INTERPRETABLE ADDITIVE MODEL FOR SOLAR IRRADIANCE ESTIMATION

A.Conte[a]

a. Ecole des Mines de Paris + alexandre.conte@mines-paristech.fr

Keywords : Explainable AI ; Solar forecasting

Abstract :
Recent studies on solar power forecasting have been using neural networks (NN) with sky-images. From these images taken by hemispheric cameras, the role of NN is to directly predict a value of either solar irradiance or PV production. Even though NN perform well, they lack interpretability, that is to say, transparency and robustness. They act as a black box since the way the prediction is made is hidden. In particular, it is hidden to field experts, who could be wary of the prediction. The ambition of this paper is to suggest a strategy to create an interpretable model, with an example in the field of solar forecasting. From domain-specific interpretability definition to fine-tuning of the model, each step is meant to stick with domain knowledge.

1 Introduction

1.1 Solar power plants and solar irradiance estimation

With the development of solar power energy, a growing need for reliable prediction of production emerged. Indeed solar panel production is highly dependent on cloud coverage, causing fluctuation of PV output. Using production time series and sky images, taken by hemispheric cameras, one tries to predict solar plant production in the near future. Many techniques have been developed to that end, some using Convolutional Neural Networks (CNN) on sky images directly [1]. A related challenge is to predict the Global Horizontal Irradiance (GHI), which is the total amount of solar radiation, received by a surface horizontal to the ground. Here again, it is possible to benefit from Neural Networks (NN) ability to convey non-linearity, for example by extracting features from sky images and then using an Artificial Neural Network for regression [2].

A sub-task of these two objectives (prediction of production or GHI) is to be able to precisely estimate GHI from a sky image. Here it is not forecast, rather a nowcast, i.e. " the prediction of the solar irradiance at the instance the frame is captured" [3]. This is motivated by two things: firstly pyranometers used to measure GHI on-site are expensive and lack precision for low irradiance. Secondly, this sub-task can be combined with any model that forecasts a sky image, using for example Long Short-Term Memory (LSTM) as in [4].

1.2 Need of interpretable models

As machine learning models are increasingly applied to various scientific domains and used for high-stakes decisions, users are right to ask for more transparency of these models. Indeed many machine learning or deep learning models used nowadays are considered black box since the way the model takes a decision is not known. As it is said in [5] "models' internal logic and inner workings are hidden to the user".Some such as CNN are too complex to provide an understandable link between input and output, and lack robustness. An invisible change for humans in the input can disrupt the output tremendously. See [6] for examples. In this context emerged the field of Explainable Artificial Intelligence (XAI). The first task at hand is to define "interpretability" in machine learning. Some consider that interpretability definition should be domain-specific [5].

Another question is how one evaluates the interpretability of a model, which depends on the task done by the model, and the public that can interact with the model's inputs and outputs. Evaluation can be for example application-grounded and thus involves " conducting human experiments within a real application" [7].

1.3 An interpretable model for solar irradiance estimation

In this paper, it was tried to address the need for interpretable models and more importantly for a methodology to conceive interpretable machine learning models. The case study chosen is the estimation of real-time Global Horizontal Irradiance from sky-images. In a few words, interpretability in this scientific field is defined, features are extracted from sky images, and then a model named Generalized Additive Model (GAM) [8] is built, with the interpretability definition as a guideline. The question of the research is as follows: Is it possible to build an interpretable model for solar irradiance estimation, whose performances are close to current black-box solutions?
This article is organized as follows: in 2. will be detailed every step required to build an interpretable model and the tools used at each step. In 3. results of the built model and main outcomes of the article will be discussed, and finally in 4. will be presented leads to improve the model.

2 Interpretable model, step by step

2.1 Interpretability definition

As a starting point, we will consider that interpretability needs to be defined relative to the domain of study. This is [5] conviction: " one should take into account the application domain and the use case of the problem in hand, as well as the type of audience that is asking questions about the model's decisions." In our problem, the use case is to be able to estimate Global Horizontal Irradiance (GHI) from sky images and it was considered that sun power experts would be the audience asking about the model's behavior.

Since model trains, one can consider that it makes experiments and learns from them. Progressively these results of experiments aggregate as knowledge about GHI estimation. Thus the criterion of interpretability is whether the model learns the same way an expert learns. Do they look at data the same way, and does the model captures the same knowledge that the one used by an expert? *In other words, a model is interpretable if every step of its estimation process refers to the expert's knowledge.*

Having discussed with sun power experts, two main scientific domains were identified: radiation physics and pattern matching. In the next steps, we will try to insufflate these two domains, to reproduce experts' estimation process as much as possible.

2.2 Data preparation

2.2.1 Irradiance data

Our model of GHI estimation requires two data series: GHI values and sky images. Thanks to the company *Solais*, I had access to these data streams from a solar panel plant in Mali. Both pictures and GHI measurements are done every minute from sunrise to sunset, every day from May 2020 to October 2020. To observe meaningful variations of the cloud coverage, it was decided to use only images and measures every five minutes.
GHI was measured on four locations of the plants, thanks to four pyranometers. Due to harsh field conditions in Mali, all pyranometers were not operational all the time, thus we kept moments when at least one pyranometer was working. Given the spatial coordinates of the camera and the sensors, it was estimated that the mean was a good estimation of the GHI at the camera location.

Furthermore, to reduce the seasonal variability of GHI values, it was chosen to work with K_c :

$$K_c = \frac{GHI_{mean}}{GHI_{clearsky}} \tag{1}$$

where $GHI_{clearsky}$ is an estimation of GHI under clear sky conditions.

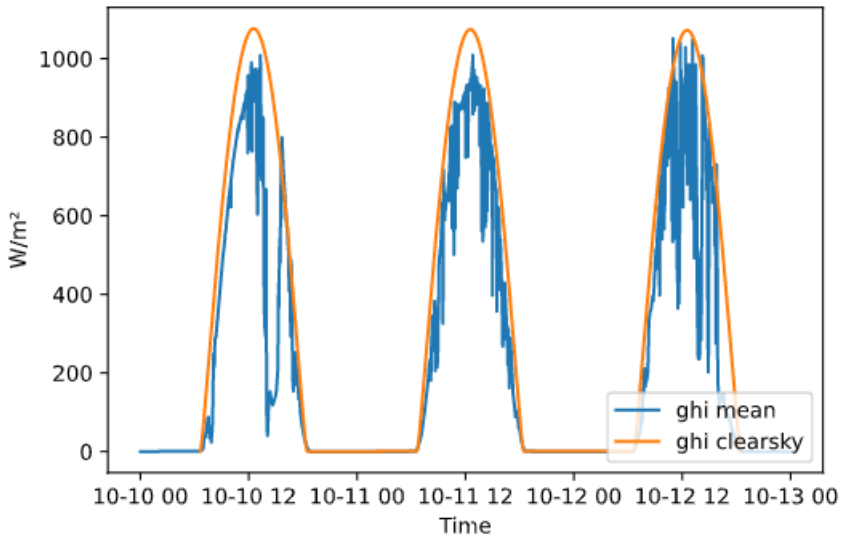

Figure 1 – *GHI mean and GHI clearsky values during 3 days*

Plenty of models exist to estimate irradiance in clear sky conditions, but we used the Ineichen-Perez model provided by the pvlib python library [9]. Figure 1 presents $GHI_{clearsky}$ and GHI_{mean} graphs for three consecutive days. This is a parametric model influenced by atmospheric turbidity which describes the transparency of the cloudless atmosphere. K_c values should always be in $[0, 1]$ since irradiance rises as the sky gets clearer. Having this in mind proved helpful because at first in more than 30% of the dataset, K_c was above 1. The reason was that the turbidity was not corresponding to real climate conditions. After adjusting this parameter, K_c values above 1 represent less than 1% of the dataset.

Then, since pyranometers have high uncertainty for low irradiance, all moments when $GHI_{clearsky} < 20W.m^{-2}$ were discarded. Following [10] recommendations on quality control procedures for solar radiation measures, and considering the time series available, K_c was clipped to 1.2. Finally, it seemed that the PV installation was not fully operational before mid-June, and to have relevant values to build our model, we should keep data corresponding to a stationary PV installation. In the end, we obtained 17815 K_c values between 20 June and 10 October 2020. K_c's distribution is shown in Figure 2.

2.2.2 Sky images

Sky images were taken from a hemispheric camera pointing at the zenith, above solar panels. The camera installed is manufactured by Mobotix, a company specialized in security cameras. First 960x1280 images were cropped into square images and then a mask has been applied to keep only the sky zone of the image. The result is shown in Figure 3 and Figure 4.

2.2.3 Image processing and sky zone delimitation

At first, it was considered to extract features from "undistorted" images. Indeed, compared to a plane projection, images taken by a fisheye camera seem distorted. It was motivated by the fact that standard features formulas are thought for plane projection, and also by the fact that "a fisheye lens produces severe non-linearities, particularly near its edges." [11]. Using the OpenCV package for Python and

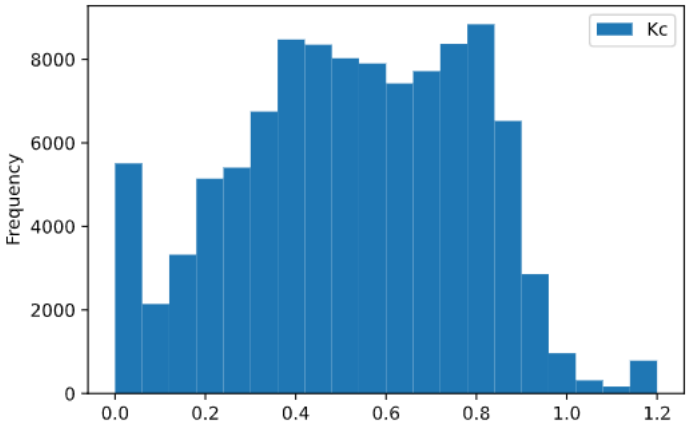

Figure 2 – *Kc distribution for 17815 measurements*

Figure 3 – *Original image*

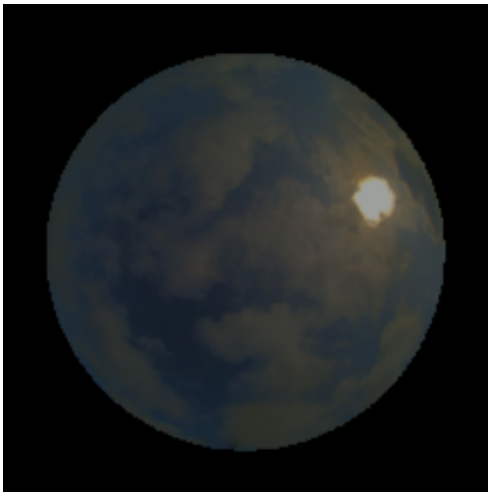

Figure 4 – *Cropped and squared image*

specific images, we obtained intrinsic parameters of the camera. It was then possible to "undistort" images, that is to say, to project them in a plane representation.
Unfortunately doing so, we inevitably lose image periphery. Moreover, since our objective is to ensure that each step in the model makes sense to an expert and that the estimation process is as close as possible from the one of an expert, a plane projection would not be relevant.

From RGB images we extracted two images: grayscale image and normalized R-B ratio image ($\frac{R-B}{R+B}$). Normalized R-B ratio is commonly used since it discriminates efficiently sky and cloud pixels. Then, to reproduce an expert's analysis, who takes into consideration the sun position on the image, three sky zones centered on the sun were delimited. For example, to observe the Direct Normal Irradiance (DNI) - the irradiance received by a surface that is held normal to the rays that come in a straight line from the direction of the sun - an expert will focus on the circumsolar disk. Thus, we computed features separately in each sky zones.

We also added three zenith-centered zones. The reason for this is that in a situation of overcast sky, Diffuse Horizontal Irradiance (DHI) - the irradiance received by a horizontal surface that has been scattered or diffused by the atmosphere - presents a distribution centered on zenith. This can be seen in Figure 5, provided under copy-permissive license by [12]. We introduce a priori knowledge about *solar radiation* in the model input.

We thus delimited six sky zones on our sky images (as in Figure 6).

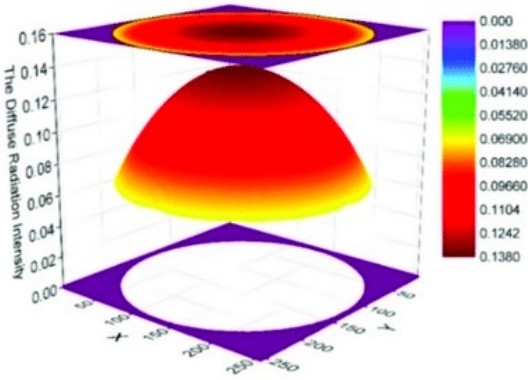

Figure 5 – *image-based diffuse radiation intensity field, for overcast sky. Image from Southern Great Plains atmospheric observatory in Oklahoma, United States, at 19:37 (UTC) on 7 July 2008*

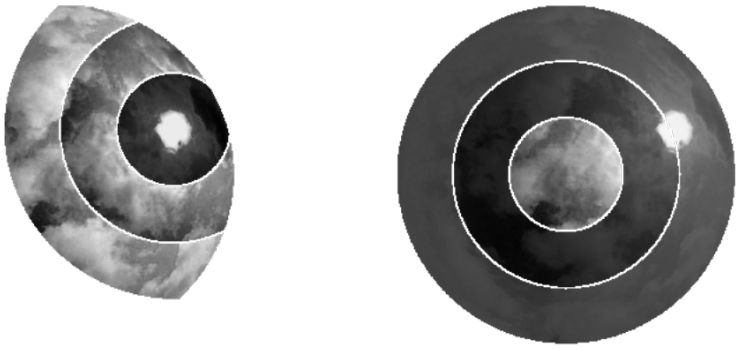

Figure 6 – *The six sky zones*

2.3 Feature extraction - Inputs of the model

Features are either statistical features (mean pixel value, standard deviation, etc..) or spectral features (computed on the image spectrum), or lastly cloud coverage-related features (cloud ratio, cloud thickness, etc..). These classes of features are proposed in [13], and in other articles that extract features from sky images to classify cloud types. It also would have been possible to compute texture features using co-occurrence matrixes, as it is made in [14].
Statistical features are computed in the six sky zones and on three versions of the grayscale image: the original, the low-filtered image, and the high-filtered image. Features computed on the blurred image capture general tendencies and features captured on the high-filtered image describe the details of the shapes.

2.3.1 Image's spectrum

Motivated by the choice of features made in [13], we computed the power spectrum of the sky images. Then we computed the relative energy in three frequency domains. We obtained low- mid- and high-frequency energy. For example, a significant high-frequency energy highlights that the image presents scattered patterns, such as a herd of small clouds. This spectrum distribution is computed on both grayscale and R-B image. See an example of spectrum in Figure 7.

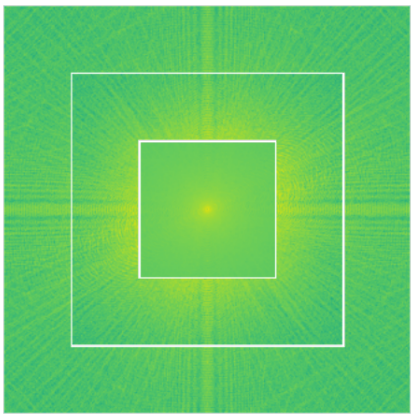

Figure 7 – *Example of image power spectrum, with three bins*

2.3.2 Cloud segmentation

Since experts use *pattern matching* to recognize a cloud coverage and match it with an estimated GHI, some of the model inputs need to focus on the cloud coverage. The task at hand was thus to distinguish cloud and sky pixel-wise. The method used was watershed: it treats pixel values as heights on a map and finds the lines that run along the tops of ridges. To segment one image, two objects are required: markers - pixels that can be identified with certainty as sky or cloud - and an elevation map - a transformation of the original image that highlights shapes boundaries. An efficient elevation map is obtained by passing the grayscale image through a high-pass filter. Concerning markers, the objective is to find a decent threshold for the R-B image, and then to allocate a class to pixels that are far greater/lower than the threshold. For pixels that are too close to the threshold, we let the watershed algorithm classify them.

Obviously, pixel values are impacted by their distance to the sun, so a fixed threshold is not feasible. One could try to determine the dependency between pixel value and distance to sun, for example by regression. Our approach was to introduce a small decision tree and local thresholds.
First, we used the standard deviation of the elevation map as an indicator of the image uniformity (the lower it is, the more uniform the image is). If the image is considered as uniform (that is to say below a fixed threshold), we used the sun isoperimetric ratio: it measures how far from circular a shape is. This ratio is defined by $r_{iso} = \frac{4\pi A}{P^2}$, with A the area and P the perimeter, and verifies $0 \leq r_{iso} \leq 1$ (reaches 1 for a circle). If this ratio is close to 1, it means that the sun is clearly visible and identified during the calculation of the ratio, and thus it is considered that we are in a clear sky situation. Otherwise, the image presents an overcast sky.
If the image is not considered uniform, we take advantage of the sun-centered sky zones introduced previously. Indeed, they can be used to calculate thresholds between sky and cloud pixels locally. We used either simple thresholds : $t_{loc} = \mu_{loc} \pm 0.1 * \sigma_{loc}$, where μ_{loc} is the local mean of pixel values and σ_{loc} the local standard deviation; or cross-entropy based thresholds. The cross-entropy maximization threshold presented in [15] divides pixels histogram in two groups to maximize the cross-entropy H between the two groups. This notion comes from information theory and cross-entropy can be defined as follows in Equation 2:

$$H(C, S) = \sum_x c(x) * log\ s(x) \tag{2}$$

where s and c are the sky and cloud pixels frequencies for the bin x. Using these thresholding techniques and the small decision tree explained above enabled us to segment images as shown in Figure 8.

Once we obtained segmented images, several key features could be extracted such as cloud ratio, cloud disparity, or cloud brightness. Cloud disparity measures the "fragmentation" of clouds. As one can see in Figure 8, the segmentation performs badly far from the sun. From one image to another, the quality of the segmentation varies a lot. In the case where the sun is surrounded by a halo because of aerosols,

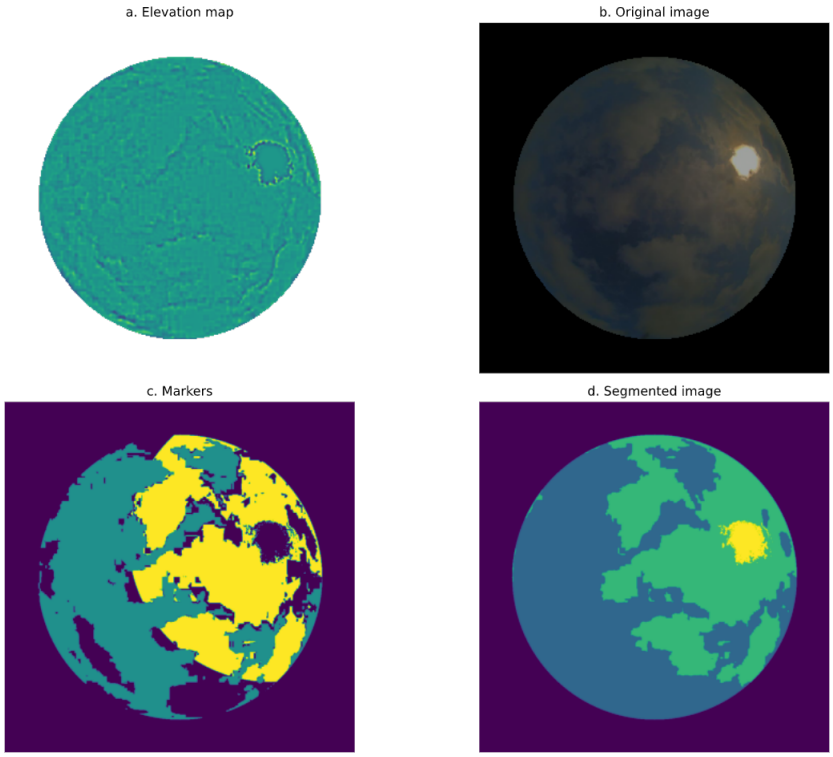

Figure 8 – *The elevation map and the markers are used to produce a cloud-segmented image*

segmentation performs poorly and identifies this halo as clouds. To counter that we also computed a cloud isoperimetric ratio, which indicates if the pixels labeled as clouds form a circular shape or not. For all these reasons, it was decided not to calculate cloud features on sky zones but rather on the whole image.

The complete list of features is displayed in Table 1, 2 and 3, with an explanation. It makes a total of 100 features because some features were calculated on sky zones and on different "versions" of the image (for example "gray concentric1 mean" is the mean of pixels contained in the first concentric zone of a grayscale image).

During the extraction process, the main objective was to collect enough relevant information from sky images. More importantly, to build an interpretable model, features extracted had to make sense for an expert. It is interesting here to make a comparison with CNN: in many CNN architectures, the first part is called encoding, and the information contained in the image is reduced to a vector that characterizes the image in a latent space. When one uses a CNN, he let it extract features "automatically", whereas our approach is close to "feature engineering". Our choice of features may prove less efficient for predicting K_c, but those features remain interpretable and are backed by a scientific approach. It is the typical trade-off between performance and interpretability underlined in [5].

2.4 Feature selection

As mentioned previously, 100 features are extracted from the sky images. To keep the prediction understandable and to reduce the time to train the GAM (Generalized Additive Model), we needed to select features that tell us the most about the sky situation (brightness, cloud cover, direct or diffuse irradiance, etc..). Furthermore, since the model is additive, we had to ensure that the features selected are not correlated with each other. It should be noted that dimension reduction techniques like Principal Components Analysis (PCA) was not an option since it creates features that do not convey a physical

Name	Description
mean	mean intensity
standard deviation	measure of contrast in the image
smoothness	equals 0 for an image of constant value, 1 for an image with large variability
skewness	measure of the asymmetry of the pixel distribution
entropy	measure of the randomness in the pixel values

Table 1 – *Statistical features, calculated on gray scale images, and on the 6 sky zones*

Name	Description
low band energy	Proportion of the image's energy located at low frequencies
mid band energy	Proportion of the image's energy located at mid frequencies
high band energy	Proportion of the image's energy located at high frequencies

Table 2 – *Spectral features, calculated on power spectrums of gray scale and R-B images*

Name	Description
cloud ratio	proportion of the image covered with clouds
cloud disparity	ratio between cloud border pixels and cloud pixels.
cloud isoperimetric ratio	measure the circularity of the cloudy zone
cloud brightness	measure the mean value of cloud pixels

Table 3 – *Cloud related features. Calculated on segmented images*

meaning. Thus, we evaluated dependencies between our features, applied different techniques to order features by relevance, and then we made these techniques vote.

2.4.1 Paired correlation - Greedy elimination

The idea of this approach is to iteratively eliminate features with respect to their correlation to other features. Therefore, the feature pair with the highest absolute correlation coefficient is selected. The feature of this pair that has the lower correlation with the K_c is eliminated. This procedure is repeated until no features are left.

The reverse order of elimination shall be the order of selecting features. For example, if five features shall be selected for prediction then the five features which were eliminated last are to be chosen. An extract of feature pairs - the least correlated ones- with their correlations, is displayed in Table 4. This gave us a ranking of feature importance, but to strengthen the selection, we decided to take into account other criteria than paired correlation.

Feature a	Feature b	Correlation
gray_low_pass_concentric2_std	gray_low_pass_circum1_skewness	0.000262
gray_concentric2_mean	gray_circum1_skewness	0.000254
high_pass_concentric2_skewness	high_pass_circum1_smoothness	0.000219
gray_low_pass_circum1_mean	gray_low_pass_circum1_skewness	0.000154
gray_concentric1_smoothness	cloud_disparity	0.000112
gray_low_pass_circum0_std	high_pass_concentric2_skewness	0.000065

Table 4 – *Least correlated pairs of features of the dataset*

2.4.2 Lasso Regularisation as feature selector

The Lasso model is a linear regression model with a regularization term that acts as a feature selector. With X the observation matrix, y the ground truth, then the Lasso estimator of the regression coefficients is :

$$\hat{\beta}^{lasso} = \arg\min_{\beta} ||y - \beta X||_2^2 + \lambda||\beta||_1 \tag{3}$$

λ is the regularization parameter. The more it increases, the more coefficients there are that are worth zero since the L^1 norm is penalized. Thus it indicates which features are less relevant for the prediction, as they are progressively set to zero. Also, the regularization reduces the influence of correlated variables on the model because the weight is shared between the two predictive variables, so neither alone would have strong weights.

Figure 8 shows how many features were selected with respect to λ. As λ raises, fewer features are

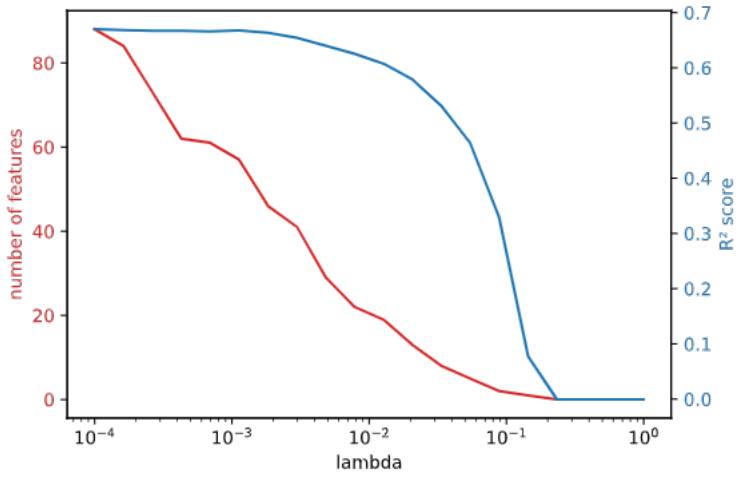

Figure 9 – *Influence of λ on non-zero regression coefficients*

selected. We can see that even if we only keep 20 features, the R^2 score would only drop from 0.68 to 0.65. We needed to determine which features are dropped first as λ increases. Since we trained multiples models with different λ: for each feature, we took the mean of the coefficient's absolute values over the models. We only kept coefficients from models whose $\mathbf{R^2}$ is higher than 0.5. Indeed, as the Lasso's performance decreases, the feature selection thanks to this model is becoming less relevant.
This method gave us another ranking of the 100 features.

2.4.3 Insights of mutual information

The mutual information between two random variables X and Y is a complementary tool to co-variance, to examine dependency. Mutual information is defined in Equation 4, with probability mass functions $p(x), p(y)$ and joint probability mass function $p(x, y)$.

$$I(X,Y) = E\left(\ln \frac{p(x,y)}{p(x)p(y)}\right) = \sum_{x,y} p(x,y)\left[\ln p(x,y) - \ln p(x)p(y)\right] \tag{4}$$

$I(X, Y)$ is thus a measure of distance from independence. Higher values mean higher dependency. For each feature of the dataset, we computed the mutual information with the target K_c. We used the algorithm proposed by [16]. This gave us the last ranking of features.

Finally, we combined the three rankings to obtain the final feature relevance order. An extract of the final feature ranking is displayed in Table 5.

Feature	Rank
gray_low_pass_circum0_std	1
rb_high_freq_energy	2
gray_low_pass_circum0_entropy	3
high_pass_circum0_entropy	4
gray_low_pass_circum0_skewness	5
sun_circularity_ratio	6
gray_circum0_mean	7
gray_low_pass_concentric0_std	8
gray_circum0_entropy	9

Table 5 – *First ten most relevant features*

As we can see in Table 5, the most important features according to the selection are calculated on the sun disk area (circum0). It is obviously reassuring since the Direct Normal Irradiance (DNI) - the main component of the GHI - locates itself in this area. The presence of features calculated on either the low-filtered, high-filtered or basic version of the images validates this choice of image filtering. The spectral feature in the second position is also worth noting: since the R-B image discriminates well sky from clouds, energy proportion in high frequencies band may characterize patchy clouds.

Being able to present to an expert a features ranking that is coherent with radiation physics theory (circum0 and DNI) and pattern matching logic (R-B high frequency and patchy clouds) contributes to the interpretability of the model.

2.5 Generalized additive model

During the previous steps we chose relevant features to extract from sky images, and we defined the target variable K_c. To keep the K_c estimation model interpretable, its inputs could not be too numerous. Therefore we had to set a small number of inputs without deteriorating the model's performance (this is detailed in 3). We also standardized them to reduce variability.

2.5.1 Presentation of GAMs

Generalized Additive Models were introduced by Hastie et al.[8] in 1999. For this work, we used the Python library developed by [17]. Formally, if we note X the matrix of features and y the target - supposedly following a distribution of the exponential family, the additive structure is defined by the following relation :

$$g(E[y|X]) = \beta_0 + \sum_n \beta_i f_i(X_i) \tag{5}$$

g is called the link function and f_i are named partial dependency functions. These functions are the strength of this model and one of the reasons it was chosen to build an interpretable model. Also, its additive nature makes it easier to grasp. In Figure 10 one can see an example of the partial dependency functions found by the model. The library pyGAM used also provides confidence intervals computed when fitting the partial dependencies.

These functions can either be linear, a tensor product between two features X_i and X_j or splines. Splines are piecewise polynomial functions often used in interpolation problems. Splines can be penalized, which means that the function is subject to various constraints: second derivative smoothing, monotonic or concave constraints, etc... This represents the opportunity to introduce prior knowledge in the model parameters by selecting the types of partial dependency functions (pdf) and the constraints/penalties. Of course, this would require the intervention of an expert - here in solar radiation - to build the model following his knowledge.
Our approach was rather to let the GAM train and observe if it "learned" the same things an expert

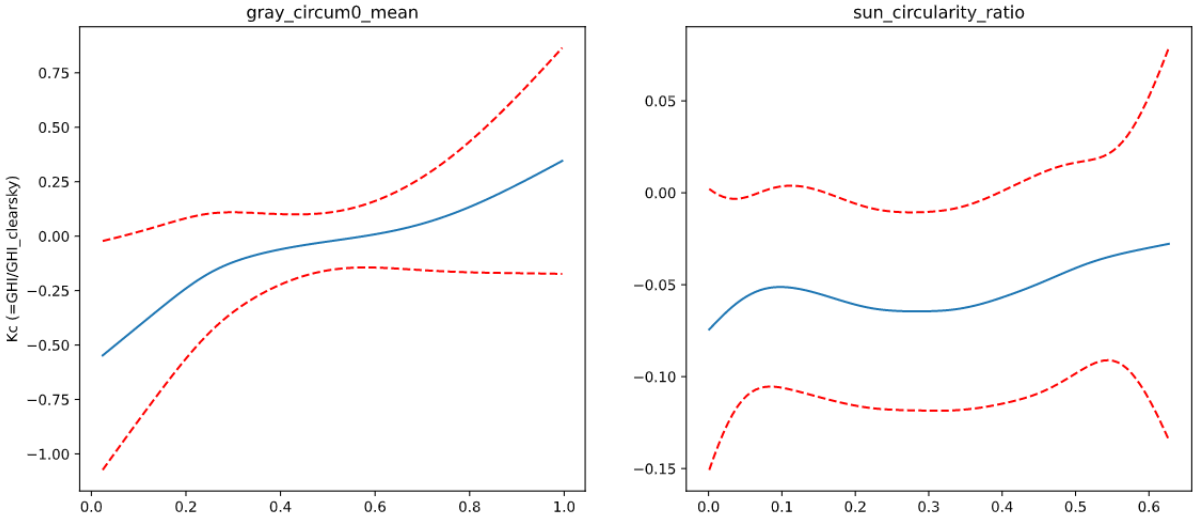

Figure 10 – *Examples of partial dependency function : variation of K_c with respect to mean pixel value in the sun area for the gray scale image (left), and to sun circularity ratio (right*

learns during his studies and experiments. Afterward, we would solicit an expert to improve the model from a baseline, making the task faster and easier for the expert. This approach comes back to one of the motivations of this paper: do machine learning models learn the same way experts do?

GAM baseline model was not performing well (parameters and performances of the model will be detailed in Section 3. Two possible reasons were identified: either the data can not be captured by a simple additive model and maybe paired interaction are needed, or the model parameters were not well chosen. Concerning paired interaction, they can be implemented when choosing the pdf. Here one drawback of GAM should be mentioned: the number of parameters to set.

2.5.2 Using Ridge Regression as a proxy

To find a solution to this problem of choosing many parameters, we used a linear regression model : Ridge regression. With X the observation matrix, y the ground truth, then the Ridge estimator of the regression coefficients is :

$$\hat{\beta}^{ridge} = \arg\min_{\beta} ||y - \beta X||_2^2 + \lambda ||\beta||_2^2 \tag{6}$$

λ is again the regularization parameter. It is close to the Lasso regression model, except that the regularization of regression coefficients uses the L^2 norm, constraining coefficients to small values, and reducing overfitting of the model.
The key aspect of this model relies on the vector of coefficients β, which gives us access to the weight of each feature in the prediction, as well as the sign of its impact.

Ridge regression could have been used as a K_c estimation model, but we rather used it as a *proxy*, in other words, a tool to enhance GAM's performances. As just mentioned, the Ridge model tells us the contribution of each feature, and we can use this to prioritize the pdf we want to improve in the GAM. For example, if the most important feature according to the Ridge model is *RB-high-freq-energy*, we examine the pdf found by the GAM for this feature first. This way we solved the challenge of the many parameters to set.

Another beneficial outcome of the Ridge model is that it performed well, and better than the GAM baseline. Firstly, it justifies that Ridge can be used as a proxy to improve the GAM. But it also answers the question of the hypothetical over-simplicity of an additive model. Indeed Ridge regression is a GAM with only linear partial dependencies and since it performs well: there is a high chance that with a good

tuning of GAM parameters we can reach better performances.
Let's mention that using the Ridge model alone is not appropriate since it lacks interpretability. Also, here Ridge was the most relevant proxy to use, but it is task-sensitive. The whole point is to find a simple model adapted to the problem: it must perform well and help investigate the GAM.

2.6 Collaboration with an expert

The final step in building an interpretable model that we propose is to confront it to an expert. This is mentioned in [7] : "application-grounded evaluation". But here it should not be seen as an evaluation step but rather as the fine-tuning step. This was not done in the context of this article, but it can still be presented to conclude our methodology of an interpretable model from start to finish.

The process would be as follows: once the GAM baseline and the Ridge model are trained on part of the selected features (for example the first twenty features, see Section 3 for more details); we use Ridge weights to choose which partial dependency function should be examined first by the expert. Using his knowledge in solar radiation, he can confirm or invalidate the pdf. More than that, he can suggest constraints, penalties, and types of dependency, that enable us to update the model parameters. For example, in Figure 10, the expert could validate the dependency with "gray-circum0-mean" (mean pixel value in sun area on the grayscale image), since it seems logical that K_c increases when this feature does. Finally, the expert plays a role in the feature selection, to improve the one made with statistical metrics in Section 2.4.

3 Main results

The results of this article are of two kinds: on the one hand: a GAM model and its Ridge proxy, which can be evaluated thanks to metrics used in regression, and on the other hand a methodology to build interpretable models.

3.1 A trade-off in performance

Metrics used to measure the performance of our model are common metrics in regression: Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and R^2 score. As mentioned previously, to keep the model interpretable and to reduce computation time, we had to choose the part of the 100 features to give to our model. It also speeds up the fine-tuning task with the expert. To choose the right number of features to keep from our ranking of features, we plotted the Ridge model performances as more and more features are used as inputs. The result is displayed in Figure 11. The regularization parameter was set to 0.01 through cross-validation, with a Leave-One Out approach. Our choice was to take 20 features, which is a trade-off we had to make between interpretability and performance.

Concerning the dataset, it was split into a training dataset and a test dataset in 80/20 proportion, with shuffling. We did not take into account the temporal aspect of the image and GHI dataset in the splitting, even though it is a valid choice. This is because our task at hand is to estimate the K_c from one image only.
Performances of the Ridge and the GAM are summarized in Table 6. The "Naive" model is a model that would estimate the K_c of any image as the mean K_c of the training dataset. The GAM parameters are relatively simple here: all partial dependency functions are regularized splines. The regularization parameters are chosen with grid search. As it was said, Ridge - a simplified GAM - performs better

Model	RMSE	MAPE	R^2
Naive	0.23	35 %	x
GAM	0.30	60 %	**0.68**
Ridge	**0.14**	**25 %**	0.60

Table 6 – *Performances of the models on the test dataset*

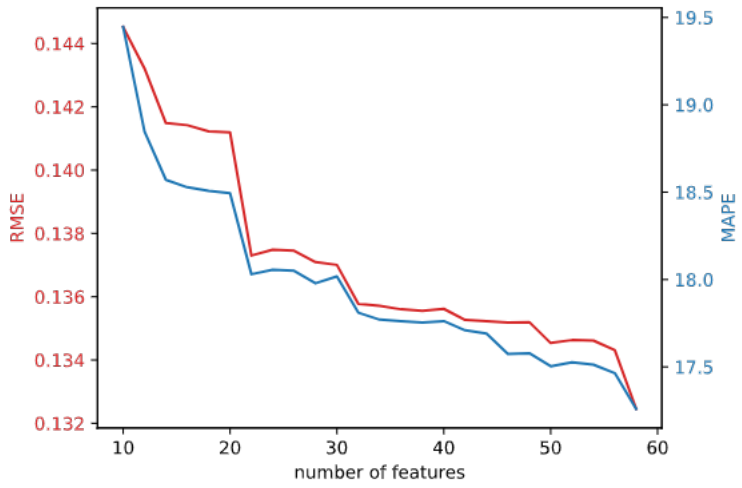

Figure 11 – *Performances of the Ridge model, with respect to the number of features given as input*

than the GAM, and thus there is a high chance that a well-tuned GAM will constitute a performing and interpretable model to estimate K_c from sky images.

3.2 A methodology for interpretable models

The second outcome of this article is to open a path for interpretable machine learning models, in technical fields. We are convinced that a model will meet interpretability requirements if the way the model predicts refers to the knowledge of experts.
The first step is to identify what constitutes this knowledge. Then one should introduce elements of this knowledge during the creation of the model. For example when defining features to extract, or when examining the relevance of features selected. Of course, one has to choose a model that we could qualify as transparent, like a GAM, or even simpler like a generalized linear model. After that, one should involve an expert in the final construction of the model.

Finally, the model's evaluation should be conducted through metrics for the performance aspect, but also through confrontation with experts to measure the interpretability of the model. This is an opportunity to measure the correspondence of the estimation process with expert knowledge, and the level of confidence in the model. This way, the model would be evaluated with both performance metrics and "interpretability metrics".

4 Outlooks and discussion

Throughout this article, we detailed how we tried to build an interpretable model, from the choice of features to the model evaluation. The outcome of this work would be to conduct the confrontation with experts once the performance of the GAM is sufficient. But are experts able to estimate K_c from sky images, or to evaluate precisely how much the model is faithful to their knowledge? Are questions formulated so that the answers provided can be used as an interpretability indicator? Indeed, there is no guarantee that a confrontation with experts - even with a well-prepared protocol - will reach meaningful conclusions. But since not much has been done in interpretability evaluation, it is worth trying.

As stated above, to proceed to a confrontation with experts, the GAM needs to reach better performances. Here we will underline some leads of what could have been done to improve it. First of all, it was quite frustrating that the cloud-related features were not ranked high by the feature selection methods. In Table 7 are listed the rank of the cloud-related features.

This is obviously related to cloud segmentation. As mentioned in Section 2.3.2, the quality of the cloud segmentation was average. It is not possible to evaluate it quantitatively since ground-truth segmented images are not available for the dataset used. To improve the segmentation, it could be possible to create a clear-sky database: this is done by extracting clear-sky images from the dataset, labeled with their date and time. Then, for each image from which cloud features need to be extracted, we find the nearest neighbor in the clear-sky database. The pixel-wise difference between those two images provides us with a cloud-only image, which is a lot easier to segment.

Cloud feature	Rank
Cloud ratio	22
Cloud disparity	27
Cloud brightness	30
Cloud isoperimetric ratio	55

Table 7 – *Rank of cloud features*

It was discussed that one can indicate paired interaction when building the additive model, thus creating a $f_{ij}(X_i, X_j)$ term in the GAM. To choose which features should be paired, we could have done a Sobol sensibility analysis. This is a variant-based analysis that decomposes the variance of the predicted feature into fractions and attributes them to inputs or sets of inputs. Second-order Sobol's indices measure how much paired interactions explain the variance of the output.
We also brought up that choosing an additive model may be a too strong assumption, even if paired interactions are permitted. One solution that could benefit from the feature extraction and be more permissive than a GAM is a Fully Connected Neural Network (FCNN). The FCNN would use the defined features as inputs, and one could compute partial derivatives with respect to each feature following the back-propagation rules. These partial derivatives then play approximately the same role as the partial dependency functions of the GAM.

Finally, there may be a case-agnostic tool to evaluate the level of learning of a model: the Minimum Description Length (MDL) principle. It was created for model selection, as a way to balance between model simplicity and the goodness of fit. According to [18], "learning" is equivalent to "finding regularities in the data". These regularities can be used "to compress the data, that is, to describe it in a short manner". To choose between two models that perform evenly, one can compute the description length of each model. The one with the shortest description length is the one that "learned" the most from the data.

4.1 Conclusion

In this paper, we addressed the issue of interpretability in machine learning. With a concrete example: K_c estimation from sky images, we first defined an interpretability criterion and then built a model that respects as much as possible this constraint. Each step of the process has been discussed through the spectrum of interpretability and transparency. The outcome is an additive model which has yet to be improved in collaboration with an expert, following the procedure detailed in Section 2.6.

Black-box models are one of the greatest challenges of Artificial Intelligence (AI) nowadays, and there is an urgent need for interpretability standards and guidelines.

Acknowledgment

I would like to warmly thank Philippe Blanc and Sébastien Travadel for their great support during the Trimestre Recherche in Sophia Antipolis. I learned a lot and enjoyed this first short experience of research. I also thank Christophe Vernay and Thomas Carrière from Solais, who took the time to discuss the subject of GHI estimation and gave me access to their company's dataset.

References

[1] A. Ryu, M. Ito, H. Ishii, and Y. Hayashi, "Preliminary analysis of short-term solar irradiance forecasting by using total-sky imager and convolutional neural network," in *2019 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia)*. Bangkok, Thailand: IEEE, Mar. 2019, pp. 627–631. [Online]. Available: https://ieeexplore.ieee.org/document/8715984/

[2] C. Crisosto, M. Hofmann, R. Mubarak, and G. Seckmeyer, "One-hour prediction of the global solar irradiance from all-sky images using artificial neural networks," *Energies*, vol. 11, no. 11, p. 2906, Oct. 2018. [Online]. Available: http://www.mdpi.com/1996-1073/11/11/2906

[3] T. A. Siddiqui, S. Bharadwaj, and S. Kalyanaraman, "A deep learning approach to solar-irradiance forecasting in sky-videos," *arXiv:1901.04881 [cs]*, Jan. 2019, arXiv: 1901.04881 version: 1. [Online]. Available: http://arxiv.org/abs/1901.04881

[4] V. Le Guen and N. Thome, "Prévision de l'irradiance solaire par réseaux de neurones profonds à l'aide de caméras au sol," in *GRETSI 2019*, Lille, France, Aug. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02948131

[5] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: a survey on methods and metrics," *Electronics*, vol. 8, no. 8, p. 832, Jul. 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/8/832

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv:1412.6572 [cs, stat]*, Mar. 2015, arXiv: 1412.6572. [Online]. Available: http://arxiv.org/abs/1412.6572

[7] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv:1702.08608 [cs, stat]*, Mar. 2017, arXiv: 1702.08608. [Online]. Available: http://arxiv.org/abs/1702.08608

[8] T. Hastie and R. Tibshirani, *Generalized additive models*. Boca Raton, Fla: Chapman & Hall/CRC, 1999.

[9] W. F. Holmgren, C. W. Hansen, and M. A. Mikofski, "Pvlib python: a python package for modeling solar energy systems," *Journal of Open Source Software*, vol. 3, no. 29, p. 884, Sep. 2018. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00884

[10] B. Espinar, P. Blanc, L. Wald, C. Hoyer-Klick, M. S. Homscheidt, and T. Wanderer, "On quality control procedures for solar radiation and meteorological measures, from subhourly to montly average time periods," Apr. 2012. [Online]. Available: https://hal-mines-paristech.archives-ouvertes.fr/hal-00691350

[11] W. Richardson, H. Krishnaswami, R. Vega, and M. Cervantes, "A low cost, edge computing, all-sky imager for cloud tracking and intra-hour irradiance forecasting," *Sustainability*, vol. 9, no. 4, p. 482, Mar. 2017. [Online]. Available: http://www.mdpi.com/2071-1050/9/4/482

[12] J. Du, Q. Min, P. Zhang, J. Guo, J. Yang, and B. Yin, "Short-term solar irradiance forecasts using sky images and radiative transfer model," *Energies*, vol. 11, no. 5, p. 1107, May 2018. [Online]. Available: http://www.mdpi.com/1996-1073/11/5/1107

[13] J. Calbó and J. Sabburg, "Feature extraction from whole-sky ground-based images for cloud-type recognition," *Journal of Atmospheric and Oceanic Technology*, vol. 25, no. 1, pp. 3–14, Jan. 2008. [Online]. Available: http://journals.ametsoc.org/doi/10.1175/2007JTECHA959.1

[14] R. Dambreville, "Prévision du rayonnement solaire global par télédétection pour la gestion de la production d'énergie photovoltaïque," phdthesis, Université de Grenoble, Oct. 2014. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01130251

[15] M. Malik, P. Spurek, and J. Tabor, "Cross-entropy based image thresholding," *Schedae Informaticae*, vol. 2015, no. Volume 24, pp. 21–29, Apr. 2016. [Online]. Available: https://www.ejournals.eu/Schedae-Informaticae/2015/Volume-24/art/7006/

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: machine learning in python," *arXiv:1201.0490 [cs]*, Jun. 2018, arXiv: 1201.0490. [Online]. Available: http://arxiv.org/abs/1201.0490

[17] "Dswah/Pygam: V0.8.0," Oct. 2018. [Online]. Available: https://zenodo.org/record/1208723

[18] P. D. Grünwald, *The minimum description length principle*. The MIT Press, 2007. [Online]. Available: https://direct.mit.edu/books/book/3813/the-minimum-description-length-principle