SHORT-TERM SOLAR IRRADIATION FROM A SPARSE PYRANOMETER NETWORK

ANNETTE ESCHENBACH

GRADO EN INGENIERÍA INFORMÁTICA, FACULTAD DE INFORMÁTICA, UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Grado en Ingeniería

8 de junio de 2018

José Ignacio Gómez Pérez
Christian Tenllado van der Reijden

Resumen

La predicción precisa de la radiación solar es necesaria para estimar correctamente la producción de energía de los sistemas solares fotovoltaicos y su integración en la red eléctrica. Este trabajo explora hasta qué punto las técnicas de *Machine Learning* pueden ser utilizadas para resolver este problema. La meta es predecir la radiación a corto plazo para un objetivo con varios horizontes de predicción. El objeto de las predicciones es una de las 22 estaciones de redes de piranómetros difusos con observaciones de muestra de resolución 30’. Se analizan las prestaciones y limitaciones de un modelo de *Support Vector Machine* simple y dos conjuntos de métodos de aprendizaje más sofisticados – *Random Forest Regression* y *Gradient Boosting*. Se muestra que todos ellos funcionan bien en condiciones climáticas constantes pero no realizan pronósticos fiables durante días en que las condiciones climáticas cambian rápidamente. Una selección inteligente de funciones es útil para hacer que el modelo sea más eficiente y rápido sin necesariamente mejorar significativamente la fiabilidad de los resultados. Con modelos agregados para escenarios específicos, se debe prestar atención a seguir algunas reglas para no aumentar innecesariamente la complejidad del modelo a expensas de la generalización de nuevos datos. Los modelos de entrenamiento en pequeñas cantidades de datos preseleccionados pueden causar sobreajuste o *overfitting*.

Palabras clave

Previsión a corto plazo de radiación solar, aprendizaje automático, minería de datos, árboles de decisiones, regresiones de bosques aleatorios, máquina de vectores de soporte, red de piranómetro difuso, Python, Pysolar, Scikit Learn, inteligencia artificial, aprendizaje de conjunto

Abstract

Accurate forecasting of solar irradiance is necessary for correct estimates of the energy output of solar photovoltaic systems and their integration into the power grid. This paper explores to which extent Machine Learning techniques can be applied to solve this problem. The objective is to predict short-term radiation for a target with various forecast horizons. The target is one of 22 stations of a sparse pyranometer network with sample observations of 30' resolution. The performance and limitations of a simple Support Vector Machine model and two more sophisticated ensemble learning methods – Random Forest Regression and Gradient Boosting are analyzed. It is shown that all of them perform well in steady weather conditions but fail to make reliable predictions for days with rapid weather changes. A smart feature selection proves useful to make the model more efficient and faster without significantly improving the reliability of the predictions. With aggregated models for specific scenarios one has to pay attention to follow some rules in order not to unnecessarily increase the complexity of the model at the expense of generalization on new data. Training models on small preselected data may cause overfitting.

Keywords

Short-term solar radiation forecasting, Machine Learning, data mining, decision trees, Random Forest regression, Support Vector machine, sparse pyranometer network, Python, Pysolar, Scikit Learn, artificial intelligence, ensemble learning

Table of Figures

List of Tables

TABLE OF CONTENTS

List of defined terms

ANN	Artificial Neural Networks
GHI	global horizontal irradiance
NREL	National Renewable Energy Laboratory
ML	Machine Learning
PCA	Principal Component Analysis
RBF	Radial Basis Function
SPA	Solar Position Algorithm
SVM	Support Vector Machine

1. CAPÍTULO 1 - INTRODUCCIÓN

1.1. Motivación y objetivos del estudio

En los últimos años, la energía solar se ha convertido en una fuente de energía barata y limpia y ha aumentado significativamente su participación en el suministro de energía global. Sin embargo, la irradiación solar depende de algunas condiciones climáticas incontrolables, como tener un cielo despejado. Como resultado, la incertidumbre aún representa un riesgo enorme para la estabilidad de la red eléctrica.

Históricamente, las infraestructuras de las redes eléctricas generalmente estaban diseñadas para que tuvieran niveles de electricidad relativamente constantes, de modo que la demanda y el suministro de energía pudieran coincidir exactamente. Sin embargo, la naturaleza de la energía solar significa que a menudo hay caídas repentinas o picos en el suministro de electricidad debido a los rápidos cambios de las condiciones climáticas, que crean grandes dificultades a los operadores de la red.

Las predicciones precisas de la irradiación solar exacta en un momento concreto pueden usarse para calcular la cantidad exacta de electricidad que se alimentará a la red. Luego, la generación de energía podría ser ajustada o se podrían activar una reserva de energía, según el caso. Las previsiones precisas pueden permitir la gestión de la capacidad de energía extra convencional (nuclear, de gas, carbón, etc.), sistemas de almacenamiento de batería y cargas controlables y también pueden ayudar a operar con electricidad fotovoltaica y con la gestión de plantas de energía.

La predicción es, por lo tanto, un factor crucial para integrar la energía solar en el sistema de energía a bajo costo. Sin embargo, las técnicas de predicción fiables siguen presentándose como un gran desafío.

Se han desarrollado modelos sofisticados de pronóstico de irradiación solar y se ha logrado una mejora significativa en la precisión de los pronósticos. Las dos grandes categorías de predicción solar son la utilización de sistemas de imágenes de nubes que rastrean el movimiento de la nube y las simulaciones/optimizaciones numéricas. Sin embargo, todavía hay mucho margen de mejora.

Este proyecto tiene como objetivo rellenar este vacío avanzando en la previsibilidad de la irradiación solar. Específicamente, los objetivos de este estudio son: (i) seleccionar un algoritmo de aprendizaje automático para predecir la irradiación a corto plazo para un objetivo basado en observaciones en tierra de una red de piranómetro difusa[1], (ii) entrenar este modelo, (iii) optimizar la selección de características, (iv) evaluar estos modelos predictivos con datos de prueba, y (v) analizar los resultados con sistemas de medición apropiados.

1.2. Estructura de la tesis

El Capítulo 2 presenta los métodos utilizados para el estudio. Después de una breve descripción general de los conceptos fundamentales del Aprendizaje Automático ("**ML**", por sus siglas en inglés), el Capítulo describe las principales categorías. A continuación describe con más detalle algunos algoritmos de ML para las tareas de regresión. De este modo, el foco principal se encuentra en los modelos ampliamente establecidos como el modelo *Support Vector Machine* ("**SVM**"), pero también en los métodos de conjunto *Random Forest* y *Gradient Boosting*, que rara vez se han utilizado para la predicción de la radiación solar. Además, se mencionan las bibliotecas de Python en relación con su implementación práctica. Finalmente, este Capítulo describe la base de datos utilizada, llamada "*Inforiego*", y explica en términos generales la idea detrás de este estudio.

El Capítulo 3 explica cómo se generan las características de entrada para el modelo utilizando Pysolar como un modelo de cielo despejado para transformar la irradiación absoluta en irradiación relativa. Se proporciona un resumen de los diferentes enfoques que se han aplicado para la selección de características y se describen los pasos que se han tomado en la preparación y en la limpieza de datos. Después de resaltar las cualidades específicas de los algoritmos elegidos, se describe su proceso de entrenamiento y evaluación, con énfasis particular de las *importancias* características derivadas.

El Capítulo 4 ilustra inicialmente la compensación sesgo/varianza en ML y propone varias mediciones para medir la precisión de los modelos ML. Se demuestra la idea de usar un

[1] Un Piranómetro es un instrumento para medir la irradiación solar utilizando una termopila generadora de voltaje que se excita por exposición a la irradiación solar [Zh17].

modelo básico como referencia. Los resultados de la evaluación de los modelos entrenados en el conjunto de prueba se muestran e ilustran con algunas parcelas. El capítulo se cierra con un enfoque para predecir escenarios fáciles y difíciles con un modelo agregado.

La conclusión resume los principales hallazgos de nuestro estudio y sugiere algunos enfoques interesantes que podrían explorarse más a fondo en el futuro.

2. CHAPTER 1 - INTRODUCTION

2.1. Motivation and Goals of the Project

In recent years solar power has emerged as a cheap and clean energy source and has significantly increased its share in the global energy supply. However solar irradiation depends on some still uncontrollable weather conditions such as having a clear sky. As a result, uncertainty still poses an enormous risk for the stability of the electricity grid.

Historically, the infrastructures of power grids were typically designed to carry relatively consistent levels of electricity such that power demand and supply could be matched exactly. However, the nature of solar energy means that often there are sudden drops or surges in electricity feed-in due to quick changes of weather conditions, which grid operators greatly struggle to cope with.

Precise predictions of the exact solar irradiation at any given time may be used to calculate the exact amount of electricity that will be fed into the grid. Power generation could then be adjusted or a reserve of power activated accordingly. Accurate forecasts may enable the management of extra conventional power capacity (nuclear, gas, coal, etc.), battery storage systems and controllable loads and may also help trading with photovoltaic electricity and scheduling powerplants.

Forecasting is thus a crucial factor for integrating solar energy into the energy system at low cost. Yet reliable prediction techniques remain a particularly challenging problem.

Sophisticated solar irradiance forecasting models have been developed and significant improvement in accuracy has been achieved. The two big categories of solar forecasting are the cloud imagery that tracks the cloud movement and numerical simulations/optimizations. However, there is still a lot of room for improvement.

This project aims at filling this gap by advancing in the predictability of solar irradiation. Specifically, the goals of this study are: (i) to select a machine learning algorithm to predict short-term irradiation for a target based on ground observations from a sparse pyranometer[2]

[2] A Pyranometer is an instrument for measuring solar irradiance using a voltage-generating thermopile that is excited by exposure to solar radiation [Zh17].

network, (ii) train this model, (iii) optimize feature selection, (iv) evaluate these predictive models on test data, and (v) analyse the results with appropriate metrics.

2.2. Thesis Structure

Chapter 2 introduces the methods used for the study. After a short overview of the fundamental concepts of ML, the Chapter outlines the main different categories, after which it describes in more detail some ML algorithms for regression tasks. Hereby the main focus lies in the widely established models like the Support Vector Machine model (“**SVM**”) but also in the ensemble methods Random Forest and Gradient Boosting that have rarely been used for solar radiation prediction. Also, the Python libraries for the practical implementation is mentioned. Finally, this Chapter describes the database used, called “Inforiego”, and explains in broad terms the idea behind this project.

Chapter 3 explains how the input features for the model are generated using Pysolar as a clear-sky model to transform absolute into relative radiation. A summary of the different approaches that have been applied for the feature selection are given and the steps of data preparation and cleaning described. After highlighting the specific qualities of the chosen algorithms, their training and evaluation process are described with a particular emphasis of the derived feature importances.

Chapter 4 initially illustrates the bias/variance tradeoff in ML and proposes several metrics to measure the accuracy of ML models. The idea of using a basic model as a reference is demonstrated. The results of the evaluation of the trained models on the test set are shown and illustrated with a few plots. The chapter is closed with an approach to predict easy and difficult scenarios separately with an aggregated model.

The conclusion summarizes the principal findings of our study and suggests a few interesting approaches that could be further explored in the future.

3. CHAPTER 2 - METHODS

3.1. Machine Learning

3.1.1. Fundamental Concepts of Machine Learning

Since the spamfilter became the first mainstream ML-application in the 1990s, machine learning ("**ML**") has conquered many domains; among them, speech recognition. Often classified as a subfield of artificial intelligence, there are several definitions of what ML exactly is. A less abstract and intuitive one is to think of ML as a "*method of building mathematical models*" ([Va16], p.332) to help understand data and give "*computers the ability to learn without being explicitly programmed*" ([Ge17], p.4). The principle is to specify some performance measure (usually a cost function) and give tunable parameters to the model that can be adapted to the real observed data. This process of "fitting" models to previously seen data is the training phase. During the test phase these models can be used to predict and understand frequent patterns of newly observed data.

Furthermore, ML expands the concepts of classical statistical inference to process huge amounts of data instead of only a small number of samples [Vo17]. These characteristics make ML particularly suited for forecasting or data mining problems that usually involve huge datasets, since ML models find solutions for problems for which it is extremely hard or impossible to implement a traditional algorithm. An algorithm-based solution would most likely end in a very long list of complex rules whereas an ML-model performs better and can automatically learn which features are good predictors of a given final result.

As ML learns from data, it is critical to: **first**, select a dataset with a sufficient quantity of quality data, and **second**, clean and prepare the data properly, so it can be fed into the algorithm. This also involves selecting and engineering the features that are most strongly related to the expected outcome. The learning algorithm searches for the model parameter values that minimize a cost function that measures the distance to the expected outcome.

3.1.2. Categories of Machine Learning

There are many different types of ML systems and algorithms, and as many approaches to categorize them. The goal here is to give a quick overview of the broad categories of ML methods at a superficial level, followed by a presentation in more detail of the methods that have been selected for our study within this context.

ML algorithms may primarily be grouped into two main categories: ***supervised*** and ***unsupervised*** learning:

(i) In **supervised** learning, labelled training data is fed into the algorithm, i.e. expected solutions. The input are called *labels*. *Classification* and *regression* are two typical supervised tasks within this category: The previously mentioned spam filter is a typical example of a *classification* supervised task, as the sample emails have to be put into two discrete predefined classes, spam or not spam. Regression models on the other hand predict continuous numeric target values given a set of features called *predictors*. The task of predicting the specific amount of solar radiation for a particular time and location falls into this category.

(ii) In **unsupervised** learning, training data comes in a more complex unlabeled form and the model has to identify a hidden structure within the data itself without anyone facilitating the "teaching/training". Clustering models detect distinct groups, called **data clusters**. Dimensionality reduction and visualization algorithms map higher dimensional data to a 2 or 3D representation without losing too much information. This model serves to make your data easy to plot, to assign labels to unknown data, and to significantly speed up computation. Many data mining methods to preprocess data are also used in unsupervised learning ([Vo17], p.12).

As in supervised learning, in unsupervised learning labels can also be discrete categories (classification) or continuous quantities.

Additionally, there are other learning methods which fall between supervised learning and unsupervised learning; they are the so-called semi-supervised learning methods with partially labeled data (See[Ge17], p.7-9, [Va16], p.332-342).

3.1.3. Most Important and Established Algorithms for Regression in Supervised Learning

The following algorithms all work equally well for classification tasks. As this study focuses on a regression problem, we show the algorithms in this context. It should be evident to the reader how to predict a class instead of a value.

(A) Linear and Polynomial Regression

Linear regression is the most commonly used regression and simply looks for a linear correlation between two features x and y, that fits the training data to a straight regression line. The Linear Regression algorithm finds the optimal parameter values for a linear equation with bias θ_0 and a slope θ_1 such that it minimizes a cost function that measures the distance between the linear model's predictions and training labels. Even though this model is limited and cannot adapt to non-linear relationships, the advantage is that it can never "overfit" the data and generalizes well, i.e. is not influenced by noisy data.

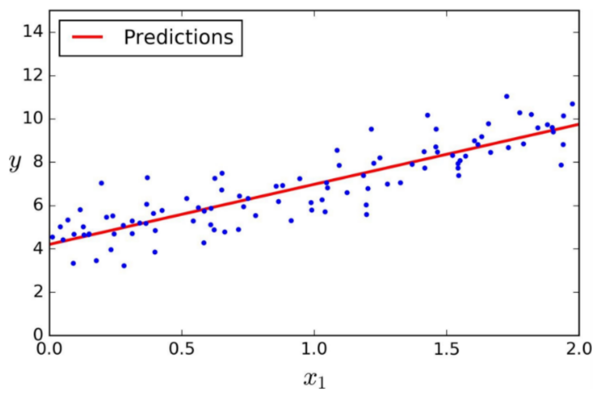


Figure 1: Linear Regression model prediction (example), source: [Ge17], p.109

Polynomial Regression is more powerful and complex. The model creates additional features from the powers of existing features and then trains a linear model on them. This technique is capable of detecting non-linear relationships but is prone to overfitting. There are regularization methods to detect and avoid this risk.

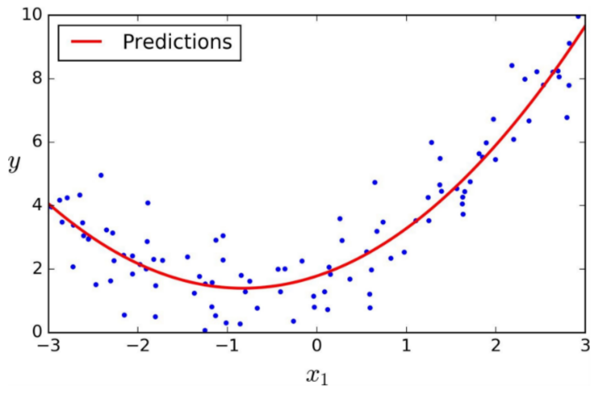


Figure 2: Polynomial Regression model prediction (example), source: [Ge17], p.122

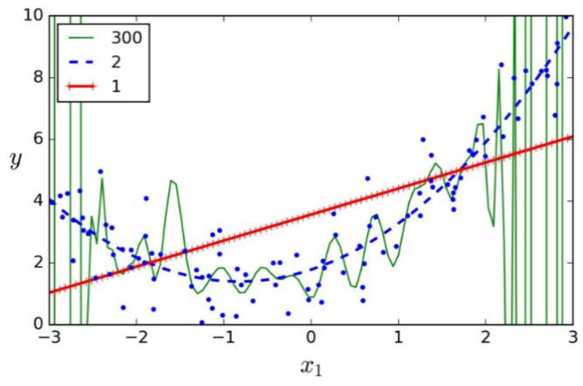


Figure 3: High degree of polynomial regression shows overfitting (example), source: [Ge17], S.123

(B) SVM Regression

The SVM Regression fits the data on a kind of broad street (large margin) and chooses the line that maximizes this margin (maximum margin estimator) while limiting margin violations. The hyper-parameter ε controls the width of the street. The boundaries and predictions are not affected by new samples as long as they fit on the street. Only the samples on the edge of the street (support vectors) matter for the fit, which provides high flexibility and fast computation.

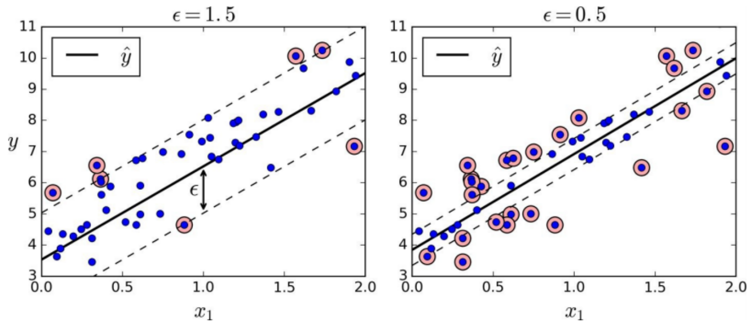


Figure 4: SVM Regression with margins (dashed lines) and support vectors (circles), source: [Ge17], p. 155

The SVM is a kernel-based model, i.e. the model can improve by transforming the inputs, which is done by projecting the features into higher-dimensional space similar to a polynomial regression.

With complex polynomial or *rbf (*'**RBF**'*)*'-kernels the data can be projected into higher-dimensional space defined by polynomials and Gaussian basis functions ([Va16], p.411). Thus, the SVM can perform linear and nonlinear regression and even outlier detection.

The hyper-parameters C and gamma γ control the flexibility of the margins, i.e. the tolerance of margin violations by outliers. Both act as regularization parameters and have a similar influence. Decreasing γ and C makes the model more general with more influence for individual samples. In case of overfitting, γ and C should be reduced [Ge152].

(C) Decision Trees

Decision Trees are typically known for performing classification tasks but they can equally be used for regression. They work very intuitively as in each node a question is asked to predict the class of the sample or a value, respectively (regression).

The samples are split (generally binary) such that their average value comes as close as possible to the target prediction value of a node such that the value in the leaf node finally represents the average target value of the samples associated with this leaf. During this process, the algorithm performs a linear regression that approximates a sine curve. Overfitting can be controlled by limiting the parameters *max_depth* for the maximum depth of a tree and *min_samples_leaf* for a minimum of samples required in a leaf. Otherwise the curve will get very dense and the algorithm will badly overfit training data without filtering out the noise [Ge17], p.175-176.

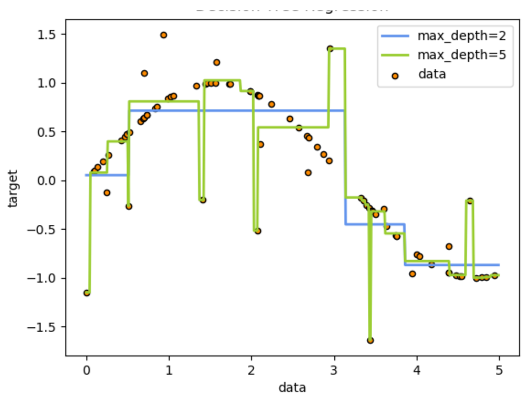


Figure 5: Decision Tree regression, source: http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

(D) Ensemble Learning and Random Forest

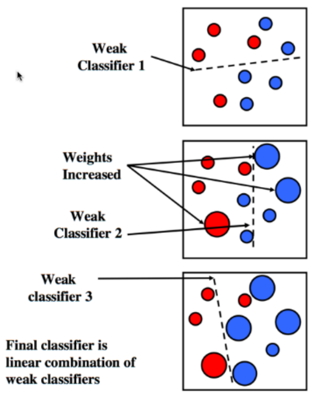


Figure 6: AdaBoost Classifier that shows adaptive boosting, source: http://vinsol.com/blog/2016/06/28/computer-vision-face-detection/

In general, ensemble learning means that a few simple predictors are combined into an even more powerful predictor. One technique is to aggregate very diverse predictors, that have been trained with different algorithms that are as independent from each other as possible. Another way is to use the same training algorithm for every prediction but train them on different random subsets of the training set. If these samples are drawn with replacement (boot-strapping), this is called *bagging*. Random forests are an example of a bagging ensemble method built on decision trees with the bootstrap-parameter set to true by default.

Contrary to the simple decision trees this model is less intuitive and more of a black box.

The accuracy of every individual tree is improved by averaging the vote of the individual subtrees. Each of the multiple decision trees is built on a random subset of the training samples. During the training process a specific number of features is selected at random to find the best split of the data. The model accuracy can be evaluated on the OOB-samples, i.e. the “out-of-bag” samples that have not been used for the tree growing and are unknown to the algorithm [Zh17].

There are several more examples for the application of the above principle.

Another ensemble method is to combine a couple of weak learners into a strong learner. Predictors are trained sequentially, each trying to improve the previous result. The most popular algorithms that use this method are *AdaBoost* and *Gradient Boosting*. AdaBoost assigns higher weights *according to the* prediction error of the instances such that the next predictor focuses more on hard cases ([Ge192], S.192).

Gradient Boosting is another sequential learning technique, where the predictors all correct their previous predictor. This in each iteration the new predictors are being fit to the unexplained errors of the regression line ([GE17], p.195-197).

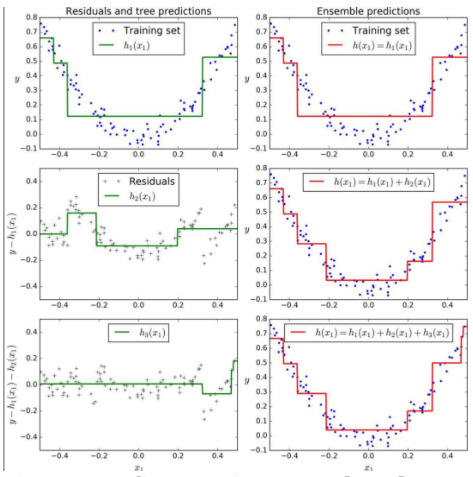


Figure 7: Gradient Boosting, source: [Ge17], p.197

3.1.4. Python packages and libraries used for the implementation

- **Jupyter**: Package for computational environment.
- **NumPy**: Efficient manipulation and storage of n dimensional homogenous data with *ndarrays* objects.
- **Pandas**: Efficient manipulation and storage of heterogeneous and labeled data in *dataframe* objects.
- **Matplotlib**: Data visualizations and plotting.
- **Scikit Learn**: Efficient and clean implementations of the most common ML algorithms.

3.2. Database

3.2.1. Research Area

The approach for this study is to choose an existing pyranometer network to obtain ground-based observations of downward solar irradiation in a given region. These real-time measurements would ideally provide a high level of accuracy with a high temporal resolution. This is very relevant for short-term irradiation-forecasting with the horizon of two hours as the accuracy of the predictions depends on the quality of the training data for the machine learning algorithms. The final model can afterwards be applied to other datasets of other networks. An alternative method consists of getting less accurate meteorological data from remote sensing via satellites, that are not restricted to a specific area which allows forecasts with longer forecast horizons across space and time [Zh17].

Inforiego is a governmentally-funded and publicly accessible platform in Castilla y León, an autonomous community in North-western Spain. They maintain a network of more than 50 stations across the region in the different provinces that provide various meteorological data mainly for agricultural purposes with a resolution of 30 minutes[3] via a FTP-server and a Rest

[3] At least the data used in this study (2015 to 2017) consists fully of 30'observations (monitored period Inforiego: 2001 - now).

API (web service): precipitation (mm.), temperature (°C), humidity (%), solar irradiation(W/m^2), wind velocity (m/s) and wind direction (°).

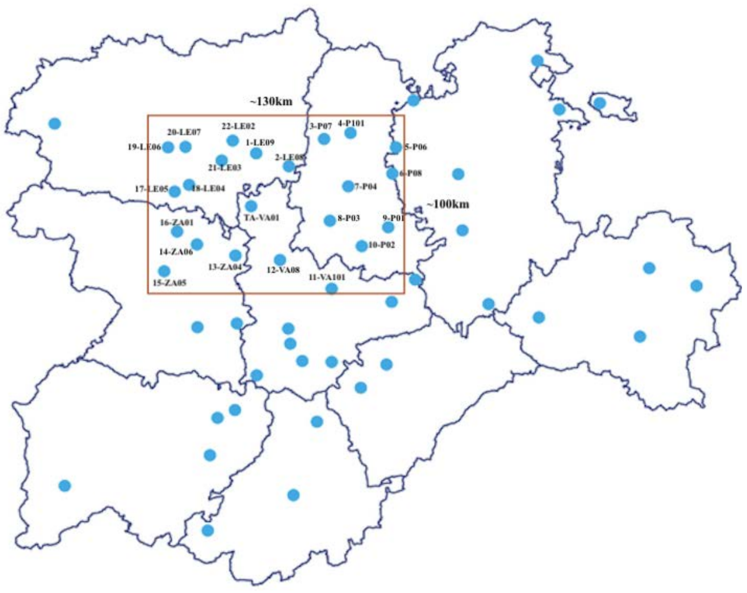


Figure 8: Inforiego: selected stations and target TA

3.2.2. Idea

Our target station is in the North of Valladolid province (see Figure 8, VA01) idea is to pick a target station in the center of a local region and the 21 surrounding stations in highest proximity (See Figure 8). The objective is to predict the solar irradiance at the target TA for a specific time, based on samples from the other stations with the following parameters. Our intuition is that choosing the closest stations may help to improve predictions.

- **Prediction horizon or offset**: defines the temporal difference between the sample closest to the predicted value and time of prediction, i.e. from 8 a.m. if we want to predict with a prediction horizon of two hours for 10 a.m. [see example in figure].
- **nsamples**: number of samples that will be included from a station, for example n samples = 3 for a given station, if we use data from 7 a.m., 7.30 a.m. and 8.00 a.m. to predict radiation at 10.

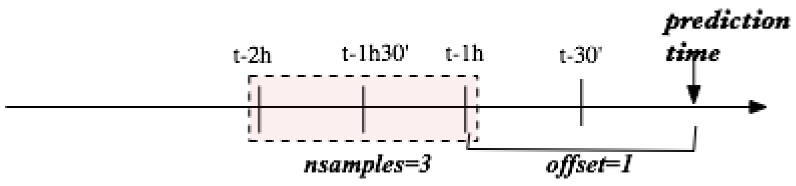


Figure 9: example for *nsamples*- and *offset*-parameters

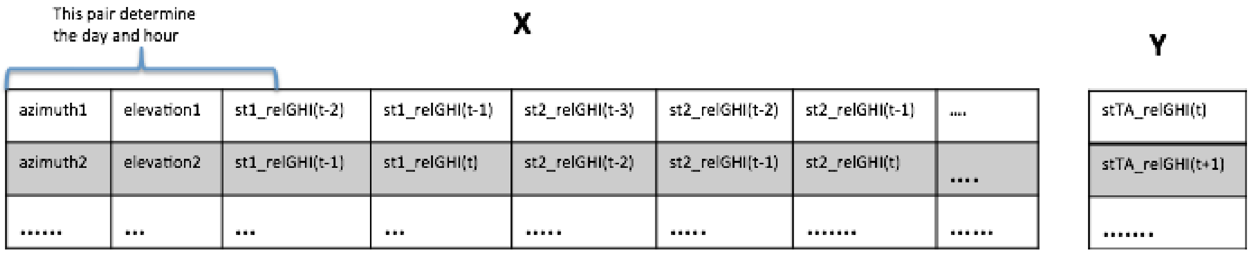
In the training **X**, **Y** matrix with known values every **X**, **Y**-row represents a sample. A row consists of all the input features **X** and the output value **Y** – the target value. Y will hold the value measured at the prediction time corresponding to **X** (See Figure 10).

The radiation is expressed as relative global horizontal irradiance ("**GHI**"), i.e. the ratio of measured GHI and the expected radiation given optimal weather conditions (clear-sky model). This ratio serves the model to indirectly derive day of year and hour.

To support this idea of a time-series, azimuth angle and zenith angle will be additionally included as extra features for the sample closest to prediction time, if necessary. All values will be normalized, i.e. converted to the range [0, 1]. Especially SVMs are sensitive to feature scales ([Ge17], p.146) whereas Random Forests do not require feature scaling.

Precipitation, temperature, humidity, wind velocity and wind direction are not further considered to be included as input parameters for the model as these measures tend to create noise in the computation.

The idea for this study is to test if a machine-learning model can detect a kind of signal similar to the motion of clouds that travels between the stations over time, or at least recognize patterns that indicate a drop of radiation. These patterns will have to be discovered and it is not known whether a drop of radiation may be due to a local storm or a kind of mist.


This pair determine the day and hour
X
Y

azimuth1	elevation1	st1_relGHI(t-2)	st1_relGHI(t-1)	st2_relGHI(t-3)	st2_relGHI(t-2)	st2_relGHI(t-1)	....
azimuth2	elevation2	st1_relGHI(t-1)	st1_relGHI(t)	st2_relGHI(t-2)	st2_relGHI(t-1)	st2_relGHI(t)	....
......	...	...	...	.....	.....	....	......

stTA_relGHI(t)
stTA_relGHI(t+1)
....

Figure 10: built X, Y matrixes (example)

4. CHAPTER 3 - MODELLING APPROACHES

4.1. Solar Model for Relative Radiation

To determine the exact GHI for all the stations given a clear sky, a model is necessary that computes the solar position as accurately as possible. The Solar Position Algorithm ("**SPA**") by the National Renewable Energy Laboratory ("**NREL**") of the United States is currently the most common solution with the highest accuracy used for PV-applications in general, including the calibration of pyranometers (See [Re08], p.1). There exist several applications based on this algorithm, including a freely available one that implements it in Python: Pysolar. Pysolar is a collection of Python libraries for simulating the irradiation of any point on earth and chosen as the best solution for this study as it is specially aimed at modeling photovoltaic systems. Sunpy for example is a similar application but focused in solar physics modelling.

Pysolar expects a timezone-aware datetime as input parameter together with longitude and latitude of the location to compute azimuth angle and zenith angle (altitude). With datetime and altitude the GHI is obtained. The ratio of the measured radiation and the 'ideal' GHI represents the relative radiation of a specific location at a specific time.

4.2. Feature Selection

As described in section [3.2.2] we create the **X**, **Y** training matrix with the n values from t-prediction horizon to t-1h-n*30' of relative radiation of all 22 stations as the samples all have a period of 30 minutes. The basic model is to take *nsamples* = 3 with a prediction horizon of 2 hours. As we start at 5 am the first predicted value is at 8 am, the last is at 10 pm. The samples of the target station itself are also included.

For this study different methods have been used to select features:

- By trial and error: just build models, select different sets of features and check with which features performance is improved;

- by looking at the standard correlation coefficient[4] to look for linear correlations between a feature and the target value; and
- by evaluating the feature importance with the Random Forest algorithm.

The first option is clearly the most intuitive. It has actually been applied many times, for example to analyse for which stations it makes sense to include azimuth angle and altitude. The best result was to also include the azimuth angle for every radiation sample from every station and the altitude only once for the target station at the target time. Naturally, not all possible constellations can be tested with this approach.

Another fast and simple method to check for a correlation between a pair of continuous variables for a regression problem is the standard correlation coefficient. It gives a value from the interval [-1,1] with -1 meaning a perfect negative, 1 a perfect positive and 0 no correlation at all ([Ge17], p.56). Unfortunately, this method only gives information about linear correlation, while features can be perfectly correlated in a non-linear way.

A less obvious, but very elegant method is to find data correlation with Random Forests and Decision Trees ([Ts10], p.11) that will be used to determine the key features for our different models (See [Training and evaluating on the Training set]). These algorithms find out the statistical usages of each feature which can be accessed with the *feature_importances* attribute that gives the relative importance of each feature where the sum of all importances is 1 ([Ge17] p.190).

Finally, with the Principal Component Analysis ("**PCA**") algorithm it is possible reduce the dimensionality of the input features (see unsupervised learning algorithms in [Categories of Machine Learning]). PCA indeed can be an efficient way to reduce complexity and perform a feature selection. For our study we did not use PCA.

[4] Also called Pearson's r, the covariance of two variables X and Y, standardized by the product of their standard deviations.

4.3. Data Preparation

4.3.1. Detection and Replacement of Outliers

A ratio for relative radiation >1.0 was considered an outlier as the GHI gives the maximum theoretical value. Most outliers were either in the early morning or late evening. This may be due to the fact that Pysolar quickly rises from 0 to high values and the same in the evening in reversed form. That makes the curve very steep, whereas the measured values go up more gradually. The samples were kept and filled backward with the next valid value. As a day had samples beginning from 5 a.m. always with a value of 0, this value was at most the first valid value.

4.3.2. Replacement of Missing Values

There are quite a lot of gaps in the dataset, sometimes only for specific stations, sometimes for each station, but usually datasets for whole days were missing. It showed the method of gap-filling strongly affected the model. If all samples containing at least one null-value were just dropped up to 100 days would have been lost for a whole year.

A sample with n individual values from 22 stations normally forms a vector of *n**22 values. It was decided to keep a sample if it had already one instance for the n individual values. At this point, some models had already been trained and the correlation coefficients of the different stations were known. The replacement algorithm works like this: If for example it comes across a missing value for t-2h at station LE02, it first checks, if the corresponding value t-2h is present for the target station VA01, and if possible, and it replaces it.

If the value is also missing, it looks at the second most correlated station and so on until all other stations have been checked. If the value is nowhere to be found the sample is finally dropped. This happens with all values and it proved to be a good solution as the correlations between the stations did not differ very much, so a lot of samples could be saved that would otherwise have created important gaps in our evaluation. This was especially disadvantageous for the graphic plots.

4.4. Model Selection

The original problem definition of this study proposed to use a Support Vector Machine ("**SVM**") model to solve the given task. This is a very powerful and flexible model for supervised learning capable of performing classification and regression tasks. Indeed, according to a paper, the SVM figures among the most effective and frequently employed algorithms for the prediction of short-term solar irradiance. Yet the paper states that the mostly applied solutions are Artificial Neural Networks ("**ANNs**"), that yield similar results compared to an SVM in terms of accuracy of the prediction. Yet it is indicated that the training of ANNs entails considerably more effort and is intense in computing power. Therefore, it is advisable to rather use SVMs.

It is concluded that aside from SVM, all algorithms related to regression trees and especially random forest, bagging and boosting, may be of great interest in the future though there is not much evidence yet and this has to be further explored ([Vo17], S.22). For this reason, our study will focus on the SVM approach and also try Random Forest and a Boosting technique.

4.5. Training and Evaluating on the Training Set

4.5.1. Results of Cross Validation

For evaluating, the *Cross Validation* method is used to avoid overfitting without having to split the training data into a smaller training and test set. Instead, using *Cross Validation*, the training set is randomly split into k distinct subsets called *folds*. Then the model is trained and evaluated k times, selecting a different fold for evaluation every time and training on the other k-1 folds. The scoring method is RMSE (see section [4.2]) for relative radiation values. Cross validation only allows to get an estimate of the model performance and the precision (its standard deviation) ([Ge17], p.70-71). For training on the whole-year-period of 2015, the following average results of 10 folds were obtained for the three models:

Model	**RMSE**	**Standard deviation**	**Nr. features**	**Nr. of samples**
Support Vector regression	0.14770	0.01422	133	10178
Gradient Boosting regression	0.13516	0.02043	133	
Random Forest regression	0.13578	0.02090	133	
Random Forest regression with best features	0.13368	0.01968	**32**	

Table 1: Results of Cross Validation of different algorithms

4.5.2. Extracting feature importances with Random Forests

In the original trainset, we trained with 133 features and samples. With the feature *importances* – attribute of the Random Forest algorithm (See [Feature selection]) we created the following feature ranking[5]:

Rank	**Feature**	**Importance**	**Rank**	**Feature**	**Importance**
1	altitude **target t**	0.541054	**17**	radiation **LE07 t-1**	0.005789
2	azimuth **ZA06 t-3**	0.069402	**18**	radiation **P07 t-2**	0.005652
3	radiation **LE02 t-1**	0.035668	**19**	radiation **P08 t-1**	0.004655
4	radiation **LE05 t-1**	0.030819	**20**	azimuth **LE09 t-2**	0.004507
5	azimuth **ZA05 t-3**	0.025496	**21**	azimuth ZA01 t-3	0.004102

[5] *t-1* is closest to the prediction time t, t-3 is the farthest.

6	azimuth **LE06 t-3**	0.024477	**22**	azimuth **P04 t-3**	0.004098
7	radiation **ZA01 t-1**	0.013982	**23**	azimuth **LE02 t-3**	0.004025
8	radiation **P07 t-1**	0.012791	**24**	azimuth **LE08 t-3**	0.003956
9	radiation **LE06 t-1**	0.012335	**25**	azimuth **LE09 t-3**	0.003824
10	azimuth **LE07 t-3**	0.008936	**26**	radiation **ZA04 t-1**	0.003812
11	azimuth **VA08 t-3**	0.008915	**27**	azimuth **P03 t-1**	0.003810
12	azimuth **VA0101 t-1**	0.008714	**28**	radiation **ZA06 t-1**	0.003742
13	radiation **LE04 t-1**	0.008707	**29**	azimuth **P02 t-3**	0.003647
14	radiation **target t-1**	0.008612	**30**	radiation **P02 t-3**	0.003483
15	radiation **LE03 t-1**	0.007769	**31**	radiation **VA08 t-1**	0.004054
16	radiation **LE08 t-1**	0.006508	**32**	radiation **VA101 t-1**	0.004036

Table 2: Extraction of the most relevant features with Random Forest

These 32 features together have an importance of 0.9 out of 1. The remaining 100 features add very little and leaving them out makes the model much more efficient. Azimuth angles and the altitude of the target station are very relevant. They could be added to the persistence model to form a somewhat advance reference for comparison. It’s noticeable that the newest radiation value from target station is only on position 14, as well as some other remarkable features that do not follow the general scheme (in yellow).

The resulting model is not only slightly more precise, the lower standard deviation also indicates more precision, as the other 100 features presumably add a lot of noise.

4.5.3. Training with different forecast horizons

Forcast horizon	**RMSE**	**Standard deviation**	**Nr. features**	**most relevant features**
1.5 h	0.02787	0.15075	133	altitude **target t** azimuth **LE06 t-3** azimuth **VA101 t-3** azimuth **ZA05 t-3**
	0.026077	0.15340	10	
1.0 h	0.02237	0.142550	133	altitude **target t** azimuth **LE06 t-3** radiation **ZA01 t-1** azimuth **LE02 t-3**
	0.02372	0.1495293	13	
0.5 h	0.1394029	0.02200	133	altitude **target t** radiation **LE02 t-1** azimuth **ZA05 t-3** azimuth **LE07 t-3**
	0.14265	0.02227	13	

Table 3: Predicting with different forecast horizons

As before, altitude at target time and station is always by far the most important feature. The next 10 to 13 features make up for around 80% of the feature importance and were chosen for a second more compact model.

5. CHAPTER 4 - RESULTS

5.1. The Bias/Variance Tradeoff

In supervised ML, optimizing one parameter often means that another gets worse – as in described in section [4.5.1] - when the (R)MSE is minimized during training and the standard deviation augments.

One has to understand that the generalization error of a model is composed of three parts: Bias, Variance and the Irreducible Error. The last one describes the noisiness of the data and we already tried to minimize it by data cleaning and intelligent feature selection. Trying to minimize Bias and Variance at the same time is impossible and will probably result in overfitting the data.

Instead one should make the right assumptions about the data in order to detect correlations while avoiding an excessive sensitivity to small fluctuations in the training data ([Ge17], p.127). To prevent overfitting the algorithms can be regularized by special parameters (See [Most important and established algorithms for Regression in supervised learning]) or increasing the training data also helps generalize more.

5.2. Metrics for Evaluation of Model Accuracy for Individual Samples

It is relatively simple to evaluate if a classifier works well by just quantifying correct and false predictions. For regression models it is quite difficult to find appropriate performance measures to evaluate the accuracy of the models or to compare their performances. We will use the standard performance measures for ML and also define a couple of own metrics.

(i) ***MAE*** *(Mean Absolute Error):* also called Average Absolute Deviation - measures the Manhattan distance between the target- and the prediction-vector, where you can only move along orthogonal paths within a grid. This measure is less sensitive to outliers than the RMSE.

$MAE(\boldsymbol{X}, h) = \frac{1}{m}\sum_{i=1}^{m} |h(x^{(i)}) - y^{(i)}|$ ([Ge17], p.39)

Where m is the number of samples, $\boldsymbol{x}^{(i)}$ is the vector of all the feature values of the i^{th} instance in the dataset, $\boldsymbol{y}^{(i)}$ the label for that instance and h the system's prediction function (hypothesis) that computes a prediction $h(\boldsymbol{x}^{(i)}) = \hat{\boldsymbol{y}}^{(i)}$

(ii) ***RMSE** (Root-Mean-Squared-Error):* a typical performance measure for regression problems that gives an idea of how much error the system typically makes in its predictions with a higher weight for large *errors:* $RMSE(\mathbf{X},h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2}$ ([Ge17], p.37)

(iii) ***MSE(X,h)***: the parameter that is usually minimized by the training algorithm (cost function). The scoring function in *Scikit Learn* usually is the opposite of this (as absolute or relative value) ([Ge17], p.70).

5.3. Metrics for measuring the prediction quality for days

(i) ***s(day)***: error vector ($\boldsymbol{s}$) generating a vector per day like: $\boldsymbol{s}(day) = [\sqrt{(real(day,h) - predicted(day,h))^2}], \forall h, h \geq 10{:}00, h \leq 22{:}00$

(ii) *Similarity measure*: to compare similarity between the signals (real and predicted) It consists on the scalar product of the two signals, divided by the product of its norms:

$$\frac{\sum_{i=10:00}^{i=22:00}(real(i) * predicted(i))}{\sqrt{\sum_{i=10:00}^{i=22:00} real(i)^2} * \sqrt{\sum_{i=10:00}^{i=22:00} predicted(i)^2}}$$

(iii) *Score*: integer number for each day that measures the quality of its sample predictions. We define a threshold parameter that determines if the difference between a predicted and a real value is too high. We evaluate the difference for each half-hour-sample and add it to the score if it exceeds the threshold. The higher the score for a day the worse are the predictions for this day. We use the number of days with high scores (depending on your threshold) as a final **"overall" metric** for the test set - i.e. number of poorly predicted days.

5.4. Validation on the test set

The test set included the whole year 2016 (10 days were missing).

Model	RMSE	Standard deviation	Nr. features	Nr. of samples
Support Vector regression	0.13227	0.04588	133	10308

Gradient Boosting regression	0.11623	0.03569	133	
Random Forest regression	0.1185	0.036436	133	
Random Forest regression with best features	0.11965	0.03708	**32**	

Table 4: Validation of different algorithms on the test set

5.5. Reference Models

Another method that can provide a rough idea of how well your model works is to compare it with other models. The most ‘trivial’ or so-called ‘naive’ forecasting model is to assume ‘things stay the same’. I.e. it just takes the last valid value from the sample vector for the target station as a prediction. This ‘persistence’ model basically demonstrate, if our forecasting has any effect at all ([Vo17], p. 21).

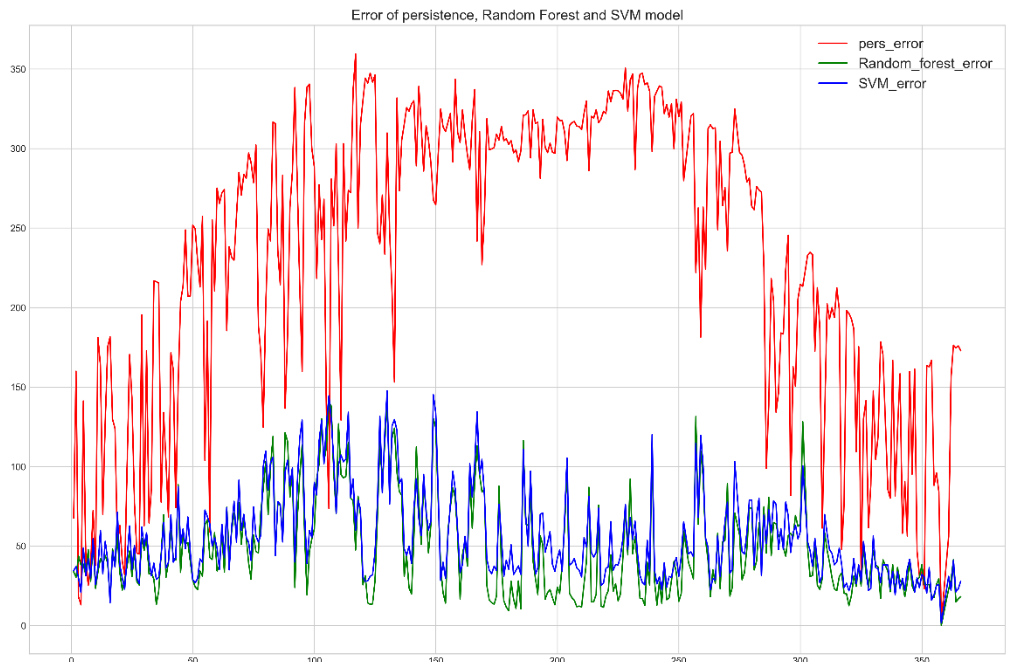


Figure 11: MAE for the Persistence model, SVM and Random Forest for the whole test set (aggregated by days)

5.6. Graphic evaluation

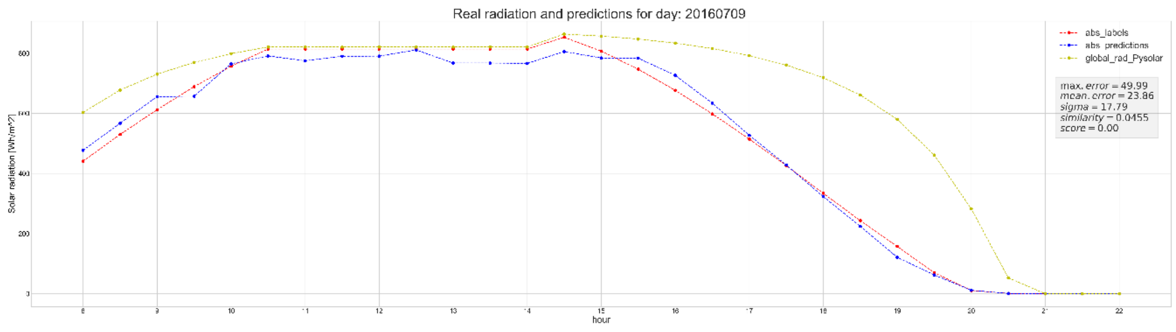


Figure 12: Random Forest model predicts radiation for a sunny day (easy scenario)

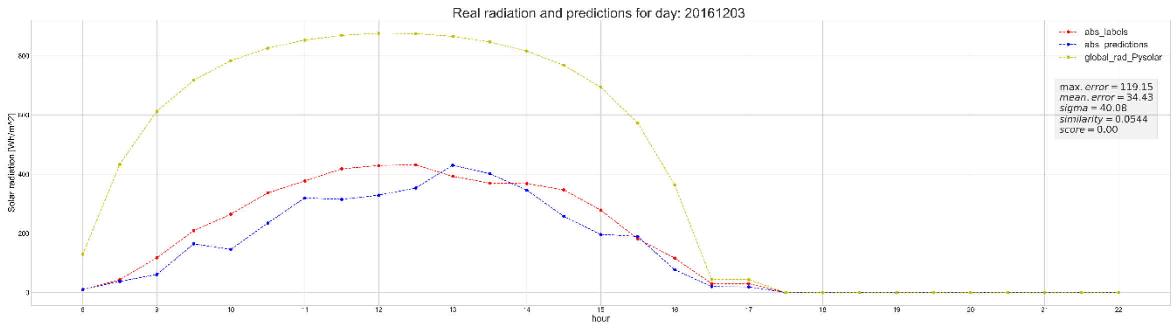


Figure 13: Random Forest predicts radiation for a cloudy day (easy scenario)

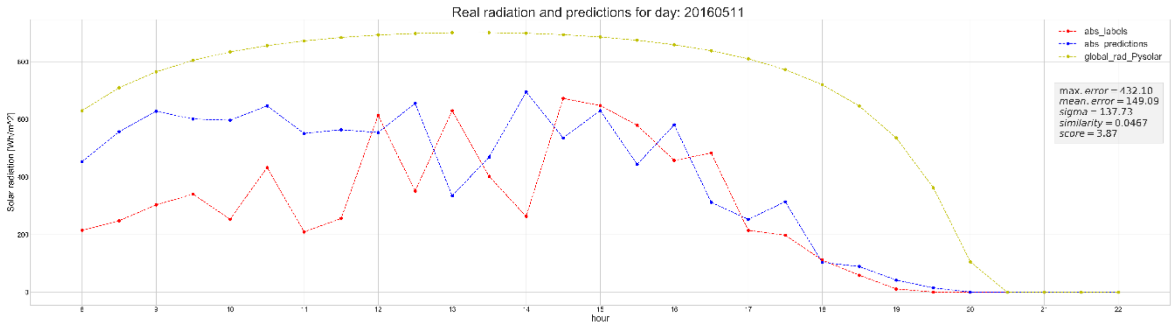


Figure 14: Random Forest predicts radiation for a day with unstable weather conditions (difficult scenario)

5.7. Aggregated model

To improve the quality of our predictions we wanted to train two different models for “difficult”, i.e. hard to predict days and for ‘easy’ to predict days. To identify those days we used the metrics for individual days, see paragraph [4.3]. For this experiment training and test set were exchanged. From the last validation set (year 2016) with the Random Forest model the samples with a score > 0.3 and a score <=0.3 respectively were selected as the two new training sets for the two separate models to predict difficult and easy days. Their feature importances were the same and both were retrained with 42 key features.

For the validation the old trainset (year 2015) was split into two categories: ‘easy’ and ‘difficult’ days. This time the GHI from Pysolar with absolute radiation values was used together with the absolute radiation labels to compute a score. To break the test set into similar proportions as the training set, the threshold for the score was 7500. The models were then individually tested on the two test sets and afterwards the results were combined.

Model	**RMSE**	**Standard deviation**	**Nr. features**	**Nr. of samples**
Random Forest for “easy” days	0.08250	0.02800	133	5229
Random Forest for “difficult” days	0.15913	0.01150	133	5079
Random Forest for “easy” days with optimized features	0.08101	0.02888	42	5229
Random Forest for “difficult” days with optimized features	0.15992	0.01020	42	5079

Aggregated Model for “difficult” and “easy” days (on test set)	**0.09963**	**0.03196**	**42**	**10178**
Random Forest for “easy” days (individual) (on test set)	0.13032	0.04672	42	4743
Random Forest for “difficult” days (individual) (on test set)	0.13454	0.04218	42	5435

Table 5: Aggregated model for “easy” and difficult predictions

Model		**Nr. of poorly predicted samples**	**Nr. of total days**
not aggregated	Training set	5079	359
Aggregated	Test set	**3653**	349

Table 6: Aggregated model: Nr. of poorly predicted samples

The number of mis-predicted samples decreased significantly with the aggregated model and the model may predict very well for some specific days. However, the variance of the data increases enormously. An explanation for this could be that the combined model overfits the data and does not generalize well. The reduced number of training instances for each individual model may also be responsible that the algorithm is more sensitive to individual data.

Unfortunately, training and test set were exchanged for this experiment that made a more detailed comparison with the non-aggregated Random Forest model impossible.

6. CHAPTER 5 - CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

The temporal tracking of signals did not really succeed as older samples did not prove useful for the predictions and only the newest ones were relevant. Nonetheless, the selected models all perform well in clear-sky situations or on completely cloudy days but do not predict well for partly cloudy days with quick weather changes. The Random Forest model facilitates the feature selection by giving direct access to their importances. This leads to the idea of a highly complex model where for every point in the dataset the best combination of features could be used to build the ideal model. Yet therefore, an instance is needed that tells us with 100% accuracy beforehand the best feature selection for the given scenario. The approach of developing aggregated models for different scenarios comes with a high risk of overfitting the data and may not generalize well on unknown data.

6.2. Future Work

A way to improve our model could be the engineering of new features. An interesting approach would be for example to create a new feature vector with the differences between consecutive samples, that kind of approximates to the derivative. As increasing the complexity of our models comes with a high risk of overfitting we should rather try to increase the quality and variety of our dataset. Though difficult to obtain a network with higher resolution and maybe a more dense grid structure may be key to realize the idea of a model that is more responsive to the signals of surrounding stations and sensitive to quick weather changes. We could try to built aggregated models with very diverse predictors that could work on the same dataset and implement a voting system among them. We could also apply the SVM on different randomly selected subsets of the training set and implement a sort of bagging method similar to the Random Forest algorithm. Also feature selection could be done randomized and then improved by the algorithm.

7. CAPÍTULO 5 - CONCLUSIONES Y TRABAJO FUTURO

7.1. Conclusiones

El seguimiento temporal de las señales no tuvo realmente éxito, ya que las muestras más antiguas no resultaron útiles para las predicciones y solo las más recientes fueron pertinentes. No obstante, los modelos seleccionados funcionan bien en situaciones de cielo despejado o en días completamente nublados, pero tienen un rendimiento deficiente para los días parcialmente nublados con cambios rápidos de clima. El modelo *Random Forest* facilita la selección de características al dar acceso directo a las importancias. Esto lleva a la idea de un modelo altamente complejo donde para cada punto del conjunto de datos podría usarse la mejor combinación de características para construir el modelo ideal. Sin embargo, se necesita una instancia que nos indique con 100% de precisión de antemano la mejor selección de características para un escenario o circunstancias concretas. El enfoque de desarrollar modelos agregados para distintos escenarios viene aparejado con un gran riesgo de *overfitting* de los datos y puede que no se generalice bien con datos desconocidos.

7.2. Trabajo futuro

Una forma de mejorar nuestro modelo podría ser la ingeniería de nuevas características. Un enfoque interesante sería, por ejemplo, crear un nuevo vector de características con las diferencias entre muestras consecutivas, que de algún modo se aproxima a la derivada. Como aumentar la complejidad de nuestros modelos implica un alto riesgo de sobreajuste o *overfitting*, deberíamos intentar aumentar la calidad y la variedad del conjunto de nuestros datos. Aunque es difícil obtener, la utilización de una red con mayor resolución y tal vez una estructura de red más densa puede ser clave para realizar la idea de un modelo que responda mejor a las señales de las estaciones circundantes y sea sensible a los rápidos cambios climáticos. Podríamos tratar de construir modelos agregados con predictores muy diversos que podrían funcionar en el mismo conjunto de datos e implementar un sistema de votación entre ellos. También podríamos aplicar el SVM en diferentes subconjuntos seleccionados al azar del conjunto de entrenamiento e implementar una especie de método de ensacado (*bagging method*) similar al algoritmo de *Random Forest*. La selección de características también podría hacerse de forma aleatoria y luego mejorada a través del algoritmo.

References

[Ge17] Géron, Aurélien: Hands-On Machine Learning with Scikit-Learn& TensorFlow – Concepts, Tools and Techniques to build intelligent Systems, first edition, O'Reilly Media, 2017

[Mc12] McKinney Wes, Python for data analysis, first edition, O'Reilly Media, 2012

[Re08] Reda, I.; Andreas, A.: Solar Position Algorithm for Solar Radiation Applications. 55 pp.; NREL Report No. TP-560-34302, 2003, revised January 2008, https://www.nrel.gov/docs/fy08osti/34302.pdf

[Ts10] Tsanas, Athanasios: A Simple Filter Benchmark for Feature Selection, in Journal of Machine Learning Research, 2010, http://www.maxlittle.net/publications/tsanas10a.pdf

[Va16] VanderPlas, Jake: Python Data Science Handbook: Essential Tools for Working with Data, first edition, O'Reilly Media, 2016

[Vo17] Voyant, Cyril; Notton, Gilles; et.al.: Machine learning methods for solar radiation forecasting: A review, University of Corsica, 2017

[Zh17] Zhou, Qingtao: A machine learning approach to estimation of downward solar radiation from satellite-derived data products: An application over a semi-arid ecosystem in the U.S., in PLOS journal, Boise State University, Boise, Idaho, USA, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5544233/, published online 04.08.2017

The code for the project can be found here:

https://drive.google.com/drive/u/1/folders/1oteOP5fSi29k57Uo_i_ScRB0erru0pQ0