Détection d'anomalies dans les séries temporelles

Maxime LAFONT-TREVISAN, Michele BONA, Rihab KHLIFI, Héloïse VIOSSAT
Polytech Sorbonne

Le projet

L'entreprise à l'origine du projet est Prévision.io, créée en 2016, qui cherche à démocratiser l'accès à l'Intelligence Artificielle et l'utilisation de l'apprentissage automatique. Elle propose des plateformes d'analyses de données pour des scientifiques des données ainsi que pour des personnes inexpérimentées.
Le but de notre projet est de mettre en oeuvre des méthodes pertinentes afin de détecter des anomalies dans des séries temporelles, c'est à dire : "Les observations qui dérivent à un tel point des autres observations que l'on peut suspecter qu'elles ont été générées par un mécanisme différent" — Hawkins (1980).
Il existe de nombreuses méthodes réalisant cette tâche. Notre objectif est de les analyser et de sélectionner les meilleures afin de concevoir une méthode efficace de détection d'anomalies dans les séries temporelles.

Méthodologie

Afin de réaliser ce projet nous avons commencé par une phase de recherche sur les multiples méthodes de détection d'anomalies dans les séries temporelles existantes qui nous à permis de faire une première sélection de celles-ci. Nous avons ensuite testé les méthodes sélectionnées et retenu les meilleures. Pour ce faire, nous avons suivie la méthodologie suivante de la Figure 1. Suite à la phase de test des méthodes, nous avons choisi la méthode du test de Grubbs, Isolation Forest et Prophet que nous détaillerons par la suite.

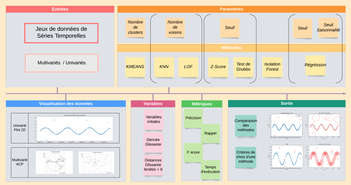

Figure1: Etapes de la méthodologie du projet

Test des méthodes

Pour chaque méthode testée nous calculons sa précision, c'est à dire le nombre d'anomalies correctement détectées par rapport aux nombre d'anomalies total; son rappel, c'est à dire le nombre d'anomalies correctement détectées par rapport aux nombre d'anomalies détectées; son F-score, qui combine les deux métriques précédentes et son temps d'exécution.

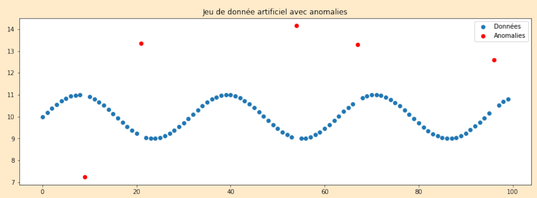

Figure2: Détection d'anomalies à l'aide d'Isolation Forest sur le jeu de données artificiel 1

Chaque méthode permet de détecter plus ou moins d'anomalies, plus ou moins correctement. Les performances d'une méthode varient beaucoup selon le jeu de données.

Isolation Forest

Nous avons utilisé la méthode Isolation Forest de la bibliothèque *sklearn*. Cette méthode permet de détecter les anomalies dans un jeu de données de manière non supervisée. Elle sépare notre jeu de données en fonction d'une valeur seuil choisie aléatoirement entre le maximum et le minimum de la série. On classe alors toutes les valeurs du jeu de données dans un arbre de décision en fonction de ce seuil, toutes les valeurs inférieures seront à gauche et celles supérieures, à droite. On réitère ensuite l'opération suivant une autre valeur seuil dans chaque sous-arbre créé, jusqu'à avoir isolé chaque donnée. Enfin les valeurs qui ont été les plus simples à isoler (les plus petites branches de l'arbre) sont considérées comme des anomalies.

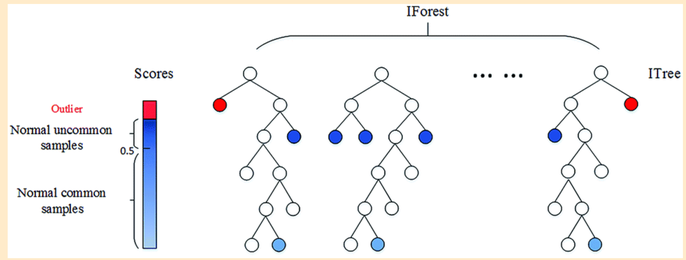

Figure3: Fonctionnement d'Isolation Forest

Test de Grubbs

Le test de Grubbs est une méthode statistique paramétrique qui consiste à tester une par une les valeurs les plus éloignées de la moyenne de la série afin de voir s'il s'agit d'une anomalie. Lorsque l'on trouve une valeur qui n'est pas considérée comme telle, la méthode s'arrête.
Pour chaque valeur x_i testée, on calcule son z-score :

$$Zscore(x_i) = \frac{x_i - \mu}{\sigma} \quad (1)$$

Où μ et σ sont respectivement la moyenne et l'écart-type de notre variable.
Puis on le compare à une valeur critique G, calculée comme suit :

$$G = \frac{N-1}{\sqrt{N}} \sqrt{\frac{(t_{\frac{\alpha}{2N}, N-2})^2}{N-2+(t_{\frac{\alpha}{2N}, N-2})^2}} \quad (2)$$

Où N est la taille du jeu de donnée et α la sensibilité du test.
Si le z-score est supérieur à G, alors la valeur est considérée comme une anomalie.

Prophet

Alibi-detect est une librairie de détection d'anomalies utilisée surtout dans le cas des séries temporelles. Cette librairie contient de nombreuses méthodes de détection. Nous utiliserons la méthode de détection Prophet. Prophet est une méthode de prévision des données dans des séries temporelles basée sur un modèle additif de régression linéaire non paramétrique. Dans la méthode Prophet, la variable à expliquer est ajustée à partir des données fournies et non pas, comme pour les autres méthodes de régression, selon une fonction d'estimation déterminée à l'avance.

Création de nouvelles variables

Pour accentuer les différences et les similarités entres les valeurs de notre série temporelle, nous avons créé de nouvelles variables qui dépendent de nos valeurs. Celles-ci nous servent à améliorer l'efficacité de nos méthodes de détections qui, appliquées sur ces variables, sont bien plus précises. Ainsi pour le test de Grubbs nous avons pris pour chaque valeur la médiane des différences des valeurs autour (les 4 précédentes et 4 suivantes), et pour Isolation Forest la différence avec la valeur d'avant et la différence avec la valeur d'après (on applique ensuite la méthode sur les deux variables).

date	vals	d_avant	d_apres
2020-09-01 00:00:00	10.000000	0.000000	-0.149438
2020-09-01 01:00:00	10.149438	0.149438	-0.146082
2020-09-01 02:00:00	10.295520	0.146082	-0.139445
2020-09-01 03:00:00	10.434966	0.139445	-0.129677
2020-09-01 04:00:00	10.564642	0.129677	-0.116996

Figure4: Fenêtre glissante de taille 2, différence avec les valeurs d'avant et après

Résultats

Notre méthode finale consiste à combiner les trois méthodes précédentes en considérant qu'une valeur détectée comme anormale par les trois méthodes est une anomalie "sûre"; par seulement deux méthodes, une anomalie "simple"; et par une méthode au plus, n'en est pas une.

Détection.py		Grubb naif	GrubbDérivé	IsolationForest naif	IsoDérivé	Prophet	Combinaison 1	Combinaison 2
Jeux de données	Performances	alpha = 0.05	alpha = 0.05	n=100	n = 100	seuil = 0.99	avec Prophet	sans Prophet
artificial_data_1	Précision	100%	100%	100%	100%	100%	83.3%	100%
	Rappel	100%	100%	100%	100%	80%	100%	100%
	fscore	100%	100%	100%	100%	88.8%	90.9%	100%
	Temps d'exécution	0.0028	0.035	0.209	0.448	3.214	3.725	0.401
artificial_data_2	Précision	100%	100%	5.56%	100%	100%	100%	100%
	Rappel	0%	60%	60%	80%	0%	60%	80%
	fscore	0%	75%	10.18%	88.9%	0%	75%	88%
	Temps d'exécution	0.00086	0.046	0.196	0.474	2.961	3.511	0.515
artificial_data_3	Précision	100%	100%	38.46%	80%	100%	100%	80%
	Rappel	0%	100%	6.25%	100%	50%	100%	100%
	fscore	0%	100%	47.62%	88.9%	66.7	100%	88.9%
	Temps d'exécution	0.0012	0.038	0.216	0.462	3.421	4.037	0.604
artificial_data_4	Précision	100%	100%	0%	88.8%	100%	100%	88.8%
	Rappel	0%	60%	0%	80%	0%	60%	80%
	fscore	0%	75%	0%	84.2%	0%	75%	84.2%
	Temps d'exécution	0.00092	0.072	0.217	0.623	3.057	3.683	0.586
real_time_serie_energy	Précision	100%	93.9%	4.48%	64%	100%	93.9%	71.1%
	Rappel	0	93.9%	66.67%	97%	87.9%	93.9%	97.00%
	fscore	0	93.9%	7.82%	77.1%	93.5	93.9%	82%
	Temps d'exécution	0.0014	0.633	0.222	1.181	18.39	19.4	1.965

Figure5: Résultats de nos méthodes

Voici les résultats obtenus sur un jeu de données réel provenant de l'entreprise EDF et représentant une consommation d'énergie :

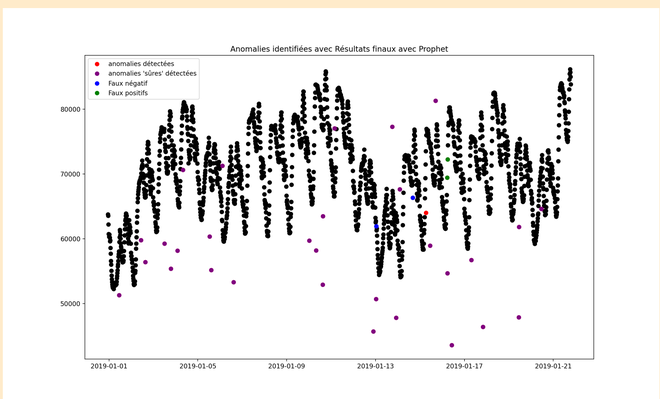

Figure6: Affichage des résultats de la méthode finale sur le jeu de données réel EDF