Le prix Zillow : Prévision de la valeur de bien immobilier de Zillow (Zestimate)

Mohamed Ben Yassine - Ellyes Khalfaoui - Imen Rezoug

Université de Lille , Sciences Humaines et Sociales UFR Mathématiques, Informatique, Management, Économie

16 janvier 2020

Plan de la présentation

1. Introduction
2. Exploration des Données
3. Applications des Méthodes :
   - A.C.P.
   - XGBoost

Introduction

Zillow qui représente une plateforme commerciale de propriétés, le concours Zillow prize consiste donc à solliciter la communauté de Data scientists à améliorer les performances de leur algorithme « **Zestimae** » (residual error).

$$logerror = log(Zestimate) - log(SalePrice)$$

le but est de prédire la log error entre le prix prédit par l'algo Zestimate et le prix de vente réel.

Description des données et du problème :

- Le dataset d'entrainement contient les transaction avant le 15 octobre 2016 et des transaction après le dataset de test contient les transactions entre le 15 et 31 décembre,le reste du dataset de test est utilisé pour faire le classement des participants et contient toutes les propriétés entre le 15 octobre et le 15 décembre 2017.

Les fichiers :

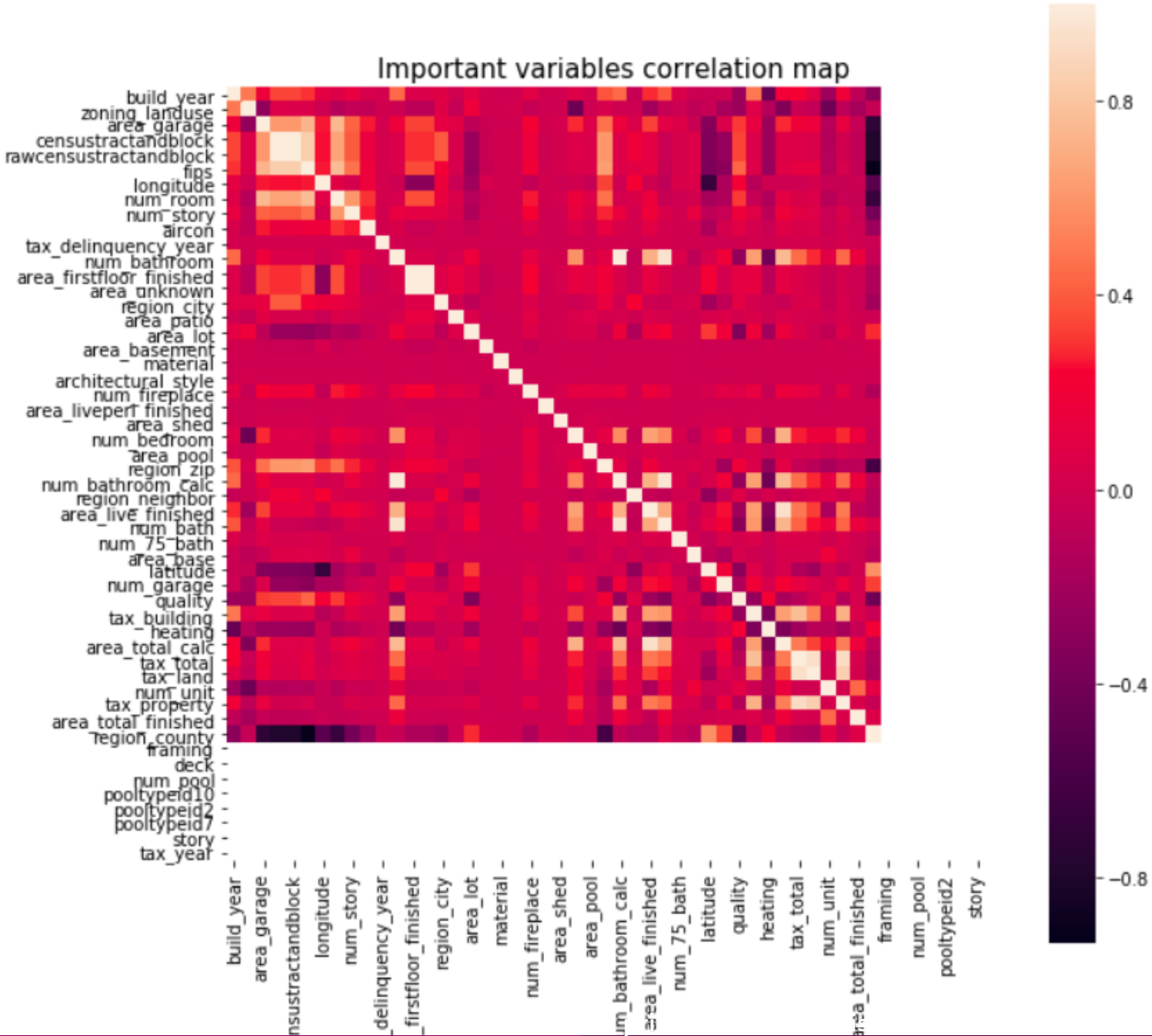
- **properties_2016.csv** : Les propriétés avec leurs caractéristiques pour l'année 2016, avec des propriété de l'année 2017 avec un manque de valeurs dans les caractéristiques.
- **properties_2017.csv** : Les propriétés avec leurs caractéristiques en 2017.
- **train_2016.csv** : l'ensemble d'entrainnement avec les transactions entre le 1er janvier 2016 et le 31 décembre 2016.
- **train_2017.csv** : l'ensemble d'entrainnement avec les transactions entre le 1er janvier 2017 et le 15 septembre 2017.
- **sample_submission.csv** : un échantillion example de la solution à soumettre zillow_data_dictionary.

Statistiques descriptives :

	aircon	architectural_style	area_basement	num_bathroom	\
count	90275.000000	90275.000000	90275.000000	90275.000000	
mean	1.260271	7.000665	1527.612074	2.279474	
std	1.721860	0.146291	20.119940	1.004271	
min	1.000000	2.000000	100.000000	0.000000	
25%	1.000000	7.000000	1528.000000	2.000000	
50%	1.000000	7.000000	1528.000000	2.000000	
75%	1.000000	7.000000	1528.000000	3.000000	
max	13.000000	21.000000	1555.000000	20.000000	

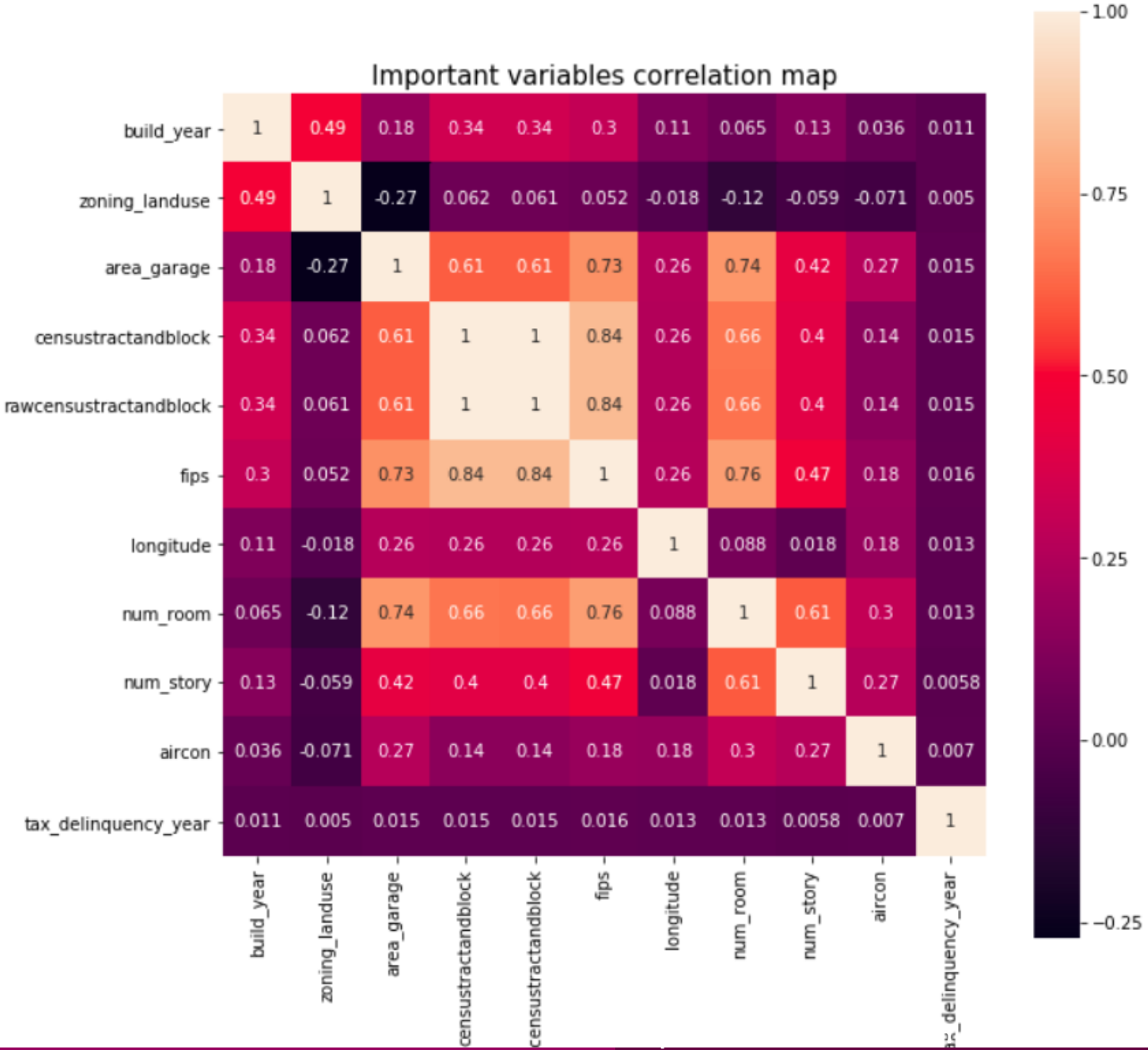
	num_bedroom	framing	quality	num_bathroom_calc	deck	\
count	90275.000000	90275.0	90275.000000	90275.000000	90275.0	
mean	3.031869	4.0	6.088408	2.305168	66.0	
std	1.156436	0.0	1.664972	0.970398	0.0	
min	0.000000	4.0	1.000000	1.000000	66.0	
25%	2.000000	4.0	4.000000	2.000000	66.0	
50%	3.000000	4.0	7.000000	2.000000	66.0	
75%	4.000000	4.0	7.000000	3.000000	66.0	
max	16.000000	4.0	12.000000	20.000000	66.0	

Statistiques descriptives : Matrice de corrélation



ÉTUDE EXPLORATOIRE

Statistiques descriptives : Matrice de corrélation



Applications des Méthodes : A.C.P

Données :

- L'A.C.P. permet d'explorer les liaisons entre variables et les ressemblances entre individus.

Résultats :

–> Visualisation des variables (en fonction de leurs corrélations)

Applications des Méthodes : A.C.P

Choix de nombre d'axe factoriels à retenir

Les valeurs permettent d'effectuer un choix du nombre de composantes principales à retenir pour l'interprétation. Le choix du nombre d'axes à interpréter se fait sur la base de règles.

- La règle de Kaiser.

Applications des Méthodes : A.C.P

Étude des variables expliquqnte mieux un axe donnée

La détermnation des variables expliquant chacun des axes est realisée en examinant (table des valeurs propres) qui sont elle-même reliées à leur contribution.

Applications des Méthodes : A.C.P

Normalisation des données La première étape pour appliquer l'analyse en composante principale est de normaliser les variables. Pour faire cela sur python il existe la librairie suivante :

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data =scaler.fit_transform(X)
```

Applications des Méthodes : A.C.P

	aircon	architectural_style	area_basement	num_bathroom
count	9.027500e+04	9.027500e+04	9.027500e+04	9.027500e+04
mean	-4.565104e-17	5.284182e-16	5.603021e-15	1.760713e-16
std	1.000006e+00	1.000006e+00	1.000006e+00	1.000006e+00
min	-1.511579e-01	-3.418318e+01	-7.095548e+01	-2.269792e+00
25%	-1.511579e-01	-4.543269e-03	1.928077e-02	-2.782868e-01
50%	-1.511579e-01	-4.543269e-03	1.928077e-02	-2.782868e-01
75%	-1.511579e-01	-4.543269e-03	1.928077e-02	7.174659e-01
max	6.818087e+00	9.569563e+01	1.361240e+00	1.764526e+01

	num_bedroom	framing	quality	num_bathroom_calc	deck
count	9.027500e+04	90275.0	9.027500e+04	9.027500e+04	90275.0
mean	5.399416e-17	0.0	2.110180e-16	-4.250269e-17	0.0
std	1.000006e+00	0.0	1.000006e+00	1.000006e+00	0.0
min	-2.621751e+00	0.0	-3.056169e+00	-1.344990e+00	0.0
25%	-8.922893e-01	0.0	-1.254327e+00	-3.144786e-01	0.0
50%	-2.755836e-02	0.0	5.475152e-01	-3.144786e-01	0.0
75%	8.371726e-01	0.0	5.475152e-01	7.160326e-01	0.0
max	1.121394e+01	0.0	3.550585e+00	1.823472e+01	0.0

	area_firstfloor_finished	...	build_year	num_story
count	9.027500e+04	...	9.027500e+04	9.027500e+04
mean	5.694574e-17	...	9.970817e-16	-1.748907e-16
std	1.000006e+00	...	1.000006e+00	1.000006e+00
min	-3.563066e+00	...	-3.520444e+00	-3.148669e-01
25%	-1.766592e-01	...	-6.507304e-01	-3.148669e-01
50%	-1.766592e-01	...	2.449637e-02	-3.148669e-01
75%	-1.766592e-01	...	7.841265e-01	-3.148669e-01
max	2.964825e+01	...	1.965773e+00	9.091027e+00

	tax_building	tax_total	tax_year	tax_land	tax_property
count	9.027500e+04	9.027500e+04	90275.0	9.027500e+04	9.027500e+04
mean	1.550561e-17	-4.718586e-17	0.0	-6.513144e-18	-1.930232e-16
std	1.000006e+00	1.000006e+00	0.0	1.000006e+00	1.000006e+00
min	-8.606183e-01	-8.247743e-01	0.0	-6.949215e-01	-8.677957e-01
25%	-4.707308e-01	-4.661345e-01	0.0	-4.896602e-01	-4.549413e-01
50%	-2.311313e-01	-2.068912e-01	0.0	-2.131690e-01	-2.107475e-01
75%	1.450821e-01	1.494338e-01	0.0	1.675003e-01	1.340778e-01
max	4.679389e+01	4.918615e+01	0.0	6.047979e+01	4.620060e+01

	tax_delinquency_year	censustractandblock	logerror

Applications des Méthodes : A.C.P

- **Détermination du nombre des axes principaux**
  La deuxième étape consiste à calculer les valeurs propres de la matrice de corrélation, La fonction PCA de la classe decomposition de la librairie sklearn sur Python fait ce calcul. On peut accéder à ces valeurs avec l'attribut suivant :
  *pca.explained_variance_ratio_*

```
variance expliquée:
[7.39964948 5.7401271  2.94522339 2.19039121 1.9870038  1.58269494
 1.28940268 1.1579138  1.05532472 1.0331848  1.00648635 1.00068814
 1.0001975  0.99985392 0.9978238  0.9893985  0.98417918 0.96996801
 0.94684986 0.90307072 0.84014942 0.79146738 0.74348514 0.72816178
 0.69044025 0.61626513 0.59612441 0.49603275 0.44809932]
```

- La première valeur propre obtenue explique la quasi-totalité de la variance.
- La décroissance des valeurs propres est ici très rapide

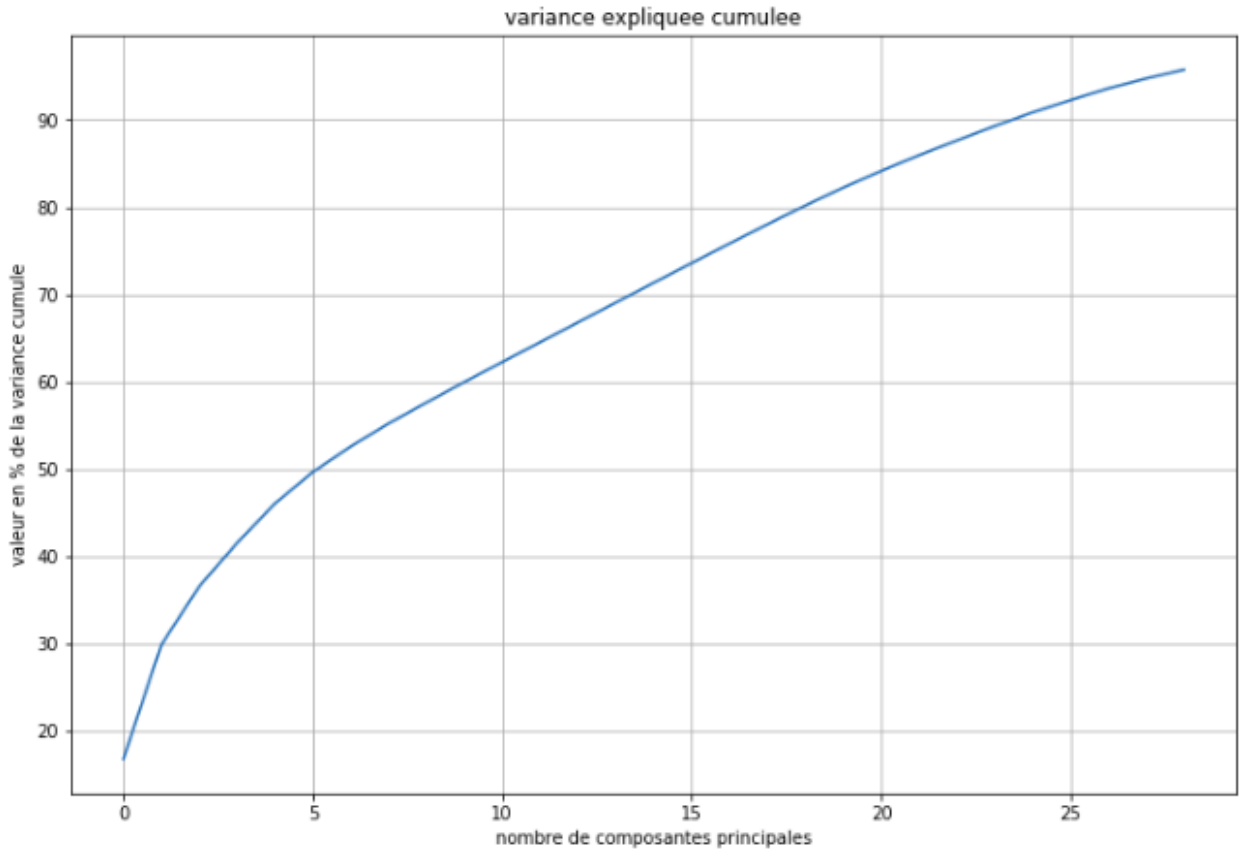
Applications des Méthodes : A.C.P

- Le pourcentage de la variable expliquée par chaque variable est donnée par le code suivant :
  pca.explained_variance_ratio_

```
pourcentage de la variance expliquée
[0.16817168 0.29862743 0.36556346 0.41534443 0.46050303 0.4964729
 0.52577713 0.55209301 0.57607735 0.59955852 0.62243292 0.64517553
 0.667907   0.69063066 0.71330818 0.73579422 0.75816164 0.78020608
 0.80172512 0.82224919 0.84134325 0.85933091 0.87622808 0.892777
 0.90846862 0.92247446 0.93602257 0.9472959  0.95747984]
```

Applications des Méthodes : A.C.P

- .Le graphe suivant présente le pourcentage de la varaince expliquée cumulé des variables.



Applications des Méthodes : XGBoost

- **Extreme Gradient Boosting** (**XGBoost**) , est un nouveau classificateur basé sur un ensemble d'arbres de classification et de régression.
- L'algorithme **XGBoost** est un algorithme ensembliste qui agrège des arbres. À chaque itération, le nouvel arbre apprend de l'erreur commise par l'arbre précédent.
- Dans XGBoost, les arbres sont optimisés en utilisant le gradient boosting.
  Soit la sortie d'un arbre :

$$f(x) = w_q(x_i)$$

où x est le vecteur d'entrée et w_q est le score du feuille q.

Applications des Méthodes : XGBoost

- Le résultat d'un ensemble de K arbres sera :

$$y_i = \sum_{k=1}^{k} f_k(x_i)$$

L'algorithme XGBoost tente de minimiser la fonction objectif suivante J au point t :

$$J(t) = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{t-1} + f_t(x_i)) + \sum_{i=1}^{t} \Omega(f_i)$$

—> Le premier terme contient la fonction de perte de train L (par exemple, moyenne erreur au carré) entre la classe réelle y et la sortie $\hat{y}$ pour les n échantillons.
—> Le deuxième terme est le terme de régularisation, qui contrôle la complexité du modèle et permet d'éviter le sur-apprentissage.

Applications des Méthodes : XGBoost

- La complexité est définie comme suit :

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

où T est le nombre de feuilles, γ est la pseudo-régularisation hyperparamètre, en fonction de chaque jeu de données et λ est la norme L2 pour les poids des feuilles.

Applications des Méthodes : XGBoost

```
variance expliquée:
[7.39964948 5.7401271  2.94522339 2.19039121 1.9870038  1.58269494
 1.28940268 1.1579138  1.05532472 1.0331848  1.00648635 1.00068814
 1.0001975  0.99985392 0.9978238  0.9893985  0.98417918 0.96996801
 0.94684986 0.90307072 0.84014942 0.79146738 0.74348514 0.72816178
 0.69044025 0.61626513 0.59612441 0.49603275 0.44809932]
```

MERCI DE VOTRE ATTENTION ! !