Analyse et interprétation Ophélie Engasser
M1 - IA - Elearning - 30 mai 2023
Co-équipiers : Shabboo Aleagha, Khedoudja Rym Merad, Mike Duran

Les données dont nous disposons pour ce projet représentent **l'évolution des facteurs météorologiques en Inde**. Nous nous intéresserons plus particulièrement à la variable **"meantemp"** qui décrit la température moyenne à des temps différents. Si nous regardons le dataset, nous observons que chaque ligne représente un jour : **les données sont quotidiennes**. Nous commençons donc par fixer un dataframe contenant en index la variable temporelle (feature), et notre target "meantemp".

Partie 1. Analyses Time Step versus Lag Feature

Cette partie a pour but de nous faire comprendre la différence entre une analyse Time step, et une analyse Lag feature.

Time step. Time step signifie 'pas de temps' et représente l'intervalle régulier entre deux observations successives dans une série temporelle. Le pas de temps est ici de 1 jour. L'analyse Time step permet de modéliser la dépendance au temps, ici nous le faisons de manière linéaire, grâce à un modèle de régression linéaire (fonction **`LinearRegression()`** de Python), qui nous permet d'obtenir une équation de droite. Cette droite décrit la tendance, soit la direction générale que prend l'évolution des températures moyennes dans le temps.
Notre équation ici est **`Y = 0.0022705303257685404 * X + 23.836898252787844`** et la direction de la tendance nous est indiquée par le coefficient directeur (pente) de la droite. Il est positif, ce qui signifie que notre tendance croît au fil du temps, mais il est faible, suggérant une croissance très légère, permettant de prédire que c'est dans le sens d'une légère augmentation que les températures iront dans le futur.
La tendance étant une direction à envisager sur le long terme, nous pouvons proposer l'insight selon lequel cette tendance reflète le réchauffement climatique.

Lag feature. Une lag feature représente la valeur précédente d'une série temporelle à un pas de temps précisé. Ici nous choisissons un lag de 1, soit la température de la veille (n-1). Pour cela, il nous suffit d'opérer un shift, à l'aide de la méthode **`.shift(n)`** de Python, pour obtenir une série décalée selon le lag souhaité. Ce qui diffère ici par rapport à Time step, c'est que nous n'observons plus une relation entre la variable et le temps, mais entre la variable et elle-même (**autocorrélation**) : pour prédire les températures du futur, nous nous servons des températures du passé. A nouveau, grâce à un modèle de régression linéaire, nous obtenons l'équation suivante :
`Y = 0.9740674782673876 * X + 0.6614381865604209`
Mais c'est surtout le nuage de points qui nous intéresse car il suggère une corrélation entre la température et la température à n-1, nous laissant penser que les valeurs de température de la veille sont un bon prédicteur de celles de demain. Nous verrons dans la partie 4 si des lags à des temps plus lointains nous permettent d'effectuer des prédictions pertinentes.

Partie 2. Analyse de la tendance

Comme nous l'avons dit, la tendance renseigne sur la direction que prend une variable dans le temps, dans une visée à long terme. La première chose utile à faire pour analyser une tendance est de tracer un **graphique de moyenne mobile (moving average)** qui permet de réduire le bruit en lissant les variations d'une série temporelle, par le calcul d'une moyenne des valeurs sur une fenêtre fixée. Ainsi, nous visualisons les données brutes et les données lissées en même temps. Le choix de la taille de la fenêtre conditionne la sensibilité du lissage : une fenêtre plus grande lisse davantage le bruit et accentue les tendances à long terme, alors qu'une fenêtre restreinte conserve les variations à court terme. Mais l'idée est de pouvoir bien percevoir les changements à long terme.

Nous avons procédé à l'aide de la fonction **`rolling()`** qui prend le paramètre **`windows`**, et dont le résultat a été stocké dans une variable nommée **`moving_average`**.
Ici nous pensons que la fenêtre idéale est 365, afin de capturer les tendances sur une fenêtre d'une année, donc un assez long terme, tout en lissant les variations à plus court terme. L'objectif étant de lisser les saisons, et il nous apparaît sur notre graphique que cela semble être le cas. Nous observons une tendance de pente assez faible mais qui semble croître légèrement, ce qui rejoint le résultat de notre régression linéaire de l'analyse time step.

Une fois la forme de la tendance identifiée, nous avons modélisé la tendance en nous basant sur plusieurs fonctions (**partie trend engineering**). Pour cela, nous avons opéré une transformation de la variable temporelle, à partir des méthodes de deterministic process de Python. La fonction dédiée **`DeterministicProcess()`** permet de spécifier l'ordre du polynôme souhaité : 1 pour une tendance linéaire **`y = a * x + b`**, 2 pour le temps au carré (tendance quadratique, d'allure parabolique), 3 pour le temps au cube (tendance cubique, d'allure sigmoïdale), et ainsi de suite. La fonction **`LinearRegression()`** nous a ensuite permis d'ajuster un modèle à partir de ces caractéristiques. Il s'agit toujours d'un modèle de régression linéaire car il se base sur la méthode des moindres carrés ordinaires (OLS) visant à minimiser la distance entre les courbes ajustées et les données réelles.

Afin de trouver le modèle idéal, nous avons confronté chaque modèle en mesurant les performances sur des données test. Ce que nous pouvons dire en nous basant uniquement sur la visualisation, c'est que la tendance linéaire, d'ordre 1, nous semble la plus à même de capturer la tendance observée sur le moving average, soit de manière intuitive, ce qui pourrait refléter une augmentation des températures liée au réchauffement climatique. D'autant que des polynômes d'ordre supérieur peuvent évoluer rapidement sur des grandes valeurs du fait de la transformation appliquée par la puissance, ce qui nous paraît peu plausible concernant les températures.
Cependant, si l'on s'en tient à nos modèles ici présentés, il se trouve que le modèle présentant la meilleure accuracy est celui qui utilise l'ordre 6 pour prédire la tendance. C'est sur sa base qu'a été réalisé le forecast sur 15 jours, à partir de la fonction **`out_of_sample()`**.

Ce que nous pouvons en conclure : les métriques sont mauvaises et cela est normal car nous essayons d'ajuster un modèle de tendance à un nuage de points qui est en majeure partie expliqué par la saisonnalité. Par conséquent, nous pouvons dire que la tendance est intéressante pour donner une idée de la manière dont les données vont évoluer sur le long terme, voire le très long terme, mais ne permet pas de réaliser de manière pertinente un forecast à court terme, comme c'est le cas ici pour 15 jours.

Partie 3. Analyse de la saisonnalité

Dès le départ de notre travail, il apparaît assez flagrant que la série est caractérisée par des fluctuations qui évoquent une périodicité, une saisonnalité selon un rythme inhérent aux variations saisonnières de températures. La partie 3 a pour but de nous faire analyser cela de plus près.

Dans un premier temps, nous avons réalisé des **seasonal plot**, graphiques nous permettant d'identifier les motifs saisonniers dans la série. L'objectif principal étant de dégager des schémas récurrents. Nous avons ici tracé deux seasonal plot, à notre sens complémentaires.
Le 1er est un **seasonal plot week/day** qui permet de visualiser les motifs saisonniers dans les jours de la semaine, le 2e est un **seasonal plot year/day of year** qui permet de capter les variations sur les jours de l'année.
Il est intéressant de les confronter ici car : (i) sur le seasonal plot week/day : nous observons peu de variation des températures sur une semaine, et ce quelle que soit la semaine observée (ce qui est assez logique pour des températures), en revanche, nous observons une différence dans les températures entre les différentes semaines (on voit distinctement les semaines "chaudes", en vert sur le graph, et les semaines "froides", en rose), ce qui reflète une saisonnalité basée non pas sur un temps court, comme ici, mais sur un temps long, et ce 1er graphique n'est pas le meilleur pour capturer cela.
(ii) Le seasonal plot day/dayofyear, quant à lui, capte la saisonnalité que l'on voit distinctement basée sur un rythme été-hiver (donc annuelle), mais le 'bruit' ne nous permet pas de distinguer grandement les courbes représentant chaque année.

A partir de ces constats, nous nous sommes basés sur les méthodes des Fourier Features. Pour savoir combien de paires de Fourier inclure dans notre modèle, nous avons construit un **périodogramme**, dans le but d'identifier les fréquences présentant le plus de variance, donc de pertinence, dans notre série. Celui-ci nous indique une importante variance pour la **périodicité annuelle**, et une variance moindre pour la **périodicité semi-annuelle** : cela signifie que ce sont les fréquences qui contribuent le plus à la variation de la série temporelle, ce sont donc les motifs récurrents. Par conséquent, dans notre méthode Fourier features sur Python, nous avons utilisé la fonction :
`CalendarFourier(freq="A",order=2)`
où le paramètre `freq="A"` correspond à une modélisation selon des fréquences annuelles, et `order=2` correspond au nombre de paires de Fourier pour modéliser cette variation

annuelle. Chaque paire de Fourier comprend un sinus et un cosinus avec des amplitudes et des phrases spécifiques.
Ensuite, nous avons à nouveau réalisé un deterministic process sur ces caractéristiques, puis avons entraîné un modèle de régression linéaire, et à partir de ce modèle, réalisé un forecast à nouveau sur 15 jours, à l'aide de la fonction **`out_of_sample()`**. Le forecast est ici nettement plus intéressant car on voit qu'il imite une nouvelle période qui reprend à partir de 2017.
Il est à noter que nous avons retranché la composante de tendance de la saisonnalité et nous avons remarqué que à la fois le périodogramme et le modèle restent identiques : nous en avons conclu que le périodogramme ainsi que le dp ont saisi uniquement la saisonnalité. En revanche, lorsque nous avons enlevé la composante de saisonnalité, le périodogramme n'était plus en mesure d'indiquer de fréquence caractéristique.

Partie 4. Analyse des cycles

La composante de cycles se fonde sur l'idée vue à la partie 1 d'une autocorrélation de la variable 'meantemp' : la série n'est plus étudiée dans sa relation avec le temps, mais elle se boucle sur elle-même pour vérifier sur quel moment des données du passé de la variable elle-même nous pouvons nous baser pour effectuer une prédiction du futur. La première chose que nous avons réalisée est de tracer des **lagplots** de lag 1 à lag 12, afin d'identifier les corrélations avec les lags passés. Sur ces graphiques, nous observons que plus nous nous éloignons des lags 1 et 2, plus il est difficile d'établir une relation entre la variable et son lag, les nuages de points deviennent chaotiques et les coefficients de régression diminuent. Cela est bien visible sur **l'autocorrélogramme** : seuls les lags 1 et 2 apportent une variance et donc une information significative (le lag 1 plus fortement). Cela suggère que la veille et le jour d'avant sont de bons prédicteurs des températures d'aujourd'hui, au-delà, cela devient aléatoire. Mais si nous devions nous en tenir uniquement aux lagplots, nous aurions retenu uniquement le lag 1 car le coefficient de corrélation du lag 2 est inférieur à 70%, suggérant de rejeter l'ajustement linéaire. Mais cela dépendra bien-sûr de l'usage que nous ferions d'une telle prédiction, le lag 2 serait peut-être utile si nous devions prévoir une tenue pour un mariage ayant lieu dans 2 jours.
C'est donc sur cette base que nous avons réalisé une modélisation et un forecast. Nous y observons le décalage caractéristique du lag, et nous voyons que les deux courbes sont assez similaires, ce qui conforte notre analyse des corrélations. Ce que nous pouvons imaginer en voyant les cycles, c'est qu'ils reflètent des variations inhérentes au rythme jour-nuit, sans que cela constitue pour autant une périodicité stricte.

Pour résumer la démarche itérative réalisée de la partie 1 à la partie 4 : nous avons analysé chaque composante et l'avons retranchée une fois le modèle entraîné, restant alors les résidus de la composante précédente, soit tout ce que le modèle n'a pas été en mesure d'apprendre de la composante en question :
Tendance → y - y_pred → série sans tendance
Saisonnalité → y_detrended - y_pred → série sans tendance ni saisonnalité
Cycles → y_deseason - y_pred → série sans tendance, sans saisonnalité, sans cycles
Résidus

Partie 5. Les modèles hybrides

Jusqu'à présent, nous l'avons vu, l'algorithme utilisé a été à chaque fois la régression linéaire. La régression linéaire est optimale pour détecter et extrapoler les tendances, mais n'est pas forcément optimale pour détecter les interactions entre des features. C'est pour cela que nous avons utilisé une méthode hybride : il s'agit d'une combinaison de modèles censés améliorer les prédictions.

Combinaison 1. Pour détecter la tendance, nous avons repris le **modèle de régression linéaire** utilisé dans la partie 2 (ordre 6), mais nous avons ajouté une variable 'humidity'. Puis nous avons appliqué un **modèle XGBoost** aux résidus privés de tendance, dans le but de pouvoir étudier par ce modèle, non linéaire, le comportement des données résiduelles. Comme XGBoost ne peut produire des prédictions pour plusieurs target, nous avons au préalable empilé les données du dataframe grâce à la fonction **`stack()`** afin d'obtenir une table pivot. Nous apercevons que l'ajustement est relativement correct, suivant la saisonnalité, mais le forecast présente tout de même une erreur suggérant que XGBoost n'a pas été en mesure de compenser la partie tendance. Leur utilisation conjointe n'est donc pas suffisante.

Combinaison 2. Nous avons voulu tester si cela pouvait être meilleur avec une combinaison d'un **réseau de neurones artificiels (ANN)** à une seule couche et d'un **XGBoost**. L'accuracy rendue par le modèle étant de 85%, nous pouvons en déduire que cette combinaison est plus performante pour réaliser une prédiction des températures en intégrant toutes ses composantes.