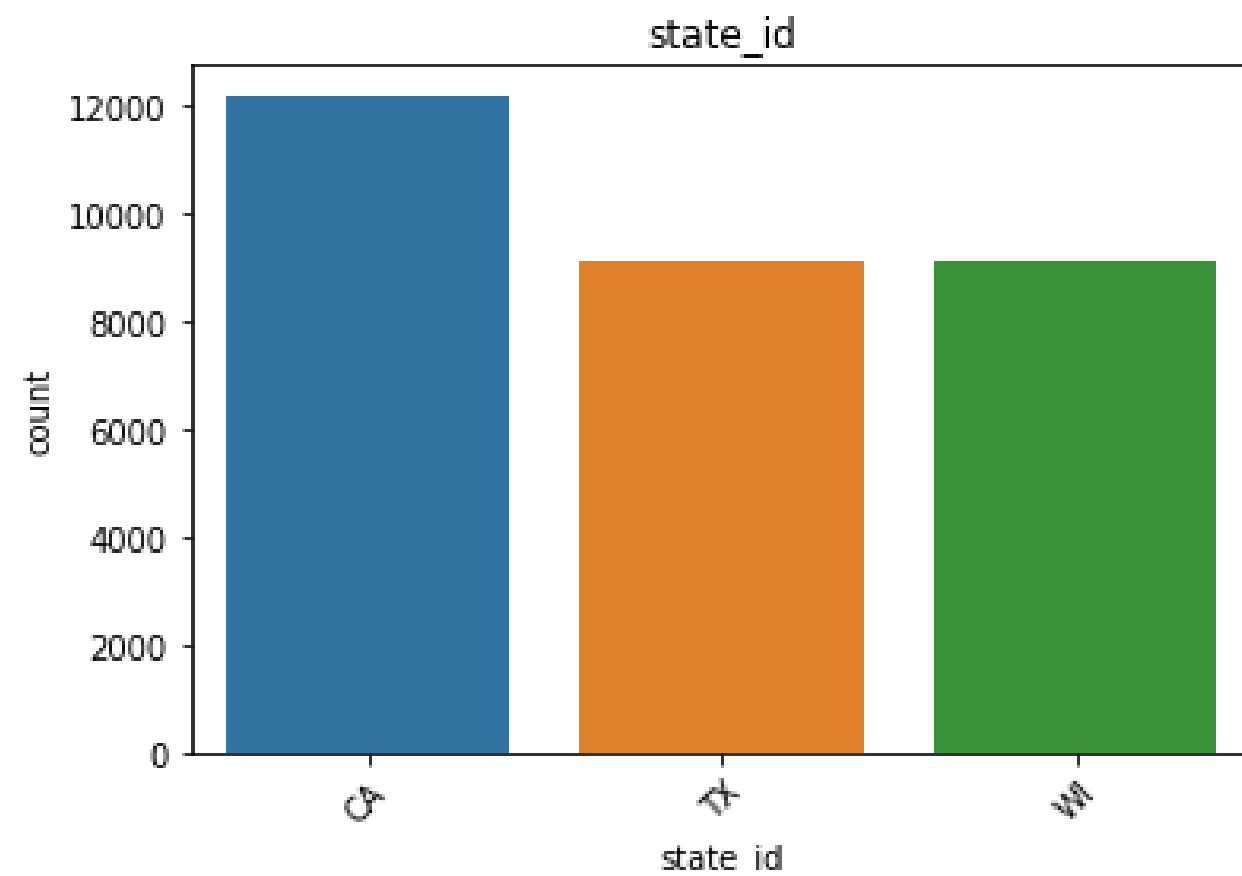


Analyse exploratoire des données (EDA) : illustration sur un jeu de données Walmart

- sales_train_validation.csv – Données de vente

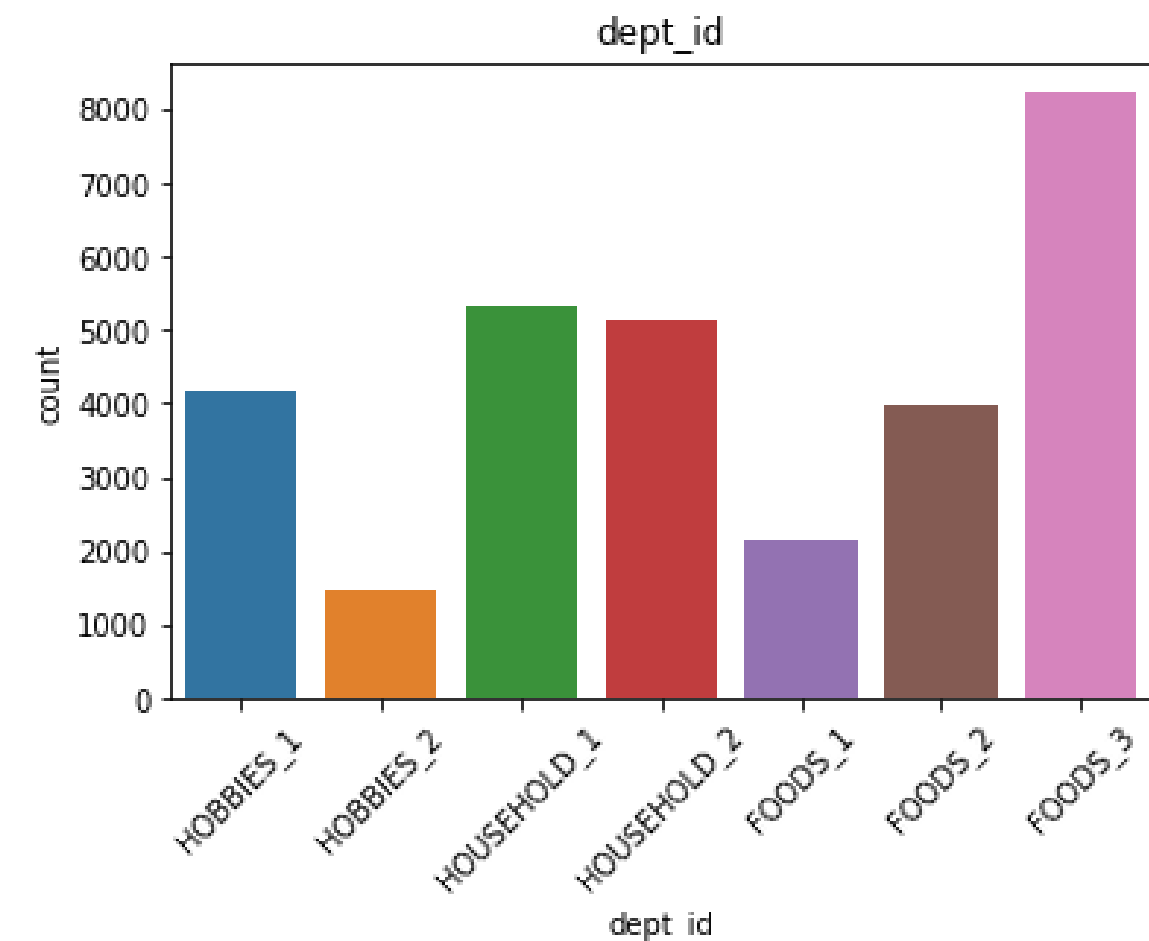
`sns.countplot(df[state_id])`

Répartition des ventes entre états?

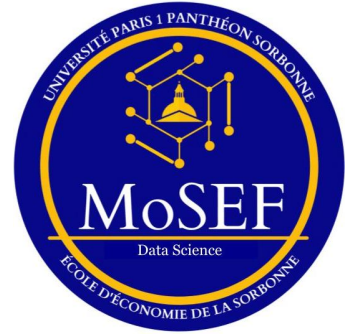


`sns.countplot(df[dept_id])`

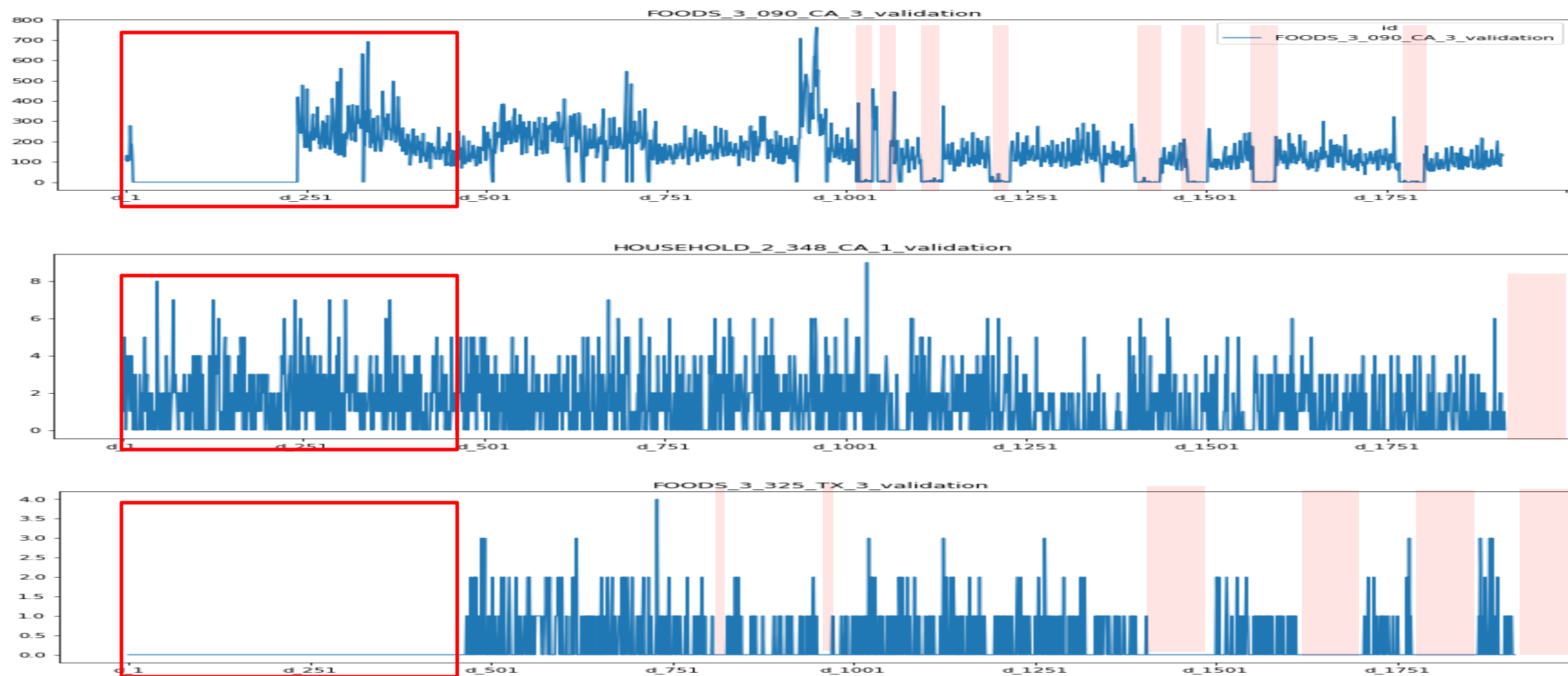
Quelle répartition des ventes entre catégories?



3.2. EDA: Exploratory Data Analysis 6/15

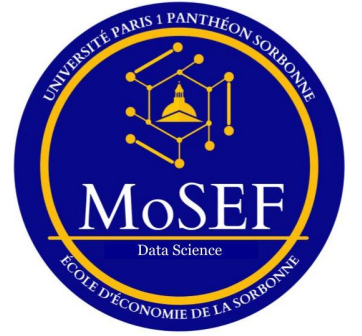


L'objectif est de prédire les prochaines ventes à 28 jours. Y a-t-il des tendances dans les ventes avant ces 28 jours ?

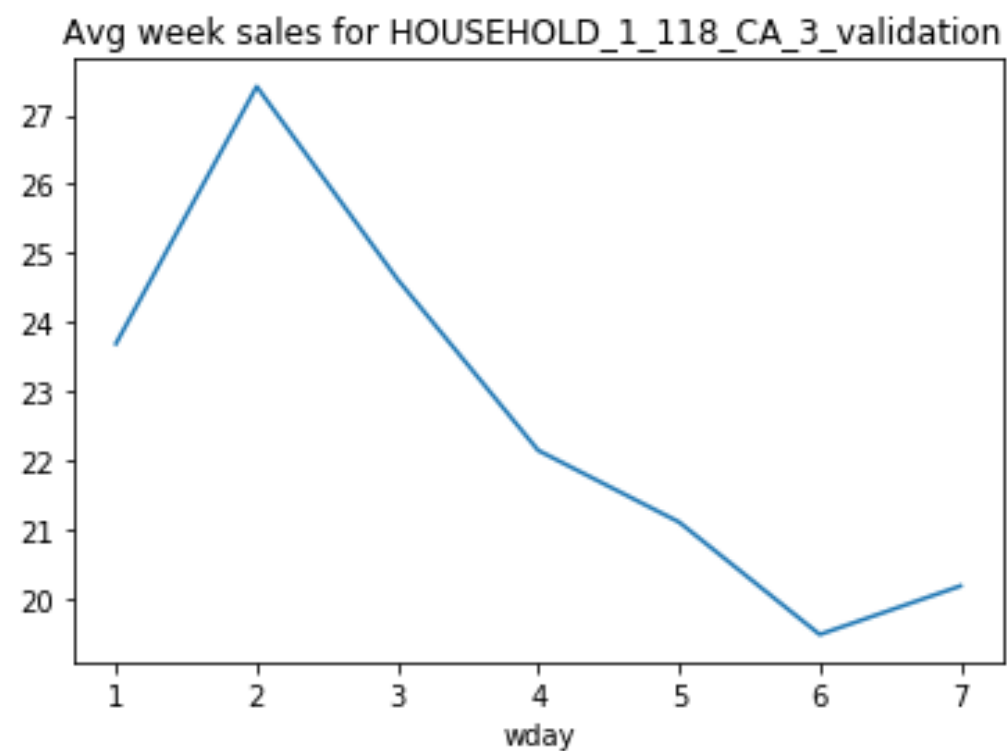


- Pas de pattern évident
- Demande intermittente pour certains produits
- Démarrages à des dates différentes
- Données manquantes : ruptures de stocks?
- Volumes très différents selon les produits

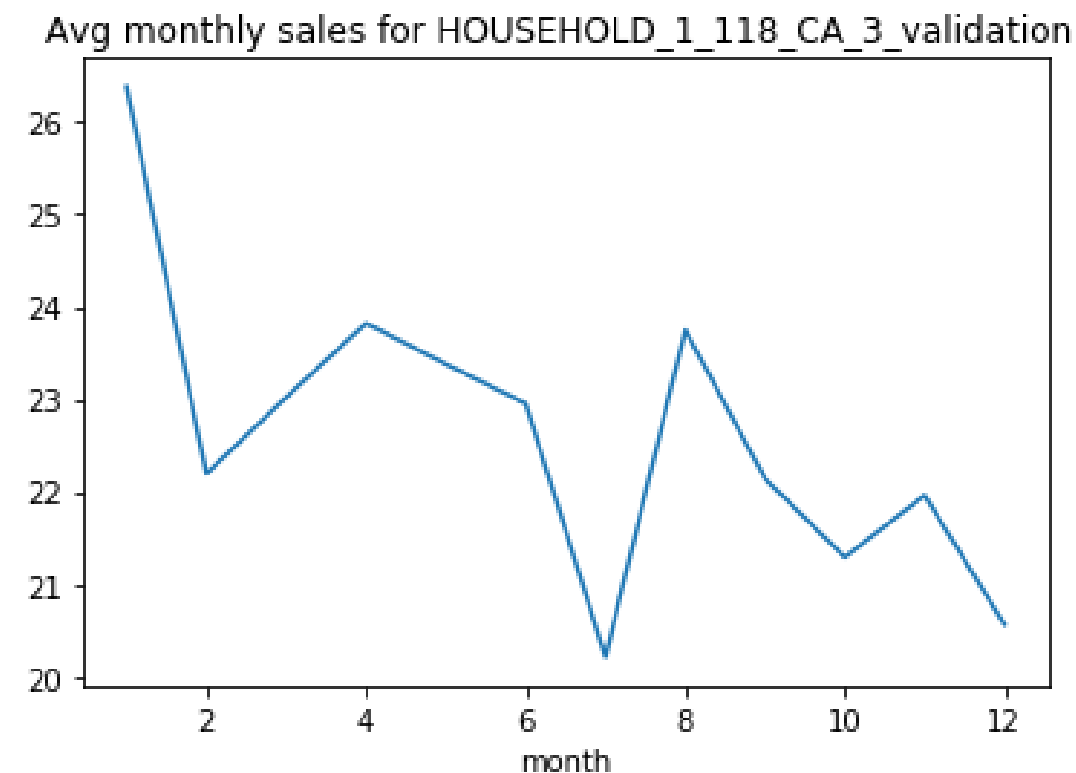
3.2. EDA: Exploratory Data Analysis 7/15



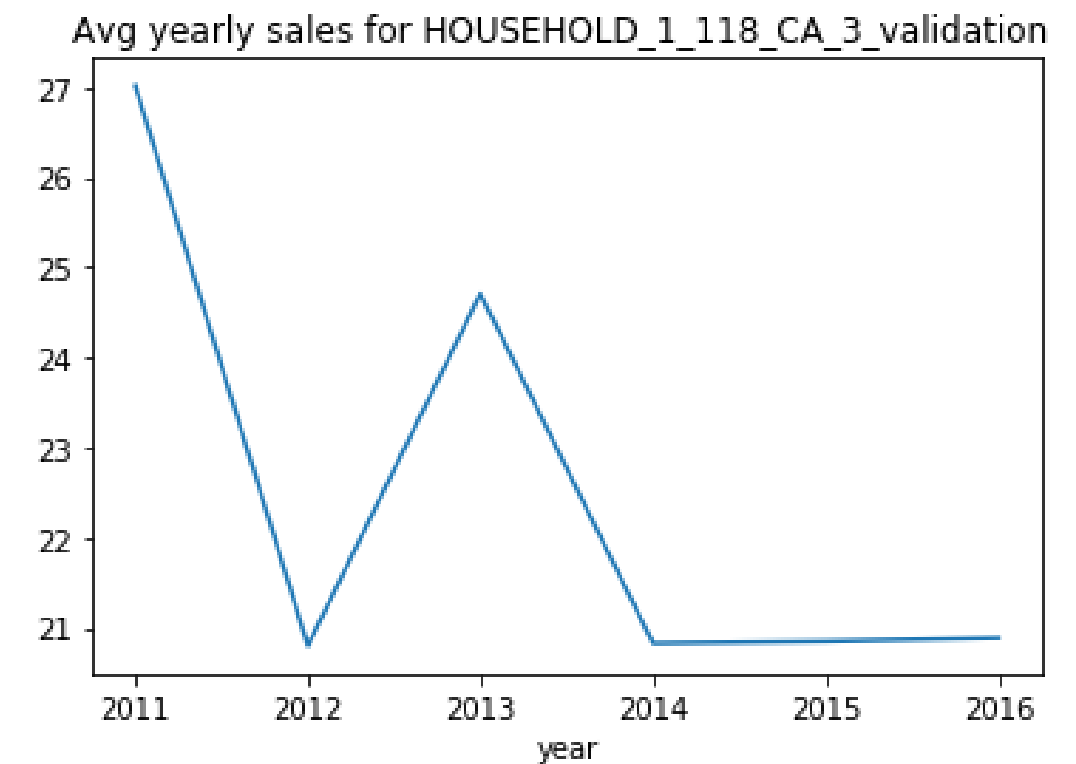
Quel est le profil des ventes sur une base hebdomadaire, mensuelle et annuelle ?



Profil hebdomadaire



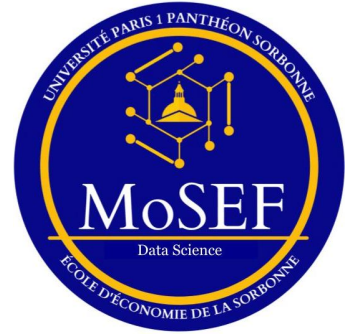
Profil annuel



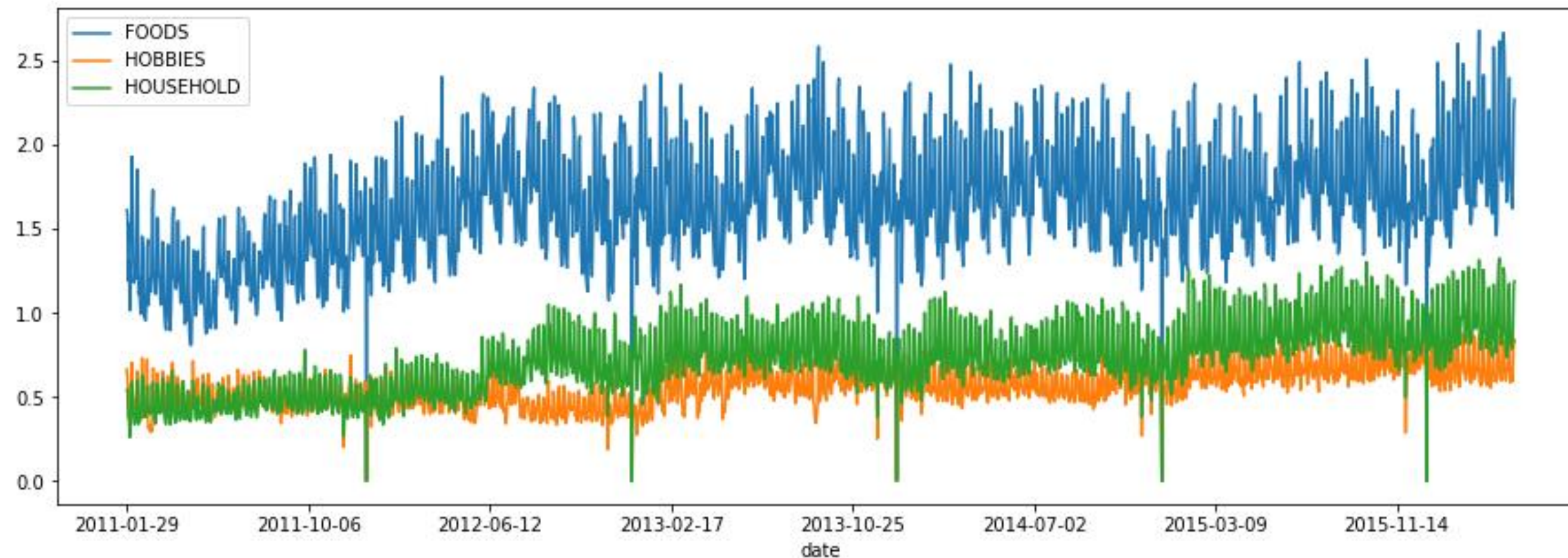
Profil multi-annuel

- Des saisonnalités marquées : illustre l'importance que vont prendre les variables jour, mois et année dans le futur modèle de prévision

3.2. EDA: Exploratory Data Analysis 8/15



Quelles sont les tendances selon les catégories ?

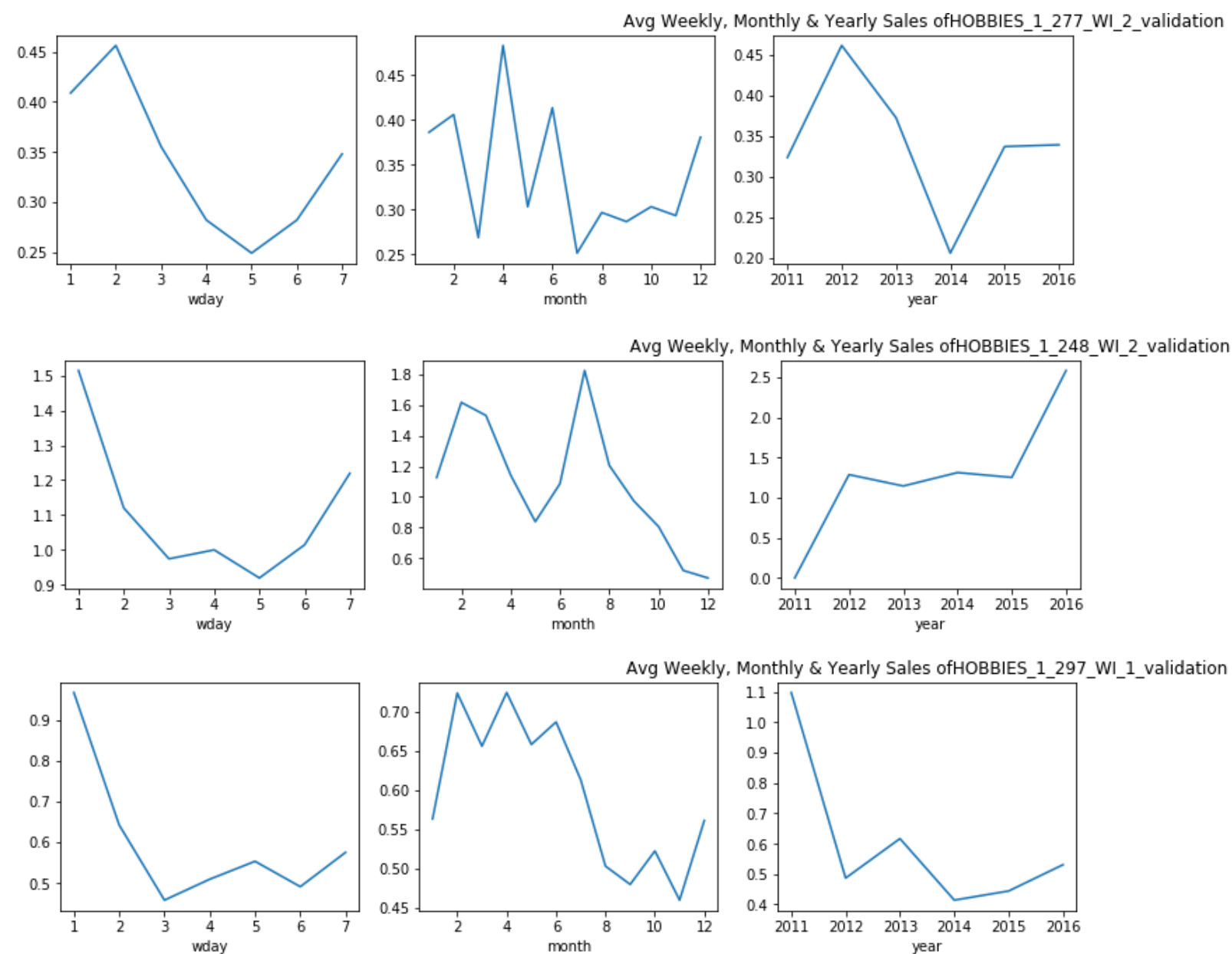


- La catégorie Foods est bien plus importante que les deux autres
- Il existe une tendance long terme pour toutes les catégories

3.2. EDA: Exploratory Data Analysis 9/15



Quelle est la distribution des ventes de produits appartenant à la catégorie des loisirs sur une base hebdomadaire, mensuelle et annuelle pour un état donné ?



Quelques observations pour l'état du Wisconsin

- Les ventes chutent en milieu de semaine
- Les ventes du dernier trimestre sont inférieures au reste de l'année
- Certains produits sont en forte croissance d'année en année, d'autres performant de moins en moins bien

3.2. EDA: Exploratory Data Analysis 10/15

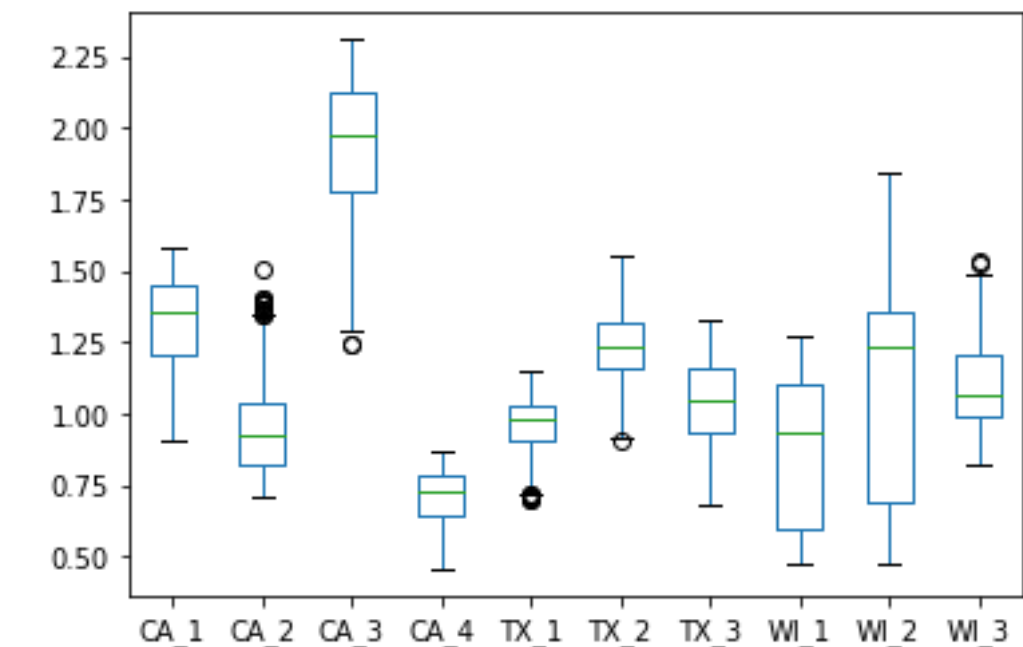
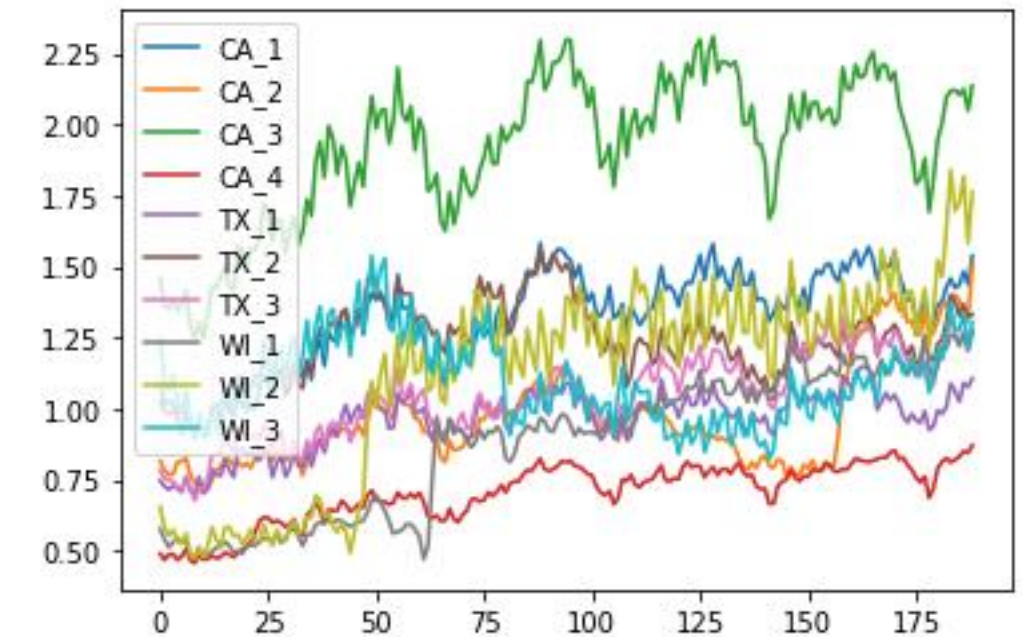


Les ventes varient-elles entre les magasins situés dans différents états ?

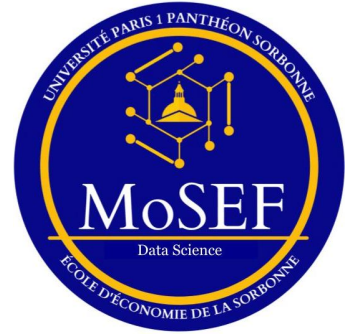
- Line plot
- Box-plot

Observations

- Les ventes du magasin CA_3 sont supérieures aux ventes de tous les autres États. CA_4 : là où les ventes sont les plus faibles
- Tendence croissante similaire, des patterns répétitifs
- Californie : ventes médianes toutes différentes. Variance élevée
- TX_1 et TX_3 ont un comportement assez similaire, TX_2 est au-dessus
- Développement rapide de TX_3 pour approcher TX_2 : croissance plus rapide



3.2. EDA: Exploratory Data Analysis 11/15



Méthodes pour filtrer les données

Existe-t-il un moyen de voir plus clairement les ventes de produits sans perdre d'information ?

- **Moving average denoising**

Technique de lissage simple : une fenêtre fixe est déplacée le long des données de la série chronologique en calculant la moyenne. Intervalle (stride) afin d'alléger les calculs

Exemple : fenêtre de 20 et un pas de 5

1^{er} point = moyenne des points du jour 1 au jour 20

Points suivants = moyenne des points du jour 5 au jour 25, puis du jour 10 au jour 30, etc

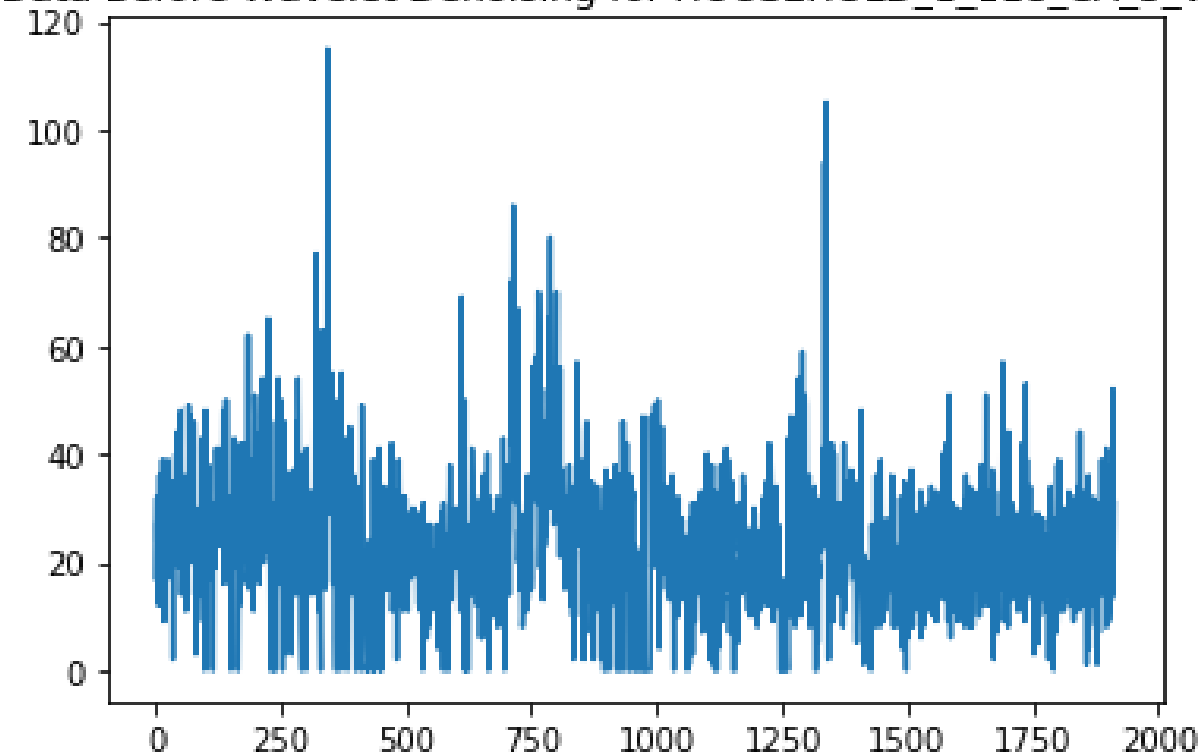
3.2. EDA: Exploratory Data Analysis 12/15



Méthodes pour filtrer les données

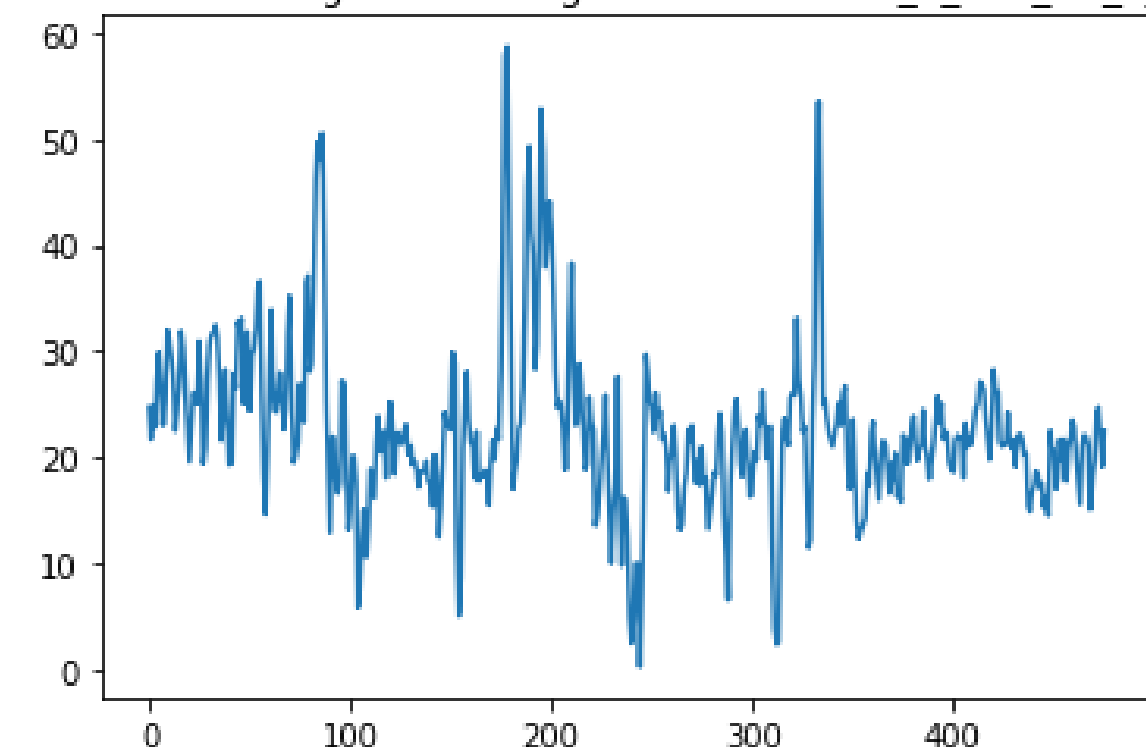
- Moving average denoising

Sales Data Before Wavelet Denoising for HOUSEHOLD_1_118_CA_3_validation



Profil des ventes original

Sales Data After Average Smoothing for HOUSEHOLD_1_118_CA_3_validation



Profil des ventes après filtrage par moyenne mobile

3.2. EDA: Exploratory Data Analysis 13/15



Méthodes pour filtrer les données

- Wavelet denoising

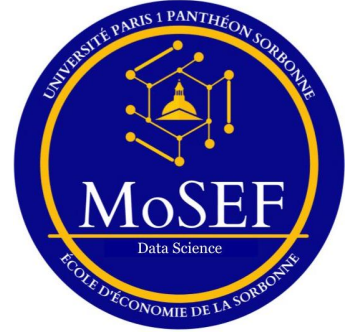
La transformée en ondelettes concentre les caractéristiques du signal dans quelques coefficients d'ondelettes de grande magnitude.

Les coefficients d'ondelettes de faible valeur sont généralement du bruit

Étapes pour réduire le bruit dans les données à l'aide de cette méthode :

- Réduction (shrinkage) de ces coefficients ou suppression sans affecter la qualité du signal
- Seuillage et reconstruction des données à l'aide de la transformée en ondelettes inverse

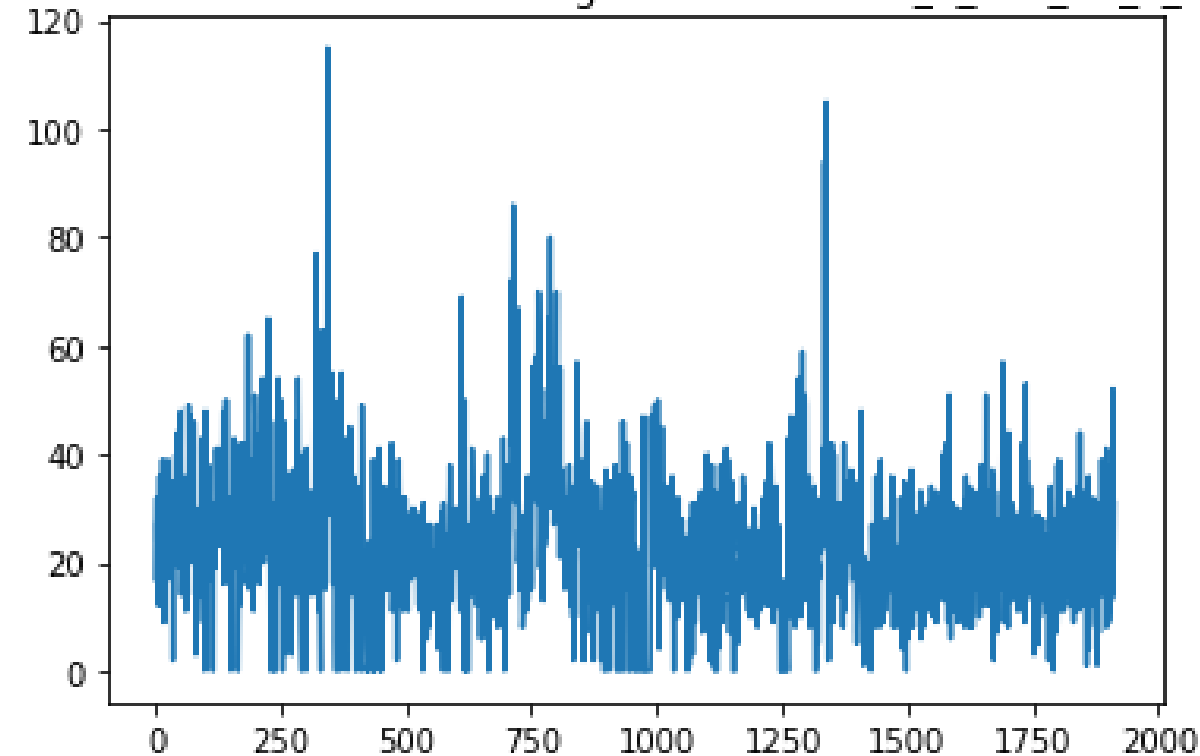
3.2. EDA: Exploratory Data Analysis 14/15



Méthodes pour filtrer les données

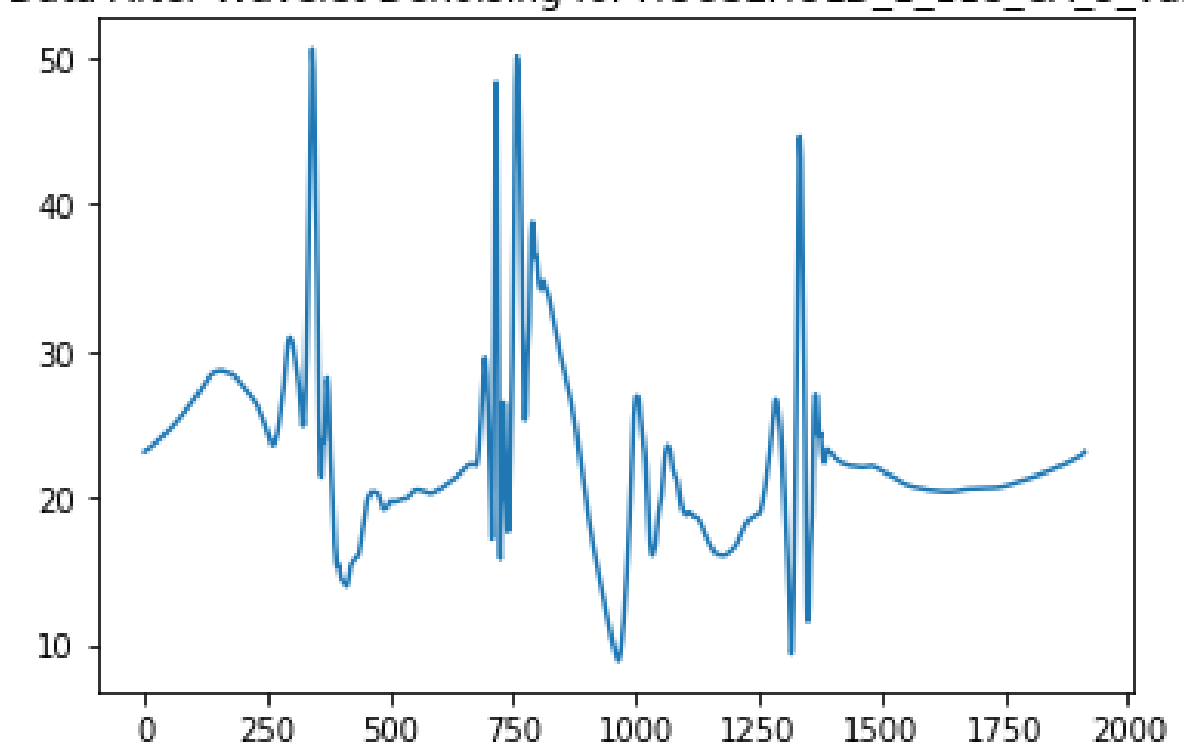
- Wavelet denoising

Sales Data Before Wavelet Denoising for HOUSEHOLD_1_118_CA_3_validation



Profil des ventes original

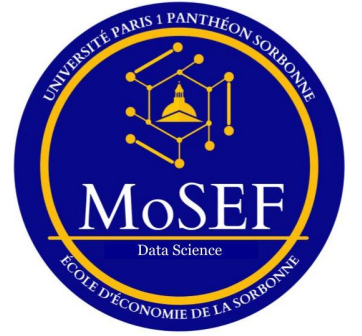
Sales Data After Wavelet Denoising for HOUSEHOLD_1_118_CA_3_validation



Profil des ventes après filtrage par ondelettes

- Interprétation plus facile que la moyenne mobile
- Identification d'un pattern tous les 500 jours

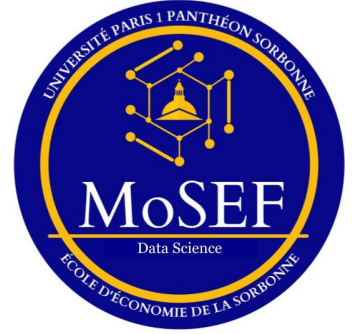
3.2. EDA: Exploratory Data Analysis 15/15



Conclusion sur l'analyse exploratoire des données (EDA) de séries temporelles

- L'EDA se construit à l'aide de **questions que l'on se pose sur le jeu de données**
- Un graph ou une série de graphs apportent la réponse à la question posée
- Si la réponse apportée est partielle, on peut essayer **un autre type de graph ou une coupe (slice) différente** dans le DataFrame pour obtenir une réponse plus précise
- **Précieux insights** sur les variables qui vont avoir de l'importance pour le modèle
- Permet également de commencer à réfléchir à la construction de variables complémentaires = **Feature engineering**

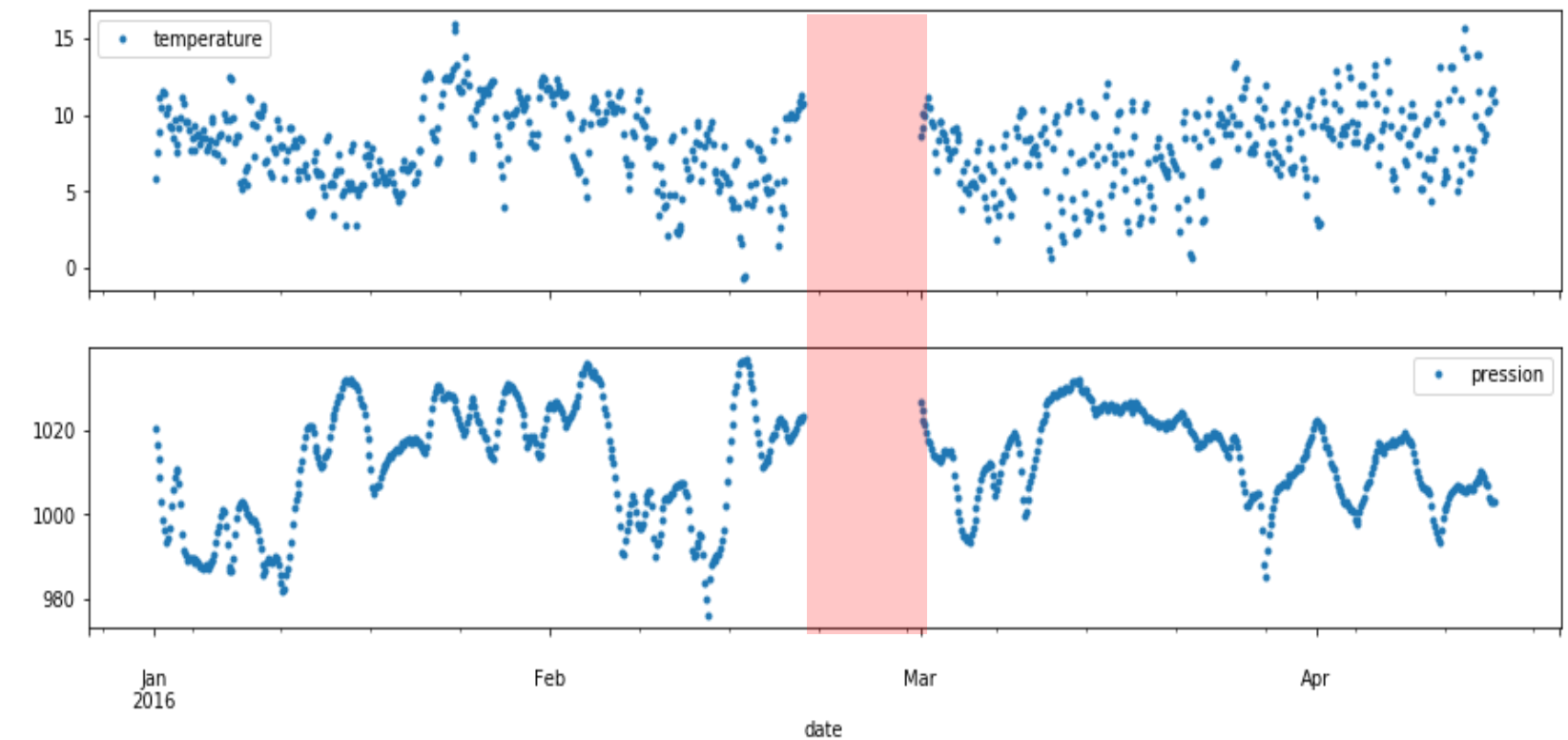
3.3. Data cleaning et data imputation 1/9



L'exploration des données permet d'obtenir des informations précieuses sur les valeurs manquantes dans la série à prévoir et dans ses covariables.

Plusieurs problèmes peuvent alors se poser :

- Echantillonnage incorrect des données
- Valeurs manquantes



Valeurs manquantes pour temperature et pression

3.3. Data cleaning et data imputation 2/9



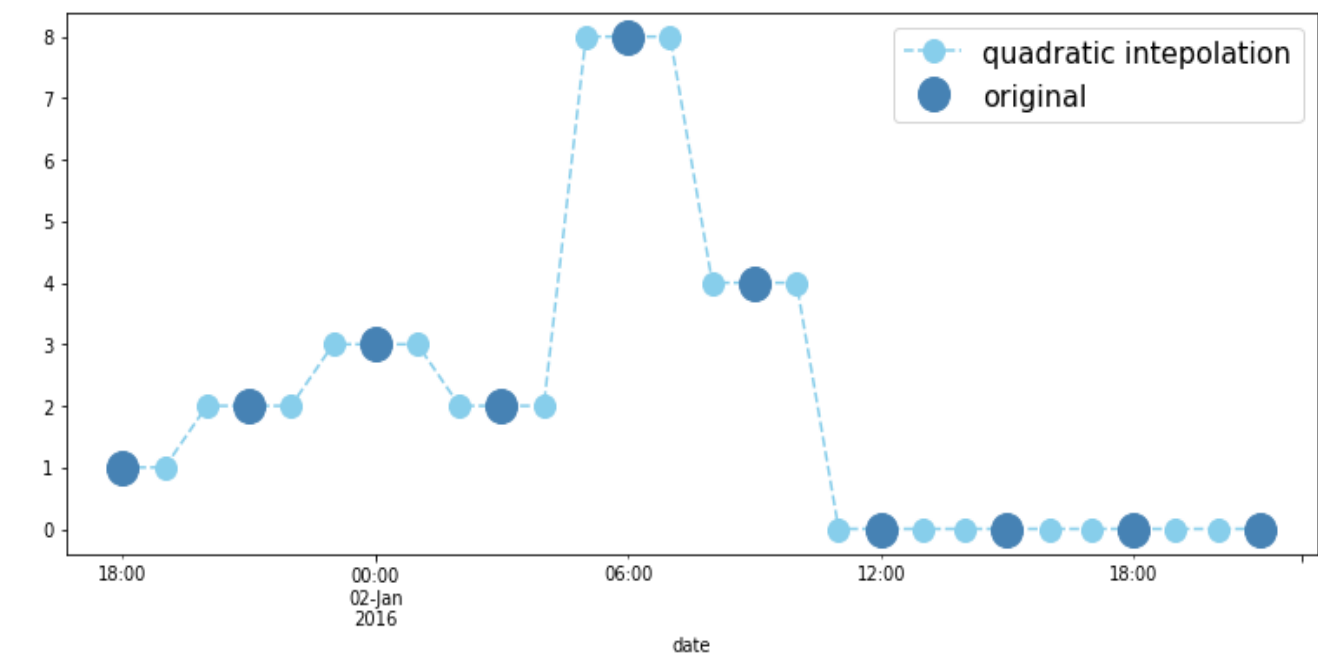
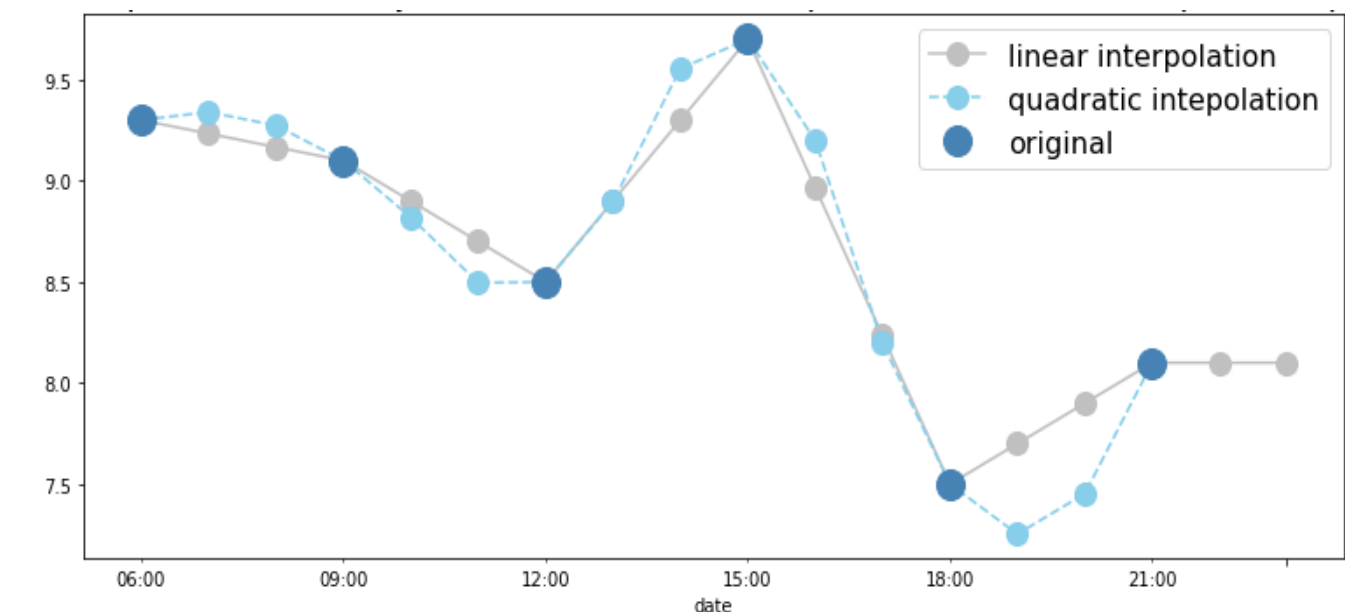
Réalignement temporel de sources de données

Lorsqu'une source de données est **échantillonnée à une fréquence différente** de la source principale, la réalignement temporel s'impose (ex : température mesurée toutes les 3h, consommation d'énergie horaire)

- Variable continue : interpolation de Lagrange

$$L(X) = \sum_{j=0}^n y_j \left(\prod_{i=0, i \neq j}^n \frac{X - x_i}{x_j - x_i} \right)$$

- Variable catégorielle : utilisation d'un algorithme KNN



3.3. Data cleaning et data imputation 3/9



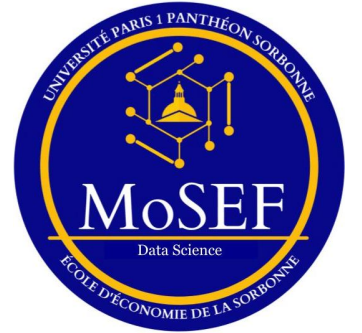
Traitement des valeurs manquantes

Il existe de nombreuses techniques d'imputation de valeurs manquantes

Les techniques classiques en machine learning ne sont pas tout à fait adaptée, du fait du caractère temporel des données

- L'imputation par la médiane ne prend pas en compte le contexte temporel (autocorrélation des données)
- Les médianes doivent être calculées sur des ensemble de données antérieurs au point considéré.
- On peut utiliser des fenêtres glissantes calculées sur les pas de temps précédant les valeurs manquantes

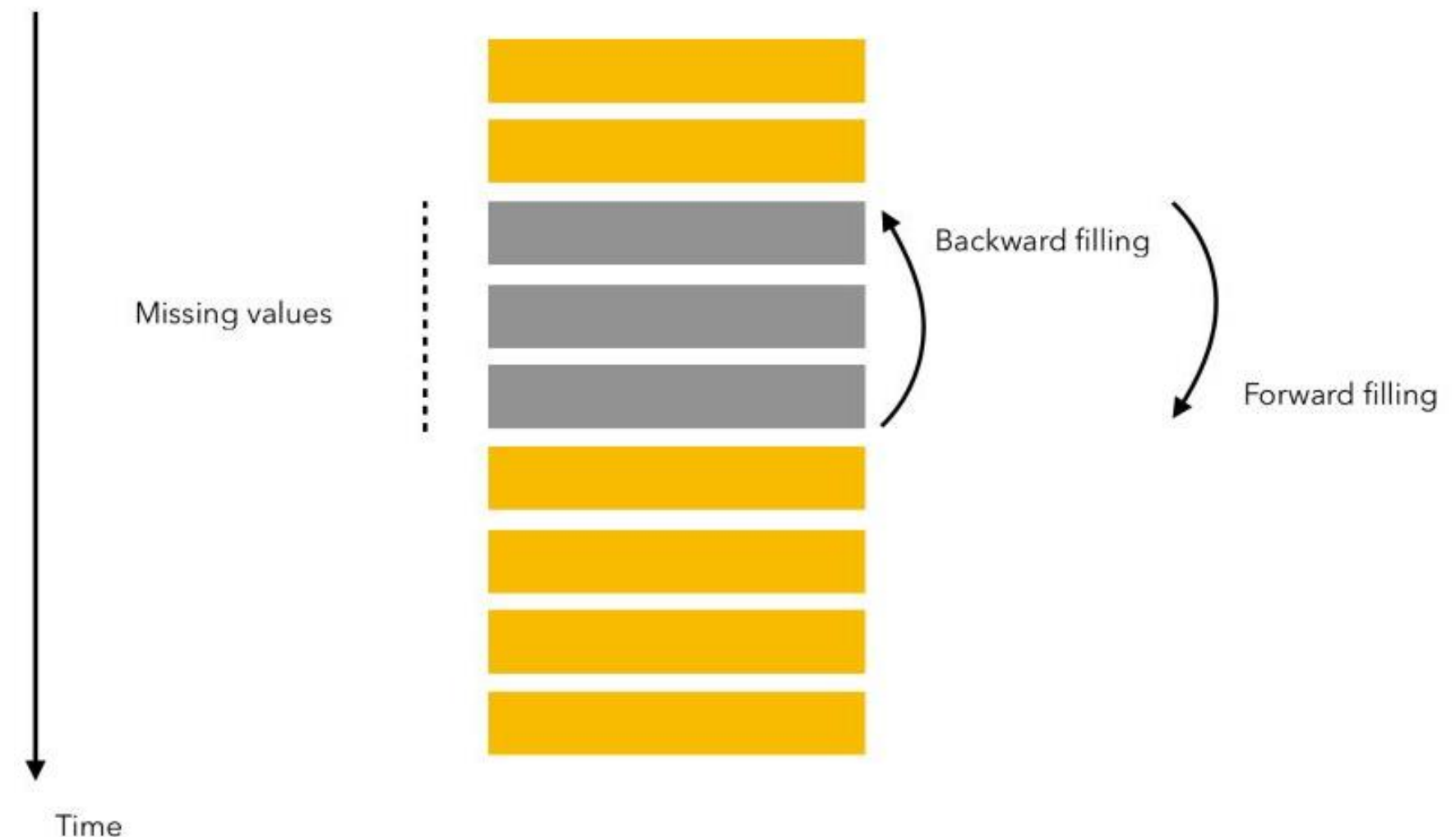
3.3. Data cleaning et data imputation 4/9



Traitement des valeurs manquantes

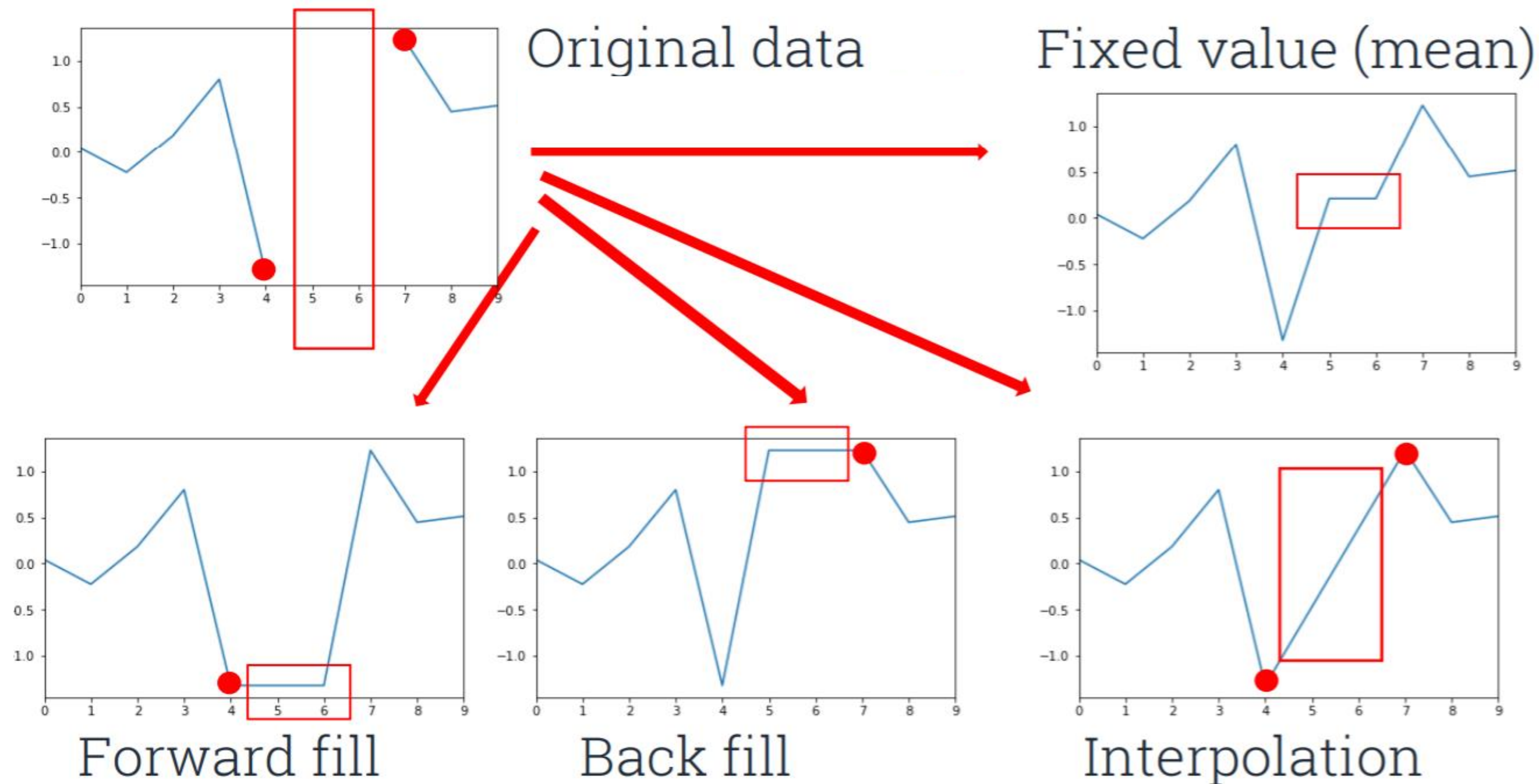
Parmi les techniques adaptées, on peut citer :

- **Forward fill / Backward fill**
- Forward fill : on remplace les valeurs manquantes **dans le futur** en se basant sur la **dernière valeur connue**
- Backward fill : on remplace les valeurs manquantes **dans le passé** en se basant sur la **première valeur connue après les manquantes**
- Hypothèses fortes sur les données : la série **ne varie pas** sur la période des données manquantes



3.3. Data cleaning et data imputation 5/9

Illustration sur un cas simple



3.3. Data cleaning et data imputation 6/9



Traitement des valeurs manquantes

Parmi les techniques adaptées, on peut citer :

- MissForest
- Basé sur Random Forests
- Algorithme itératif
- Utilisation d'une variable pour en prédire une autre
- L'algorithme va "interpoler" les données manquantes, il ne vise pas à faire des prévisions

Algorithm 1 Impute missing values with random forest.

Require: \mathbf{X} an $n \times p$ matrix, stopping criterion γ

```
1: Make initial guess for missing values;  
2:  $\mathbf{k} \leftarrow$  vector of sorted indices of columns in  $\mathbf{X}$   
   w.r.t. increasing amount of missing values;  
3: while not  $\gamma$  do  
4:    $\mathbf{X}_{old}^{imp} \leftarrow$  store previously imputed matrix;  
5:   for  $s$  in  $\mathbf{k}$  do  
6:     Fit a random forest:  $\mathbf{y}_{obs}^{(s)} \sim \mathbf{x}_{obs}^{(s)}$ ;  
7:     Predict  $\mathbf{y}_{mis}^{(s)}$  using  $\mathbf{x}_{mis}^{(s)}$ ;  
8:      $\mathbf{X}_{new}^{imp} \leftarrow$  update imputed matrix, using predicted  $\mathbf{y}_{mis}^{(s)}$ ;  
9:   end for  
10:  update  $\gamma$ .  
11: end while  
12: return the imputed matrix  $\mathbf{X}^{imp}$ 
```

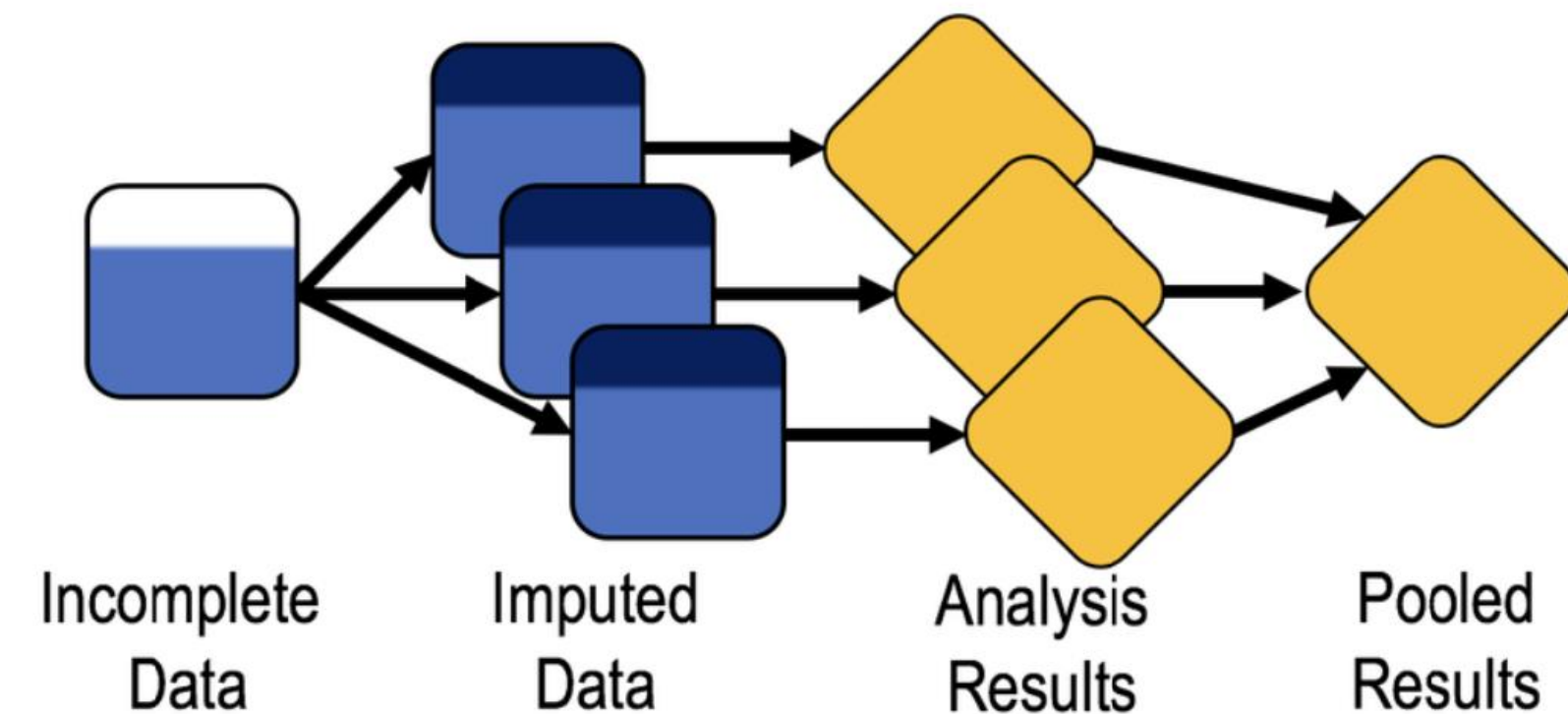
3.3. Data cleaning et data imputation 7/9



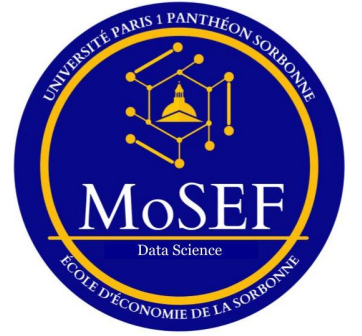
Traitement des valeurs manquantes

Parmi les techniques adaptées, on peut citer :

- Multiple Imputation by Chained Equations (MICE)
- Approche permettant de prendre en compte la **variance causée par l'imputation** (prend en compte la variabilité de chaque imputation, mais aussi entre les imputations)
- Des **composantes aléatoires** sont incorporées à ces valeurs estimées pour montrer leur incertitude
- **Même analyse individuelle** de plusieurs ensembles de données afin d'obtenir un ensemble d'estimations de paramètres
- **Combinaison des estimations** pour obtenir un ensemble d'estimations de paramètres



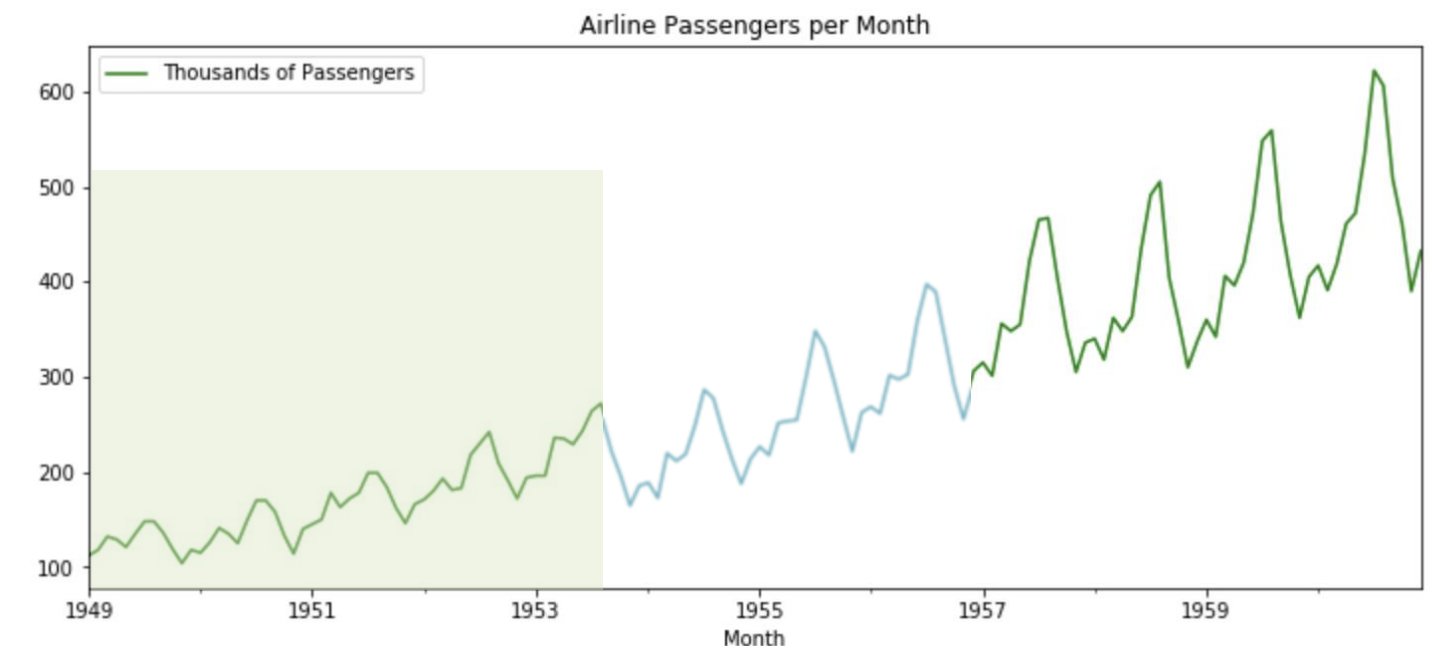
3.3. Data cleaning et data imputation 8/9



Traitement des valeurs manquantes

Parmi les techniques adaptées, on peut citer :

- **ARIMA**
- Utilisation d'un **modèle statistique pour prévoir** les données manquantes
- **Entraînement du modèle** jusqu'aux données manquantes
- **Prévision sur les horizons** qui contiennent des données manquantes
- Méthode simple; utilisée en production chez un grand hébergeur de données



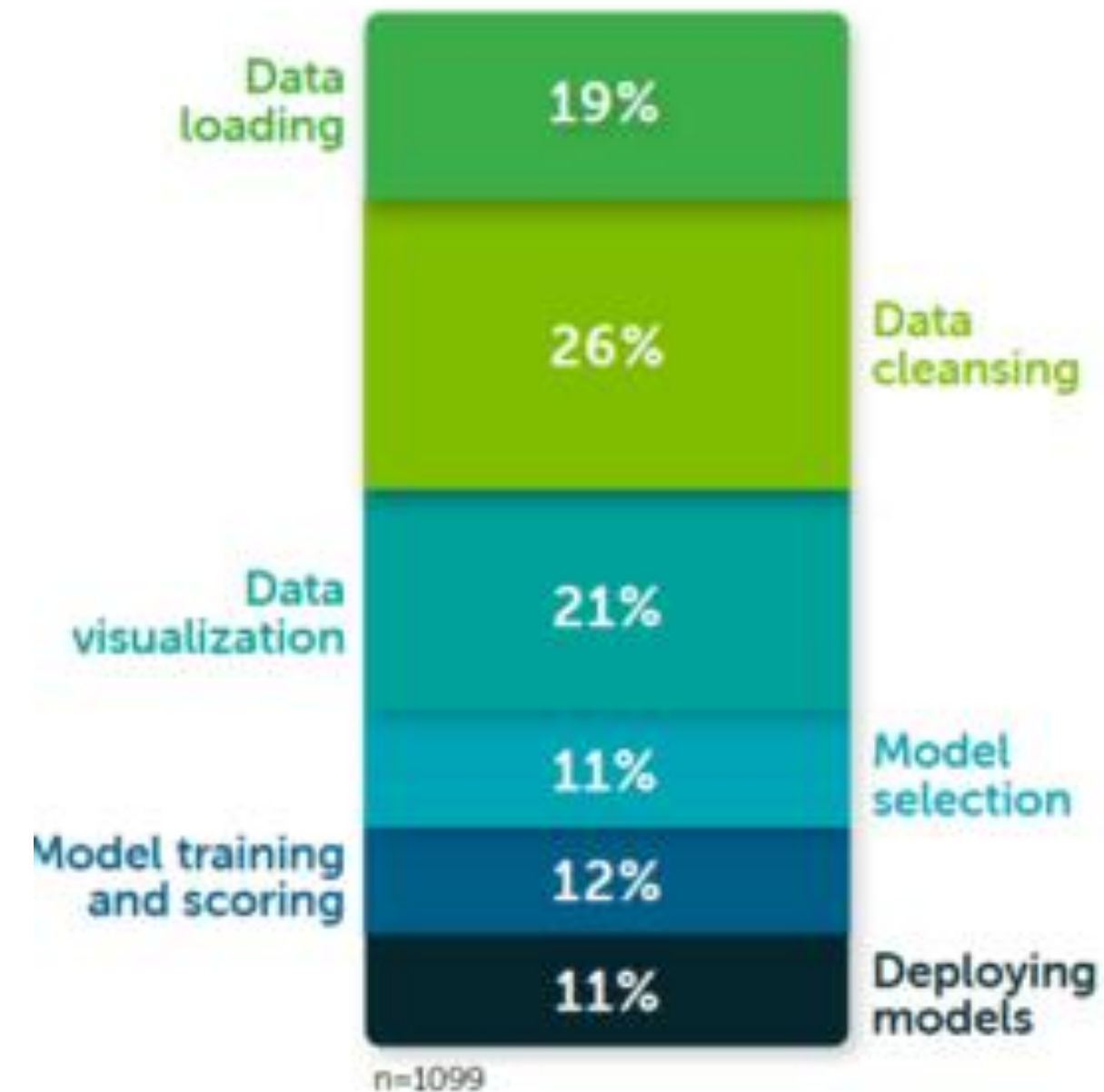
3.3. Data cleaning et data imputation 9/9



Traitement des valeurs manquantes

Parmi les techniques adaptées, on peut citer :

- Forward/backward & mean/median fill
- MissForest : basé sur Random Forests
- Multiple Imputation by Chained Equations (MICE)
- ARIMA : utilisation d'un modèle statistique pour prévoir les données manquantes



How data scientists spend their time (Image Anaconda [“2020 State of Data Science: Moving From Hype Toward Maturity.”](#))

3.4. Feature engineering 1/11



La phase d'exploration des données (EDA) apporte des éclairages précieux sur les **variables utiles et qui ont du sens** d'un point de vue métier à intégrer dans un algorithme d'apprentissage supervisé de régression

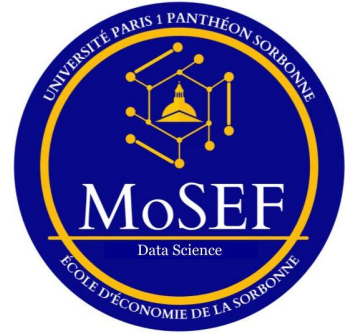
Les questions posées lors de la phase d'exploration peuvent aider à donner des idées de variables à créer

Il y a toujours un **compromis** entre ajout de variables et surapprentissage du modèle.

Ajouter des variables qui ne portent pas d'information aura tendance à dégrader le modèle (GIGO = Garbage In, Garbage Out)

Discuter avec des experts métier n'est jamais une perte de temps, il peuvent vous donner « LA » variable qui va vous faire gagner des points de RMSE

3.4. Feature engineering 2/11



Quelques considérations importantes

- **Stationnarité** : de nombreuses caractéristiques (features) de séries temporelles supposent la stationnarité et sont inutiles à moins que les données sous-jacentes soient stationnaires ou au moins ergodiques.
- Par exemple : utilité de la moyenne si la série n'est pas stationnaire, si présence de cycle ou de saisonnalités ?
- **Longueur de la série** : certaines variables sont sensibles à la longueur de la série
- Par exemple : min et max peuvent dériver dans le temps plus la série est longue
- **Temps de calcul** : certaines variables sont très coûteuses à obtenir, et allongent la durée totale d'exécution d'un pipeline de machine learning

3.4. Feature engineering 3/11



Features retardées (lags)

Il peut être utile d'incorporer comme variable des valeurs retardées de la cible ou d'autres covariables .

- Utile pour capter des composantes saisonnières (journalières, hebdo, mensuelles, ...)
- Capture le caractère autorégressif de la série (une analyse de la PACF peut aider à trouver les bons lags à intégrer)
- Attention : toujours vérifier si vous pourrez utiliser ce lag (si je prédis à un horizon de 13 semaines, le lag à 7 jours n'est pas disponible !)

Syntaxe en python pour le lag 1

```
df[y_lag1] = y.shift(1)
```

Time	Load	lag (t-1)	lag (t-2)	lag (t-3)	lag (t-4)
12:00	320.4	NaN	NaN	NaN	NaN
13:00	325.67	320.4	NaN	NaN	NaN
14:00	326.8	325.67	320.4	NaN	NaN
15:00	335.3	326.8	325.67	320.4	NaN
16:00	385.94	335.3	326.8	325.67	320.4
17:00	346.1	385.94	335.3	326.8	325.67
18:00	332.55	346.1	385.94	335.3	326.8

Exemple de lags sur la variable Load

3.4. Feature engineering 4/11



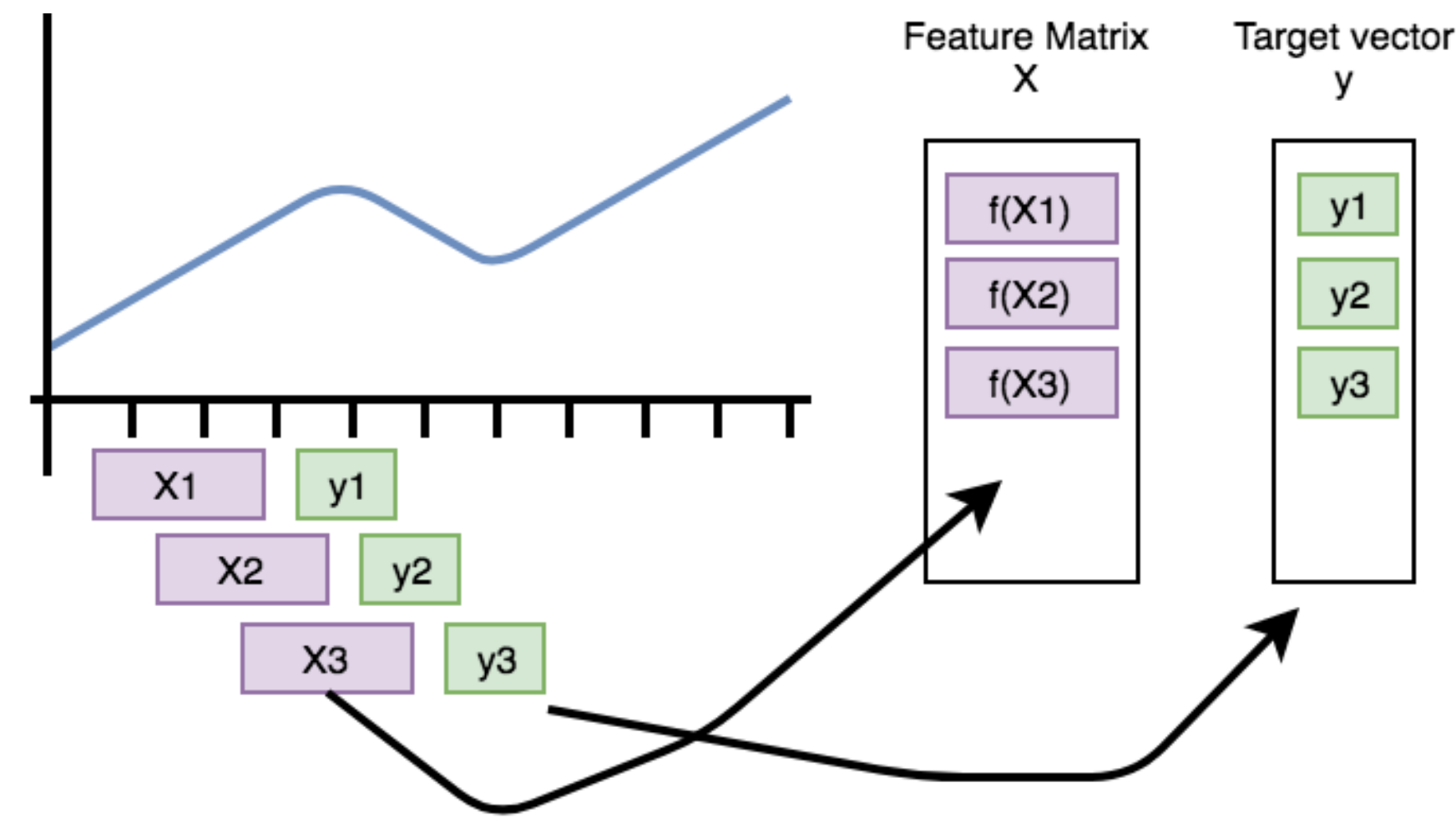
Features sur fenêtres glissantes (variables numériques)

Les variables suivantes sont calculées sur des fenêtres glissantes de taille adaptée :

- Moyenne / variance
- Maximum / minimum
- Différence entre la dernière et la première valeur
- Nombre de minima/maxima locaux
- Autocorrélation
- Mesure de non-linéarité (c3), complexité (cid_ce)
- Coefficients principaux de transformée de Fourier ou ondelettes
- ...

Syntaxe en python

```
df[var_mean2]= df[var].rolling(window=2).mean()
```



3.4. Feature engineering 5/11



Décomposition du Timestamp

A partir de l'horodatage de la série (au format **datetime**), on peut extraire beaucoup d'informations :

- Jour (numéro du jour, nom du jour, position dans la semaine)
- Mois
- Année
- Heure, minute, seconde ...

On peut aussi créer la variable booléenne *is_weekend*.

Syntaxe en python

```
data['Datetime'] =  
pd.to_datetime(data['Datetime'],format='%d-%m-%Y %H:%M')
```

```
data['day']=data['Datetime'].dt.day  
data['dayofweek_num']=data['Datetime'].dt.dayofweek  
data['dayofweek_name']=data['Datetime'].dt.weekday_name
```

```
data['month']=data['Datetime'].dt.month
```

```
data['year']=data['Datetime'].dt.year
```

```
data['Hour'] = data['Datetime'].dt.hour  
data['minute'] = data['Datetime'].dt.minute
```

```
data['is_weekend'] = data['dayofweek_num'].apply(lamda x :  
True if x>4 else False]
```

3.4. Feature engineering 6/11

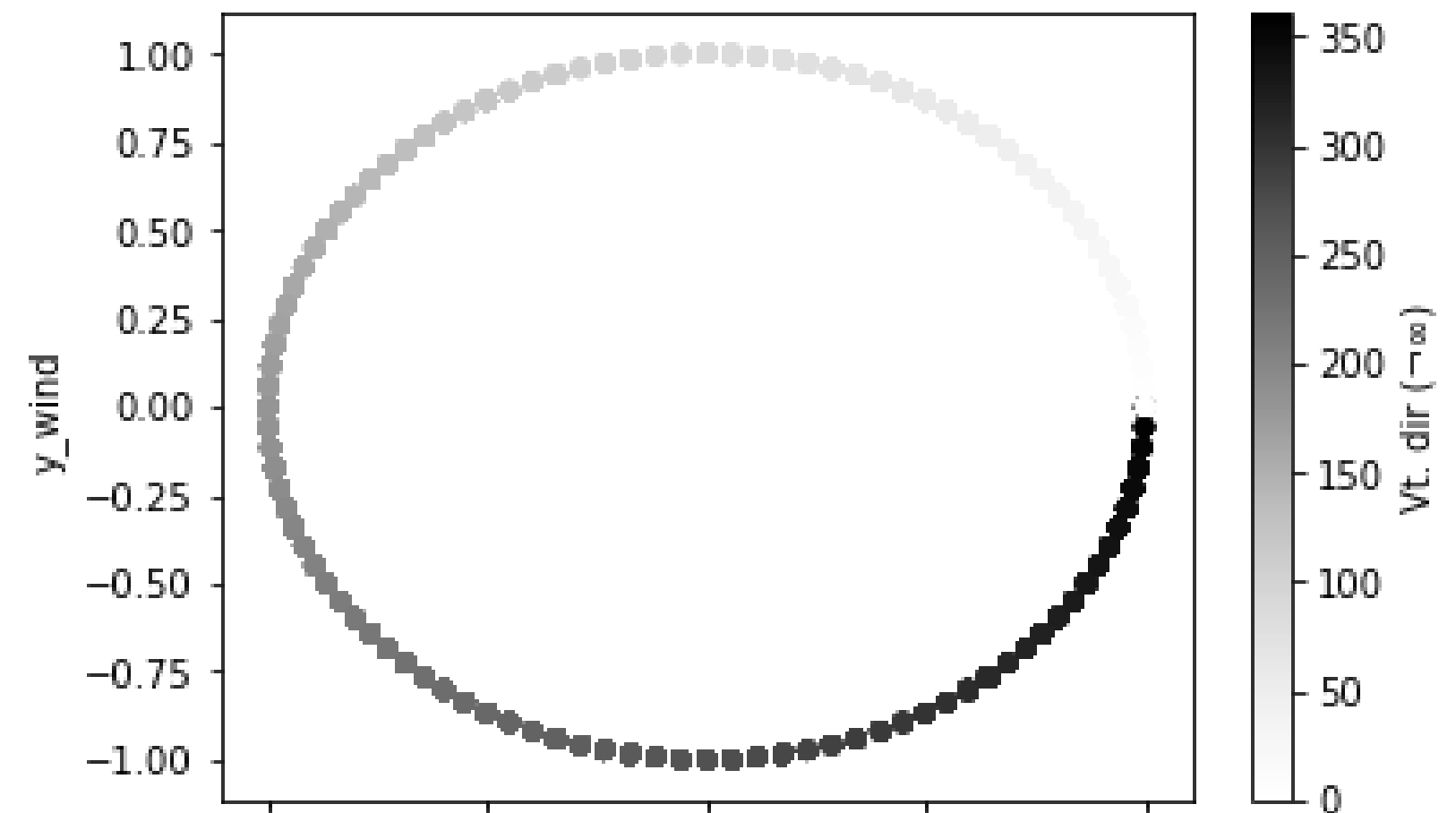


Encoding

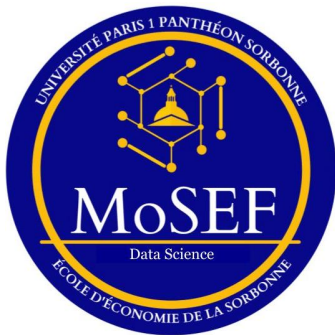
Pour des **features cycliques** (heure de la journée, mois de l'année), on peut créer un **encodage cyclique** à l'aide d'une **transformation cosinus/sinus** qui vient encoder la variable sur un cercle (comme le cadran d'une horloge)

- En effet, 23h est proche de 1h dans la réalité, mais très éloignés dans l'espace des features « heure » allant de 0 à 23.
- Autre exemple : direction du vent, 359° est proche de 1°

Cosine/sine encoding de la direction du vent



3.4. Feature engineering 7/11



Encoding

Pour des variables catégorielles à haute cardinalité, on peut procéder à un **target encoding** plutôt qu'à un One Hot encoding qui va nous emmener du côté du fléau de la dimensionnalité

But : Remplacer une variable catégorielle par la valeur cible moyenne de tous les points de données appartenant à la catégorie

Attention au data leakage ! Deux solutions :

- **Leave-One-Out Target Encoding** : implique de prendre la valeur cible moyenne de tous les points de données de la catégorie, à l'exception de la ligne actuelle.
- **Leave One fold out Target Encoding** : extension de Leave One Out où, au lieu de prendre la valeur cible moyenne de tous les points de données appartenant à la catégorie actuelle, à l'exception de la ligne actuelle, nous laissons de côté le fold de cross_validation auquel appartient la donnée.

One Hot encoding

	Salt Lake City	San Francisco	Seattle	Years OF Exp	Yearly Salary in Thousands
0	1	0	0	10	120
1	0	0	1	5	120
2	0	1	0	5	140
3	0	0	1	3	100
4	0	0	1	1	70
5	0	1	0	2	100
6	1	0	0	1	60
7	0	1	0	2	110
8	0	0	1	4	100
9	1	0	0	2	70

Target encoding

	City	Years OF Exp	Yearly Salary in Thousands
0	85.200846	10	120
1	97.571139	5	120
2	114.560748	5	140
3	97.571139	3	100
4	97.571139	1	70
5	114.560748	2	100
6	85.200846	1	60
7	114.560748	2	110
8	97.571139	4	100
9	85.200846	2	70

3.4. Feature engineering 8/11



Données calendaires

Pour les séries temporelles, les **données calendaires** peuvent être particulièrement importantes selon le contexte.

- Week end
- Vacances scolaires
- Jour férié, pont
- Nombre de jours avant/après un évènement (avant les vacances, après un jour férié) la donnée.
- Fêtes religieuses
- Fêtes commerciales (Fêtes des pères, Saint Valentin, ...)
- ...



3.4. Feature engineering 9/11



Données exogènes et open data

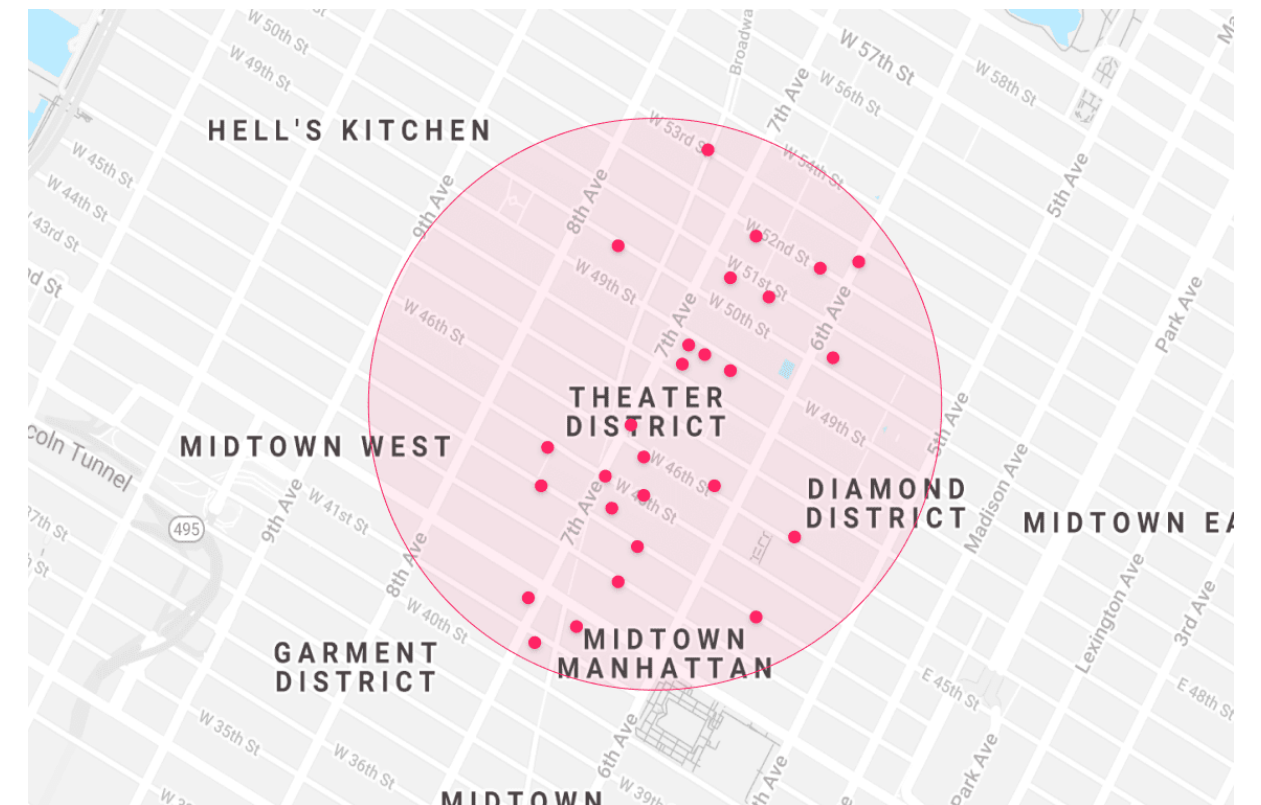
Pour enrichir un jeu de données, on peut avoir recours à des données exogènes complémentaires, à disposition gratuitement ou auprès des prestataires payants.

Open data

- A l'international : 50 Best Open Data Sources Ready to be Used Right Now (learn.g2.com/open-data-sources)
- En France : data.gouv.fr/fr/

Prestataires payants

- Predict HQ (predicthq.com) : événements sportifs, concerts, etc..
- Nombreux prestataires pour données météorologiques...



3.4. Feature engineering 10/11



Création de variables

Pour enrichir un jeu de données, on peut aussi créer des variables issues de la combinaison de plusieurs variables.

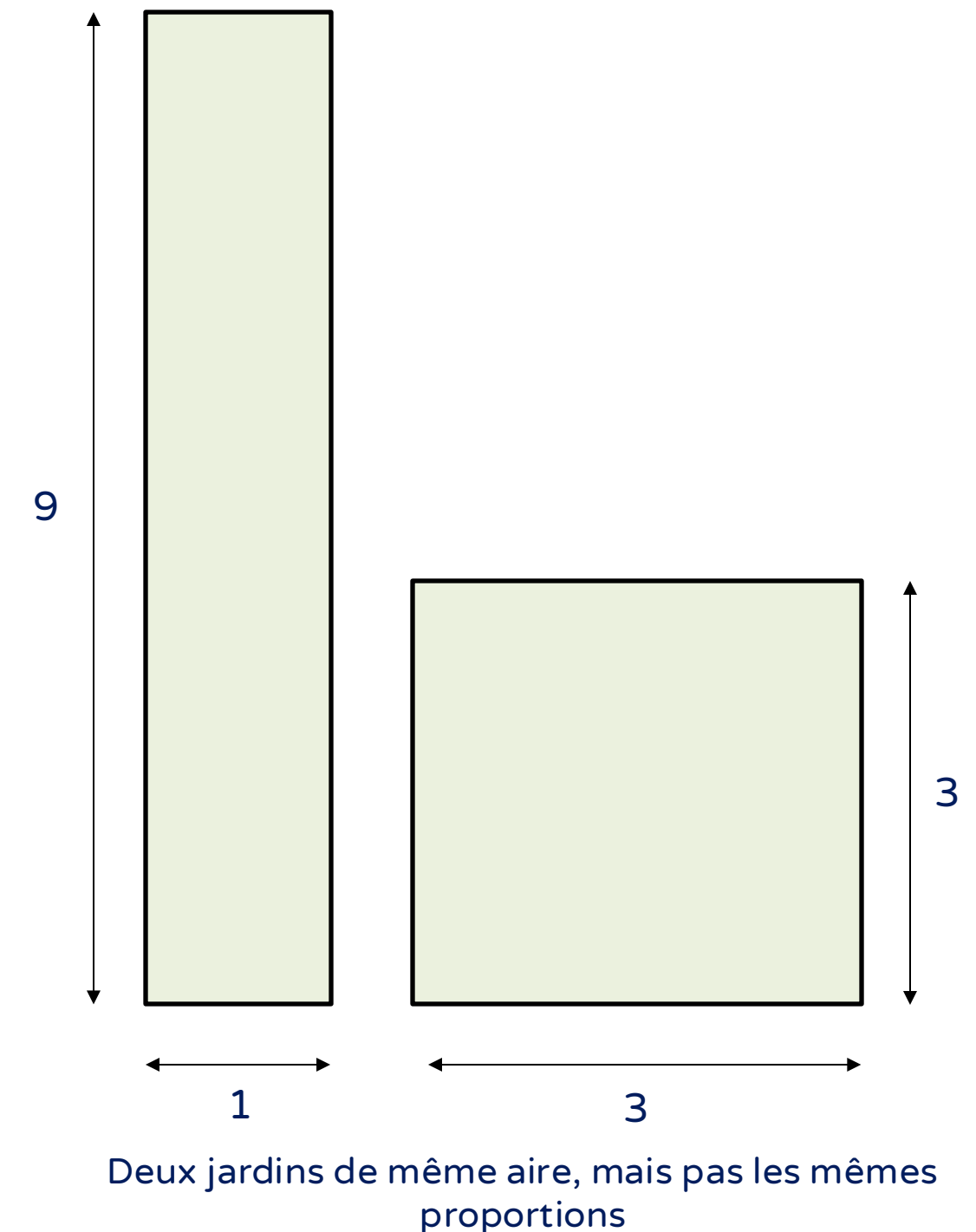
Variables « métier »

Exemple : température ressentie

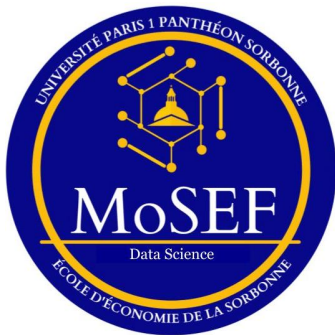
$$T_R = 13,12 + 0,6215 T_A + (0,3965 T_A - 11,37) \times V^{0,16}$$

Rapports entre variables

- Permet d'exprimer des relations non linéaires entre variables
- Particulièrement efficace avec des algorithmes de Boosting ou Random Forests
- Exemple : proportion d'un jardin et impact sur le prix d'un bien immobilier
- Autre variante : variables au carré, rapport d'une variable au carré et une autre variable



3.4. Feature engineering 11/11



Feature engineering automatique

L'étape de feature engineering peut être automatisée à l'aide de librairies spécialisées comme TSFresh.

TSFresh

- *extract_features* calcule un nombre exhaustif de variables extraites du signal
- *select_features* qui permet de sélectionner un ensemble de variables issues de *extract_features* selon des tests d'indépendance avec la cible qu'on cherche à estimer
- Il est possible d'intégrer cette extraction de variables directement dans une pipeline scikit-learn via le transformer *RelevantFeatureAugmenter*

Attention : rien ne remplace une feature créée « à la main » bien pensée !

Type	Variables
Descriptions purement statistiques du signal	<ul style="list-style-type: none">• length• has_duplicate• maximum• minimum• mean• median• quantile• variance
Descriptions du signal avec des métriques de série temporelle	<ul style="list-style-type: none">• abs_energy• absolute_sum_of_changes• agg_autocorrelation• agg_linear_trend• augmented_dickey_fuller
Décomposition du signal (ondelletes, transformation de Fourier, décomposition spectrale...)	<ul style="list-style-type: none">• cwt_coefficients• fft_aggregated• friedrich_coefficients• spkt_welch_density

3.7. Target engineering 1/2



Transformations Box-Cox

L'application d'une transformation de type Box-Cox sur la cible à prédire peut faciliter la tâche de l'algorithme de prévision et améliorer ses performances.

- Stabilisation de la variance
- En cas de distribution à longue queue, une transformation log recentre les données et rend la tâche de prévision plus facile (pour le calcul des splits)
- La transformation inverse permet d'obtenir des prévisions dans l'échelle originale.

$$w_t = \begin{cases} \log(y_t) & \text{si } \lambda = 0 \\ (y_t^\lambda - 1)/\lambda & \text{si } \lambda \neq 0 \end{cases}$$

Exemples de transformations

$\lambda = 1$: pas de transformation

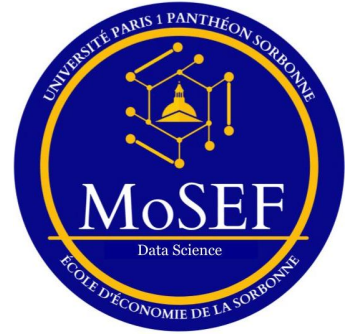
$\lambda = 1/2$: racine carrée (plus transformation linéaire)

$\lambda = 0$: logarithme

$\lambda = -1$: inverse (+1)

$$w_t = \begin{cases} e^{w_t} & \text{si } \lambda = 0 \\ (\lambda w_t + 1)^{1/\lambda} & \text{si } \lambda \neq 0 \end{cases}$$

3.7. Target engineering 2/2



Différentiation

Transformer la cible à prédire comme étant une **différence en quantité** entre l'instant t et $t+h$ peut avoir un impact significatif en terme de performance.

$$y_{diff}(t+h) = y(t+h) - y(t)$$

Cette différenciation tend à **rendre les données à prédire stationnaires**, mais :

- Sans pour autant en avoir la garantie
- Un test statistique ADF peut être effectué afin de le vérifier

On **prédit alors la cible différenciée** $y_{diff}(t+h)$ à l'horizon h , et l'on peut **reconstituer la prévision** $\hat{y}(t+h)$ sur la cible y non différenciée :

$$\hat{y}(t+h) = y(t) + \hat{y}_{diff}(t+h)$$

3.8. Time to practice



Rendez-vous sans plus attendre sur Python pour la mise en pratique !





Séries temporelles

Méthodes ML et DL

Modèles Machine Learning

Intervenant

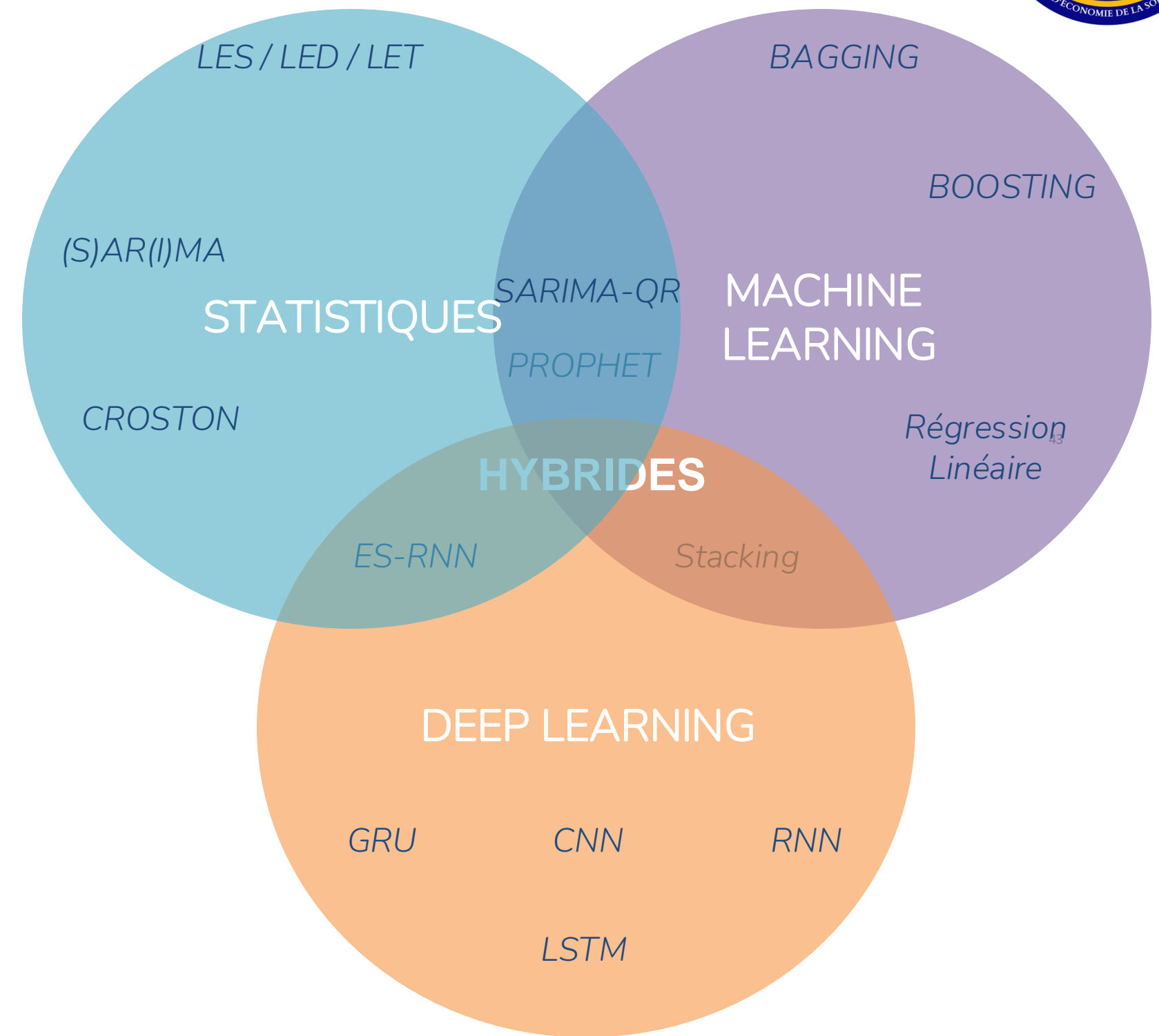
Guillaume Hochard

4.1. Introduction 1/3



Panorama des méthodes

- **Méthodes statistiques**
- ARIMA, Holt-Winters, Croston
- **Méthodes Machine Learning**
- Régression linéaire, Bagging, Boosting, ...
- **Méthodes Deep Learning**
- LSTM, GRU, CNN, ...



4.1. Introduction 2/3

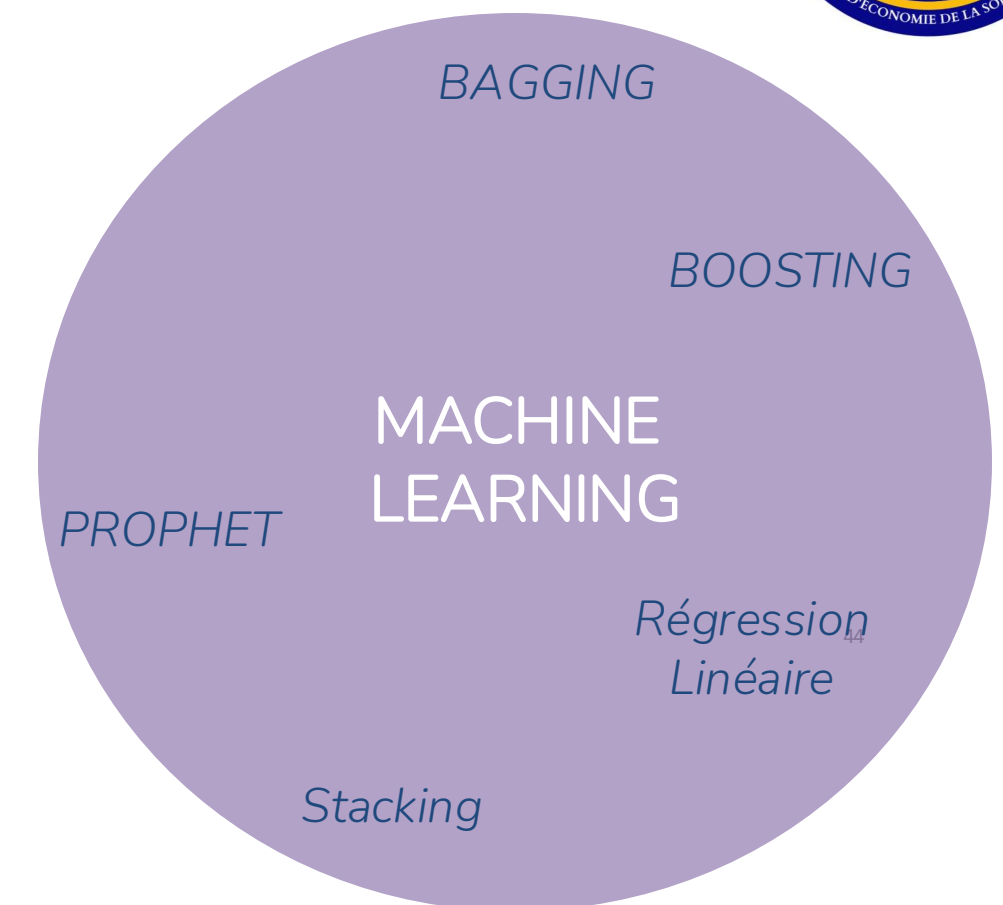


Méthodes Machine Learning

Tout modèle de régression peut être appliqué à un problème de prévision de série temporelle

Les prérequis :

- Passage d'un problème de série temporelle à un **problème de régression** (création de la cible)
- Créer des **variables qui sont ergodiques** (ex : convergence de la moyenne vers une valeur finie au fur et à mesure que les données sont collectées)
- Vérifier **les hypothèses de stationnarité** (en fonction du choix du modèle, une tendance ne peut pas être modélisée)



4.3. Introduction 3/3

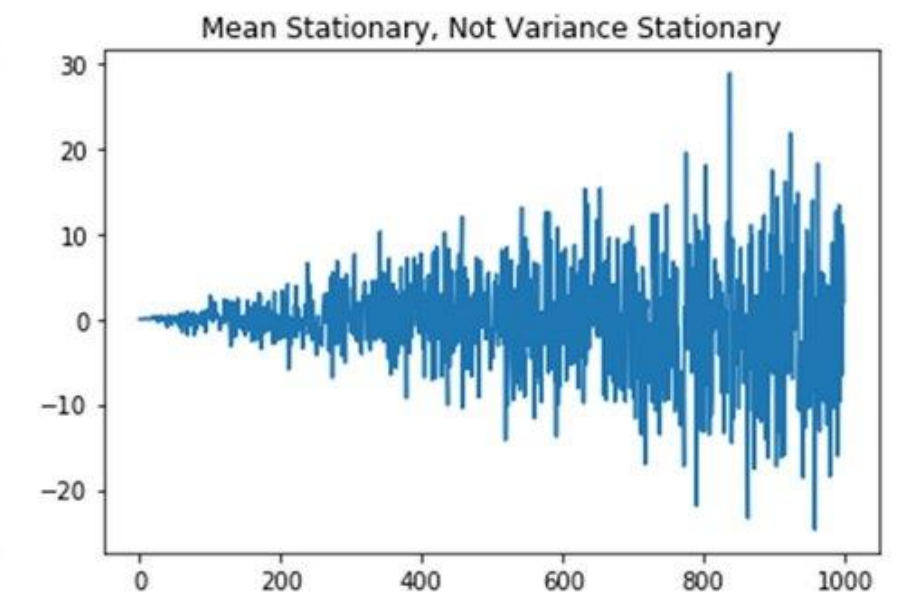
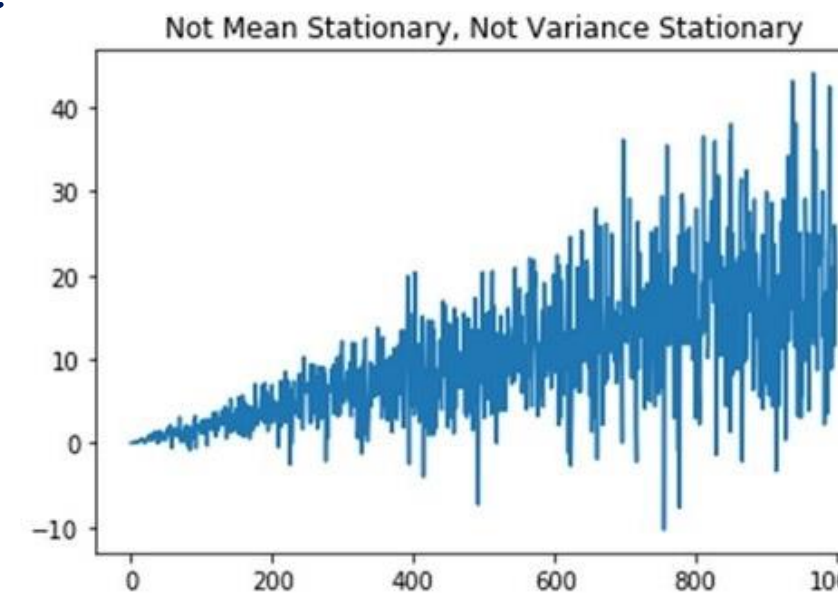
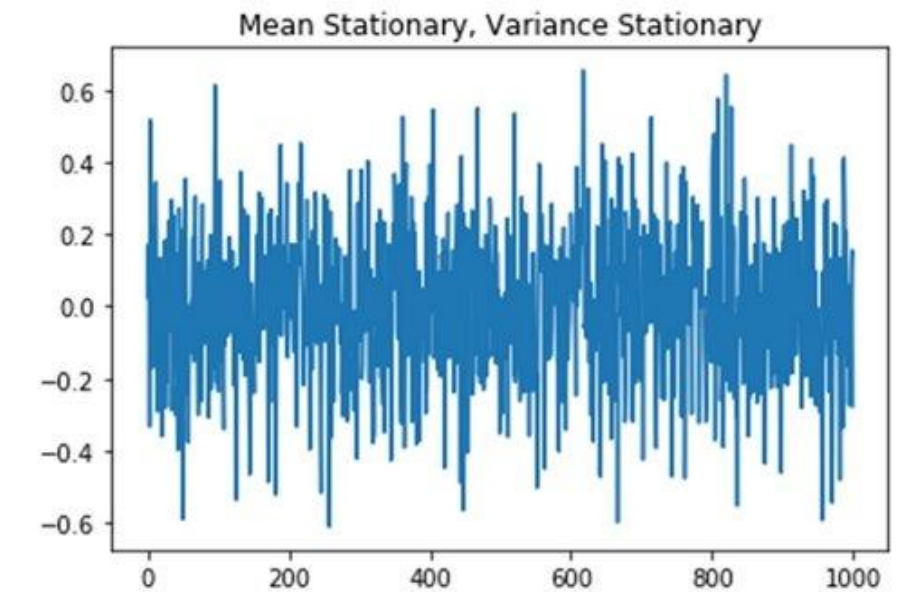
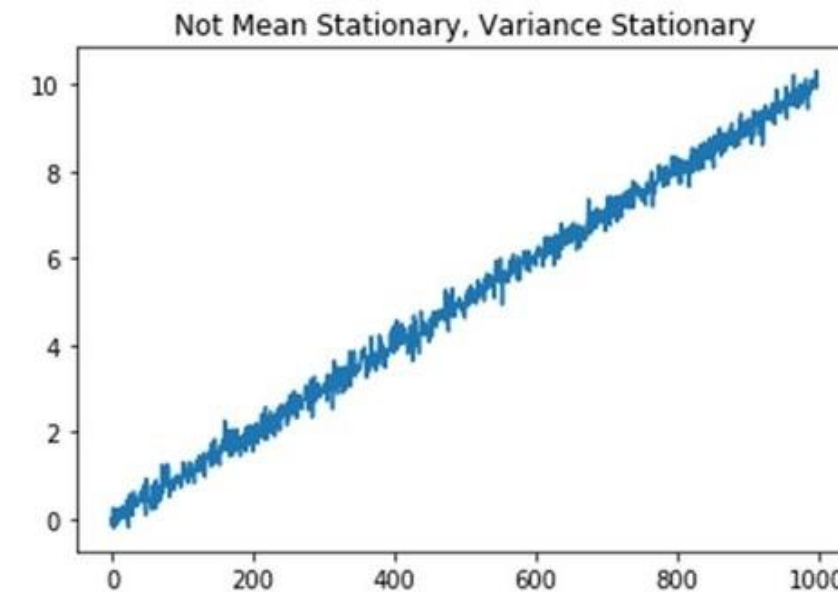


Différents types de non stationnarités

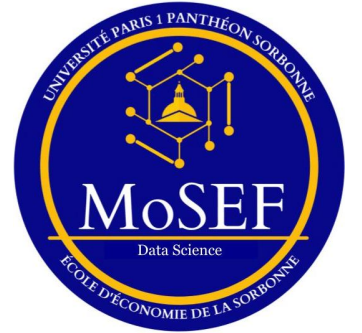
- Moyenne : il y a une tendance dans les données
- Variance : l'amplitude des saisonnalités ou la volatilité augmente au cours du temps
- La stationnarité est utile car elle soulage le modèle d'une partie de son travail

Méthodes pour rendre une série stationnaire

- Différentiation, modéliser la tendance à part
- Transformation Box-Cox (variance)



4.2. Stratégies d'entraînement 1/6



Un modèle ML intégrant des variables retardées (lags) ne peut prédire plusieurs pas de temps différents, il est spécialisé pour prédire un pas de temps spécifique, par exemple à $t+1$, à $t+2$ ou à $t+\text{horizon}$. En effet, les features de lag ne sont accessibles que pour les horizons pour lesquels nous pouvons les calculer

Pour un modèle à horizon $t+10$, on ne peut inclure la feature de lag $t-1$, puisque elle ne sera disponible que dans $t+9$ pas de temps.

Différentes stratégies pour prédire sur plusieurs horizons

- Direct
- Recursive
- DirRec

4.2. Stratégies d'entraînement 2/6

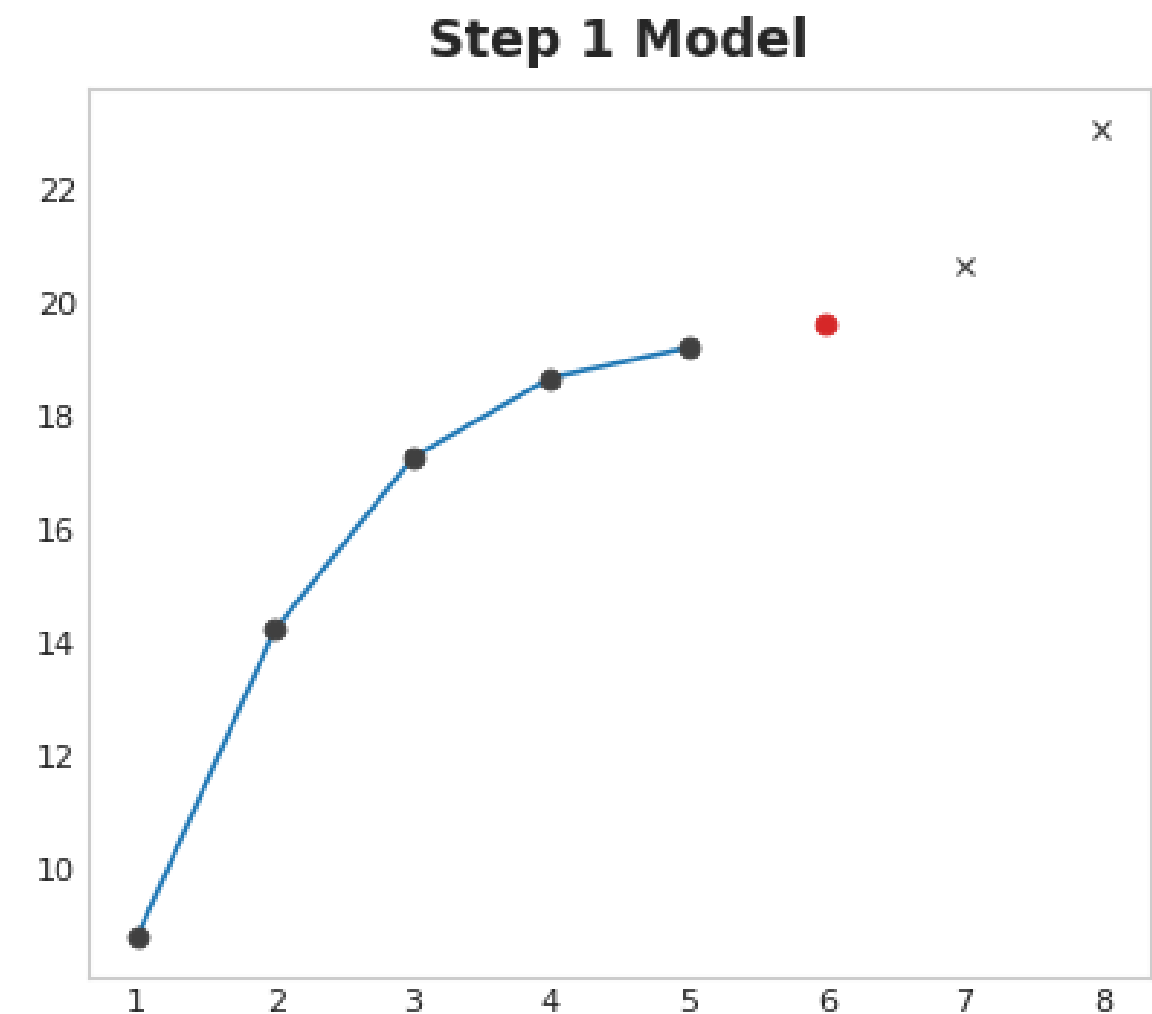


Stratégie directe

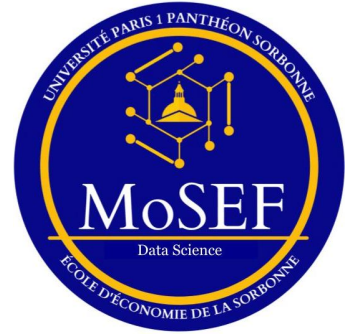
- Pour chaque pas de temps, on entraîne un modèle spécifique
- Prévoir à T+3 est différent de prévoir à T+2, T+1

Inconvénients

- Entraînement **de multiple modèles** si beaucoup d'horizons sont demandés (exemple forecast S&OP : 13 semaines)
- **Couteux en temps de calcul**
- **Mise en production** et suivi de la **dérive des modèles** plus complexes



4.2. Stratégies d'entraînement 2/6



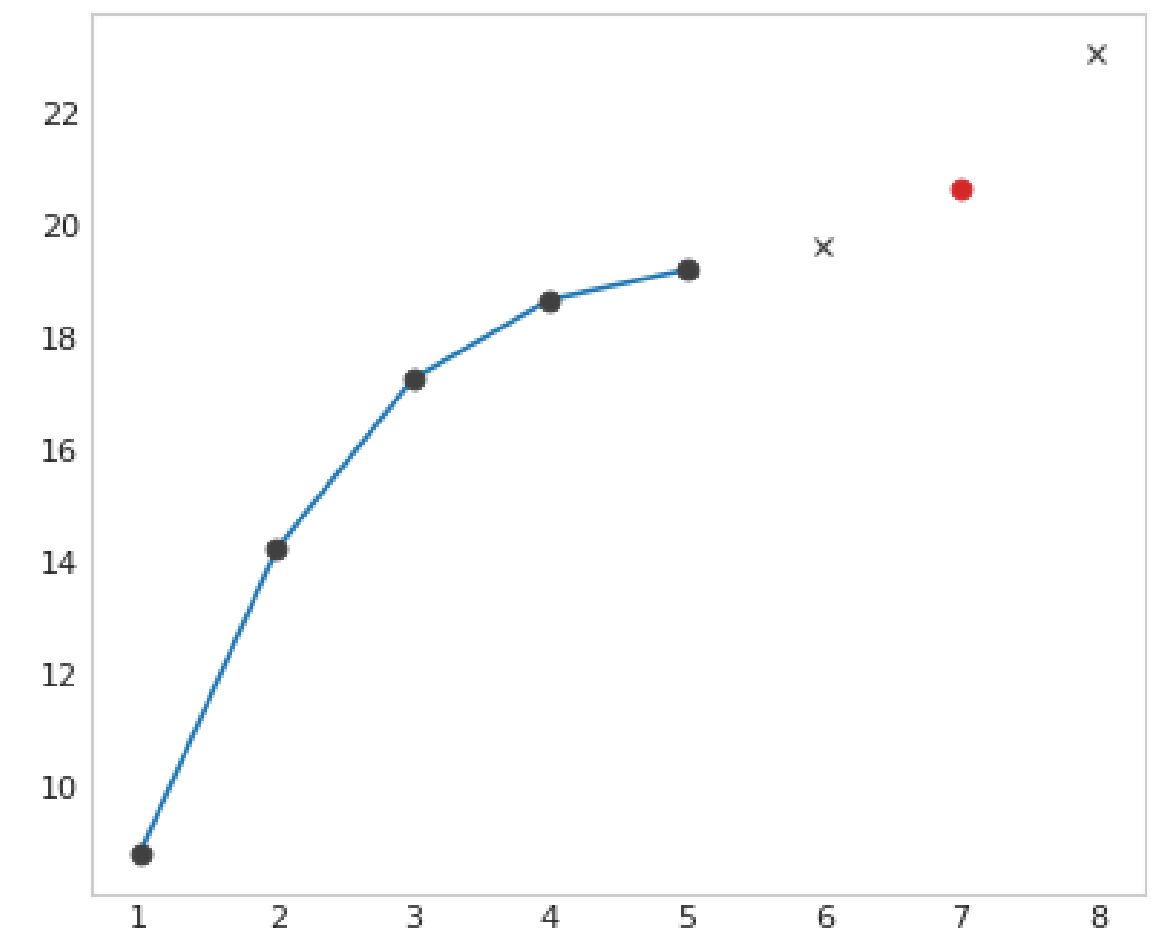
Stratégie directe

- Pour chaque pas de temps, on entraîne un modèle spécifique
- Prévoir à T+3 est différent de prévoir à T+2, T+1

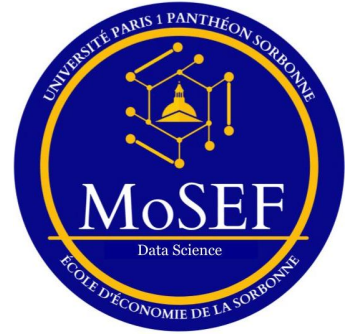
Inconvénients

- Entraînement **de multiple modèles** si beaucoup d'horizons sont demandés (exemple forecast S&OP : 13 semaines)
- **Couteux en temps de calcul**
- **Mise en production** et suivi de la **dérive des modèles** plus complexes

Direct Strategy
Step 2 Model



4.2. Stratégies d'entraînement 2/6

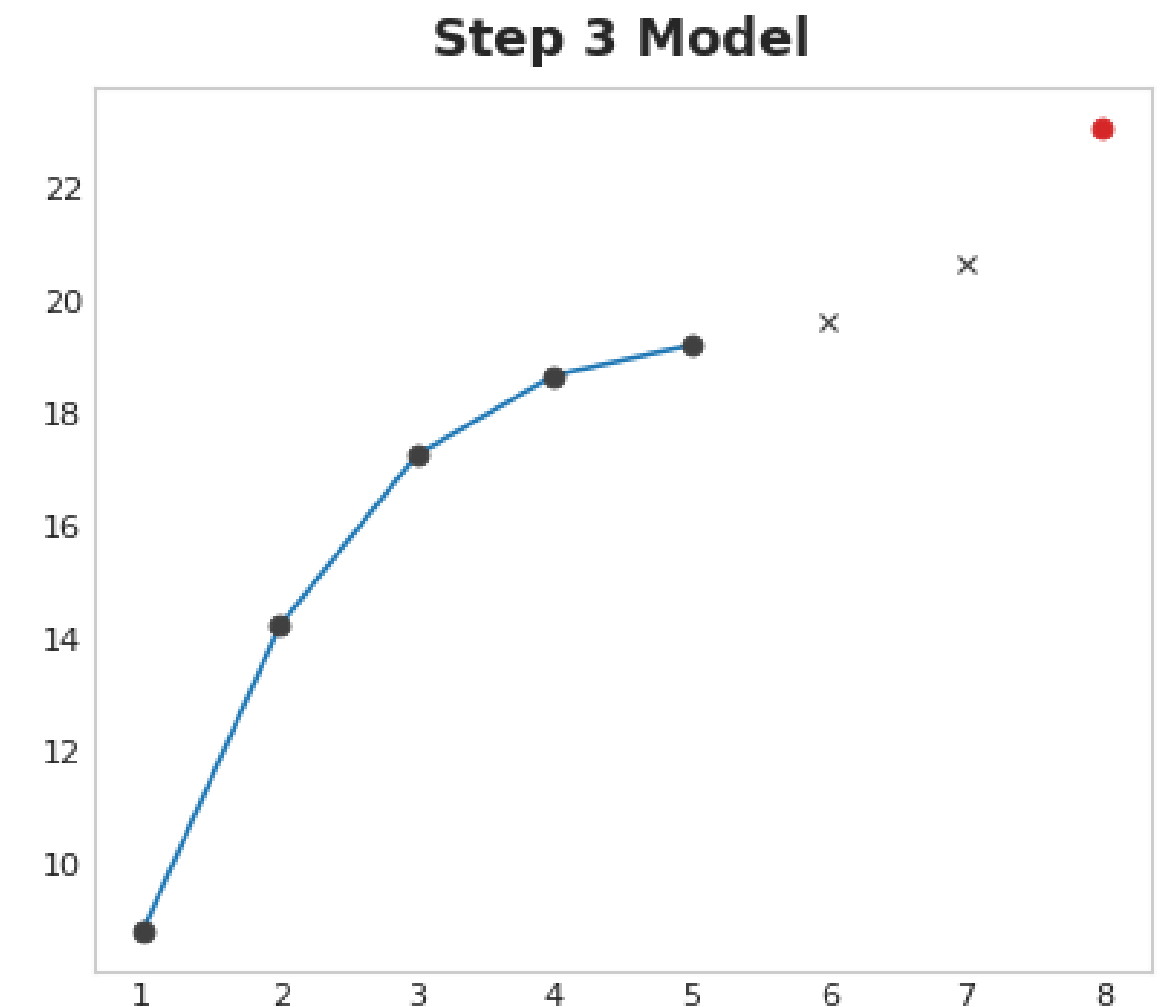


Stratégie directe

- Pour chaque pas de temps, on entraîne un modèle spécifique
- Prévoir à T+3 est différent de prévoir à T+2, T+1

Inconvénients

- Entraînement **de multiple modèles** si beaucoup d'horizons sont demandés (exemple forecast S&OP : 13 semaines)
- **Couteux en temps de calcul**
- **Mise en production** et suivi de la **dérive des modèles** plus complexes



4.2. Stratégies d'entraînement 3/6



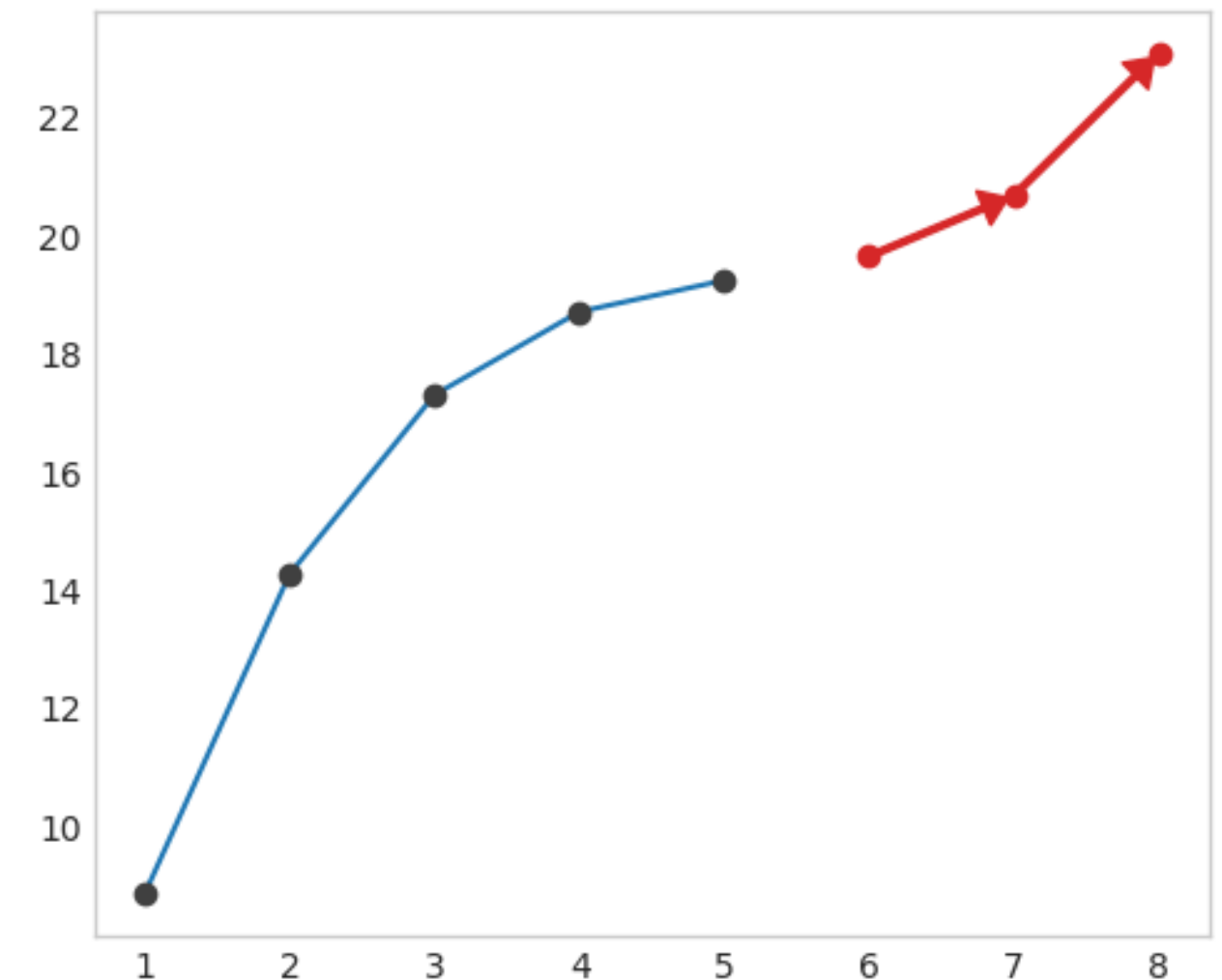
Stratégie réursive

- Un seul modèle, mono-étape (prévoir à $T+1$)
- Utilisation du forecast pour mettre à jour les valeurs de lag

Inconvénients

- **Propagation d'erreur** : plus l'horizon à prévoir est long, plus grande est l'erreur
- **Valable si on connaît les valeurs dans le futur des régresseurs externes**, i.e. variables d'entrée (sinon il faut aussi les prévoir)

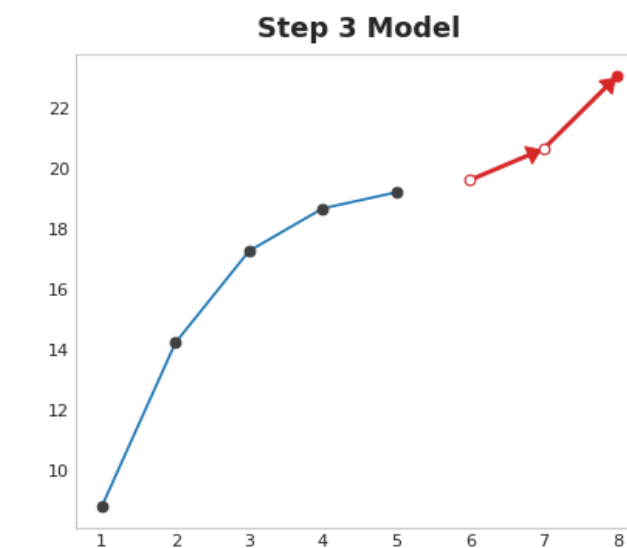
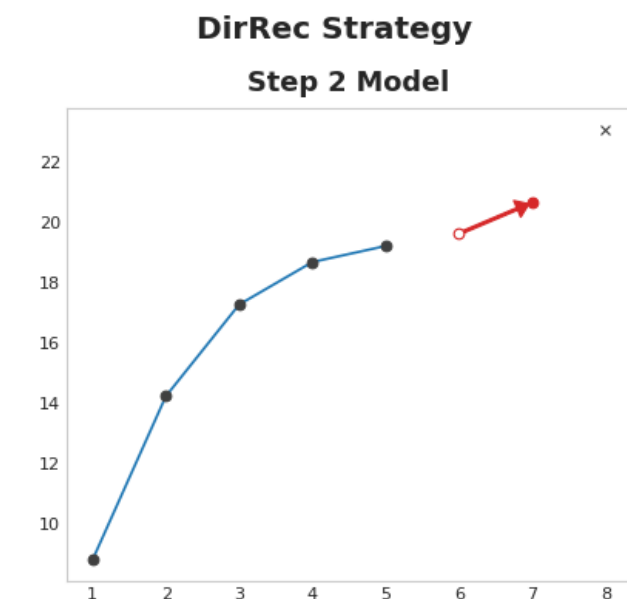
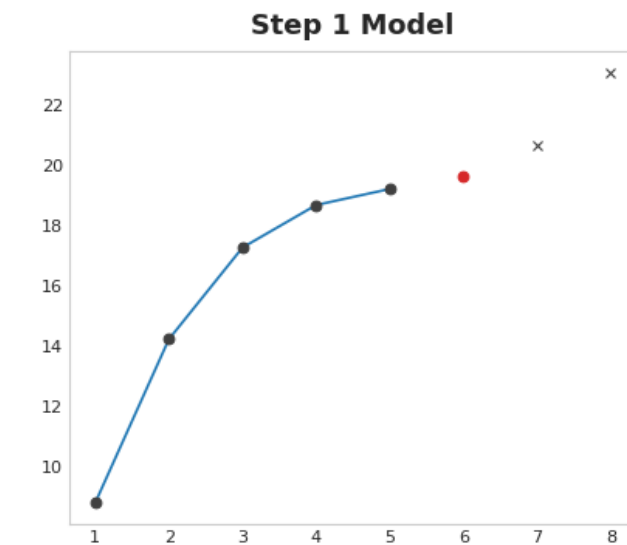
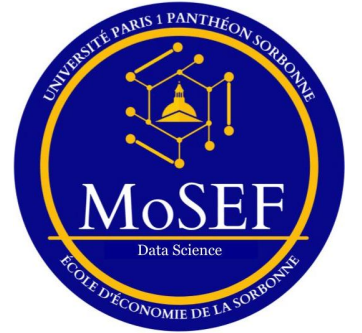
Recursive Strategy



4.2. Stratégies d'entraînement 4/6

Stratégie DirRec (hybride)

- Combinaison des deux méthodes précédentes
- Entraînement d'un modèle pour chaque horizon à prévoir
- Utilisation des forecasts précédents pour mettre à jour les valeurs de lag
- Capture mieux les dépendances que l'approche Directe
- Propagation d'erreur comme pour l'approche récursive

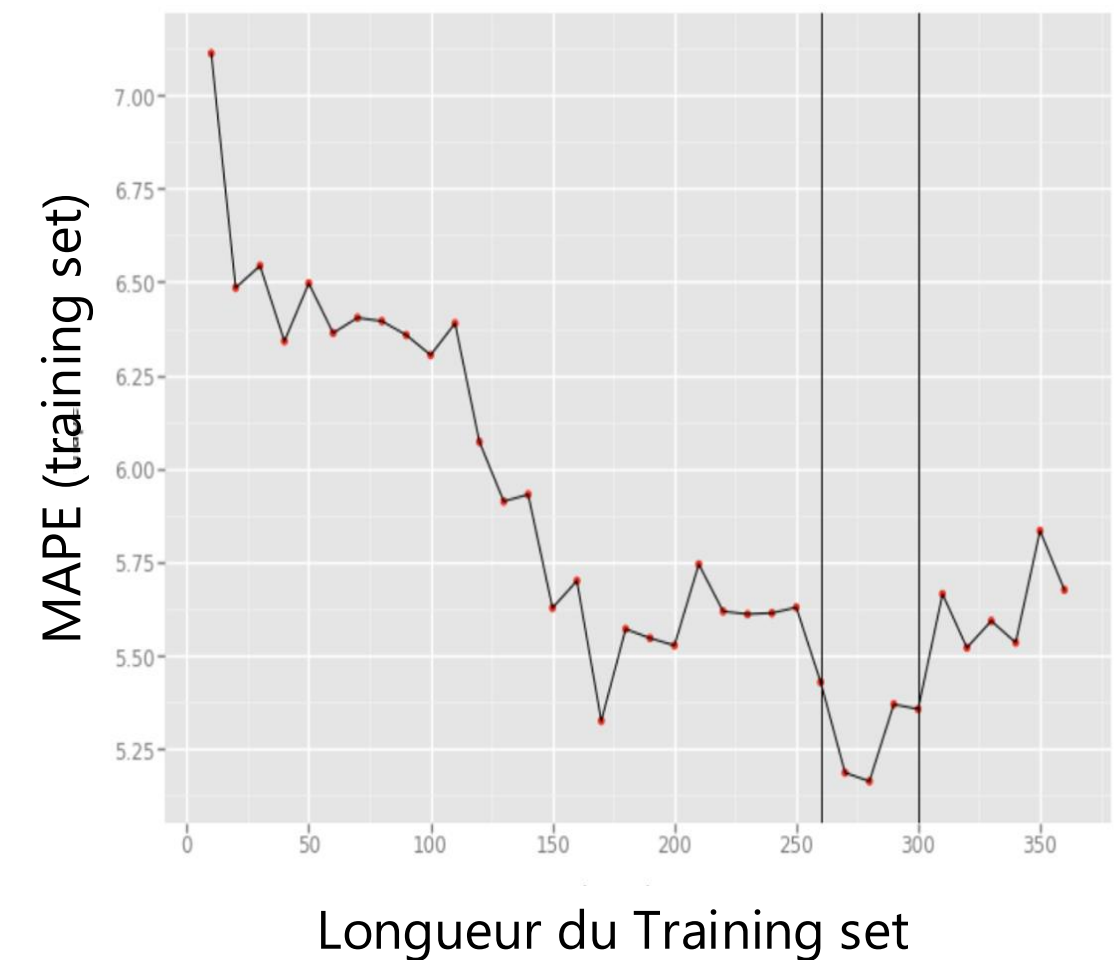
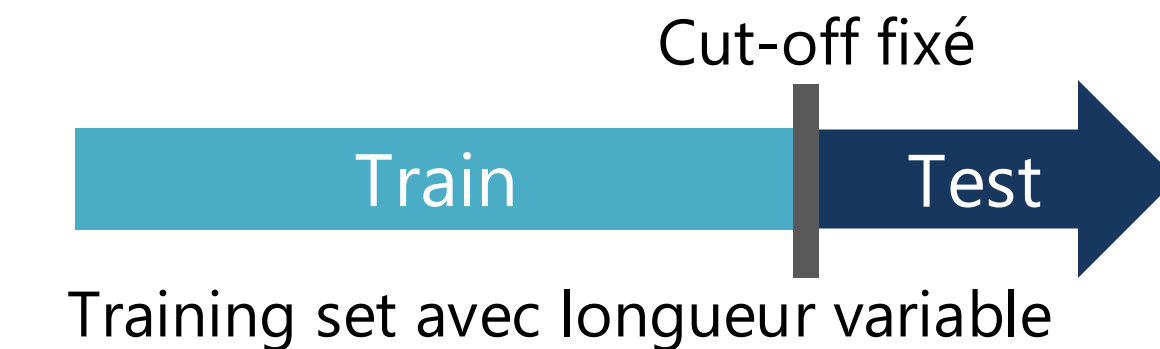


4.2. Stratégies d'entraînement 5/6



Quelle profondeur d'historique dois-je inclure dans le jeu d'entraînement?

- Méthodes par sliding/expanding window
- La **taille du jeu d'entraînement** peut également avoir un impact sur la performance du modèle
- Tout dépend si l'information utile pour prédire est contenue dans des **patterns courts termes ou longs termes**
- A évaluer en fonction des **différents horizons à prévoir**
- Compromis entre avoir des **échantillons plus représentatifs/moins bruités** et le **nombre d'échantillons** dans le jeu d'entraînement

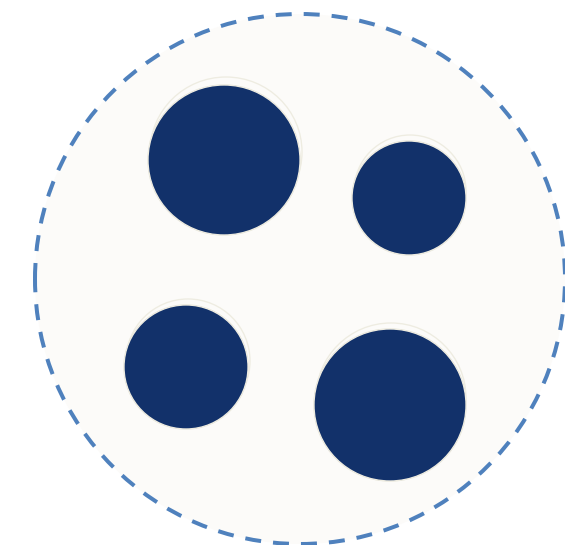
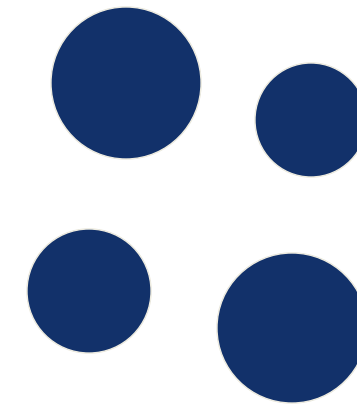


4.2. Stratégies d'entraînement 6/6



Modèle local / modèle global

- **Modèle local** : les paramètres du modèle de prédiction sont estimés individuellement pour chacune des séries temporelles du système étudié (ARIMA, Prophet, ...)
- **Modèle global** : les paramètres du modèle de prédiction sont estimés conjointement grâce à l'ensemble des séries temporelles du système étudié. (modèles ensembliste, modèles deep learning)



4.3. Prophet 1/3



Prophet utilise **un modèle additif** de série chronologique **décomposable** avec trois composantes principales : **la tendance**, **la saisonnalité** et les **vacances**.

Elles sont combinées dans l'équation suivante :

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

- $g(t)$: courbe de croissance linéaire ou logistique par morceaux pour la modélisation des changements non périodiques dans les séries temporelles
- $s(t)$: changements périodiques (par exemple, saisonnalité hebdomadaire/annuelle)
- $h(t)$: effets des vacances (fournies par l'utilisateur)
- ε_t : le terme d'erreur tient compte de toute modification inhabituelle non prise en compte par le modèle

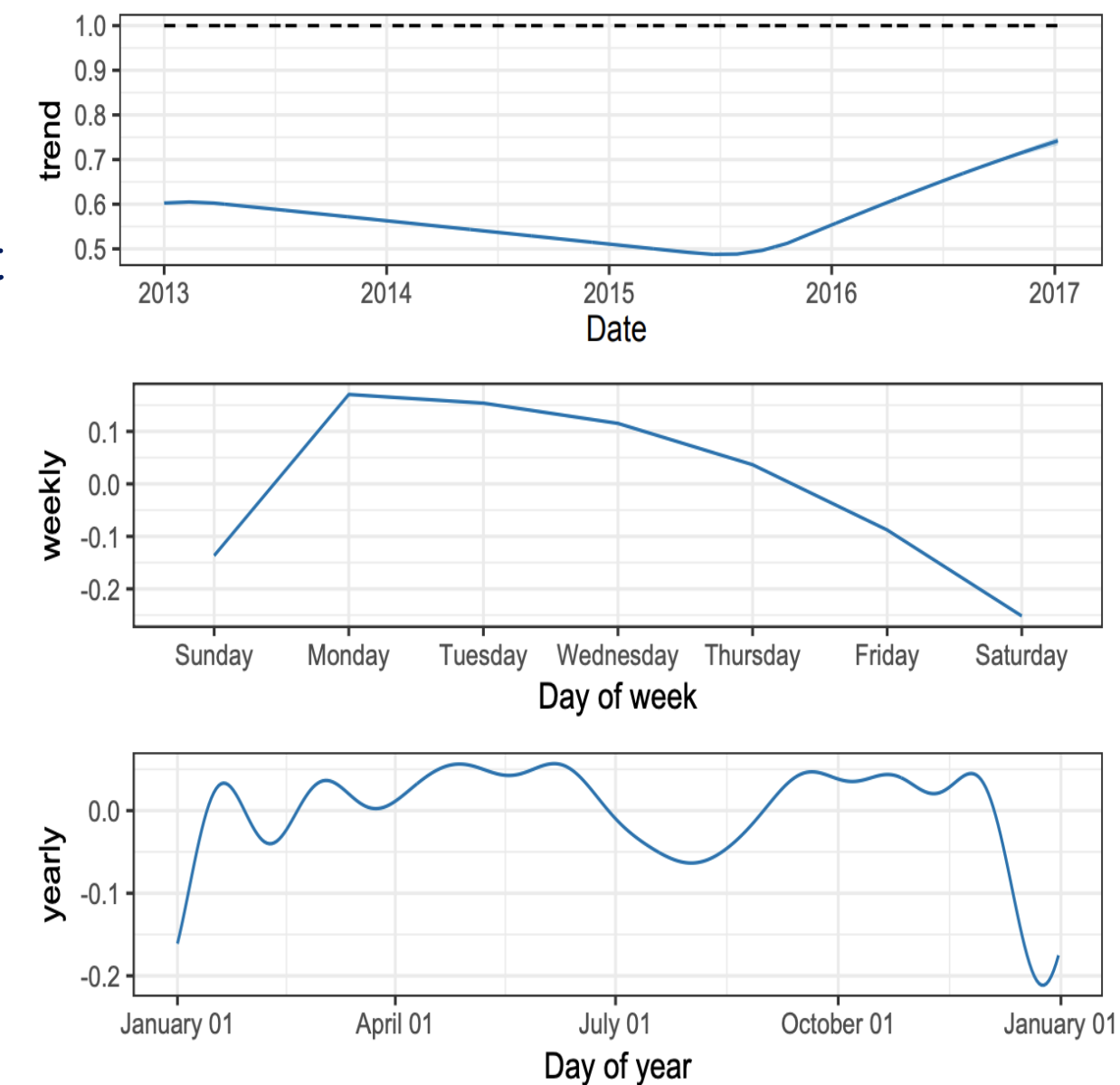


4.3. Prophet 2/3



Avantages de Prophet

- Fonctionne bien avec des TS qui ont de **fortes saisonnalités et plusieurs historiques de ces saisonnalités**.
- Bonne **résistance aux données manquantes**, aux valeurs aberrantes et aux changements de tendance.
- Facilement configurable par un non initié au forecast :
 - Capacities
 - Changepoints
 - Holidays and seasonality
 - Smoothing parameters
- Interprétabilité facilitée du fait de la construction du modèle
- Intervalles de confiance disponibles



4.3. Prophet 3/3



Critiques de Prophet

- Nombreuses critiques sur sa performance
- Les papiers originaux **ne se comparent pas** à d'autres modèles
- Facebook affirme que l'algorithme convient à leur besoin et leur contexte sans fournir d'éléments quantitatifs
- De nombreuses études montrent une **sous performance face à des algorithmes statistiques** (ARIMA)

[Comparing Prophet and Deep Learning to ARIMA in Forecasting Wholesale Food Prices](#)
[A Worrying Analysis of Probabilistic Time-series Models for Sales Forecasting](#)

- Conclusion : **toujours benchmark** un modèle à un ensemble d'autres modèles, ne jamais partir dans une voie unique de modélisation !



4.4. Méthodes ensemblistes 1/2



Bagging, boosting

Bagging

- Méta-algorithme faisant partie des méthodes ensemblistes
- Partant d'un algorithme de Machine Learning, il utilise de multiples fois cet algorithme pour obtenir un résultat plus fiable
- Dans le cas des forêts aléatoires, l'algorithme utilisé de multiples fois est celui de l'arbre de décision

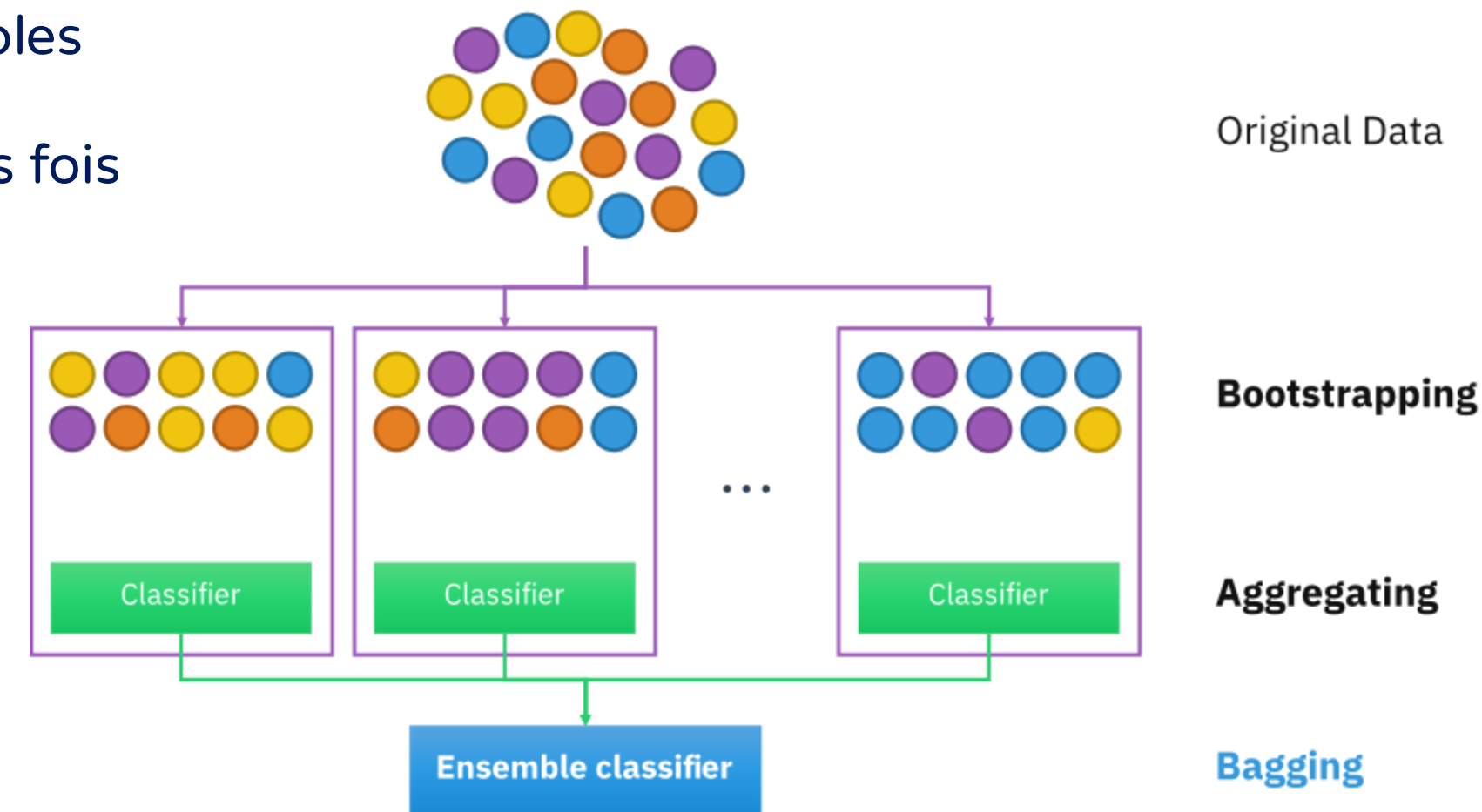
Bootstrapping

On réalise un échantillonnage des données

On entraîne l'algorithme de façon séparée sur chacun de ces échantillons

Aggregating

- Assemblage des résultats des modèles obtenus
- Classification : vote majoritaire
- Régression : Moyenne des prédictions



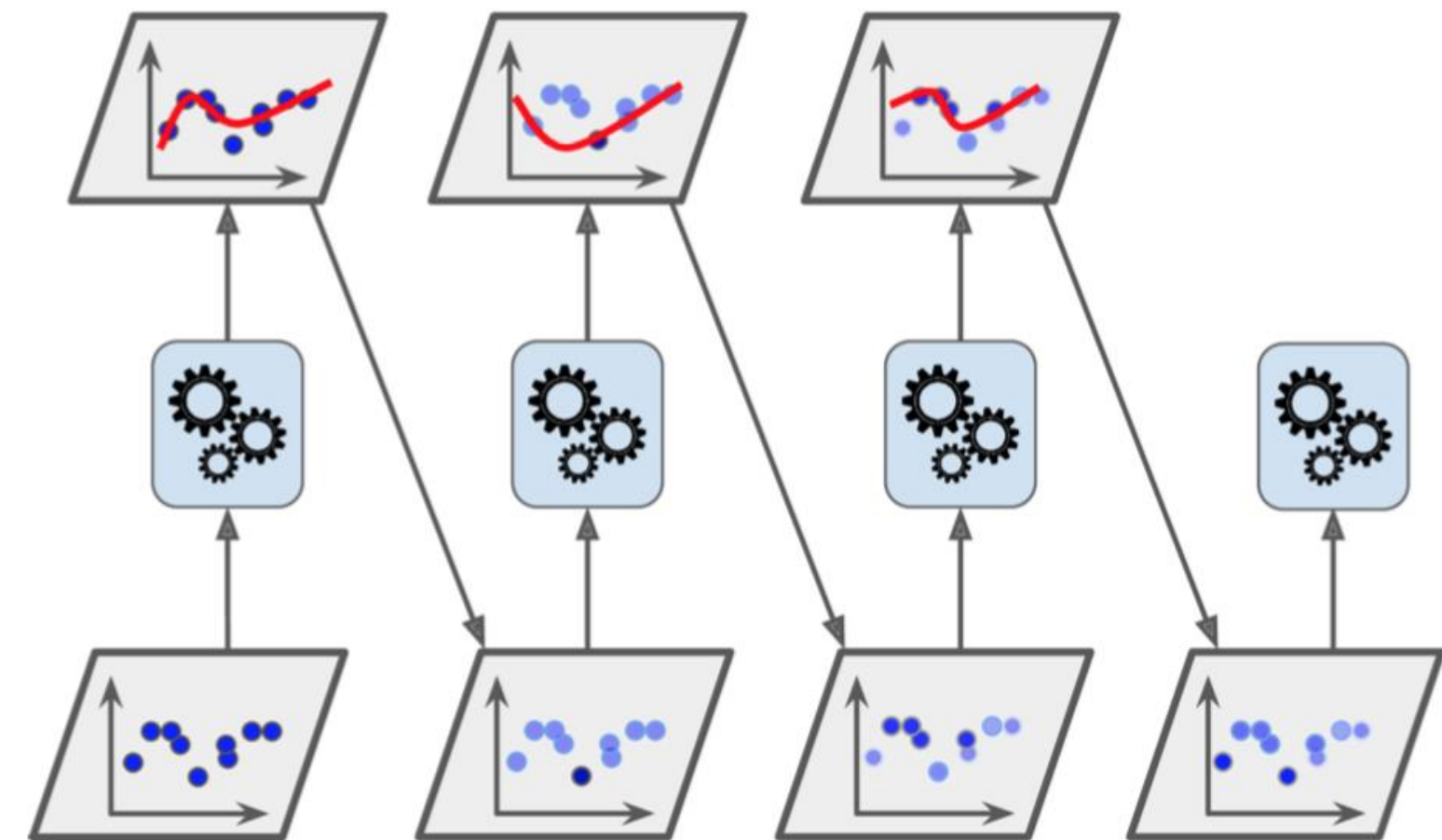
4.4. Méthodes ensemblistes 2/2



Bagging, boosting

Boosting

- Méthode ensembliste
 - Utilise des weak learners (meilleurs que l'aléatoire, mais pas de beaucoup) complémentaires
 - Apprentissage itératif du weak learner basé sur les résultats du précédent
 - Dérive un modèle robuste à partir de cette itération
-
- Pour XGBoost, l'algorithme utilisé de multiples fois est celui de l'arbre de décision ou modèle de régression linéaire



*Le modèle 1 commet une erreur importante sur le point A.
Son poids w_A est plus grand pour l'entraînement du modèle 2.
Le processus est itératif ...*

4.4. Gradient Boosting 1/9



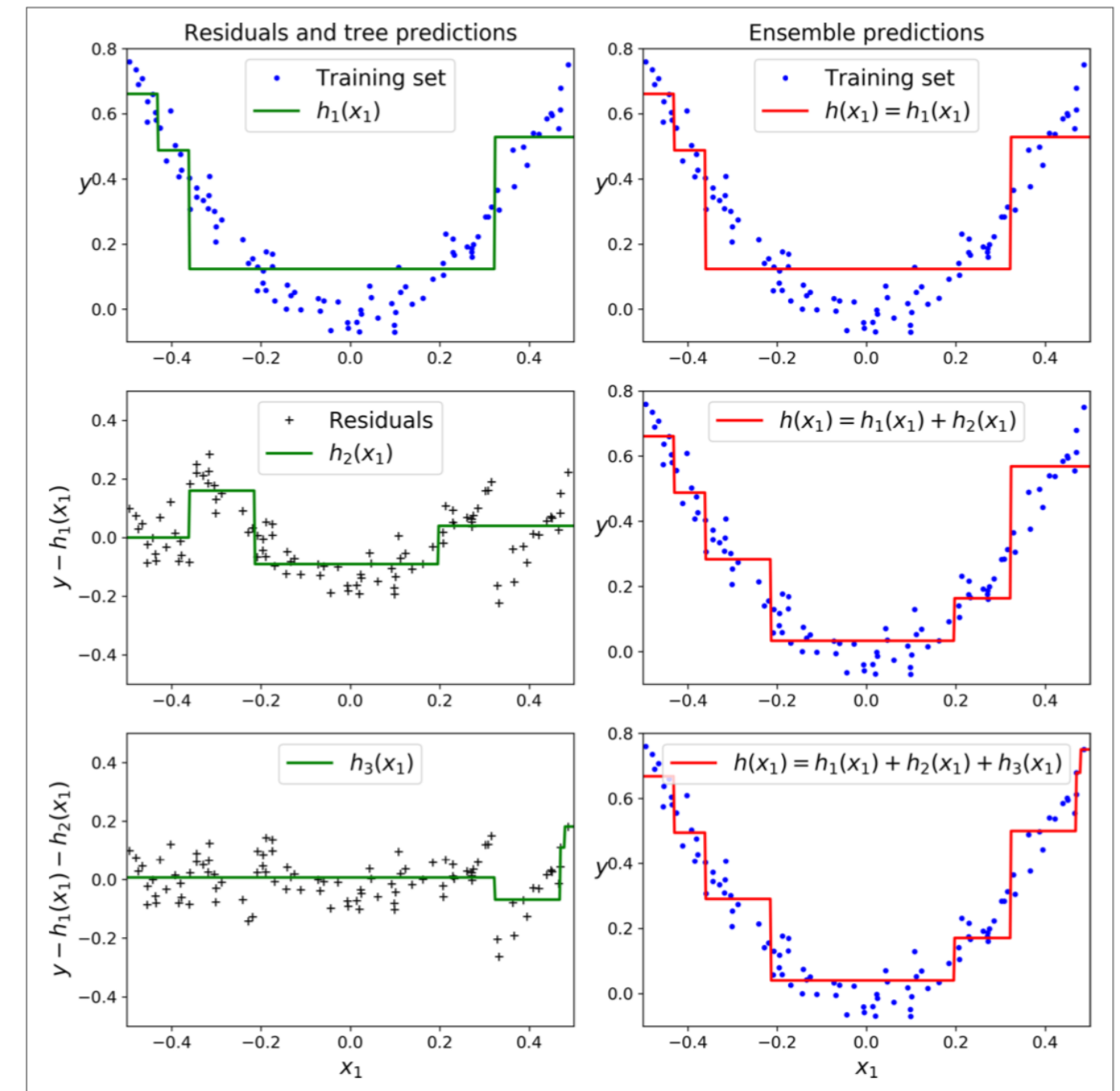
Gradient Boosting : Rappels

Entraînement séquentiel à partir d'arbres de décision

- Le premier arbre construit est un classificateur faible.
- Le deuxième arbre est entraîné sur les résidus du premier arbre
- ...
- Avec K arbres construits : $\hat{y}_i = \sum_{k=1}^K \text{Arbre}_k(x_i)$

Caractéristiques

- De nombreux hyperparamètres : taux d'apprentissage, profondeur d'arbres (risques d'overfitting), ...
- Des variantes plus performantes : XGBoost, LightGBM, CatBoost



4.4. Gradient Boosting 2/9



Gradient Boosting : XGBoost

Principe de XGBoost

- Déclinaison du Gradient Boosting qui propose des modèles très performants pour des temps d'exécution plus courts
- À la différence de GBM, XGBoost estime le weak learner suivant en minimisant directement la loss (*) en passant par un développement limité d'ordre 2

$$(*)L(y_i, f_{m-1}(x_i) + \gamma) \approx L(y_i, f_{m-1}(x_i)) - r_{im}\gamma + \frac{1}{2}h_m(x_i)\gamma^2$$

- Le modèle comporte de nombreux paramètres (nombre d'itérations, régularisation L1 et L2...)
- Très populaire suite à une série de victoires sur Kaggle en 2017

1. Initialize $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$.

2. For $m = 1$ to M :

(a) For $i = 1, 2, \dots, N$ compute

$$r_{im} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}.$$

(b) Fit a regression tree to the targets r_{im} giving terminal regions

$$R_{jm}, j = 1, 2, \dots, J_m.$$

(c) For $j = 1, 2, \dots, J_m$ compute

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma).$$

(d) Update

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}).$$

3. Output $\hat{f}(x) = f_M(x)$.

Algorithme GBM

4.4. Gradient Boosting 3/9

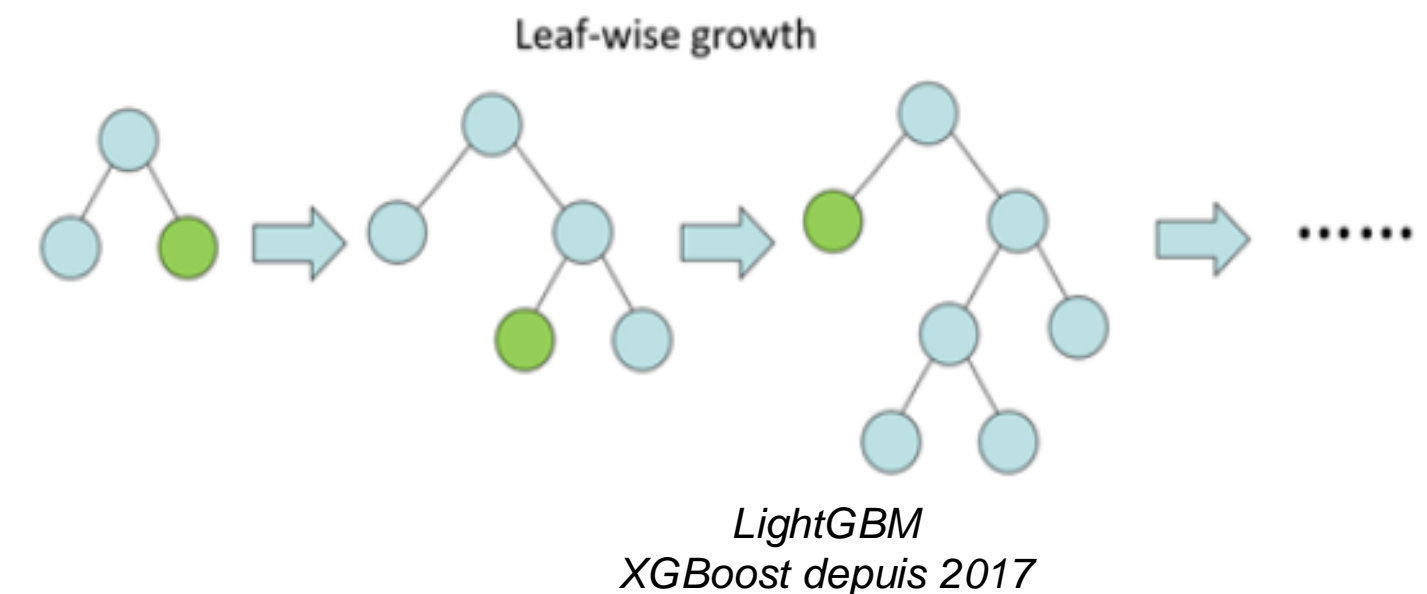
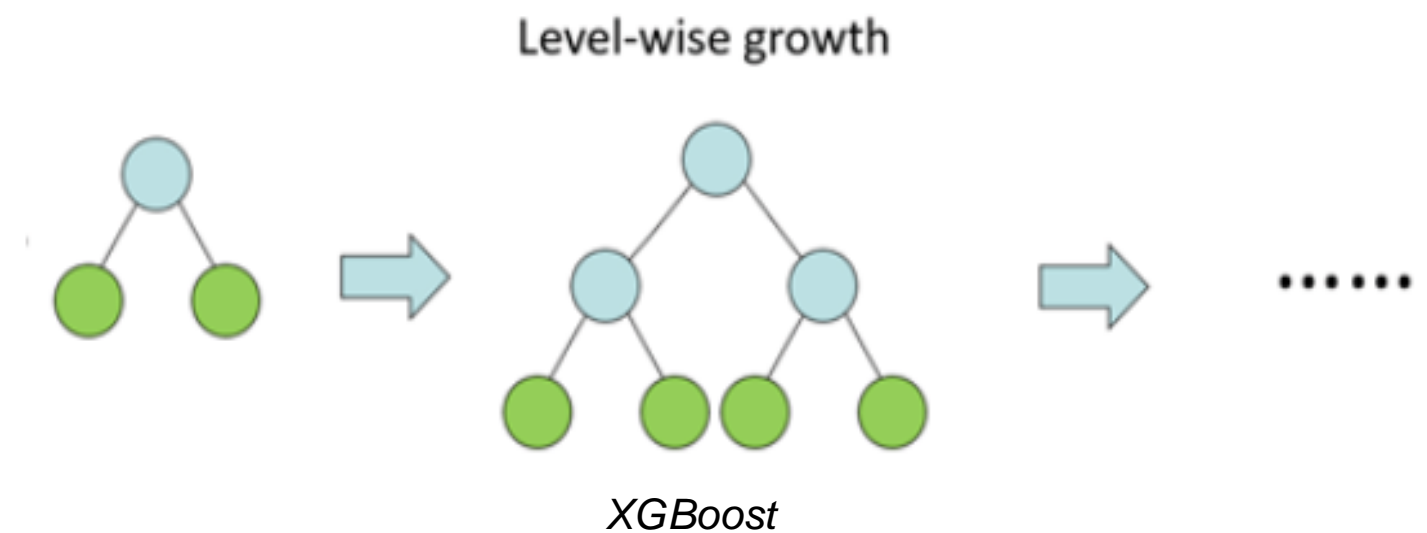


XGBoost, LightGBM, CatBoost : Quelles différences ?

- Construction des arbres
- XGBoost : Level-wise growth
- LightGBM : Leaf-wise growth
- CatBoost : Level-wise symmetric Growth

La stratégie Leaf-wise a tendance à obtenir une *loss* plus faible que la stratégie Level-wise, mais elle a tendance à *overfitter*, en particulier pour les petits datasets

Dans ce cas, la croissance Level-wise agit comme une *régularisation* pour limiter la complexité de l'arbre

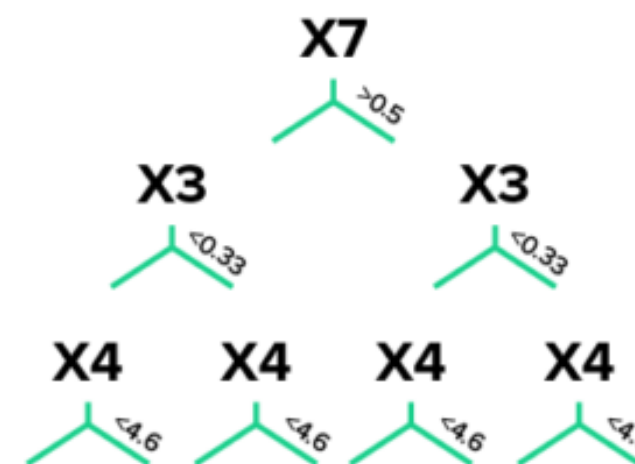


4.4. Gradient Boosting 4/9

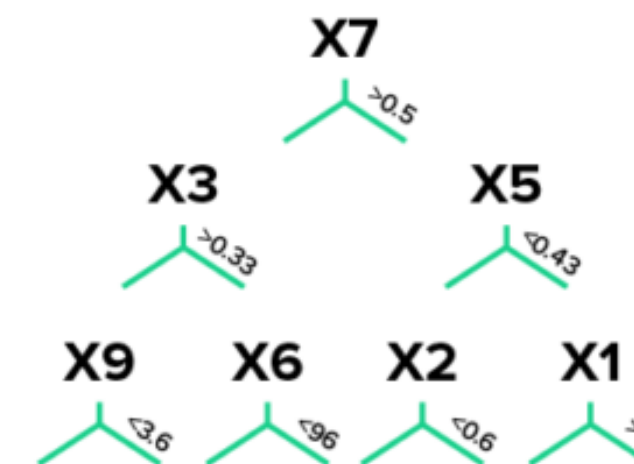


XGBoost, LightGBM, CatBoost : Quelles différences ?

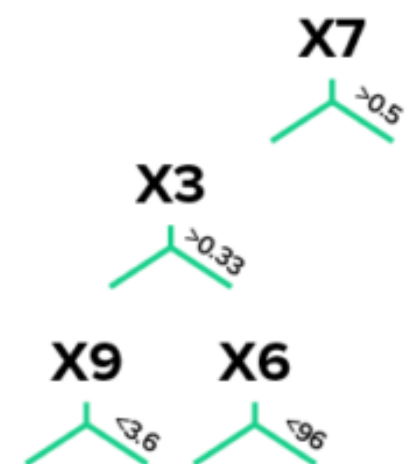
- Construction des arbres
- XGBoost : Level-wise growth
- LightGBM : Leaf-wise growth
- CatBoost : Level-wise symmetric Growth



CatBoost



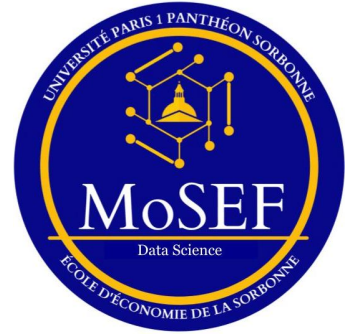
XGBoost



LightGBM

CatBoost construit des **arbres symétriques**, les mêmes splits étant effectués de part et d'autre d'un même nœud de l'arbre.

4.4. Gradient Boosting 5/9



XGBoost, LightGBM, CatBoost : Quelles différences ?

- Splits

LightGBM : échantillonnage unilatéral basé sur le gradient (GOSS)

- Sélection du split en utilisant **toutes les instances avec de grands gradients** (c'est-à-dire une grande erreur) et un échantillon aléatoire d'instances avec de petits gradients
- **Conservation de la distribution des données** lors du calcul du gain d'information, via un multiplicateur constant pour les instances de données à faible gradient
- **Bon équilibre entre l'augmentation de la vitesse** (en réduisant le nombre d'instances de données) et le **maintien de la précision** des arbres de décision appris

4.4. Gradient Boosting 6/9



XGBoost, LightGBM, CatBoost : Quelles différences ?

- Splits

CatBoost : Minimal Variance Sampling (MVS)

- Version à **échantillonnage pondéré** du Stochastic Gradient Boosting
- L'échantillonnage pondéré se fait **au niveau de l'arbre et non au niveau du split**
- Les observations pour chaque arbre de boosting sont échantillonnées de manière à maximiser la précision du score de split

Xgboost

- Pas d'optimisation particulière, ce qui rend plus lent cette étape

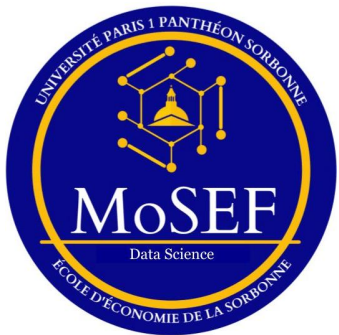
4.4. Gradient Boosting 7/9












XGBoost, LightGBM, CatBoost : Quelles différences ?

- Gestion des variables catégorielles
- CatBoost : combinaison d'un *one-hot encoding* et d'un *mean encoding* avancé, en fonction du nombre de modalités de la variable catégorielle
- LightGBM : sépare les variables catégorielles en *partitionnant leurs catégories en deux sous-ensembles*. L'idée sous jacente est de d'ordonner les catégories en fonction de l'objectif d'entraînement à chaque split
- Xgboost : pas d'incorporation par défaut (sauf mode experimental). L'*encodage doit être fait manuellement par l'utilisateur*, par *one hot* ou *target encoding* par exemple.

4.4. Gradient Boosting 8/9



XGBoost, LightGBM, CatBoost : Quelles différences ?

Algorithme	Arbres	Variables catégorielles	Précision	Rapidité
XGBoost	Asymmetric trees (Pre-sorting)	 (Experimental)		
LightGBM	Asymmetric trees (GOSS)			
CatBoost	Oblivious trees (Symmetric trees)			

4.4. Gradient Boosting 9/9



Benchmark de performance des méthodes de Boosting

	CatBoost		LightGBM		XGBoost		H2O	
	Tuned	Default	Tuned	Default	Tuned	Default	Tuned	Default
🔗 Adult	0.26974	0.27298 +1.21%	0.27602 +2.33%	0.28716 +6.46%	0.27542 +2.11%	0.28009 +3.84%	0.27510 +1.99%	0.27607 +2.35%
🔗 Amazon	0.13772	0.13811 +0.29%	0.16360 +18.80%	0.16716 +21.38%	0.16327 +18.56%	0.16536 +20.07%	0.16264 +18.10%	0.16950 +23.08%
🔗 Click prediction	0.39090	0.39112 +0.06%	0.39633 +1.39%	0.39749 +1.69%	0.39624 +1.37%	0.39764 +1.73%	0.39759 +1.72%	0.39785 +1.78%
🔗 KDD appetency	0.07151	0.07138 -0.19%	0.07179 +0.40%	0.07482 +4.63%	0.07176 +0.35%	0.07466 +4.41%	0.07246 +1.33%	0.07355 +2.86%
🔗 KDD churn	0.23129	0.23193 +0.28%	0.23205 +0.33%	0.23565 +1.89%	0.23312 +0.80%	0.23369 +1.04%	0.23275 +0.64%	0.23287 +0.69%
🔗 KDD internet	0.20875	0.22021 +5.49%	0.22315 +6.90%	0.23627 +13.19%	0.22532 +7.94%	0.23468 +12.43%	0.22209 +6.40%	0.24023 +15.09%

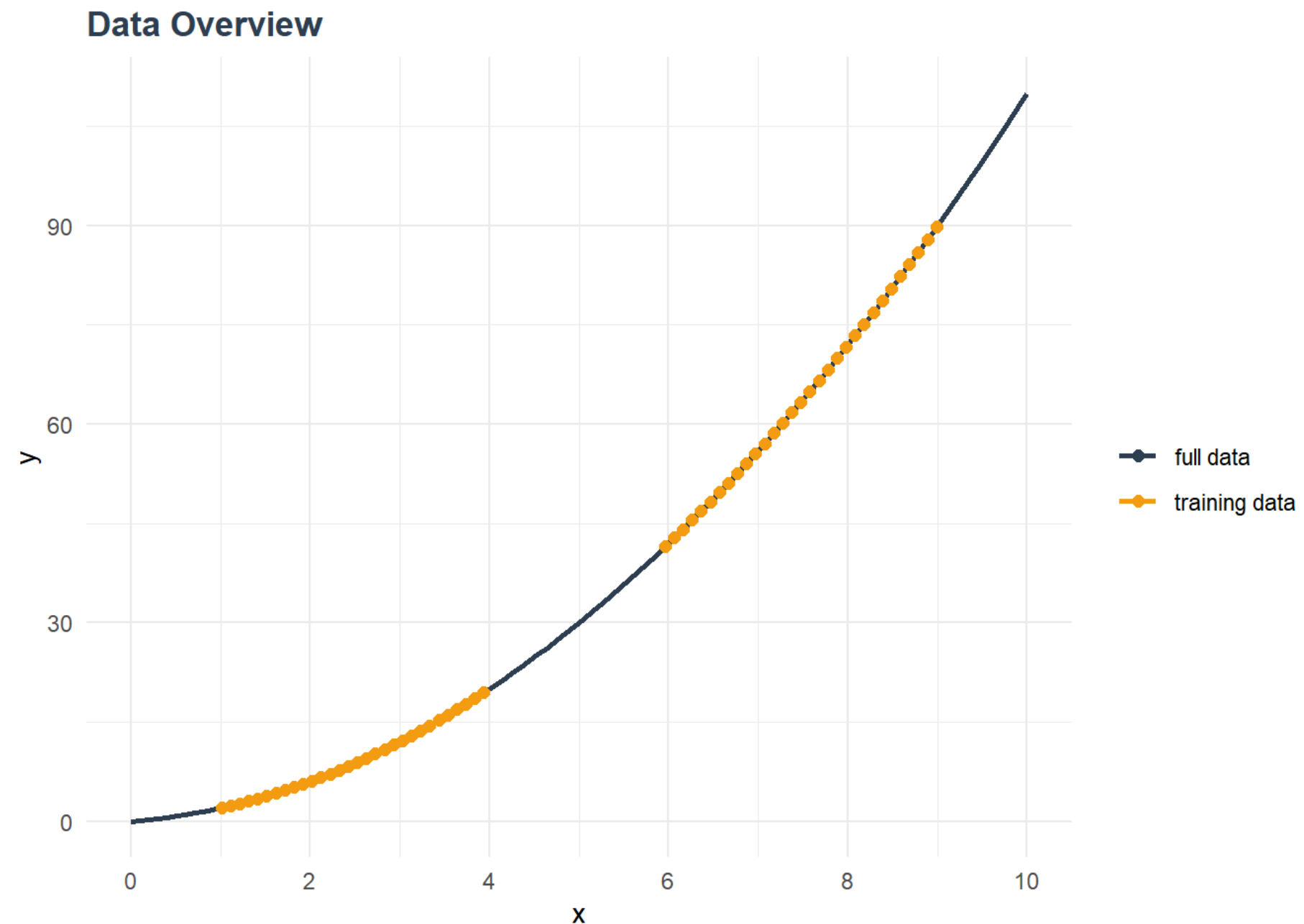
4.5. Limites des RF et Boosting 1/4



Illustration de l'apprentissage avec un modèle linéaire et un modèle basé sur des arbres

Données générées

- X distribué uniformément sur $[0; 10]$
- $Y = x + x^2$
- Données d'entraînement
- Deux sous-ensembles : $[1;4]$ et $[6;9]$
- **Modèle**
- XGBoost avec deux régresseurs faibles différents
- Modèle linéaire
- Arbre de décision



4.5. Limites des RF et Boosting 2/4

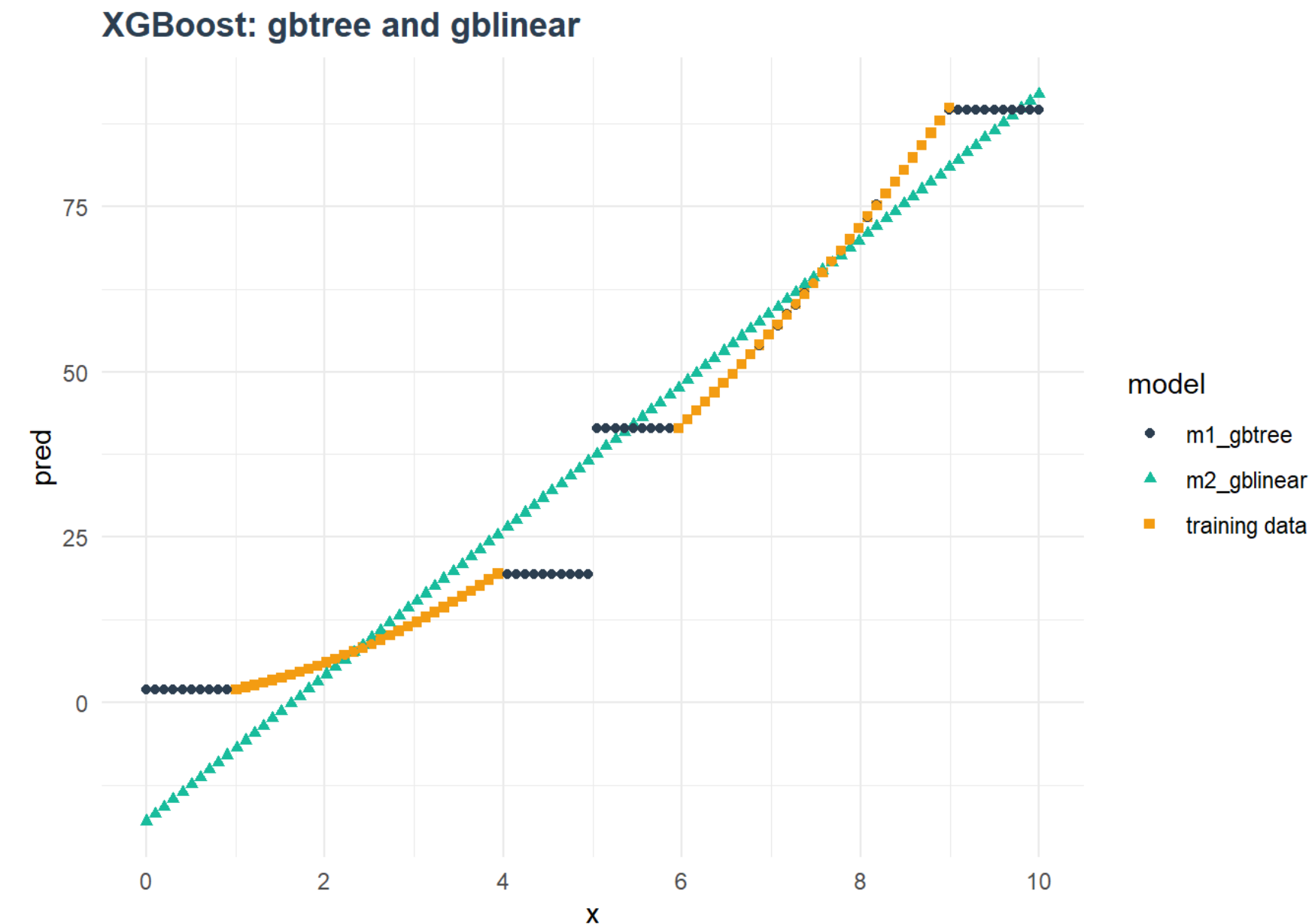


Illustration de l'apprentissage avec un modèle linéaire et un modèle basé sur des arbres

- Performance

model	RMSE (full data)	MAE (full data)	RMSE (train data)	MAE (train data)
m1_gbtree	4.91	2.35	0.05	0.03
m2_gblinear	7.74	6.39	4.50	3.89

- Le modèle linéaire capture mal les données
- Le modèle par arbres ne généralise pas sur les données qu'il ne voit pas lors de l'entraînement du modèle : pas d'inter- ou d'extra-polation.

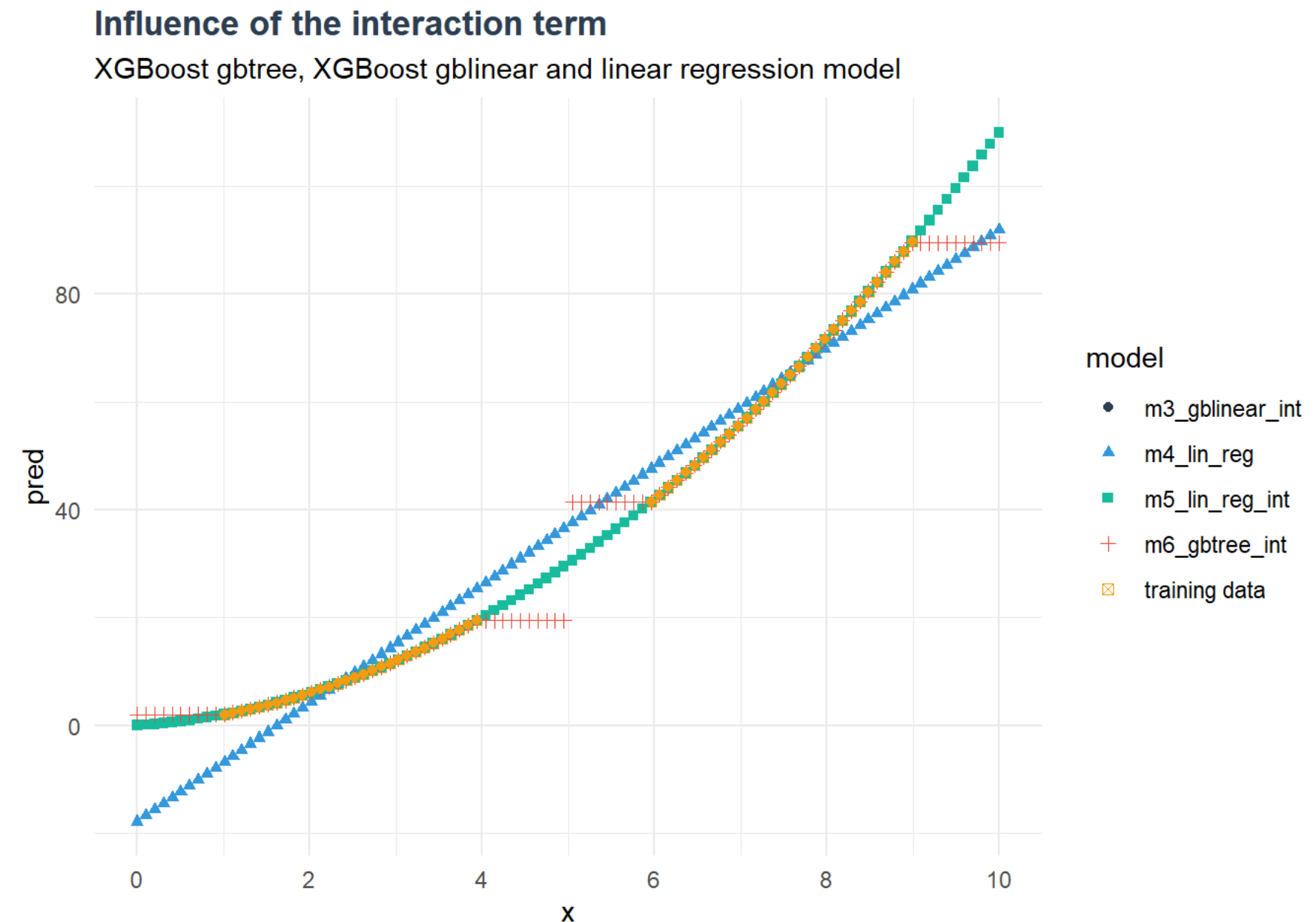


4.5. Limites des RF et Boosting 3/4

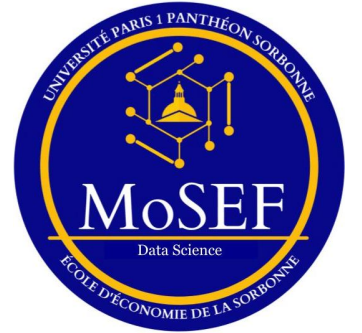


Illustration de l'apprentissage avec un modèle linéaire et un modèle basé sur des arbres

- Introduction d'une variable supplémentaire
- x^2 en complément de x
- Benchmark avec une régression linéaire, contenant ce terme d'interaction supplémentaire
- Le modèle linéaire capture très bien les données et généralise bien sur les données non vues à l'entraînement
- Pas de changement sur le modèle basé sur les arbres : pas d'information apportée permettant de faire un meilleur split



4.5. Limites des RF et Boosting 4/4

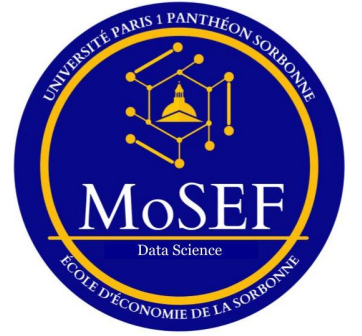


Contraintes et limites des modèles RF et de Boosting

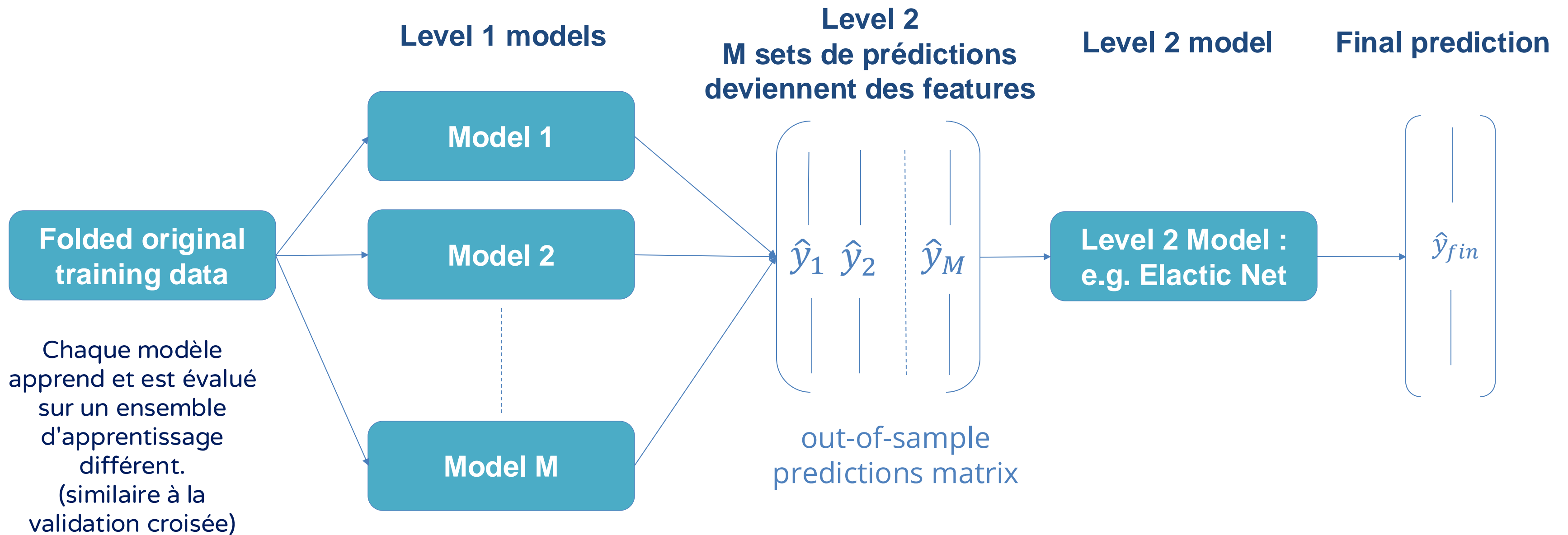
Limites en ce qui concerne leur capacité à extrapoler ou à gérer la non-linéarité ou si les données d'apprentissage ne couvrent pas toujours l'ensemble des données du cas d'utilisation.

- **Modèles par arbres (RF et Boosting)**
- **Bonne représentation** de tous les types de **données non linéaires**, car aucune formule n'est nécessaire pour décrire la relation entre la variable cible et les variables d'entrée.
- Avantage énorme si ces relations et interactions sont inconnues
- Le bagging des forêts aléatoires peut avoir pour effet d'avoir des arbres construits sur des features qui apportent peu d'information (ajout de bruit dans les résultats)
- Pas de capacité d'interpolation ou d'extrapolation dans les domaines non couverts par les données d'apprentissage
- **Modèles linéaires**
- Ne peuvent pas apprendre d'autres relations que les relations linéaires pures
- Si ces interactions supplémentaires peuvent être fournies (rapports entre variables, mise au carré de certaines variables) , les modèles linéaires deviennent très puissants.
- Possibilité d'interpolation et d'extrapolation si **les données d'apprentissage ne couvrent pas toujours l'ensemble des données du cas d'utilisation**

4.6. Performance : Stacking de modèles



Pour améliorer la performance d'un modèle, on peut aussi chercher à combiner plus régresseurs forts (strong learners) en faisant du *stacking*



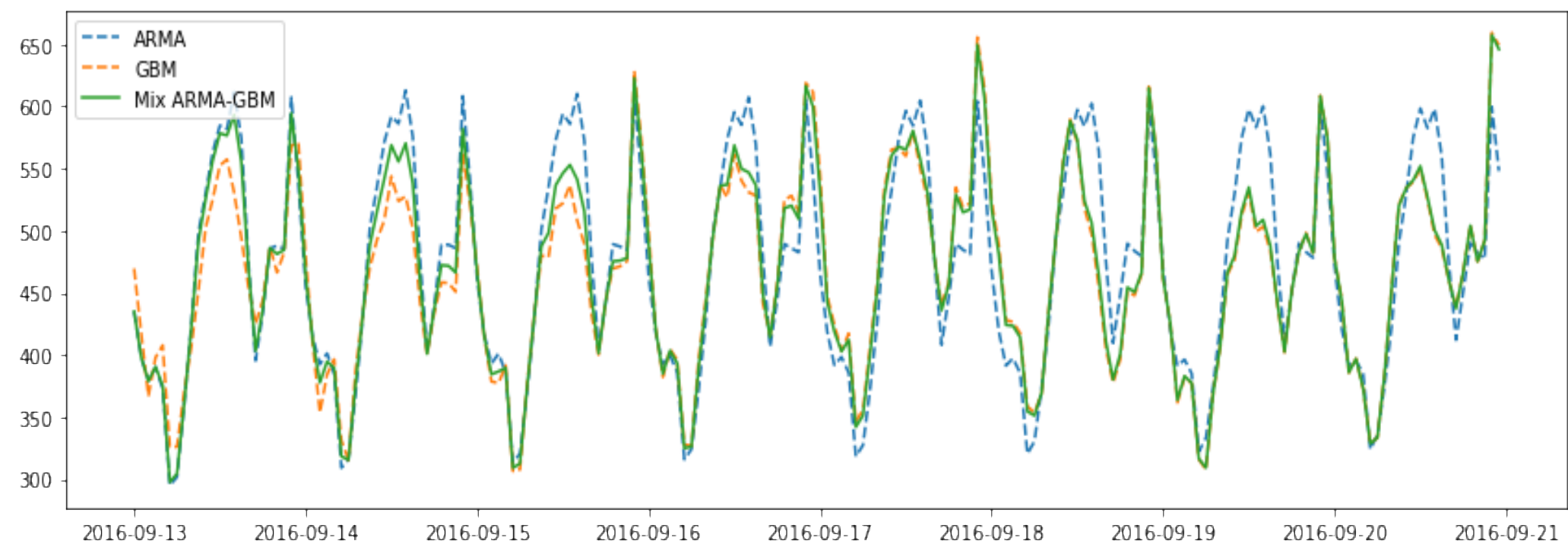
4.6. Performance : Modèles hybrides 1/2



Afin d'améliorer la performance d'un modèle, on peut créer un modèle hybride entre approche machine learning et approche statistique.

Combinaison ARMA / XGBoost

- Exemple illustratif : ne peut pas être généralisé, dépend du jeu de données
- ARMA est meilleur au pour les horizons proches de la date pivot
- $y_{pred} = \alpha(t)y_{pred,ARMA} + (1 - \alpha(t))y_{pred,GB}$
- $\alpha(t) = e^{-\frac{t}{\lambda}}$
- λ est déterminé par optimisation



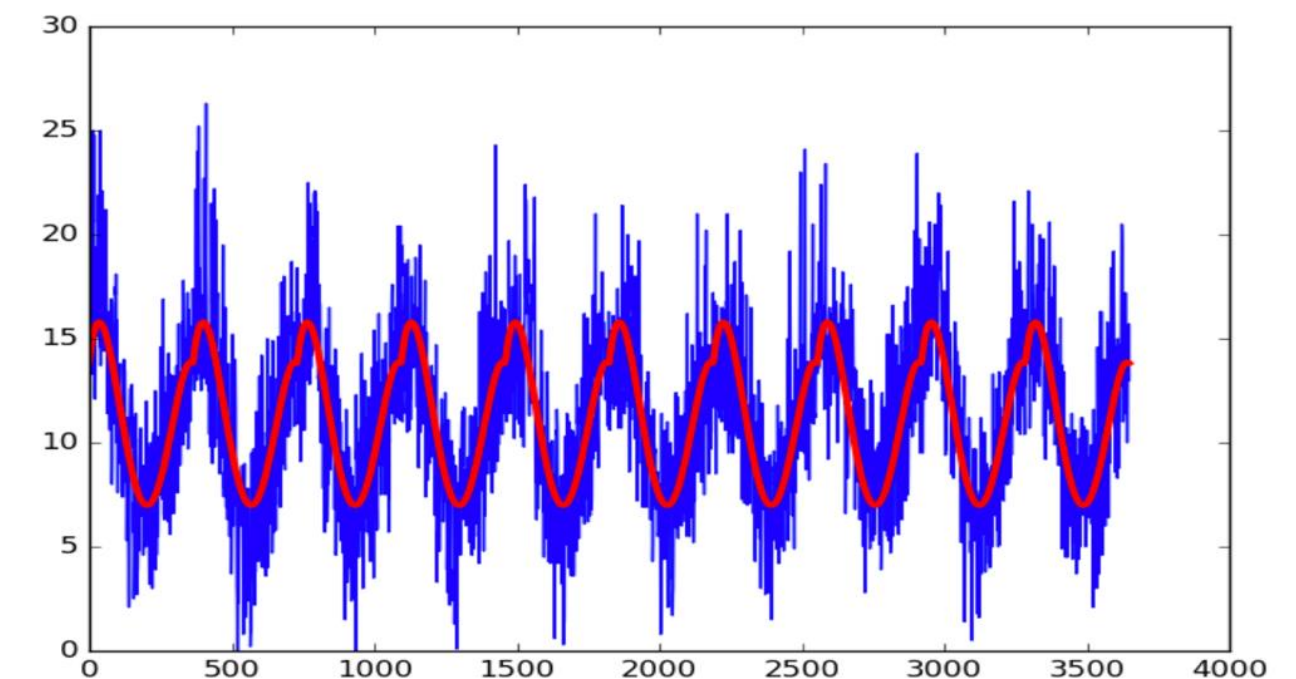
4.6. Performance : Modèles hybrides 2/2



Afin d'améliorer la performance d'un modèle, on peut créer un modèle hybride entre approche machine learning et approche statistique.

Autre idée : modèle additif

- $P = P_s + P_m + N$
 - Partie saisonnière : P_s
 - Partie sensible aux régresseurs externes: P_m
 - Bruit : N
-
- Modèle statistique pour prédire la composante saisonnière P_s
 - Modèle ML ou DL pour prédire la composante résiduelle P_m
 - Inclusion des effets calendaires, météorologiques, etc., non pris en compte dans P_s



4.7. Time to practice



Rendez-vous sans plus attendre sur Python pour la mise en pratique !

