

Automatisation de l'extraction et de l'analyse de données électriques de centrales photovoltaïques

Izem El Mourabit

▶ **To cite this version:**

Izem El Mourabit. Automatisation de l'extraction et de l'analyse de données électriques de centrales photovoltaïques. Gestion et management. 2021. dumas-03546607

HAL Id: dumas-03546607

https://dumas.ccsd.cnrs.fr/dumas-03546607

Submitted on 28 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License



Mémoire de stage de M1

Automatisation de l'extraction et de l'analyse de données électriques de centrales photovoltaïques.

Présenté par : EL MOURABIT Izem

Entreprise d'accueil :
Ener-Pacte
23, boulevard Jules Favre
69006 Lyon

Date de stage : du 06/04/21 au 30/08/21

Master 1 – Formation Initiale
Master Management des Systèmes d'Information
Sans spécialité
2020 - 2021

Avertissement :

Grenoble IAE, au sein de l'Université Grenoble Alpes, n'entend donner aucune approbation ni improbation aux opinions émises dans les mémoires des candidats aux masters en alternance : ces opinions doivent être considérées comme propres à leur auteur.

Tenant compte de la confidentialité des informations ayant trait à telle ou telle entreprise, une éventuelle diffusion relève de la seule responsabilité de l'auteur et ne peut être faite sans son accord.

RESUME

Ce stage s'est déroulé dans le cadre de mon Master 1 en Management des Systèmes d'Information à Grenoble IAE. Il a duré 5 mois (du 6 avril 2021 au 30 août 2021) et a eu lieu au sein d'Ener-Pacte, une petite entreprise de 15 personnes qui sécurise le patrimoine photovoltaïque de clients particuliers. Mon objectif durant ce stage a été d'automatiser deux processus primordiaux pour l'entreprise : l'extraction et l'analyse de données électriques de centrales photovoltaïques. Les opérationnels ont besoin de ces données pour observer l'évolution des centrales dans le temps pour quantifier les pertes de production possibles afin d'agir en conséquence. J'ai développé des programmes Python qui permettent d'extraire des données sur plusieurs années en quelques minutes et ce, de manière automatique. Cela aura été très long à faire « à la main » (voir impossible pour certaines plateformes web). Une fois ces données extraites, j'ai développé un algorithme qui analyse le rendement des centrales. Il se base sur le package Prophet et détecte des tendances, des ruptures de tendance, un possible effet de la saisonnalité et permet de prédire le rendement futur des centrales. Cette méthode d'analyse a l'avantage d'être automatique, objective et prédit les rendements futurs plus précisément qu'une simple régression linéaire utilisé jusque-là (~25% plus précise sur des données non linéaires, et équivalente sur des données linéaires). Mes programmes permettent donc aux opérationnels d'éviter de passer du temps sur des tâches d'extraction longues et fastidieuses et analyse les données de manière automatique et ce, de manière plus précise. (253 mots)

MOTS CLÉS : centrale photovoltaïque / onduleur / extraction de données / web-scraping / automatisation / tendance / rupture de tendance / modèle statistique

SUMMARY

This internship took place as part of my Master 1 in Information Systems Management at Grenoble IAE. It lasted 5 months (from April 6, 2021 to August 30, 2021) and took place within Ener-Pacte, a small company of 15 people which secures the photovoltaic assets of private customers. My objective during this internship was to automate two processes that are essential for the company: the extraction and the analysis of electrical data from photovoltaic power plants. The operational staff needs this data to observe the evolution of the power plants though years to quantify the possible production losses to act accordingly. I have developed Python programs that can extract data over several years in a few minutes automatically. This would have been very time consuming to do it "by hand" (even impossible for some web platforms). Once these data are extracted, I developed an algorithm that analyzes the performance of the solar plants. It is based on the Prophet package and detects trends, trend breaks, possible seasonality effects and allows to predict the future performance of the plants. This analysis method has the advantage of being automatic, objective and predicts future performance more accurately than a simple linear regression that was used until now (~25% more accurate on non-linear data, and equivalent on linear data). My programs allow operational staff to avoid spending time on long and tedious extraction tasks and analyze the data automatically and more accurately. (238 words)

KEY WORDS: solar plant / inverter / data mining / web-scraping / automation / trend / trend break / statistical model

Table des matières

RESUME **4**
SUMMARY **4**
REMERCIEMENTS **5**
TABLE DES MATIERES **6**
GLOSSAIRE **7**
INTRODUCTION **8**
CONTEXTE DANS LEQUEL ENER-PACTE A ETE CREEE 8
ROLE D'ENER-PACTE 8
CONTEXTE DE MES MISSIONS 9
I. EXTRACTION ET NETTOYAGE DE DONNEES **10**
A. PLATEFORME METEOCONTROL.DE : AUTOMATISATION D'UNE TACHE LONGUE ET FASTIDIEUSE 10
Présentation du problème rencontré 10
Solution proposée 12
Résultats obtenus 14
B. RECUPERATION DE DONNEES SUR DEUX AUTRES PLATEFORMES 17
solarmax.com 17
solarlog-portal.fr 18
CONCLUSION 20
II. ANALYSE DE DONNEES SUR DES SERIES TEMPORELLES **21**
PRESENTATION DU PROBLEME RENCONTRE 21
Détection de tendances et rupture de tendance 21
Prédiction du rendement futur 21
SOLUTION PROPOSEE 22
Prophet : un modèle de prédiction sur des données temporelles 22
Fonctionnement 23
Formatage des données 25
Ajustement des paramètres du modèle 26
RESULTATS OBTENUS 27
Cross-validation 30
Conclusion 32
CONCLUSION GENERALE **32**
BILAN DU STAGE 32
PROLONGEMENT DU STAGE 34
BIBLIOGRAPHIE **35**

GLOSSAIRE

API (Application Programming Interface) : une interface qui permet à deux programmes informatiques de communiquer / échanger de l'information.

Centrale (ou centrale PV) : c'est une centrale photovoltaïque, c'est-à-dire un ensemble de panneaux photovoltaïques placés sur des toitures (souvent de toitures de fermes agricoles dans le cas des clients d'Ener-Pacte).

Dataframe : type de format de données (utilisé par la bibliothèque Python *pandas*) qui présente les données en 2 dimensions : des colonnes contenant de différents types de donnée (*string*, *integer*, *float*, etc.) et un indice associé à chaque ligne.

Grandeur : dimension physique mesurée par un onduleur donnant des indications sur le fonctionnement de la centrale PV (ex. production (en Watt/h), puissance de courant alternatif (Watt), tension de courant alternatif (Volt)).

Onduleur : dispositif électronique de puissance qui convertit le courant continu provenant des panneaux photovoltaïques en courant alternatif (pour aller sur le réseau EDF).

Opérationnels : personnes en charge de faire l'audit des (nouvelles) centrales PV chez Ener-Pacte. Ils se rendent sur place, discutent avec les propriétaires de la centrale, récupèrent les informations et font des analyses avec les données récupérées (des onduleurs, de la thermographie, etc.) afin de définir le contrat et les actions à faire en cas de signature.

Requête HTTP : action effectuée sur une ressource web identifiée par une URL. Il existe différentes méthodes de requête HTTP comme GET ou POST (Nielsen et al., 1996). GET demande au serveur de renvoyer une représentation d'une ressource donnée. POST en revanche, demande au serveur d'accepter des valeurs encapsulées dans la requête comme par exemple, un identifiant et un mot de passe afin d'accéder à un contenu sécurisé.

URL (Uniform Resource Locator) **:** adresse d'une ressource internet (souvent un site internet) qui spécifie sa localisation sur un réseau informatique et une manière de la retrouver.

Web scraping : extraction de données de pages internet soit via des requêtes http, soit directement via un navigateur internet. Ce processus est généralement automatisé via un robot informatique. L'idée est de récupérer des données d'un site web afin de pouvoir les utiliser plus tard pour les analyser par exemple.

Introduction

Contexte dans lequel Ener-Pacte a été créée

Le changement climatique est l'enjeu majeur du XXIe siècle. Depuis l'avènement de l'ère industrielle, les émissions de CO_2 (et autres gaz à effet de serre) ont augmenté de manière exponentielle (Friedlingstein et al., 2020) causant un réchauffement de la température à la surface de la Terre (terres émergées et océans) de 1°C par rapport aux niveaux préindustriels (et devraient atteindre +1.5°C d'ici 2030 à 2050 d'après le cinquième rapport du GIEC de 2013 (Groupe d'experts intergouvernemental sur l'évolution du climat et al., 2013). Cette hausse de la température a de multiples conséquences néfastes pour le système Terre : réchauffement et acidification des océans, fonte des glaces, élévation du niveau de la mer, élévation de la température moyenne, augmentation des extrêmes de chaleurs, épisodes de fortes précipitations, etc. Il est urgent pour les États d'agir et de diminuer drastiquement leurs émissions de gaz à effet de serre et 190 d'entre eux ont signé l'accord de Paris en décembre 2015 qui fournit un cadre juridique mondial visant à limiter le réchauffement climatique à 1.5°C.

Dans ce contexte de crise climatique, l'État français encourage le développement des énergies renouvelables, dont l'énergie solaire. Depuis 2007 et la libéralisation du marché de l'énergie, chaque particulier peut consommer et produire sa propre énergie. Et afin de garantir une rentabilité des investissements photovoltaïques, l'État a mis en place des obligations d'achat solaire. Il s'agit d'un contrat d'une durée de 20 ans qui oblige EDF à racheter la production électrique des particuliers à un tarif préférentiel. Cela a eu pour conséquence de stimuler les ventes d'applications photovoltaïques de 2007 et 2010. Après ce boom de la fin des années 2000, certains constructeurs de panneaux PV et certaines sociétés d'installation et de maintenance ont fait faillite et ne peuvent plus assurer de service de maintenance en cas de problème. Les propriétaires de panneaux PV se retrouvent alors sans expert pour les aider à gérer au mieux leur centrale.

Rôle d'Ener-Pacte

Ener-Pacte est une petite entreprise de 15 personnes, créée en 2016, qui propose aux possesseurs de centrales photovoltaïques (souvent des agriculteurs, car ils possèdent de grandes surfaces exploitables soit au sol mais également en toitures sur hangars) de faire un audit/état des lieux de leur centrale. A l'issue de l'audit, Ener-Pacte estime la qualité de la centrale et proposera au futur client de lui garantir un rendement sur le reste de la durée de contrat d'obligation d'achat avec EDF et d'optimiser les performances de la centrale. En effet, il faut savoir qu'une centrale PV perd au mieux 0.5% de rendement par an (à cause du vieillissement du matériel) mais perd bien souvent quelques pourcents par an (à cause d'un mauvais entretien de la centrale) et fonctionne rarement au maximum de sa capacité (encrassement des panneaux, ombrage, mauvaise gestion de fortes chaleurs, de cellules PV mortes, etc.). Une fois le contrat signé, l'entreprise récolte des informations électriques de la centrale en permanence et s'occupe de la maintenance et des travaux à effectuer. Le propriétaire de la centrale n'a plus à se soucier de sa centrale. Ener-Pacte se rémunère sur les rendements qui dépassent le rendement garanti à la signature du contrat : Ener-Pace récupère 80% du surplus de rendement et les 20% restants sont reversés au client (et elle paye la différence au propriétaire de la centrale si celle-ci produit finalement moins que prévu).

Actuellement l'entreprise surveille une quarantaine de centrales réparties dans la moitié sud de la France. Pour superviser en temps réel ces centrales, Ener-Pacte se base sur les données onduleurs de ces centrales (un onduleur renvoie par exemple la puissance en Watt (W), heure par heure issue des panneaux auxquels il est relié). Cela permet aux opérationnels d'observer de potentielles anomalies de production sur certaines centrales et de lancer une opération de maintenance si besoin.

En plus du monitoring en temps réel, les opérationnels ont également besoin des données historiques de rendement des centrales. Premièrement, cela leur permet de voir l'évolution « macroscopique » de celles-ci : si une centrale connaît un déclin rapide de rendement, il est important de pouvoir le repérer et de connaître la date à laquelle cela est arrivé et de trouver la cause du problème (et donc potentiellement la solution). Deuxièmement, les données historiques permettent aux opérationnels de quantifier les pertes de production de la centrale, et donc de faire un plan d'action pertinent pour équilibrer les gains potentiels de production avec les investissements nécessaires pour améliorer le rendement de la centrale.

Le lecteur l'aura compris, une des problématiques des équipes opérationnelles est donc l'acquisition et l'exploitation de ces données onduleur (les données historiques d'une part et les données en temps réel, d'autre part). Et avec un parc de centrales qui tend à grandir rapidement (+15 depuis le début de l'année scolaire, et une prévision de +50 d'ici à la fin de l'année) il devient primordial voire impératif pour Ener-Pacte d'automatiser la récupération et l'analyse de ces données onduleurs qui, à terme, ne pourront plus être faites « à la main » par les opérationnels.

Contexte de mes missions

Pour vingt-cinq de ses centrales, Ener-Pacte utilise un intermédiaire de récupération et de surveillance de données électriques : Épices Énergie. Épices Énergie est une entreprise qui permet à un propriétaire d'installations d'énergie renouvelable de superviser et de gérer leurs installations via une plateforme web (mais Épices Énergie ne propose pas de maintenance ni d'effectuer des travaux comme Ener-Pacte le fait). Épices Énergie fait un travail de récupération des données onduleurs de ces centrales, soit via des API constructeur, soit, quand ce n'est pas possible, via des dataloggers (appareils électroniques qui enregistrent les données onduleur) posés sur les centrales. Pour les autres centrales (une quinzaine), Ener-Pacte accède aux données onduleurs via d'autres plateformes web (par exemple : meteocontrol.de, solarlog-portal.fr ou solarmax.com).

La volonté d'Ener-Pacte est double : récupérer les données éparpillées sur diverses plateformes web (hors Épices), afin de les formater, les agréger afin de pouvoir les analyser. C'est à cette étape que j'interviens. Dès lors mon stage s'articulera autour de la problématique suivante : **comment automatiser la récupération de données onduleurs et l'analyse de ces dernières ?** Concernant la récupération de données onduleurs, je devrai automatiser la récupération de données onduleurs présentes sur différentes plateformes web. L'idée est que cette tâche d'extraction de données ne soit plus faite « à la main » par les opérationnels. Concernant l'analyse de ces données, je devrai proposer une méthode qui permette de détecter des pertes de rendement de centrale PV et de faire de la projection afin de prédire le rendement futur de telle ou telle centrale. Voici mes missions décomposées

dans un diagramme de Gantt. J'y ai aussi ajouté l'écriture du rapport et le rendu des différents livrables.

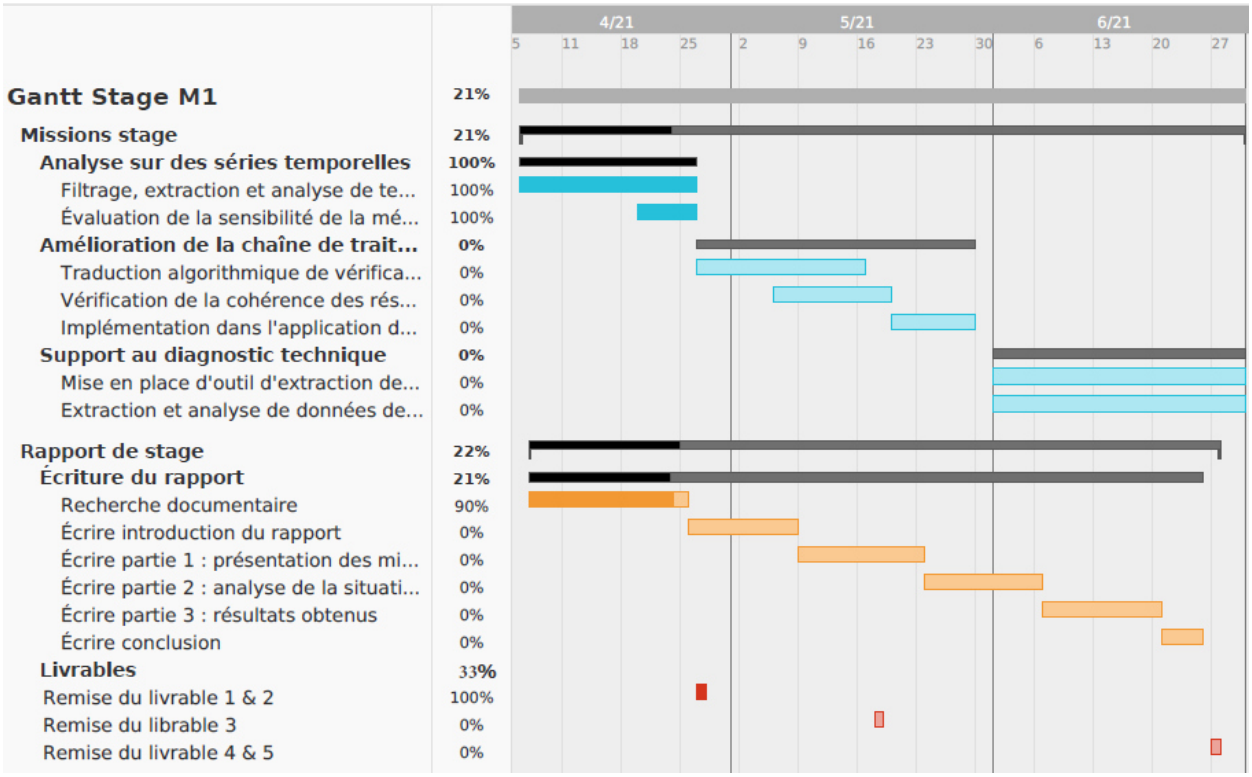


Figure 1. Gantt initial avec la temporalité des tâches liées à mes missions et à la réalisation de mon rapport de stage (date de réalisation de ce Gantt : 27 avril 2021).

I. Extraction et nettoyage de données

A. Plateforme meteocontrol.de : automatisation d'une tâche longue et fastidieuse

Présentation du problème rencontré

L'extraction de données sur la plateforme meteocontrol de deux centrales (deux actuellement, mais possiblement beaucoup plus dans un futur proche) est longue et fastidieuse car pour exporter les données, les opérationnels doivent le faire « à la main » en cliquant sur le bouton « Export Excel » (voir Figure 2) grandeur par grandeur (ex. Énergie, Puissance courant continue, Tension courant continu, etc.) jour par jour et ce, pour plus de 10 ans de données pour certaines centrales.

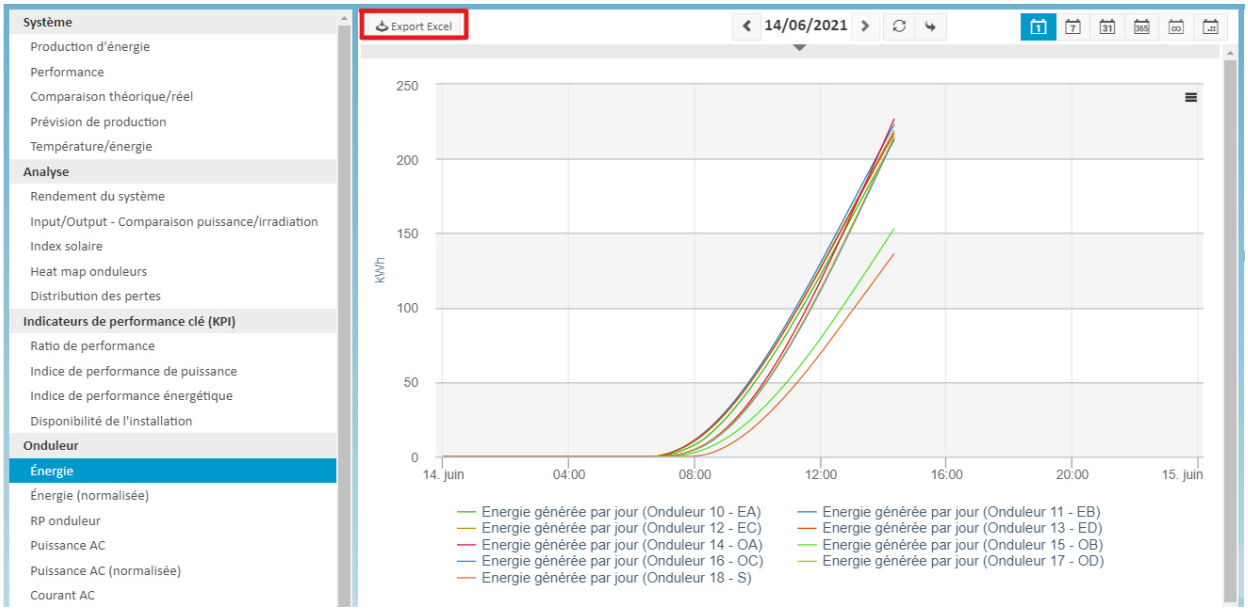


Figure 2. Capture écran de la plateforme meteocontrol pour une centrale spécifique. Sur la gauche, sous « Onduleur », on peut voir plusieurs grandeurs comme Énergie, Puissance AC ou encore Courant AC (il y en a ~15 par centrale). Au centre, on voit les courbes représentant les données de la grandeur en question (ici Énergie, exprimée en kW/h), par tranche de 15 minutes, pour chaque onduleur (ex. Onduleur 10 – EA, Onduleur 11 – EB, Onduleur 12 – EC, etc.). En haut, encadré en rouge, il y a le bouton « Export Excel » qui permet d'exporter ces données en format csv.

Voici le fichier téléchargé lorsque l'on clique sur « Export Excel ».

nergie_2021_06_14.csv - Notepad

File Edit Format View Help

"Période: 14/6/2021 0:00:00 - 12:59:42"

"Date"	"Energie générée par jour (Onduleur 10 - EA) [kWh]"								"Energie générée par jour (Onduleur 11
"0:00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:05"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:10"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:15"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:20"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:25"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:30"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:35"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:40"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:45"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:50"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"0:55"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:05"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:10"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:15"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:20"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:25"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:30"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:35"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:40"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"
"1:45"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"	"0,00"

Ln 1, Col 1 100% Windows (CRLF) UTF-16 LE

Figure 3. Fichier csv téléchargé lorsqu'on clique sur « Export Excel ». En titre, la période des données affichées (ici, le 14 juin 2021, à partir de 00:00:00). « 12:39:52 » représente l'heure à laquelle le fichier a été téléchargé (en format 'UTC' donc 2h de moins que l'heure du fuseau horaire Europe/Paris en été). En colonnes, on retrouve l'heure de l'observation, puis la grandeur d'intérêt associée à chaque onduleur (ex. « Energie (Onduleur 1) », « Energie (Onduleur 2) », etc.).

A noter qu'il s'agit d'un seul fichier csv (format standard qui facilite la lecture et l'exploitation ultérieure des données) contenant les données pour un seul jour et pour une seule grandeur. Une grossière estimation du temps que prendrait l'extraction « à la main » de 10 ans de données sur une centrale avec 20 grandeurs va comme suit : si l'on compte 5 secondes par export (le temps de cliquer sur le jour désiré, cliquer sur le bouton « Export Excel », et enregistrer le fichier), qu'il y a 20 grandeurs pour ladite centrale et 3650 jours à extraire (365 [jours dans une année] x 10 [ans]), on obtient (5 x 20 x 3650) /60 ~= 6083 minutes, soit ~101h (en continu, sans pause, jour et nuit). Après avoir extrait toutes ces données, un humain aurait en plus, à rassembler toutes ces données (par grandeur par exemple) afin de pouvoir analyser une évolution macroscopique de la centrale pour telle ou telle grandeur. La tâche s'annonce extrêmement laborieuse.

Solution proposée

La solution que je propose est une automatisation de ce processus à l'aide d'un programme Python.

Etat des lieux

Les données sont hébergées sur une plateforme web « meteocontrol.de ». Lorsqu'un utilisateur arrive sur la plateforme, il doit rentrer un identifiant et un mot de passe afin d'accéder à son espace personnel. Il peut alors cliquer sur le jour et la grandeur de son choix et ensuite, cliquer sur « Export Excel ». Le serveur renvoie alors à l'utilisateur un fichier en format csv. Cela signifie que lorsque l'utilisateur clique sur « Export Excel », il envoie une requête HTTP au serveur, lequel renvoie en réponse un fichier csv avec la date et la grandeur désirée.

Idée de départ

Mon idée de départ a alors été de simuler ces requêtes envoyées par le client au serveur grâce à la bibliothèque *requests* de Python (bibliothèque qui permet de faire des requêtes HTTP facilement). Puisque pour obtenir chaque fichier csv il suffit d'envoyer une requête, il m'a semblé facile de répéter l'opération autant de fois qu'il y a de grandeurs et de jours à récupérer.

Problème rencontré

Le problème est que pour pouvoir envoyer des requêtes à un serveur il est nécessaire d'avoir : une méthode (ici GET), un *header* contenant des informations relatives au contexte de la requête (ex. le nom du navigateur utilisé (ex. Chrome/91.0.4472.77)), une URL de base, et enfin des paramètres qui viennent s'ajouter à cette URL de base). Une URL de base avec des paramètres est de la forme suivante :

```
https://example.fr/chemin?nom=brassens&prenom=georges
```

Dans l'url suivante, les paramètres venant compléter l'URL de base sont au nombre de deux : « nom=brassens » et « prenom=georges », ils sont séparés par une esperluette « & ». Lorsque le serveur recevra l'url et les paramètres, il renverra un réponse spécifique propre à la combinaison de cette URL et de ses paramètres. Or, en inspectant les paramètres envoyés au serveur lorsque l'on clique sur « Export Excel » (puis en allant dans l'onglet « Network » de l'outil d'inspection de Chrome), on remarque qu'il n'y a pas de paramètres envoyés au serveur mais seulement une URL de base contenant un chemin d'accès vers un fichier csv :

```
https://www1.meteocontrol.de/vcom/default/file/download/get/excel_fc62
19f767f503c26a2831777636f5e7.csv/name/nergie_2021_06_14.csv
```

En analysant l'url, on peut supposer que le serveur génère un fichier csv sur demande car le nom du fichier « excel_6c6df4cd85646e8822865c45e873aefa.csv » semble être un nom généré aléatoirement ou arbitrairement (par exemple un regroupement de différentes infos (nombre de requêtes + client +, etc.)) via un algorithme d'encodage inconnu. En effet, lorsque je clique une nouvelle fois sur « Export Excel » pour le même jour, voici l'URL générée :

```
https://www1.meteocontrol.de/vcom/default/file/download/get/
excel_1f2b447dd666089261554a509d0b2a9f.csv/name/nergie_2021_06_14.csv
```

Comme la suite de caractères « 1f2b447dd666089261554a509d0b2a9f » n'est pas la même (pour le même jour), je me suis dit qu'il n'y avait pas possibilité de décoder la suite de caractères. J'ai donc suivi une autre approche.

Idée alternative

Comme on peut le voir sur la Figure 2, la plateforme web affiche des courbes (représentant les données d'une grandeur pour chaque onduleur pour le jour sélectionné) à l'utilisateur. C'est-à-dire qu'à la place de télécharger les données, on peut simplement les visualiser. Ce qui signifie que d'une manière ou d'une autre, le serveur renvoie les données au client, afin que le navigateur puisse les lire et les afficher.

Au lieu de récupérer les fichiers csv directement, ma seconde idée a été de récupérer les données brutes que le serveur envoie au navigateur. Pour ce faire, il me fallait envoyer au serveur la même requête que le navigateur envoie au serveur lorsqu'on clique sur une grandeur sur le côté gauche de la page afin de visualiser les courbes. Voici l'URL envoyée par le navigateur lorsqu'on clique sur « Énergie » à la date du 14 juin 2021 :

```
https://www1.meteocontrol.de/vcom/default/static/graph-with-table?
url=%2Fdefault%2Fstatic%2Fgraph-with-
table&displayType=highcharts&systemId=15028&type=Anlage&key=RPP09&char
tType=Wechselrichter&width=undefined&height=478.09275&date=2021-06-
13T00%3A00%3A00&leftTeilanlagen=&leftWechselrichter=&leftDataDevices=&
rightTeilanlagen=&rightWechselrichter=&rightDataDevices=&inv=Id14798.1
%2CId14798.2%2CId14798.3%2CId14798.4%2CId14798.5%2CId14798.6%2CId14798
.7%2CId14798.8%2CId14798.9&displayMinuteValues=false&period=tag
```

Bien que très longue, cette URL composée d'une URL de base et de beaucoup de paramètres contient toutes les informations nécessaires (surlignées en jaune) pour récupérer les données de la grandeur « Énergie » du 14 juin 2021. En voici un résumé succinct :

- URL de base :

```
https://www1.meteocontrol.de/vcom/default/static/graph-with-table?
```

- Paramètres

Signification	Paramètres d'intérêt de l'URL
L'identifiant de la centrale	systemId=15028

Grandeur concernée	key=RPP09
Jour concerné	date=2021-06-13T00%3A00%3A00 (soit la version encodée de 2021-06-13T00:00:00)
Les onduleurs de la centrale (*inverters* en Anglais)	inv=Id14798.1%2CId14798.2%2CId14798.3%2CId14798.4%2C Id14798.5%2CId14798.6%2CId14798.7%2CId14798.8%2CId14798.9

Tableau 1. Tableau montrant les différents paramètres encodés dans l'URL que le navigateur envoie au serveur lorsqu'on clique sur une grandeur pour en afficher les données du jour.

Ayant toutes ces informations, je n'avais plus qu'à construire un programme Python qui envoie au serveur une requête HTTP pour une centrale donnée (systemId), une grandeur (key), un jour donné (date) et un lot d'onduleurs (inv), et de répéter/boucler ce processus pour toutes les grandeurs disponibles sur cette centrale ainsi que pour tous les jours nécessaires.

Résultats obtenus

Après quelques jours de travail, j'obtiens un programme Python très automatisé pour l'utilisateur. L'utilisateur entre en paramètres d'entrée le login et le mot de passe de la centrale concernée, une date de début et une date de fin correspondants à l'intervalle de temps qu'il souhaite récupérer et le programme exporte les données désirées dans un unique fichier csv. Voici les étapes qu'effectue le programme :

Étape 1 : login

L'utilisateur entre le login et le mot de passe de la centrale en question ainsi que les dates auxquelles récupérer les données. Le programme se connecte au serveur via une requête POST en utilisant le login et le mot de passe entrés par l'utilisateur. A noter une petite subtilité : initialement, le serveur refuse l'authentification via cette méthode car il détecte que l'on se connecte via Python et non via un navigateur web. L'astuce est de changer l'*user-agent* envoyé au serveur, de :

```
"user-agent": "python-requests/2.25.1"
```

à

```
"user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93
Safari/537.36"
```

Ainsi le serveur croit que c'est un navigateur web (et donc un utilisateur humain) qui fait la requête et non un programme Python.

Étape 2 : navigation vers la page principale et récupération de métadonnées

Une fois connecté, le programme fait une requête pour arriver sur la page principale (là où tous les éléments relatifs à la centrale sont visibles :systemeId, les grandeurs et leur clé ainsi que les onduleurs et leur clés). Il suffit ensuite de récupérer les éléments de la page (en format html) qui lui serviront à faire les requêtes nécessaires.

Étape 3 : requêtes http

Lorsqu'on a les éléments nécessaires, le programme effectue une série de requêtes avec en URL de base :

```
base_url = "https://www1.meteocontrol.de/vcom/default/static/graph-with-table?"
```

et les paramètres suivants (pour ne citer que ceux qui changent selon les requêtes et les centrales) :

```
parameters = {
    "systemId": "15028",
    "key": "RPP09",
    "date": "2021-06-13T00:00:00",
    "inv":
"Id14798.1%2CId14798.2%2CId14798.3%2CId14798.4%2CId14798.5%2CId14798.6%2CId14798.7%2CId14798.8%2CId14798.9",
}
```

L'idée est que `"systemId"` et `"inv"` ne changent pas mais que `"key"` et `"date"` changent autant de fois que nécessaire (à chaque itération de la boucle). Par exemple, pour récupérer les données d'une semaine de la centrale « 15028 » qui, supposons, a 20 grandeurs, le programme effectuera 20 x 7 requêtes soit 140 requêtes.

Étape 4 : formatage des données récupérées

La réponse renvoyée par le serveur à ces requêtes est un format html de la page web demandée. Il suffit de récupérer les données présentent sur la page (les données sont sous le format « 00:00 » : [timestamps, valeur], « 00 :15 » : [timestamps, valeur], etc.) et de les mettre en forme (en format dataframe, c'est-à-dire un tableau prêt à l'export vers un format csv). Voici un dataframe exemple pour mieux comprendre le processus de formatage des données :

Données brutes (extrait) (j'ai indiqué en gras les valeurs à extraire) :

"data":[{"name":"**Energie g\u00e9n\u00e9r\u00e9e par jour (Onduleur 10 - EA)**","id":"","color":"#2cbd08","unit":"kWh","turboThreshold":25000,"isCosPhi":false,"tooltip":{"decimals":2},"zones":[],"visible":true,"yAxis":0,"data**":[[[1623739200000,0.03],[1623739500000,0.28],[1623739800000,0.61]]** … {"name":"**Energie g\u00e9n\u00e9r\u00e9e par jour (Onduleur 11 - EB)**","id":"","color":"#1a78d1","unit":"kWh","turboThreshold":25000,"isCosPhi":false,"tooltip":{"decimals":2},"zones":[],"visible":true,"yAxis":0,"data":**[[1623739200000,0.01],[1623739500000,0.14],[1623739800000,0.33]]** …

Données formatées dans un dataframe :

		Energie (kWh)	Puissance AC (W)	Tension AC (Volt)
Onduleur 10	2021-06-14 08:40	0.03	3200	239
	2021-06-14 08:45	0.28	4300	238
	2021-06-14 08:50	0.61	2600	238
Onduleur 11	2021-06-14 08:40	0.01	1300	237
	2021-06-14	0.14	1600	235

	08:40			
…	…	…	…	…

Tableau 2. Données formatées dans un dataframe avec en indices les onduleurs et la date, et en colonnes, les différentes grandeurs disponibles pour une centrale donnée.

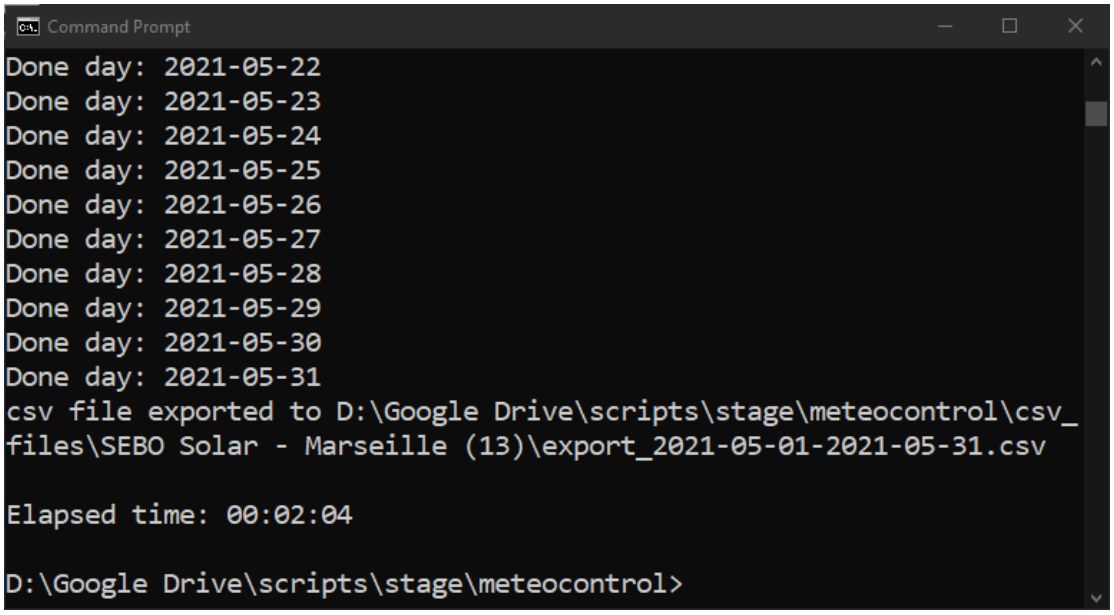


Figure 4. Terminal de commande montrant le temps d'exécution du programme pour 1 mois de données. Comme indiqué, le programme a mis 2:04 min pour extraire et exporter 1 mois de données de la centrale en question (du 1 mai 2021 au 31 mai 2021). A titre de comparaison, comme estimé précédemment, une extraction de 10 ans de données pour 20 grandeurs prendrait ~101h. Ici, le programme aurait mis (124 [s] x 12 [mois] x 10 [ans]) /60 = 248 minutes soit ~4h. Le programme est donc ~24x plus rapide qu'une extraction « à la main » (et ne se fatigue pas !).

Étape 5 : export des données en format csv

Une fois les données de toutes les grandeurs, de tous les jours et de tous les onduleurs agglomérés dans un et unique dataframe, on peut les exporter facilement sous le format csv. Voici le résultat :

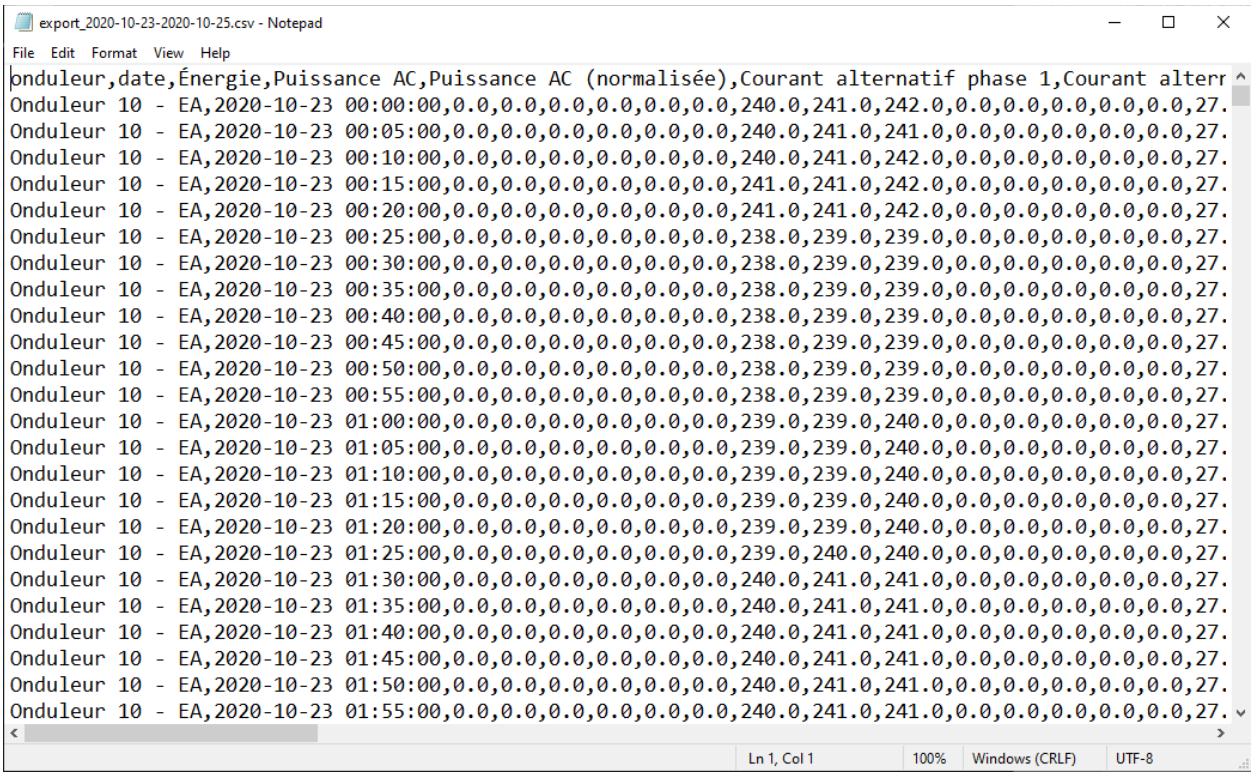


Figure 5. Fichier csv en sortie de mon programme Python. En sortie, on récupère un fichier csv avec en ligne chaque observation et en colonne le nom de l'onduleur, l'heure de l'observation, et chaque grandeur disponibles (Énergie, Puissance AC, Puissance AC (normalisée), etc.). L'utilisateur futur voulant exploiter ces données n'aura qu'à les importer sur Python, Excel ou de toute autre manière.

B. Récupération de données sur deux autres plateformes

J'ai effectué un travail similaire sur deux autres plateformes : solarmax.com et solarlog-portal.fr (travail que je résumerai ici rapidement par manque de place et parce que le principe est sensiblement le même que pour meteocontrol, à quelques subtilités près).

solarmax.com

J'ai créé un programme qui demande à l'utilisateur les mêmes paramètres d'entrée (login, mot de passe, date de début et date de fin) et qui renvoie une fichier csv sous le même format. La différence avec la plateforme meteocontrol est que solarmax permet d'exporter les données mois par mois (au lieu de jour par jour). J'ai donc utilisé cette opportunité pour éviter d'avoir à faire du *web-scraping* pour récupérer les données affichées sur la page web (dont la partie formatage et agrégation de données est assez laborieuse). J'ai utilisé la bibliothèque Python *selenium* qui permet de simuler des actions qu'un utilisateur humain ferait sur son navigateur web (naviguer de page en page, cliquer sur un bouton, etc.).

Le lecteur pourrait se demander pourquoi je n'ai adopté la même stratégie que pour la plateforme meteocontrol. La raison est la suivante : sur la plateforme solarmax, il est possible d'exporter un csv des données, mois par mois et ce pour toutes les grandeurs. Même sur 10 ans de données, cela revient à simuler ~120 (12 [mois] x 10 [ans]) clics pour exporter l'intégralité des données, ce qui est raisonnablement rapide lorsque c'est effectué par un programme. Il suffit ensuite d'agréger les fichiers csv pour obtenir un seul et unique fichier

csv. En revanche, meteocontrol ne permet pas d'exporter les données mois par mois pour toutes les grandeurs. Si j'avais voulu adopter la même méthode pour meteocontrol j'aurais dû simuler 20 [grandeurs] x 365 [jours] x 10 [ans] = 73000 clics, ce qui n'est pas raisonnable en termes de performance. Le fait d'envoyer des requêtes HTTP permet d'outrepasser le fait de devoir cliquer sur un bouton pour exporter, et donc d'être au moins aussi rapide (en réalité, beaucoup plus rapide) qu'en simulant des clics. Le temps d'exécution pour 1 mois est d'environ 2 min.

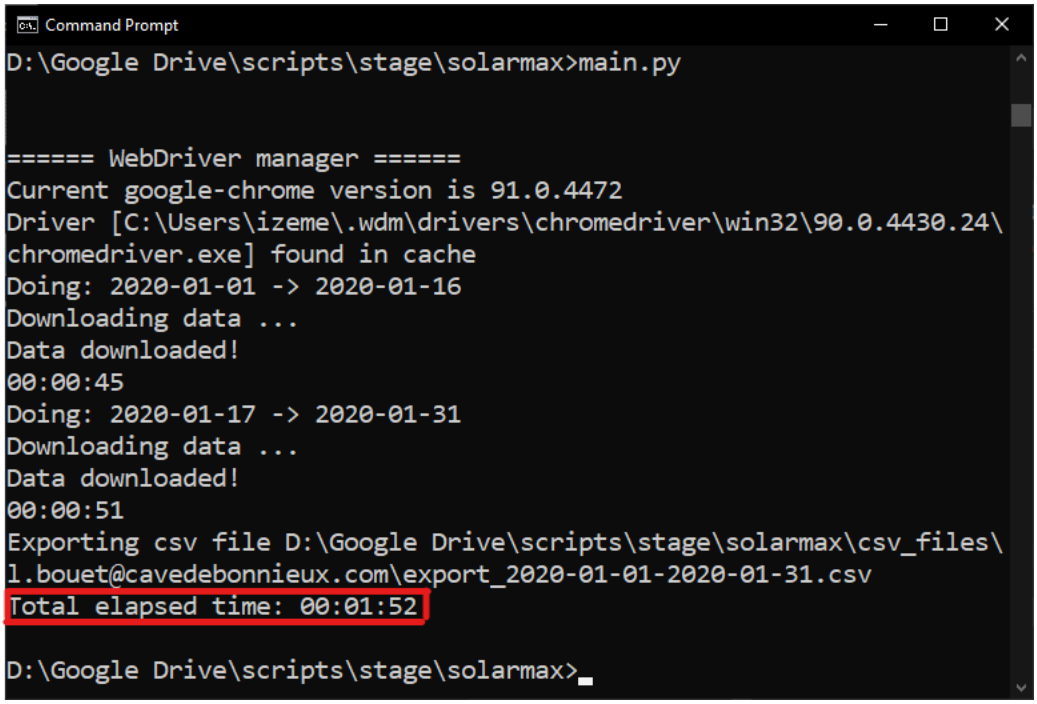


Figure 6. Terminal de commande depuis lequel j'ai exécuté le programme pour récupérer les données de la plateformes solarmax.com. Comme indiqué, le programme a mis 1:52 min pour extraire et exporter 1 mois de données de la centrale donnée (du 1 mai 2021 au 31 mai 2021). A titre de grossière comparaison, j'ai testé d'exporter un mois de données à la main sur la plateforme et cela m'a pris 1:45 minutes csv (et ce sans compter, le temps de login, de navigation pour aller jusqu'à la page où l'on peut exporter les données et de formatage du/des fichiers csv exportés). Le programme met donc sensiblement le même temps qu'une extraction « à la main » pour récupérer un mois de données (il est en réalité plus rapide puisqu'il agrège automatiquement les différents fichiers csv exportés ; un utilisateur exportant plus d'un mois de données aurait à le faire manuellement une fois tous les mois exportés), mais il a l'avantage de le faire automatiquement, aussi grand soit l'intervalle de temps désiré.

solarlog-portal.fr

La dernière plateforme sur laquelle j'ai automatisé la récupération de données est solarlog. Il s'agit bien de récupération de données (*web scraping*) et non d'automatisation d'une tâche faisable « à la main ». En effet, sur la plateforme en question il est tout simplement impossible d'exporter les données (il n'y a pas de bouton pour exporter les données en format csv par exemple) (voir Figure 7).

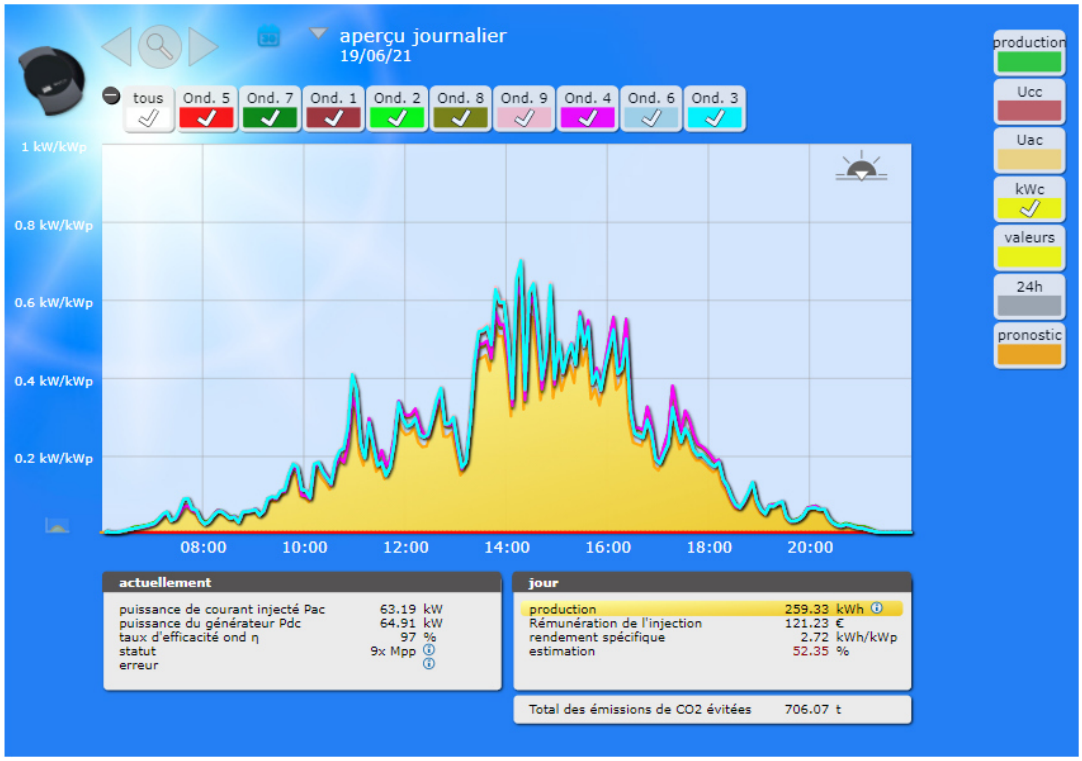


Figure 7. Aperçu de données de production au 19 juin 2021 telles qu'elles sont affichées sur la plateforme solarlog. En haut, on voit les différents onduleurs sur lesquels on peut cliquer. A droite, les différentes grandeurs mesurées. On notera l'absence de bouton pour exporter les données.

J'ai donc adopté une approche similaire que pour la plateforme meteocontrol : premièrement, je récupère les métadonnées spécifiques à chaque centrale (id centrale, noms onduleurs, grandeurs disponibles). Ensuite, je fais une série de requêtes HTTP pour récupérer le code html des pages qui m'intéressent (une requête par jour, par onduleur et par grandeur). J'extrait du code source les valeurs qui m'intéressent et les agrège dans un dataframe. J'exporte le dataframe dans un fichier csv.

Le temps d'exécution de la première version de mon programme était très grand (de l'ordre de 20-25 minutes par mois) car il fallait effectuer beaucoup de requêtes pour récupérer les données d'un seul jour sur cette plateforme : de l'ordre d'une vingtaine de requêtes par jour car la plateforme ne permet pas d'afficher les valeurs de tous les onduleurs d'un coup pour certaines grandeurs, ce qui oblige à faire une requête pour chaque onduleur et pour chaque grandeur. De plus, le serveur a un temps de réponse assez lent (1-2s) comparé aux autres plateformes. J'ai néanmoins trouvé un moyen de réduire ce temps d'exécution en utilisant du multithreading : l'idée est d'exploiter la capacité du processeur de l'ordinateur à exécuter plusieurs tâches (ou thread) en parallèle (Lamport, 1979) (versus exécuter plusieurs tâches de manière sérielle, les unes après les autres). Au lieu d'attendre la réponse du serveur pour exécuter la requête suivante, le programme envoie plusieurs requêtes en même temps sans attendre que le serveur réponde. En utilisant du multithreading, j'ai réussi à diviser le temps d'exécution par 3 (passant de 20-25 minutes à 7-8 minutes par mois) (Figure 8).

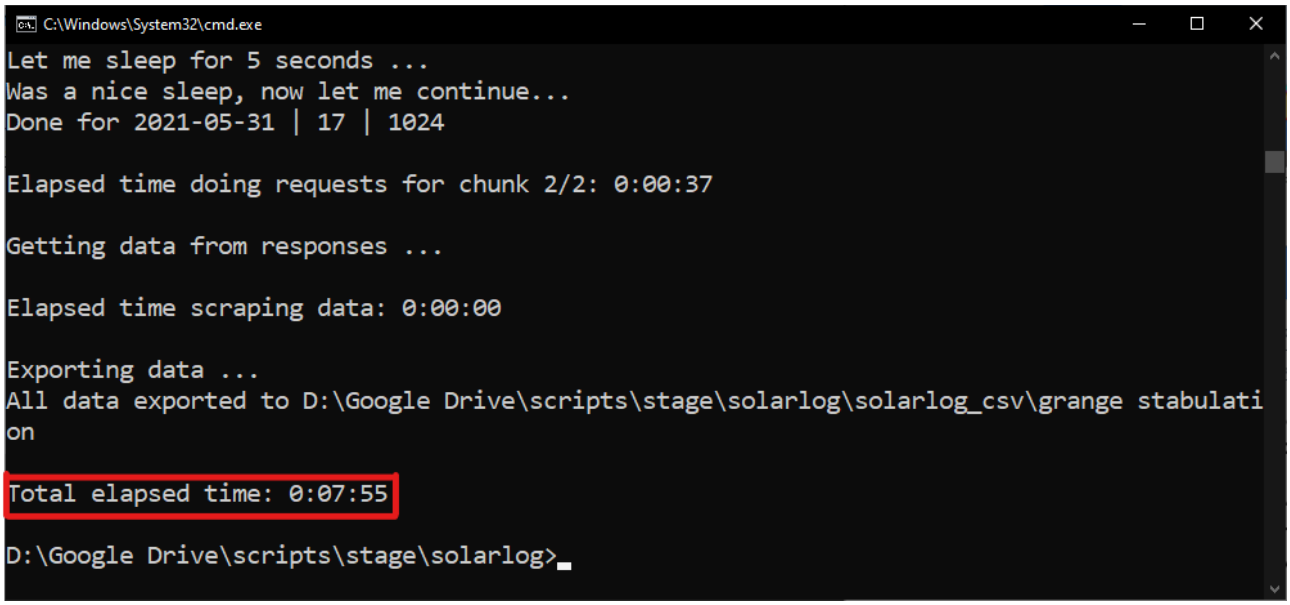


Figure 8. Temps d'exécution de la deuxième version de mon programme de web scraping *sur la plateforme solarlog pour 1 mois de données (1 juin – 30 juin 2021) en utilisant du multithreading.*

A noter que, au-delà de la récupération de données historiques, certains de mes programmes ont été adaptés au sein de l'entreprise pour compenser l'absence d'une API sur certaines de ces plateformes afin d'assurer le suivi instantané de production. Mes programmes ont donc une double utilité.

Conclusion

Les équipes opérationnelles d'Ener-Pacte ont besoin des données historiques des centrales non gérées par Épices Energie. Or, la plupart des plateformes sur lesquelles sont hébergées les données de centrales PV ne propose pas d'exporter l'intégralité des données historiques en une seule fois ; l'utilisateur doit le faire mois par mois, voire jour par jour. A l'aide de quelques connaissances de bases en Python et de beaucoup d'heures de recherche sur internet, j'ai mis au point des programmes permettant d'exporter de manière automatique toutes les données historiques des centrales présentes sur les plateformes meteocontrol.de, solarmax.com et solarlog-portal.fr. Le temps d'exécution des programmes est relatif à la période des données à extraire mais prend environ 2 min par mois pour les deux premières plateformes et 8 minutes pour solarlog, ce qui est tout à fait convenable pour l'utilisation qui en sera faite : à savoir, une extraction de données ponctuelle (ce n'est pas tous les jours que les opérationnels ont besoin d'extraire 10 ans de données). Ces différents programmes éviteront aux opérationnels de perdre du temps à effectuer des tâches d'extraction de données qui sont fastidieuses (et permettront tout simplement d'avoir la possibilité de récupérer les données pour solarlog, ce qui était impossible pour eux jusqu'ici), et leur permettront de passer plus de temps sur d'autres tâches à forte valeur ajoutée (analyse des données obtenues, expertise technique, etc.).

II. Analyse de données sur des séries temporelles

Présentation du problème rencontré

Une fois les données historiques récupérées, les opérationnels passent à une étape d'analyse leur permettant d'évaluer l'état de dégradation des centrales solaires.

Détection de tendances et rupture de tendance

Premièrement, ils vont analyser les données dans le but de détecter des anomalies de rendement[1]. Par exemple, en regardant le rendement de la centrale sur les dernières années, un opérationnel se rend compte qu'à partir d'un moment donné, le rendement s'est mis à lentement diminuer de manière anormale. L'opérationnel va alors essayer de comprendre ce qu'il s'est passé à cette date (encrassement des panneaux, ombrage, dégradation de certaines cellules etc.) pour tenter de corriger le problème.

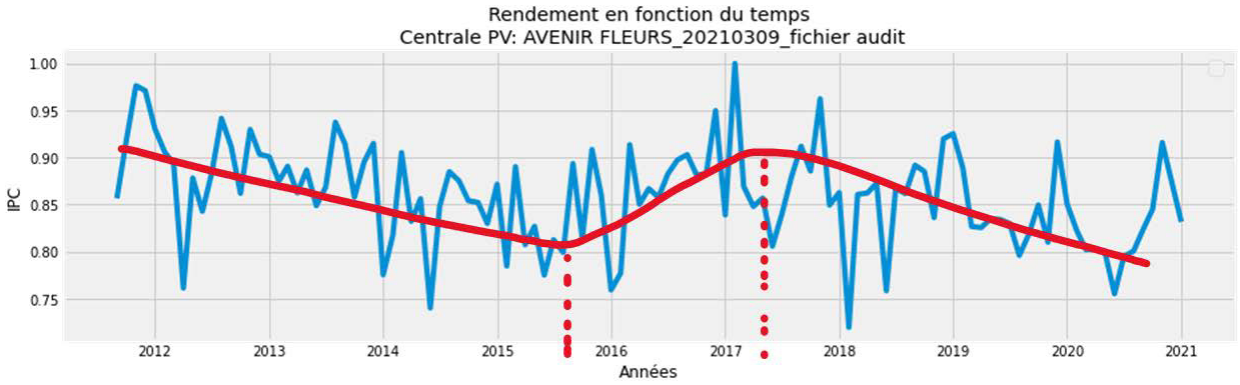


Figure 9. Exemple de données de rendement d'une centrale photovoltaïque (en bleu) en fonction du temps et d'une analyse de tendance grossière faite à la main (en rouge). On observe que le rendement de la centrale en question a diminué de manière quasi linéaire de mi-2011 à mi-2015 ; puis que ce dernier a augmenté jusqu'à quelque part entre 2017 et 2018 ; puis qu'il a diminué à nouveau jusqu'en 2021. J'ai tracé en rouge les tendances et ruptures de tendances qu'un opérationnel pourrait détecter « à l'œil nu ». En pointillé sont les dates auxquelles on observe une rupture de tendance : quelque part entre 2015 et 2016, puis au premier semestre de l'année 2017. Le lecteur pourra s'apercevoir que ce genre d'analyse grossière n'est pas très fiable et dépend de l'appréciation de chacun : y a-t-il rupture de tendance aux endroits que j'ai tracés ? Ou un peu avant ? Un peu après ? Il est difficile de trancher de manière objective. De plus, même si l'on s'accorde sur un moment donné, on n'aura pas la date précise à partir de laquelle s'est produit la rupture de pente.

Prédiction du rendement futur

Deuxièmement, toujours à partir des rendements historiques, les opérationnels vont tenter de prédire le rendement futur de la centrale, l'intérêt est d'évaluer la dégradation attendue des performances de la centrale dans le cas où aucune action corrective ne serait entreprise. Or, aujourd'hui, ils utilisent une simple régression linéaire qui se base sur l'intégralité des données historiques (ce qui donne des résultats faux lorsque le rendement ne diminue pas de manière stable et linéaire).

---

[1] Le rendement est analysé à travers de l'IPC (Indice de Performance de Centrale). L'IPC est un KPI propre à Ener-Pacte calculé en utilisant différentes grandeurs notamment la production (en W) de la centrale.

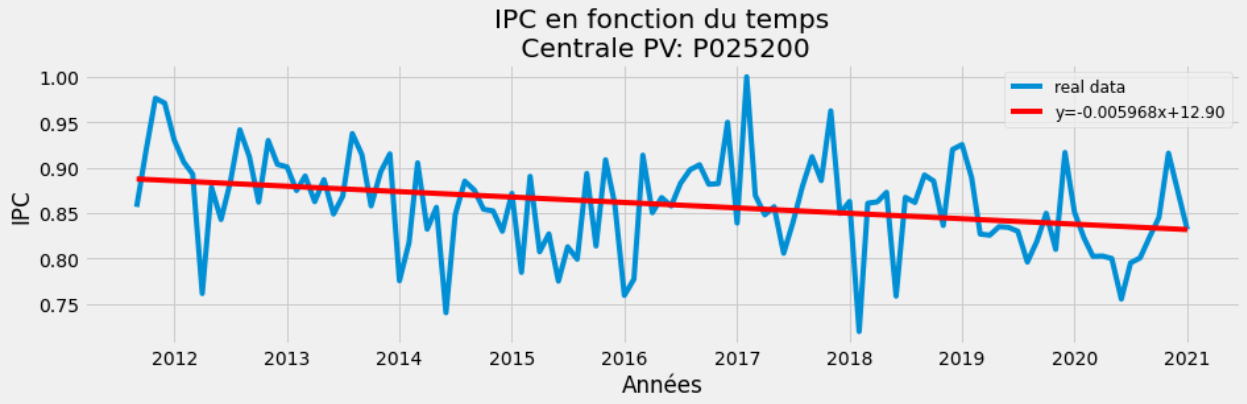


*Figure 10. Prédiction du rendement futur telle qu'elle est faite aujourd'hui : avec une régression linéaire. La régression linéaire est calculée à partir de l'ensemble des données historiques de rendement de la centrale. Il en ressort deux coefficient m et p (m = le coefficient de pente et p = l'ordonnée à l'origine). Avec ces deux coefficients, il est possible de tracer une droite de régression des données mais aussi de prédire le rendement futur (exemple, le rendement de la centrale en 2025 serait de : 2025 [année x] * -0.005968 [m] +12.90 [p] = 0.8148).*

Mon objectif est donc de trouver une méthode qui permettra d'éviter les détections manuelles (et subjectives) des ruptures de pentes et fera une projection du rendement de la centrale plus précise qu'une régression linéaire.

Solution proposée

Prophet : un modèle de prédiction sur des données temporelles

La méthode que je propose est l'implémentation du modèle statistique *Prophet* (Taylor & Letham, 2018) sous Python. Il a été développé pour pallier les manquements des modèles de prédiction de données temporelles existants. Dans leur article, les auteurs prennent l'exemple du package *forecast* développé sous R (Hyndman & Khandakar, 2008) qui fait de mauvaises prédictions sur des séries temporelles quand celles-ci sont composées de fortes ruptures de tendances et possèdent de saisonnalités. Par saisonnalité j'entends un effet produit par un évènement qui peut se répéter toutes les semaines, tous les mois voire toutes les années. Par exemple, la fréquentation des bars a de fortes chances d'être impactée par le jour la semaine (saisonnalité hebdomadaire) ou encore les ventes de maillots de bain impactées par les mois de l'année (saisonnalité annuelle). Selon les auteurs, lorsque la prédiction est mauvaise, il est également important de pouvoir ajuster les paramètres du modèle facilement sans avoir une compréhension avancée de son fonctionnement, ce qui n'est pas le cas pour le modèle *forecast* pris en exemple par les auteurs. Par exemple, les premiers paramètres à ajuster sont : *maximum orders of the differencing*, *auto-regressive components*, et *the moving average components*. Un analyste lambda ne saura pas comment ajuster ces paramètres pour affiner son modèle.

Prophet est un modèle de données temporelles décomposables (*decomposable time series model*) avec trois composantes : la tendance (*trend*), la saisonnalité (*saisonality*) et les vacances (*holidays effect*). Ces composantes sont combinées dans l'équation suivante :

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

avec $g(t)$ une fonction de tendance qui modélise les changements linéaires ou polynomiaux (ou les changements prévisibles non périodiques) des valeurs de la série temporelle, $s(t)$ les changements périodiques (ex. hebdomadaires ou mensuels), et $h(t)$ l'effet des vacances[2]. Le terme d'erreur ε_t représente les changements non capturés par le modèle. Cette approche est similaire à un modèle additif général (*generalized additive model*) : un mélange entre les propriétés d'un modèle linéaire généralisé (*generalized linear model*) et d'un modèle additif (*additive model*).

Fonctionnement

Concrètement, on donne à ce modèle une série 'ds' composée de dates (du type '2020-01-01', '2020-01-02' etc.) et des valeurs 'y' correspondantes (par exemple, nos valeurs de rendement). Premièrement, le modèle va autoriser un grand nombre de points de ruptures (*changepoints)* potentiels : 25 (auxquels la tendance est autorisée à changer) qu'il va placer uniformément sur les premiers 80% des données. Entre chacun de ces 25 points de rupture, le modèle calcule une droite de tendance.

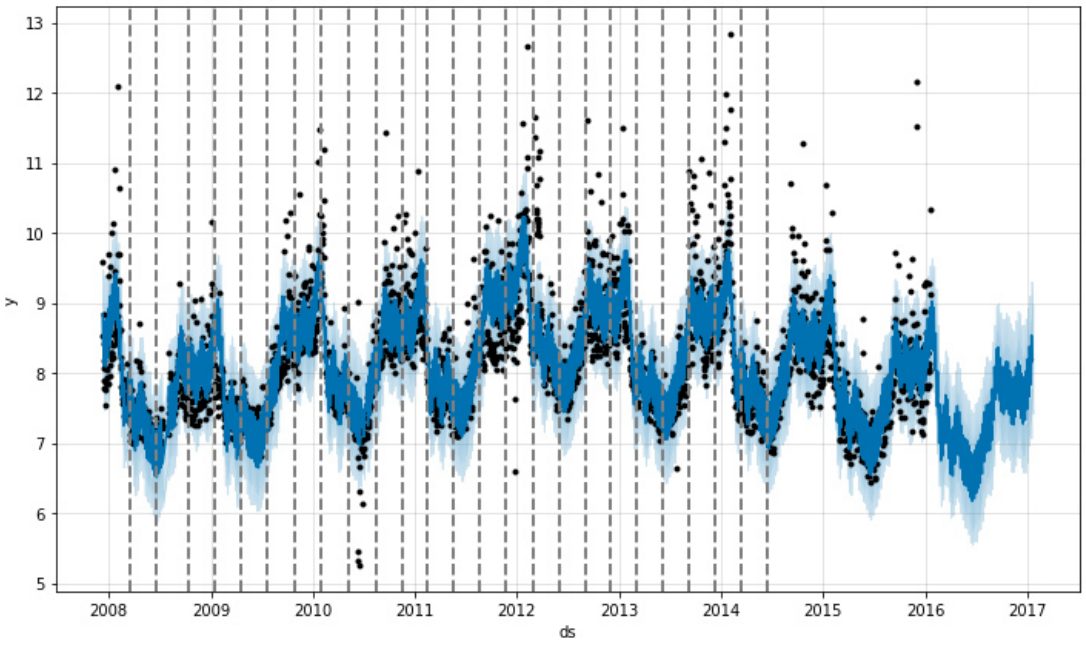


Figure 11. Illustration des 25 changepoints uniformément placés sur les premiers 80% des données. A chacun de ces points la tendance est autorisée à changer.

Ensuite, selon la valeur du paramètre principal 'changepoint_prior_scale' (plus la valeur est grande, plus la droite tendance va être autorisée à changer (valeur par défaut : 0.05), le modèle va garder seulement les *changepoints* où la rupture de la pente/tendance est la plus forte. Les auteurs indiquent que le paramètre 'changepoint_prior_scale' peut être ajusté à convenance selon que la droite de tendance *overfit* ou *underfit* les données (ces termes seront illustrés/détaillés plus loin dans le rapport).

[2] Toutes ces fonctions sont détaillées dans l'article original de Taylor & Letham, 2018.

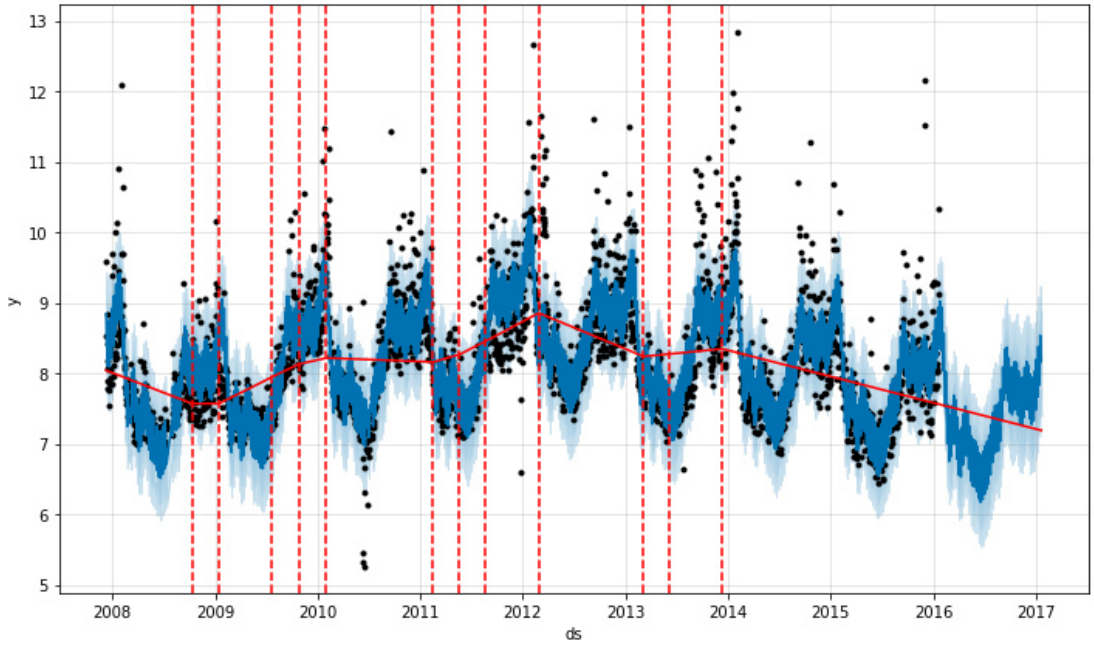


Figure 12. Seuls les points de rupture (ici en pointillés) où le changement de pente est le plus fort sont conservés par le modèle. Cette fonction du modèle est modifiable en ajustant le paramètre 'changepoint_prior_scale' (voir plus bas).

Le modèle permet également d'afficher ses éléments indépendamment les uns des autres, à savoir la tendance et les différentes saisonnalités (par défaut hebdomadaire et annuelle).

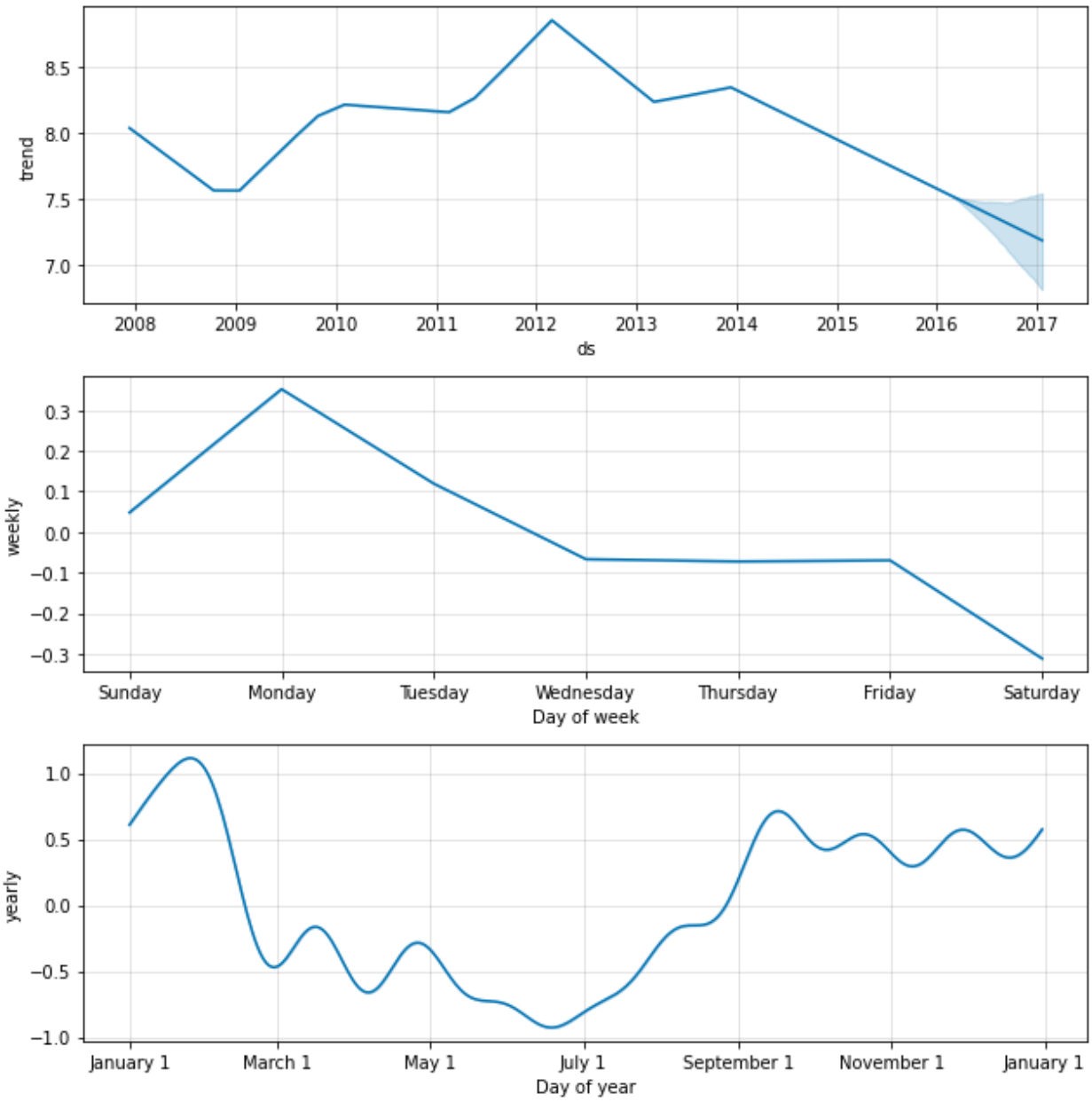


Figure 13. Le modèle affiche également ses différents éléments indépendamment, à savoir la tendance et les différentes saisonnalités (par défaut, hebdomadaire et annuelle). Dans cet exemple on peut voir que la tendance augmente de 2009 à début 2012 puis décroit fortement jusqu'en 2017. On voit également qu'il y a un effet fort du jour de la semaine sur ce jeu de données, à savoir, un pic le lundi et un déclin progressif jusqu'en fin de semaine. Enfin, on voit également un fort effet de la saisonnalité annuelle, à savoir, un pic en janvier-février et un creux en été (juin, juillet, août).

Formatage des données

Son implémentation sous Python est assez simple puisque tout le travail de formatage des données a été fait en amont lors de mon travail d'extraction et de nettoyage des données. Il suffit d'importer les données dans un *dataframe* et de renommer la colonne contenant les dates en 'ds' et celle contenant les valeurs de rendement en 'y'.

```
# importer les données
df = pd.read_excel(file_path, sheet_name='Production')
# renommer les colonnes "date" et "ipc"
df_prophet = df.rename(columns={'date': 'ds', 'ipc': 'y'})
# visualiser le début des données
df_prophet.head()
```

début du dataframe fournit à Prophet (avec les colonnes ‘ds’ les dates, et ‘y’ le rendement):

	ds	y
0	2011-09-01	0.856736
1	2011-10-01	0.917723
2	2011-11-01	0.976373
3	2011-12-01	0.971080
4	2012-01-01	0.930397

Ajustement des paramètres du modèle

Ensuite, on peut directement entraîner le modèle avec les paramètres par défaut ou bien, ajuster ces paramètres à convenance, à savoir :

- changepoint_prior_scale : le paramètre le plus impactant. Il détermine la flexibilité de la tendance c’est-à-dire, à quel point la droite de tendance est autorisée à changer au niveau de chaque point de rupture. Plus la valeur est grande, plus la droite de tendance va être autorisée à changer (valeur par défaut : 0.05, intervalle conseillé par les auteurs : [0.001, 0.5])
- seasonality_prior_scale : contrôle la flexibilité de la saisonnalité. Une grande valeur autorise la saisonnalité à suivre de larges fluctuations tandis qu’une petite valeur contraint l’amplitude de la saisonnalité (valeur par défaut : 10, intervalle conseillé par les auteurs : [0.01, 10]).
- seasonality_mode : mode de la saisonnalité (‘additive’ ou ‘multiplicative’, par défaut : ‘additive’). La saisonnalité est additive si l’amplitude des fluctuations saisonnières reste stable dans le temps. L’effet de la saisonnalité est alors ajouté (et non multiplié) à la tendance par Prophet. Au contraire, si l’amplitude des fluctuations saisonnières augmente avec le temps, la saisonnalité sera multiplicative (et donc multipliée par la tendance) (voir Figure 14). Dans notre cas, l’amplitude des fluctuations saisonnières n’a pas de raison d’augmenter/diminuer dans le temps puisque d’une année sur l’autre, l’effet des mois de l’année sur le rendement n’a pas de raison d’augmenter/diminuer avec le temps.
- holidays_prior_scale : contrôle la flexibilité de la droite de tendance à *fitter* l’effet des vacances. Ce paramètre ne m’intéressera pas car les valeurs de rendement d’une centrale PV ne dépendent pas des vacances.

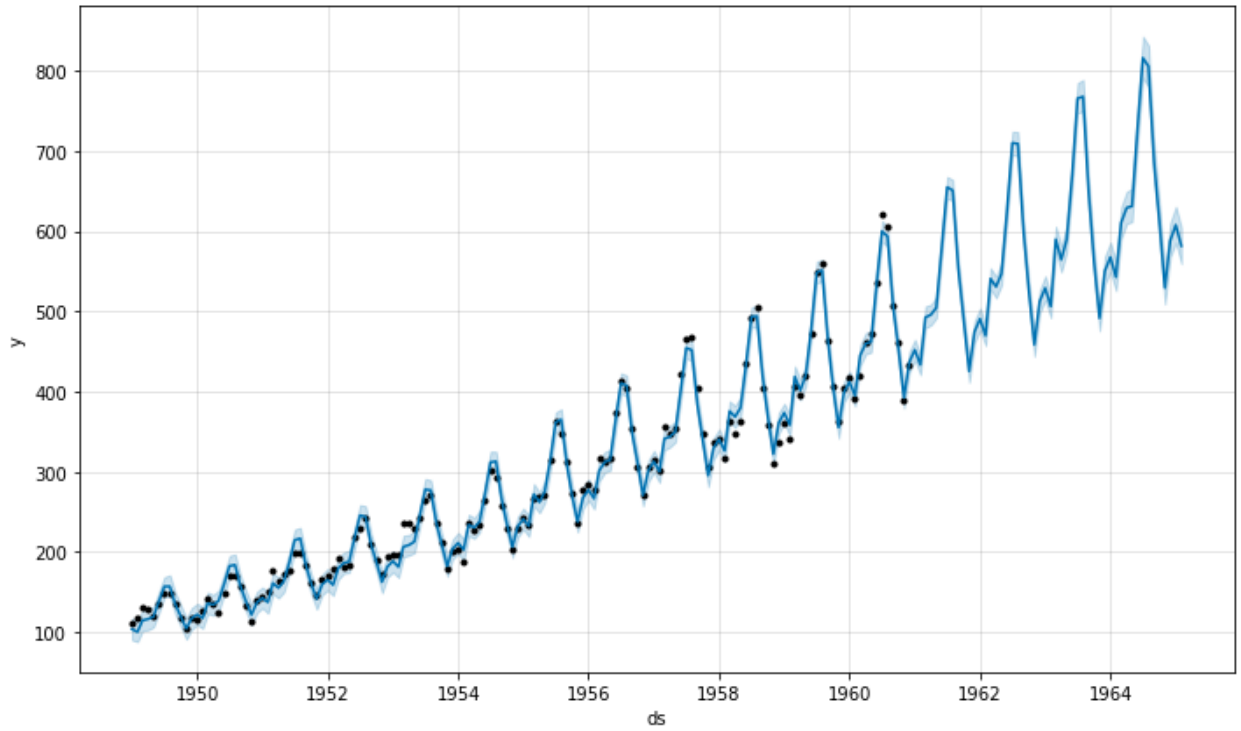


Figure 14. Exemple d'une saisonnalité multiplicative (nombre de passagers aériens à travers le temps). Ce jeu de données à un clair cycle annuel (avec une augmentation du nombre de passagers en été) et l'effet de cette saisonnalité augmente avec le temps. La saisonnalité s(t) n'est donc plus une constante ajoutée à la tendance g(t) (additive) comme Prophet le fait par défaut, mais elle est multipliée à la tendance (multiplicative).

Résultats obtenus

Après avoir essayé beaucoup de combinaisons des différents paramètres, voici les valeurs que j'ai sélectionnées :

- changepoint_prior_scale : 0.1 (défaut : 0.05)
- seasonality_prior_scale : 10 (défaut : 10)
- seasonality_mode : 'additive' (défaut: 'additive')

J'ai obtenu ces valeurs de paramètres en inspectant visuellement le « bon » ou le « mauvais » ajustements des données par le modèle. Par exemple, avec une valeur de changepoint_prior_scale trop petite, la flexibilité de la tendance est trop petite, c'est-à-dire que la droite de tendance n'est quasiment pas autorisée à changer de direction niveau de chaque point de rupture. Ce qui donne une droite qui ne change pas de direction et donc une simple droite de tendance : on fait face à de l'*underfitting* (Figure 15).

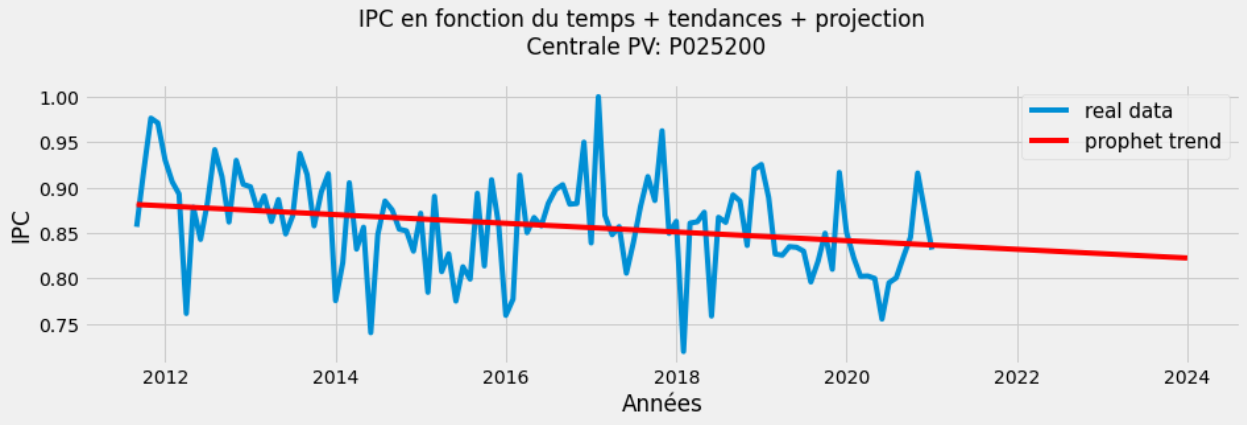


Figure 15. Exemple d'underfitting. Le paramètre changepoint_prior_scale étant trop petit (0.01), la droite de tendance n'est pas autorisée à changer de direction entre chaque point de rupture. Le résultat est une simple droite de tendance ne prenant pas en compte les changements de tendance dans les données.

Au contraire avec une valeur de 'changepoint_prior_scale' trop grande, la flexibilité de la tendance est trop grande, c'est-à-dire que la droite de tendance est autorisée à changer de direction de manière trop fréquente. Ce qui donne une courbe qui change constamment de direction : on fait face à de l'*overfitting* (Figure 16).

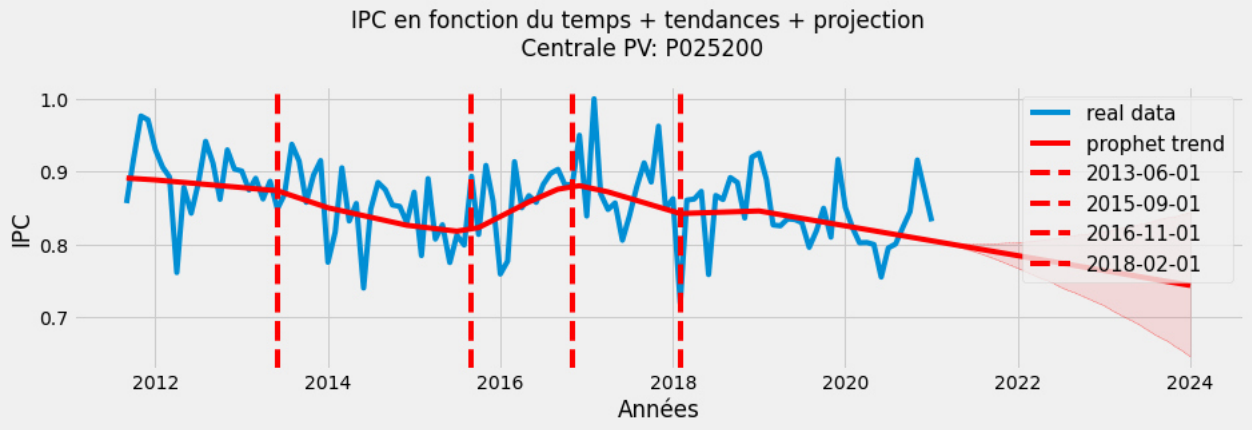


Figure 16. Exemple d'overfitting. Le paramètre changepoint_prior_scale étant trop grand (=1), la droite de tendance change de direction entre chaque changepoint (pointillés) de manière abusive. Le résultat est une droite de tendance qui fit trop les données et qui produit des ruptures de pentes de manière excessive. La zone rouge transparente indique un intervalle d'incertitude de 80% dans la projection faite dans le futur. A noter que cette incertitude augmente avec le temps.

J'ai sélectionné les valeurs avec lesquelles j'obtiens le bon juste milieu entre de l'*overfitting* et l'*underfitting*. C'est-à-dire que le modèle autorise suffisamment la pente à changer de direction sans pour autant changer de pente à tous les points de rupture. J'ai vérifié que ces paramètres donnaient des résultats cohérents sur plusieurs jeux de données de rendement de différentes centrales PV et également sur des jeux de données de « bruit » généré aléatoirement, censés représenter le comportement d'un rendement de centrale PV (changement de pente plus ou moins grossier, données très bruitées etc.).

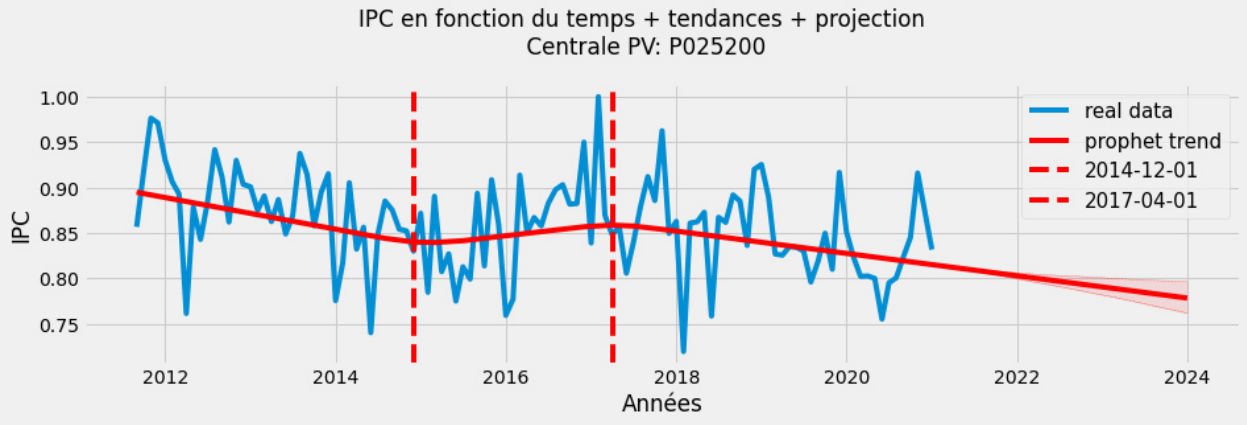


Figure 17. Fit et prédiction du modèle avec les paramètres optimums suivants : changepoint_prior_scale=0.1, seasonality_prior_scale=10, seasonality_mode='additive'. On retrouve les deux changements de pente qu'un opérationnels aurait pu tracer « à la main », mais avec en plus, les dates des changements de pente (décembre 2014 et avril 2017). Après le dernier changement de pente, le modèle projette la tendance du dernier segment de données (2017-04-01 jusqu'à la fin des données disponibles) sur les 3 prochaines années (période modifiable).

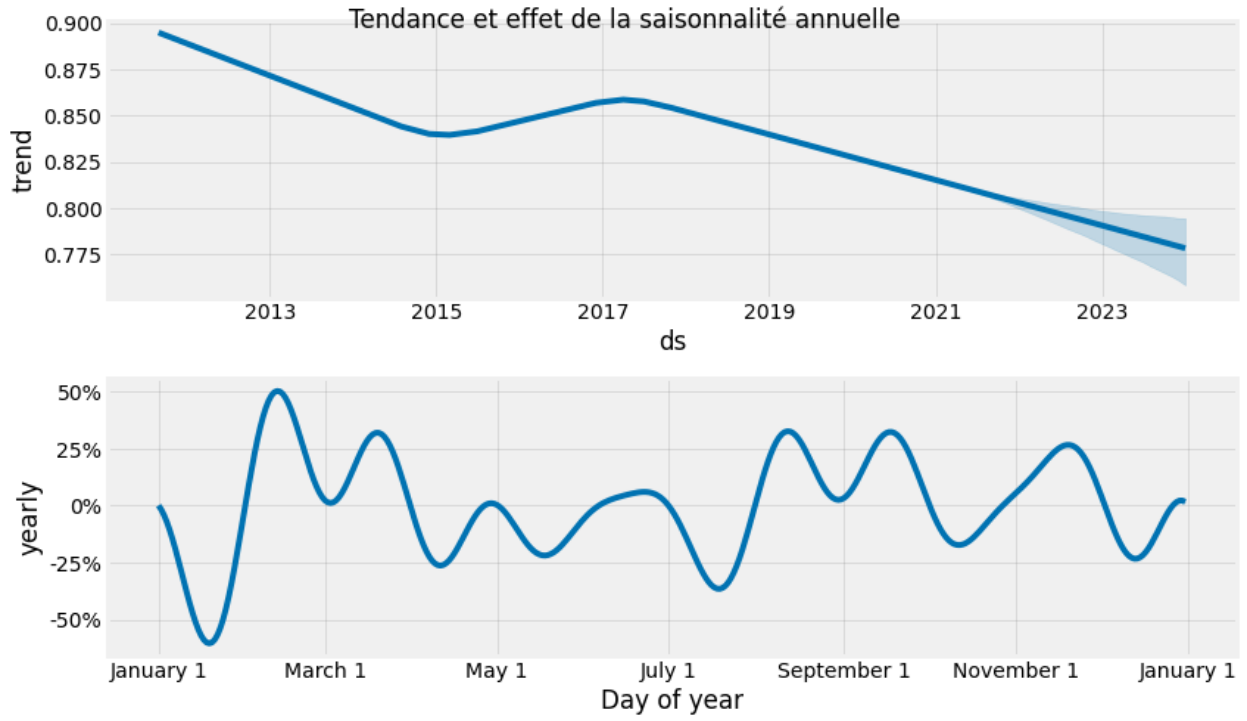


Figure 18. Tendance et saisonnalité annuelle affichée indépendamment. Dans le graphique du haut, on retrouve la tendance avec ses deux changements de pente de décembre 2014 et d'avril 2017 ainsi que l'intervalle d'incertitude dans la projection qui est faite. Dans le graph du bas, on voit la saisonnalité annuelle des données avec un pic de rendement aux alentours du mois de mars et un creux au mois de février.

Enfin, pour pouvoir quantifier l'évolution du rendement dans le futur, j'ai calculé une droite de régression linéaire sur la dernière partie des données (du dernier point de rupture

jusqu'à la fin des données) telle que $y = mx + p$ avec m le coefficient de pente de la droite et m l'ordonnée à l'origine. On obtient l'équation suivante : $y = -0.012036x + 25.13$.

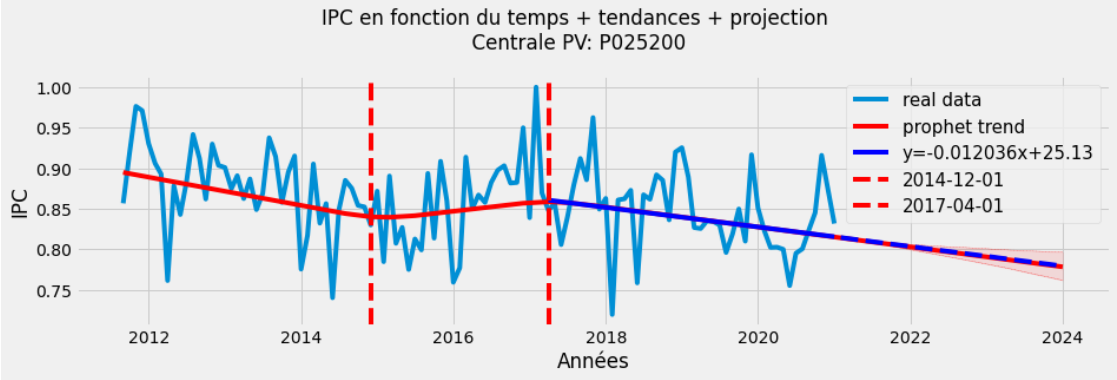


Figure 19. La régression linéaire vient s'ajouter au résultat du modèle. La régression linéaire a été calculée à partir du dernier rupture de tendance (avril 2017) jusqu'à la fin des données disponibles. Ses paramètres m et p permettent de calculer une valeur de rendement à une date donnée.

Une fois les coefficients m et p connus, il convient de remplacer x par une date pour obtenir un rendement. Par exemple, cette régression linéaire nous prédit un rendement de 0.76 en 2025 (-0.012036 * 2015 + 25.13 ~= 0.76).

Cross-validation

Une manière de vérifier que ma méthode prédit mieux les données qu'une simple régression linéaire comme utilisée jusqu'ici par les opérationnels est d'effectuer une cross-validation des deux méthodes. C'est-à-dire que pour chaque méthode, on va séparer les données en n blocs (chaque bloc de données suivant le précédent étant plus grand que le précédent). Sur chacun de ces blocs, on va prendre par exemple 80% des données pour entraîner le modèle et 20% pour tester le modèle. On calcule alors la RMSE (*root-mean-square error* ou erreur quadratique moyenne en Français) entre les valeurs prédites par le modèle et les données réelles pour juger de la précision du modèle. On répète cette opération sur chacun des n blocs. La précision finale du modèle est évaluée en calculant la moyenne de la performance du modèle sur chacun des x blocs (autrement dit, la moyenne de tous les RMSE). Cette méthode a l'avantage d'éviter l'*overfitting* et d'évaluer la performance du modèle de manière plus robuste qu'un simple *train-test* effectué une seule fois sur l'ensemble des données (Figure 20).

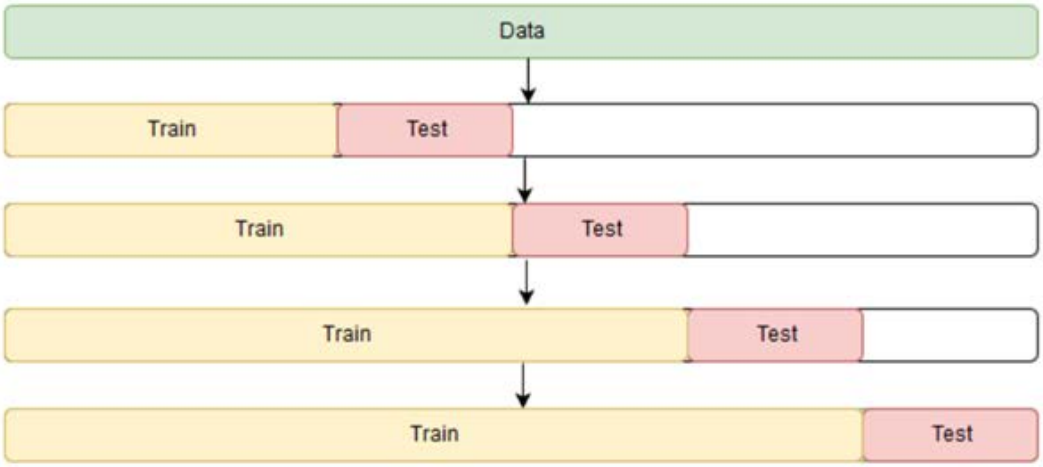


Figure 20. Exemple de séparation de données temporelles en 4 blocs pour effectuer une cross validation du modèle. Les données sont d'abord séparées en 4 blocs. Puis chaque bloc est divisé en deux avec une partie des données qui servira à entrainer le modèle et l'autre partie qui servira à mesurer la précision des prédictions du modèle. A noter que chaque bloc suivant le précédent contient plus de données que le précédent. (Source image : https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4).

Les RMSE finales que j'obtiens avec cette méthode d'évaluation de performance (n_blocs=3) sont les suivantes : 0.0634 pour la régression linéaire simple et 0.0535 pour ma méthode (soit une précision meilleure de 27% pour ma méthode). A noter que ces RMSE ont été calculées sur un seul jeu de données. D'autres jeux de données donneront des résultats différents :

Centrale	Type de données	Nb points de rupture	RMSE Régression	RMSE ma méthode	% performance Prophet (p/r à régression)
1	Non linéaire	2	0.0634	0.0535	+27%
2	Non linéaire	0	0.116	0.1255	-8%
3	Linéaire	0	0.0692	0.0692	+0%
4	Linéaire	0	0.0388	0.0388	+0%
5	Non linéaire	2	0.0787	0.0725	+8%
6	Non linéaire	1	0.1063	0.0583	+45%
7	Linéaire	0	0.1081	0.1081	0%

Tableau 3. Tableau récapitulatif des différences de performance de prédiction entre la régression linéaire simple et ma méthode pour différents jeux de données (n_blocs=3). Les couleurs verte, rouge et grise représente une meilleure, une moins bonne ou précision identique de ma méthode comparée à une simple régression linéaire.

Malgré des jeux de données différents, on remarquera une tendance de ma méthode à produire des prédictions plus précises qu'une simple régression linéaire lorsque les données ne sont pas linéaires (et donc qu'il y a au moins un point de rupture de pente (centrales 1, 5 et 6). Lorsque les données sont linéaires, mon algorithme fait aussi bien qu'une régression linéaire (ce qui est parfaitement logique puisque, quand il n'y a pas de point de rupture de pente, mon algorithme fait également ses prédictions en utilisant une simple régression linéaire sur toutes les données). On peut supposer que moins les données seront linéaires et

plus elles auront des points de rupture de tendance, plus ma méthode fera des prédictions précises comparées à une simple régression linéaire.

On notera toutefois une performance plus mauvaise de mon algorithme (-8% par rapport à une simple régression linéaire) pour la centrale 2 dont les données de rendement sont non linéaires alors qu'on pourrait s'attendre à un résultat équivalent aux centrales 1, 5 et 6 qui ont des données de rendement également non linéaires. Cette légère sous performance peut s'expliquer par du bruit dans les données de cette centrale. Ma méthode n'est donc pas parfaite mais semble fonctionner mieux qu'une régression linéaire la plupart du temps (à vérifier sur un plus grand nombre de centrales afin d'avoir un gain de performance plus fiable/reproductible).

Conclusion

Je propose une méthode algorithmique qui permet de détecter les tendances, les ruptures de tendances et les saisonnalités d'une série temporelle (ici, les valeurs mensuelles de rendement d'une centrale PV). Cette méthode d'analyse va plus loin que la simple régression linéaire utilisée jusqu'à présent puisque (i) celle-ci ne permettait pas de capturer les changements de tendance dans les données et donc, (ii) elle ne pouvait faire que de très grossières (voire fausses) projections dans le futur (quand les données ne sont pas linéaires).

La cross validation montre que ma méthode produit de meilleures prédictions qu'une régression linéaire classique lorsque les données sont non linéaires (lorsqu'il y a des points de ruptures de tendances dans les données). Lorsque les données sont linéaires, ma méthode simulera une régression linéaire. Cette méthode sera également facile d'utilisation pour les opérationnels puisque les paramètres du modèle ont déjà été sélectionnés pour être efficaces avec différents jeux de données de rendement et n'auront pas à être changé. Les opérationnels n'auront donc plus qu'à donner une date au modèle pour avoir le rendement futur de la centrale à la date donnée et pourront analyser le taux de dégradation annuel de la centrale.

Cette mission d'analyse de données souligne l'avantage d'utiliser un langage de programmation open source et tournée data science tel qu'est Python (ou R). En introduction je parlais d'une communauté Python riche et active et ce modèle Prophet en est la parfaite illustration. Ce modèle statistique a été implémenté sur Python et R seulement donc je n'aurais en aucun cas pu faire ce genre d'analyse sous un autre langage (Java par exemple).

Conclusion générale

Bilan du stage

Venant du milieu de la recherche, je n'avais eu aucune expérience professionnelle dans le milieu de l'entreprise avant cela. Ce premier stage en entreprise a donc été un premier aperçu de ce milieu dans lequel j'ai toutes les chances de travailler plus tard (en tant que data scientist, je l'espère). J'ai eu la chance d'être encadré par deux personnes ayant chacune eu une expérience dans le milieu de la recherche (Pascal Raux et Laurent Sauvage ont tous les deux un doctorat) et donc nous partageons en partie les mêmes codes, ce qui nous a permis de bien nous entendre et d'être en harmonie la façon de travailler.

Les missions qui m'ont été confiées rentraient pleinement dans mes aspirations professionnelles et m'ont permis de progresser en analyses de données sous Python et m'ont donné une première expérience concrète en data science. La liberté que l'on m'a laissé dans ce stage à chercher et expérimenter différentes méthodes a été très appréciable et a porté ces fruits puisque sans cela je n'aurais sans doute pas trouvé le package Prophet par exemple, qui convient parfaitement aux besoins métier. Cela m'a également permis d'apprendre par moi-même et de progresser considérablement en programmation Python.

Le Gantt final a beaucoup évolué depuis le Gantt initial. Premièrement, les missions se sont précisées pour ne s'articuler qu'autour deux grands axes : (i) l'extraction de données et (ii) l'analyse de données (versus (i) l'analyse de données, (ii) l'amélioration de la chaîne de traitement automatisée d'analyse de données onduleurs et (iii) support au diagnostic technique, comme indiquées dans le Gantt initial). Il s'agit en réalité plus d'une évolution de terminologie que de contenu de mission en tant que tel puisque j'ai bien contribué aux trois axes prévus dans le Gantt initial. Deuxièmement, l'écriture du rapport ne s'est pas faite tout du long du stage (comme généralement conseillé par les enseignants) mais en un bloc, les deux dernières semaines avant le rendu. Cela correspond à ma manière habituelle de travailler : je préfère ne faire qu'une seule tâche à la fois, sur laquelle je peux me concentrer pleinement, que de m'éparpiller sur plusieurs tâches et d'être moins efficace sur chaque tâche individuellement. J'ai procédé comme cela pour toutes mes tâches durant ces trois mois de stage (que ce soit pour l'extraction de données sur chacune des plateformes web, pour l'analyse de données avec Prophet ou pour l'écriture de mon rapport) et cela s'est très bien passé.

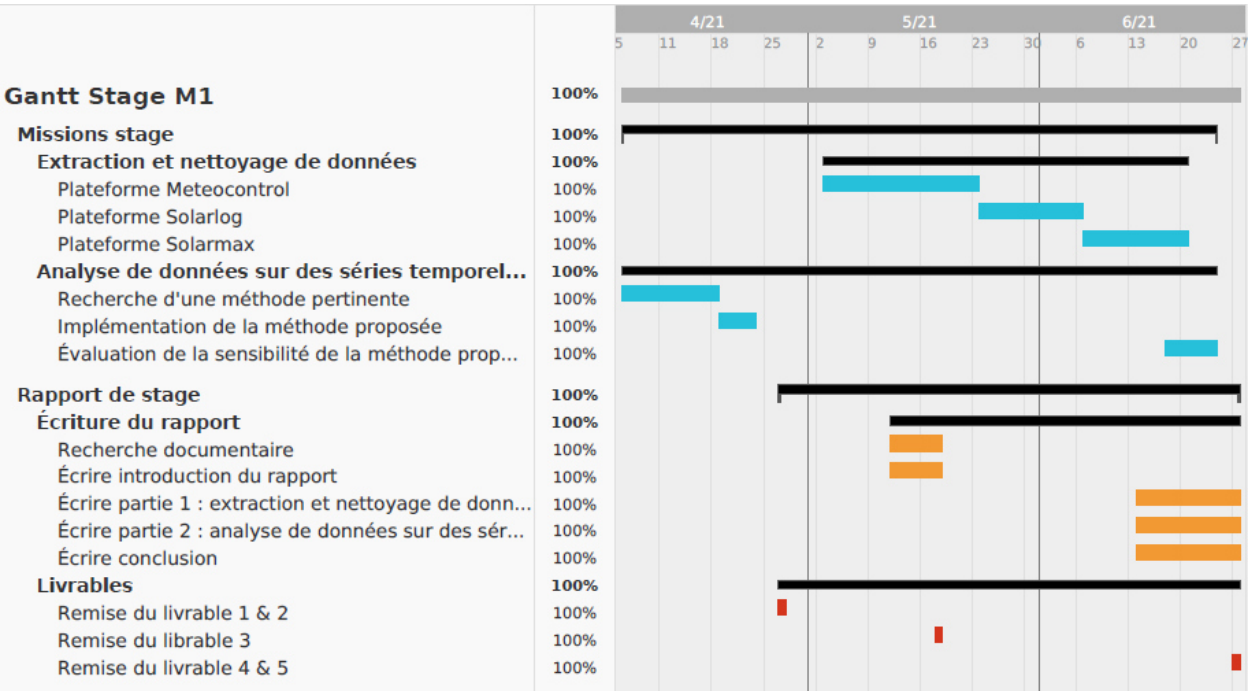


Figure 21. Gantt final montrant le planning de mes missions et de l'écriture de mon rapport de stage. On notera de grandes différences avec le premier Gantt notamment dans le contenu des missions et dans l'écriture du rapport (réalisation de ce Gantt final : 22 juin 2021).

Prolongement du stage

A l'issue de ces 3 premiers mois de stage, j'ai développé des algorithmes d'extraction de données et une méthode pour analyser les tendances et faire de la prédiction de rendement de centrales PV. Ces outils seront bientôt utilisés par les opérationnels pour les aider dans leur travail quotidien de gestion de centrales PV.

La suite de mon stage (jusqu'à fin août) se déroulera autour de sujets similaires, à savoir :

- l'exploitation d'une API dans la continuité du travail sur l'extraction de données onduleur sur une autre plateforme
- l'analyse et l'éventuelle adaptation d'un algorithme du NREL (laboratoire américain sur les énergies renouvelables) sur l'encrassement des panneaux PV
- et éventuellement, un travail sur un prototype d'un algorithme d'apprentissage (par exemple de l'apprentissage non supervisé pour faire de la détection d'anomalies de fonctionnement d'onduleurs)

BIBLIOGRAPHIE

Friedlingstein, P., O'Sullivan, M., Jones, M. W., Andrew, R. M., Hauck, J., Olsen, A., Peters, G. P., Peters, W., Pongratz, J., Sitch, S., Le Quéré, C., Canadell, J. G., Ciais, P., Jackson, R. B., Alin, S., Aragão, L. E. O. C., Arneth, A., Arora, V., Bates, N. R., … Zaehle, S. (2020). Global Carbon Budget 2020. *Earth System Science Data*, *12*(4), 3269–3340. https://doi.org/10.5194/essd-12-3269-2020

Groupe d'experts intergouvernemental sur l'évolution du climat, Stocker, T., Qin, D., & Plattner, G.-K. (2013). *Changements climatiques 2013: Les éléments scientifiques : résumé à l'intention des décideurs : rapport du groupe de travail I du GIEC : résumé technique : rapport accepté par le Groupe de travail I du GIEC mais non approuvé dans le détail et foire aux questions : extraits de la contribution du groupe de travail I au cinquième rapport d'évaluation du Groupe d'experts intergouvernemental sur l'évolution du climat*. Groupe d'experts intergouvernemental sur l'évolution du climat.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic Time Series Forecasting: The **forecast** Package for *R*. *Journal of Statistical Software*, *27*(3). https://doi.org/10.18637/jss.v027.i03

Lamport, L. (1979). *How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programm*. 2.

Nielsen, H., Fielding, R. T., & Berners-Lee, T. (1996). *Hypertext Transfer Protocol – HTTP/1.0*. RFC Editor. https://doi.org/10.17487/RFC1945

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, *72*(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080