TH Rosenheim

Fakultät für Ingenieurswissenschaften

Master Thesis

Use and adaption of ensemble learning methods to develop an irradiation nowcasting model with probabilistic output

•

Verwendung und Anpassung von ensemble learning methoden zur Entwicklung eines nowcasting modells für probablistische Einstrahlungsvorhersagen

Autor:	Andreas Brader
Studiengang:	AFE - Master
Matrikelnummer:	865178
Erstprüfer:	Prof. Michael Zehner
Zweitprüfer:	Prof. Dr.-Ing. Benedikt Dietrich
Betreuung DLR:	M.Sc Yann Fabel
Abgabedatum:	10.12.2022

Acknowledgment

With this words I want to thank my parents who supported me during my academic studies, as well as my girlfriend Christina who was very understanding with all late and night-time work on the project.
Furthermore I want to thank Prof. Zehner who supported me within recent projects, gave me the opportunity to participate at the PV-Symposium and brought me in contact with the DLR where the idea for this thesis was born. Since this project is based on a dataset from the DLR, I want to thank the team of Stefan Wilbert and especially Yann Fabel who helped me on my questions and provided useful information and references within the weekly meetings. Last but not least I also want to thank Prof. Dietrich who provided advantageous information and best practice guidelines within the topic of machine learning.

Abstract

This master thesis proposes a method of creating probabilistic irradiance predictions for intra-hour situations by combining two ensemble prediction methods.
These ensembles consist of historic measurements, which are chosen by similarity of the environmental situation and prediction results from machine-learning (ML) models, performing best with similar weather situations as the given data point. Hence the so accumulated data is processed by a natural-gradient-boost (NGB) model [13] to estimate a deterministic prediction, as well as an confidence interval. Since the location and width of such an interval is supposed to help estimating future irradiance values including uncertainty.
The probabilistic predictions, generated by the NGB approach, generate superior results compared to the base ensembles by the continous ranked probability score (CRPS) as well as the mean-absolute-error (MAE). Mentionable is that the performance of the proposed method increases, by higher forecast horizon of 15 and 20 minutes with respect to the reference ensemble. This consists of an accumulation from an analog-ensemble [2] and an prediction-ensemble by a dynamic selection [11] process.

Abbreviations

PV photovoltaic

NWP numerical weather prediction

nm nanometer

AI artificial intelligence

HDF hierarchical data format

PI prediction interval

MAE mean absolute error

DNI direct normal irradiance

DHI diffuse horizontal irradiance

GHI global horizontal irradiance

ML machine-learning

NaN not a number

RMSE root-mean-squared-error

MSE mean-squared-error

MAE mean-absolute-error

MBE mean-bias-error

MIW mean interval width

ICP interval coverage probability

CRPS continous ranked probability score

ICON ICOsahedral Nonhydrostatic

SVM support-vector-machine

CSI clear-sky-index

CSIR	clear-sky-irradiance
ANEN	analog-ensemble
NGB	natural-gradient-boost
MLP	multi-layer perceptron
ReLu	rectivied linear units
SHAP	shapley additive explanation
SSP	simple-smart-persistence
KNN	k-nearest-neighbor
LR	linear regression
RF	random forest
XGBoost	extreme gradient boost
sklearn	scikit-learn
SGD	stochastic gradient descent

Contents

1. Introduction

1.1. Motivation

In recent years an increasing trend in mounted photovoltaic (PV) power plants can be observed in figure 1.1 [15]. According to that, a steadily raise of the power-generation-mix share, figure 1.2, is achieved as well.

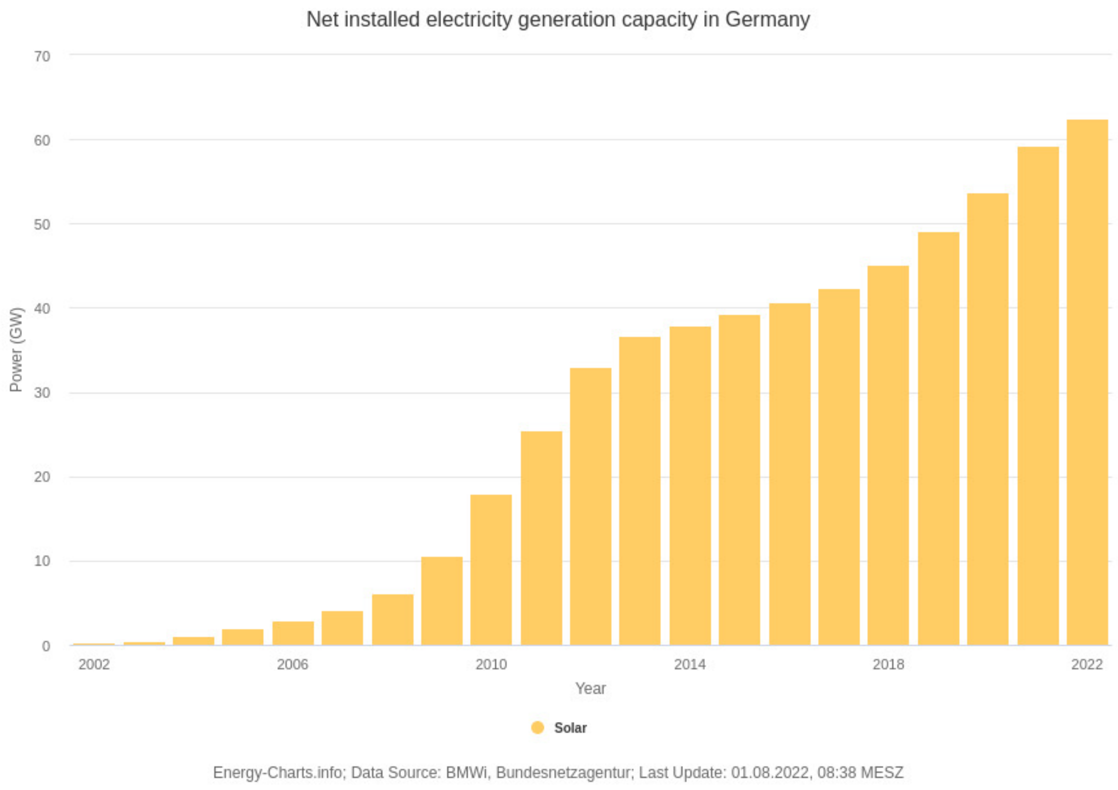


Figure 1.1.: Installed PV-netto-power in 2022 compared to previous years [15]

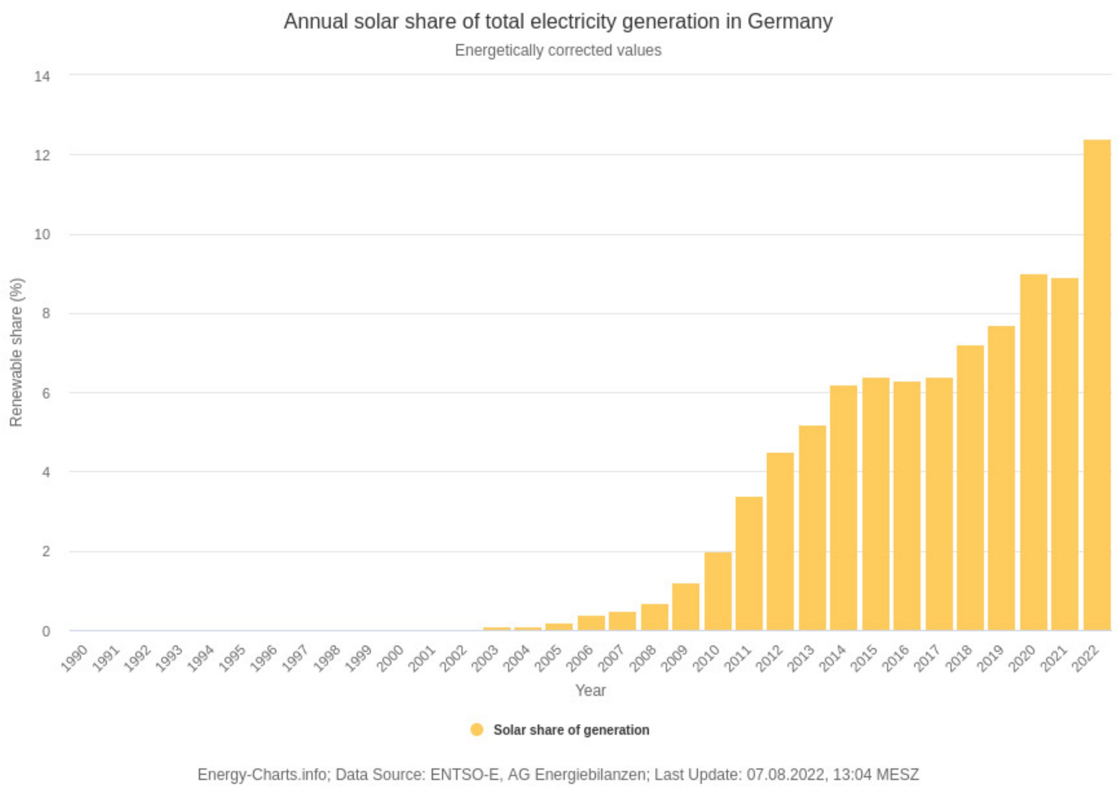


Figure 1.2.: Annual share of photovoltaic outcome within the power mix of Germany [14]

This change leads to a higher use of renewable energies that is necessary to reach the goal of reducing the carbon-dioxide footprint [12]. As a downside on the higher share, the energy outcome from PV-plants is more variable, in comparison to fossil power-plants. This is caused by changing weather and atmospheric conditions [24]. Confronted by those circumstances energy providers have to stabilize the network by adding or removing short-response-systems like gas turbines or pumped storage power stations [45].
To hit those problems, various efforts are made to predict the expected irradiation for a specific forecast-horizon, chapter 3 So by knowledge of future irradiance the composition of energy-sources for a stable network might be achieved more precisely [12].

1.2. Task and Limitations

Therefore the goal of this thesis is to create an ML-based model to forecast the irradiance within a time interval of 5, 10, 15 and 20 minutes in the future. Used information for this task are the environmental conditions of the current situation and past hour. But as a prediction is always defective in a way, as the quote ”All models are wrong, but some are useful” from George E.P.Box [9] implies. The goal is to create a probabilistic prediction for a certain confidence, to have an idea about the forecast-variability to be expected.
In order to create such an probabilistic interval, the approach of an ensemble-model, sec-

tion 2.2.2, is chosen. The outcome of this kind of model is a certain number of forecasts, which can be interpreted as a distribution to estimate a confidence interval [34]. As a measure of quality, the model has to perform better than a reference model in respect to selected evaluation metrics, section 6.1. Therefore the specific goals are listed in table 1.1.

Table 1.1.: Goals for the generated system

Type	**Goal**
Overall prediction process-time	less than a minute
Single predictions from dynamic selection	Skill-score, section 6.1, greater than zero
Probabilistic prediction	Outperforming the reference model, section 4.3 by error metrics of chapter 6.1

But even after the performance evaluation, an absolute superiority of a model can not be determined. The major reason therefore is the dependency of a model to the amount of input-features and the properties of training data [23]. Therefore as the models performance is just evaluated by the dataset on hand, chapter 4.1, the exact result is only valid for the given circumstances. Hereby in different climatic regions, the composition of input variable values does not necessarily results in similar relation to the target value.
Hence all results achieved in this thesis, are with respect to the tuned models and used data. Consequently different training data may result in deviation of performance on metrics, section 6.1.

2. Methodology and Fundamentals

2.1. Irradiance Forecasting

2.1.1. Solar Irradiance

After arguing the specific reason and goals of this thesis, the fundamentals for the project are presented within this chapter.
So to create PV irradiance predictions, input features like environmental measurements and a corresponding target value are necessary to estimate a relation model [23]. Within this project the solar irradiance is chosen as independent value, representative as a generalized measure to the power outcome of a PV-system. The total solar irradiance, emitted by the sun, is measured by a set of three different variables, table 2.1. Those are the most meaningful to the PV-power-generation [46]:

Table 2.1.: Irradiance types in $\frac{W}{m^2}$

Name	**Description**
Direct normal irradiance (DNI)	Share of irradiance perpendicular to a specified surface
Diffuse horizontal irradiance (DHI)	Scattered part of the whole visible sky without, DNI
Global horizontal irradiance (GHI)	Geometric sum of the direct and diffuse horizontal components [46]

A common way of measuring those values is in use of a pyranometer, image 2.1. Within this device a sensor converts the irradiance to small voltage outcomes. The measured parameter hereby is defined by the type of pyranometer and installation position [46].

Figure 2.1.: Pyranometer next to a PV plant [26]

There are two common types of sensors [46]:

- Thermoelectric, a black coated semi-conductor, which converts the irradiance to a measurable heat within a bandwidth of 300 nanometer (nm) to 3000 nm and a response-time from 1sec to 30sec
- Photoelectric, a optical sensor with common bandwidth of 350 nm to 1100 nm and high response-time within milliseconds

2.1.2. Clear-sky-irradiance (CSIR) Model

The irradiance can not be only measured by sensors, as discussed in section 2.1.1, or derived via a PV systems. For the case of estimating irradiance in clear-sky conditions, physical models with respect to sun position, terrestrial location and air-mass are used [48]. Simple models on this topic are mainly based on geometric calculations for extraterrestrial irradiance [38]. Relevant air-mass, passed within the layers of atmosphere is defined as persistent, so those kind of models needs to be calibrated on a dedicated location.
More complex models, considering also environmental parameters such as temperature, air pressure, relative humidity and aerosol content. A common used example is the Ineichen Perez mode. As during their studies within different model on several locations, the linke turbidity shows high influence on the model accuracy [22]. For computational calculation this model is available, implemented in software libraries as pysolar [50].

2.1.3. Deterministic versus Probabilistic Forecasts

As mentioned in section 1.1, not only irradiance for clear-sky conditions, but estimation on real weather, like overcast or misty conditions, are desirable. So forecasting methods for a certain point of time up front are necessary. Those can be achieved by using the input data of weather related sensors within prediction models. There are two common types of prediction results. First the deterministic forecasting, where a relation between the selected input features and a target feature is used to predict the most likely result [19].

The second method, probabilistic forecasting, not only returns a most likely result, but a probability measure to the forecast-target-value. Those distributions can be generated by a model itself, where the values of input features can be set in relation to an expected

model error [47]. Alternatively to that approach, a confidence interval to a prediction can be calculated by the result of many parallel predictions on the same origin. As a consequence,a parametric distribution can be assumed on those individual forecasts [34].
Aside of the by ensemble predictions, exist the more specific prediction interval (PI). Those PI considers the information of input condition variance, to quantify a single predictions uncertainty [6]. As main difference between confidence interval and PI is that a confidence interval quantifies the population of an estimated variable, where the PI estimates the uncertainty of an single observation, estimated from the population [6]. Taking this into account a PI results in an wider interval, as it includes the confidence interval of the result, as well as the output variance in general. A common used method for PI estimation is the Boostrap Method. In this approach a single neuronal network is trained by a number of data subsets to estimate the models variance. Hence a model error can be calculated by the prediction variation of all model instances to the target [25].

2.1.4. Simple-smart-persistence (SSP)

An example for deterministic forecasting is the SSP method [21], where the future state of irradiance is determined to depend on the clear-sky-model and clear-sky-index (CSI). This method assumes that all relevant parameters, such as cloud coverage, temperature or composition of air-mass are persistent over time. Therefore formula 2.2 consists of two parts. First, formula 2.1, the CSI denoted as $K_t(t)$, as factor of measured irradiance $I_{mes}(t)$ to CSIR $I_{cs}(t)$, section 2.1.2, calculated for the time of measurement. In the second part of formula 2.2, the actual prediction takes place. In that case the calculated CSI *K_t(t)*, is used as a factor of correction to the future CSIR $I_{cs}(t+h)$. As by local climatic and weather deviations, with respect to the estimated CSIR may be observed, correction factor C is used to minimize the error by comparison to measured clear-sky-conditions. C serves also to adapt the SSP method by use of direct PV systems and their efficiency factor. This method is a common benchmark to ML models [32] [10]. Since the computing effort for ML models pays off only, by generating better prediction than SSP.

$$K_t(t) = \frac{I_{mes}(t)}{I_{cs}(t)} \tag{2.1}$$

$$SP(t,h) = C * K_t(t) * I_{cs}(t+h) \tag{2.2}$$

While SSP is, as the name indicates quite simple, more complex derivatives of this method are developed as well. A more sophisticate approach is presented within a publication by Kumler and Xie [28], which decomposes the GHI value to two components.

The first part is extraterrestrial solar radiation. Secondly cloud albedo and cloud fraction are extracted with respect to the extraterrestrial radiation from the measured GHI. Those values are estimated by the assumption of persistence via an exponential moving average to generate predictions within an forecast-horizon up to 60 minutes.

2.1.5. Numerical weather prediction (NWP) Model

Even more physical basics are used to predict future weather conditions, by NWP models. Those models are build on mathematical equations, which depend on different physical laws, e.g. fluid-dynamics and thermodynamics [31]. Therefore a certain terrestrial area is mapped to a grid with nodes and connections, displayed in figure 2.2 [43].

By considering those points with respective environmental data and the difference to their neighbors, complex dependencies can be derived. In use of fluid-dynamics and thermodynamics laws, this dependency is used to estimate future conditions. Based on those geophysical processes, forecast within minutes or climate-change over years can be predicted [31]. To name an example, the used NWP ICOsahedral Nonhydrostatic (ICON) [43] by the *Deutscher Wetterdienst (DWD)* is constructed by a triangular-grid with a node distance of 13km. All input parameters, by the network, are subdivided by their location $(Atmosphere, surface, sea, lake, ice)$

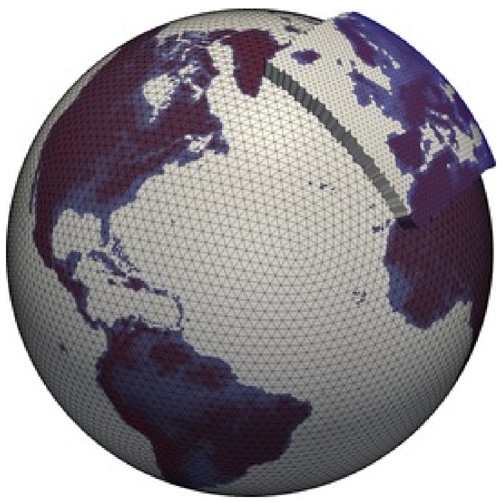


Figure 2.2.: Triangle-grid of ICON [43]

2.1.6. Image based Prediction

In comparison to a physic based model, image processing forecast approaches rely on the camera-visible weather situation. Hence not only the measured irradiance in comparison to the estimation of a clear-sky-model provides information about the actual cloud situation. In this case the more detailed position of cloud-fragments or brightness can be used to predict the behavior and thereby the overcast-rate and irradiance for future situations [37].

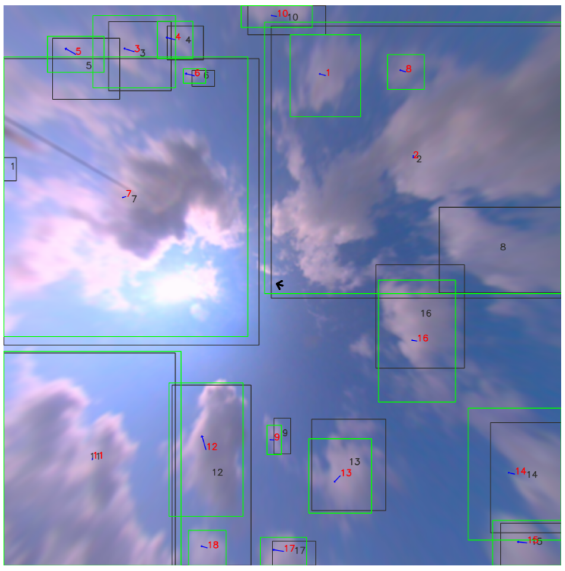


Figure 2.3.: Cloud motion vectors by tracking cloud movement [5]

This method generates more detailed information of weather and cloud development, but is limited to the field of view. Hence a cluster of stations would be needed to cover a wider area by cloud-shadow-maps, crucial for short-time irradiance forecasting [4].

2.1.7. ML Approach

Alternatively to physic based models and image processing algorithms, the machine learning approach relies on the relation of input features to a target value [19]. Where these features may include environmental sensor data, NWP-data [29] or also information gathered by image processing [37]. The target can be a single parameter, for example the power of a PV-system [32] or two parameters, like limits for the interval of possible outcome value [36].

2.2. ML Basics

2.2.1. ML Types

Since previously in section 2.1.7 mentioned, a ML prediction can be performed by using various different methods [19]. A way to categorize ML approaches, is by the training procedure. One group is the supervised training, which provides input feature along with a corresponding result. Hence the model is trained to estimate the underlying relation of those values, which is the case in most regression models [23].
In contrary, by the unsupervised approach a model just receives the input data, where the property and distribution of values or categories are used to classify by just providing the number of classes [23].
Reinforcement learning is compared to the supervised learning not training by comparing

the inputs to a result, but creating a prediction which is penalized or rewarded by a loss function. This method is used in situation, where a direct result is not available, but the prediction impact is measurable [23].

For the estimation of continuous numbers, for instance power outcome in relation to irradiance and temperature, a regression approach with supervised learning needs to be chosen. In contrary to a classification model, the regression is not only designed to decide between cases, but estimate a floating point value [19].

2.2.2. Ensemble-learning Methods

A more specific way of prediction-generation by machine-learning is the ensemble-learning method. Many simple models are trained on the same dataset. Thus not only one result is achieved, but as many as used models. This approach is also called ”wisdom of the crowd” [19]. Those results can then be handled by a hard voter as the ensembles median to get the final result. Alternatively a soft voter can be used, as *Blending*, with an additional trained model on the individual predictions and target results [19].

2.2.3. Optimization

Training an ML-model, is meant to optimize the specific parameters to reach a minimal prediction error. A boundary condition to this goal is, that the relation of input to target should not be learned by heart, what is called *Overfitting*. In detail, the performance on the known training-set is increasing, but stagnating on unknown data. So the models shows bad generalization ability [19]. In contrary *Underfitting* is the effect of not training sufficient enough, so the prediction error is higher than necessary. Alternatively the model may not be capable of creating the relation of input and result by complexity or dimension [23].
In any case, for model training a loss function is required, which quantifies the prediction error while training a model. A common used example are the squared-error-loss or the mean-squared-error (MSE) [23].
Based on those fundamentals depending on the type of ML approach, the loss function differs to optimize all model parameters. Therefore an iterative parameter-optimization tactic is the gradient descent. The gradient descent works iterative, after random initialization of the model parameters with a first evaluation of the prediction error according to the loss-function [23]. So the goal for each iteration is to minimize the error by adjusting the weights.

A crucial parameter in this approach is the step size, denoted as α, for each iteration. By a to small α, the progress for each step is very little or the function can get stuck in a local minimum, figure 2.4 with parameter vector θ. When setting α to big, the function may step over the minima and is also not converging to the best result.

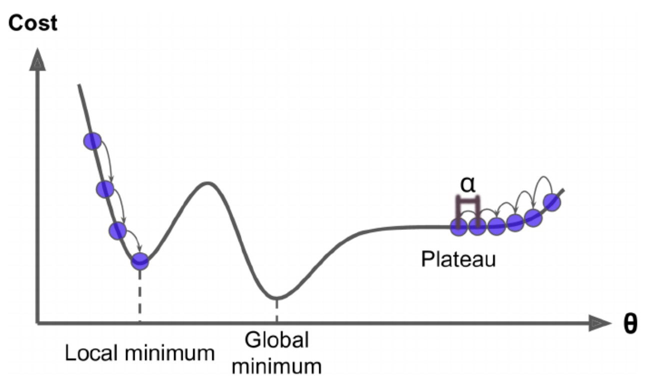


Figure 2.4.: Example of not reaching the global minima [19]

2.2.4. ML Examples

2.2.4.1. Linear regression (LR)

To name some common individual- and ensemble-models with relation to the project: One quite simple method is predicting the target by a linear function of all input features, the so called LR. Hereby the parameters have to be trained with the dataset are the specific weights. A common method is fitting by use of the least squares method [1], or approaching the optimal weights with a gradient descent approach [23]. As loss function, the average squared error or mean squared error are commonly used [23].

2.2.4.2. Support-vector-machine (SVM)

A more advanced approach than the linear regression is a SVM. Within this method all input features are non-linear-transformed into a higher dimension ($kerneltrick$) to fit hyperplanes and reach the best fitting linear solution for the prediction, displayed in figure 2.5. So the prediction result is defined, by the support-vector correspondence in the transformed space.
Hereby the transformation function can be chosen either to use a linear, polynomial or a radial basis function, depending on the properties and number of input features [23].

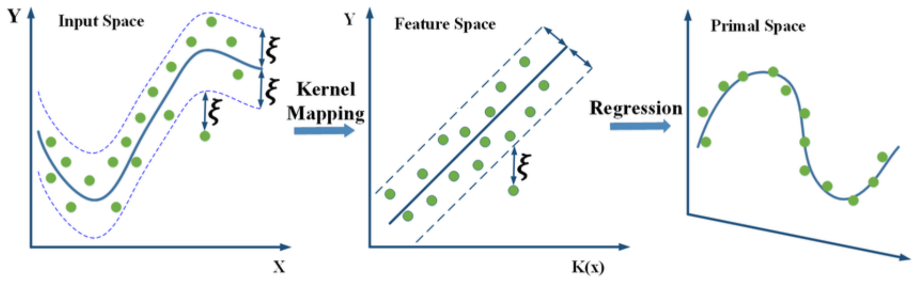


Figure 2.5.: Working principle of a simple SVM with one input feature [35]

For the loss-function a gradient descent algorithm can be utilized, similar to the linear regression. As a consequence, the best number and position of support vectors can be approximated, depending on the used transformation-type and dimension [23]. But compared to the linear regression, by feature-transformation, the SVM is computationally demanding.

2.2.4.3. K-nearest-neighbor (KNN)

Compared to the SVM, the KNN method, is based on an quite simple idea. The prediction is formed on the result of k data-points which approximates best the relation of input values to the target. While fitting the number, denoted as k, and feature-values for the stored data-points are chosen by converging the lowest prediction error. So in the act of passing a new data point to the function, a number of closest neighbors with respect to the input feature value are considered for estimating the resulting prediction [23]. Deviating to other approaches, like SVM or linear regression, KNN prediction, is based on real data points of the training set. Therefore the time, needed for fitting the parameters is much lower, than more complex functions e.g SVM. A disadvantage, which has to be mentioned is that the stored data points and consequently a possible exploit of the training data is feasible [23].

2.2.4.4. Decision Trees

An additional approach is the decision tree method, which is not based on weights or stored data as previously mentioned methods. Here, the prediction is generated by a flowchart-like path of decisions. The estimation of a target depends on the state of input parameters [23]. Based on that decisions are made which approach the resulting prediction value by every decision, figure 2.6.

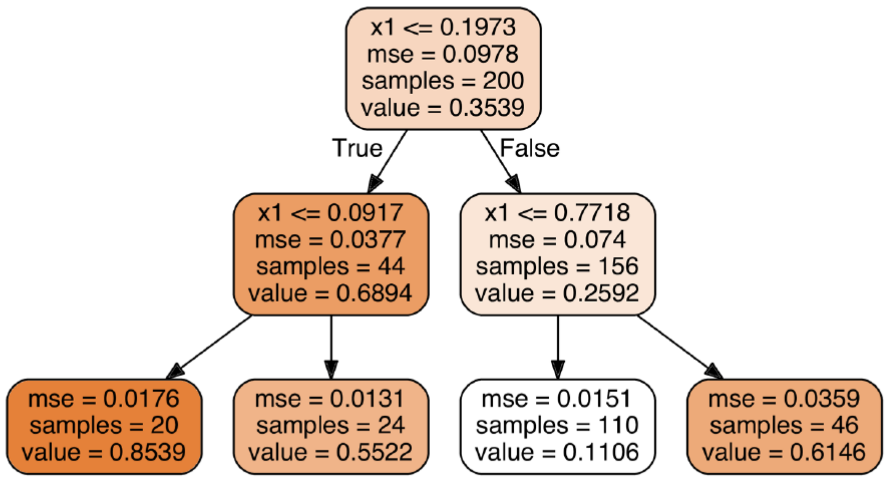


Figure 2.6.: Simple example for a regression task, using a decision tree [19]

To grow a tree implies the generation process of a tree-structure by adding decision nodes. Such nodes are generated while training the model. In the process the goal for a new decision is to separate the given data, based on a input-feature measure. So formula 2.5 result j minimizes by best separation of data m to subset m_{left} and m_{right}. A weighting factor to these separations is the MAE, by formula 2.4with formula 2.3. Here the averaged target values, denoted by $\hat{y}_{node}$ are compared to the decision related outcome value y^i. Those two error metrics calculate for each subset the average-target offset to the node prediction [19].

$$\hat{y}_{node} = \frac{1}{m_{node}} \sum_{i \in node} y^i \tag{2.3}$$

$$MSE_{node} = \sum_{i \in node} (\hat{y}_{node} - y^{(i)})^2 \tag{2.4}$$

$$j = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right} \tag{2.5}$$

In case of complex situations these trees may become big while optimizing for the best performance on the training set, which can result in large computation time in correlation to available resources [23].

2.2.4.5. Random forest (RF)

One ensemble method, section 2.2.2, of the decision-tree-approach is RF by use of bagging-method [19]. Hereby the procedure is to train several small decision trees with subsets of the given training data. For a final deterministic result, these predictions are accumulated by the mode or average. An advantage of this approach is the reduction of bias and variance, caused by the data-splitting and model aggregation [19]. Also the training procedure can be processed in parallel. Therefore a lower amount of time can be expected, than by training a single decision tree for the same task [19].

2.2.4.6. Boosting Method

As mentioned in the previous chapter 2.2.2, RF creates an ensemble with parallel performing models. In an opposite way, the Boosting method aligns all individual learner instances. Herby the goal is to create an iterative prediction. So the first module in the boosting-method creates a prediction. Afterwards the following instance is trying to minimize the forecast error of the first module. For that, each model is corrected by its predecessor [19]. An adaption of Boosting is the Gradient Boosting, which uses a gradi-

ent descent algorithm, section 2.2.3, scoring metric.
This gradient descent method can also be used within Riemannian structure of the parameter space [3], denoted by Amari and Douglas [3] as Natural Gradient. This approach results in better convergence behavior, referenced to the basic gradient descent method. Additionally the proposed NGB approach of Amari and Douglas [3] is utilizing a selected base learner model to estimate the parameters of a given distribution with respect to a scoring rule as displayed in figure 2.7. In this flow-chart, *x* denotes the input-vector, and *y* the predicted distribution parameters.

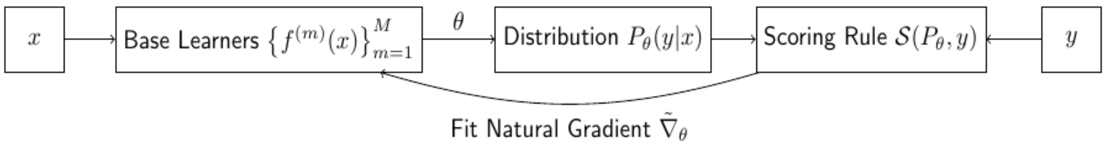

x
Base Learners $\{f^{(m)}(x)\}_{m=1}^{M}$
θ
Distribution $P_\theta(y|x)$
Scoring Rule $\mathcal{S}(P_\theta, y)$
y
Fit Natural Gradient $\tilde{\nabla}_\theta$

Figure 2.7.: NGB process components with

About the boosting method in general needs to be mentioned, that high computational cost are to expect, as parallelization is not feasible.

2.2.4.7. Multi-layer perceptron (MLP)

Compared to other methods, a MLP is designed by the method as a human brain is working on processing information. It consists of nodes as neurons and weighted connections [23] , figure 2.8.

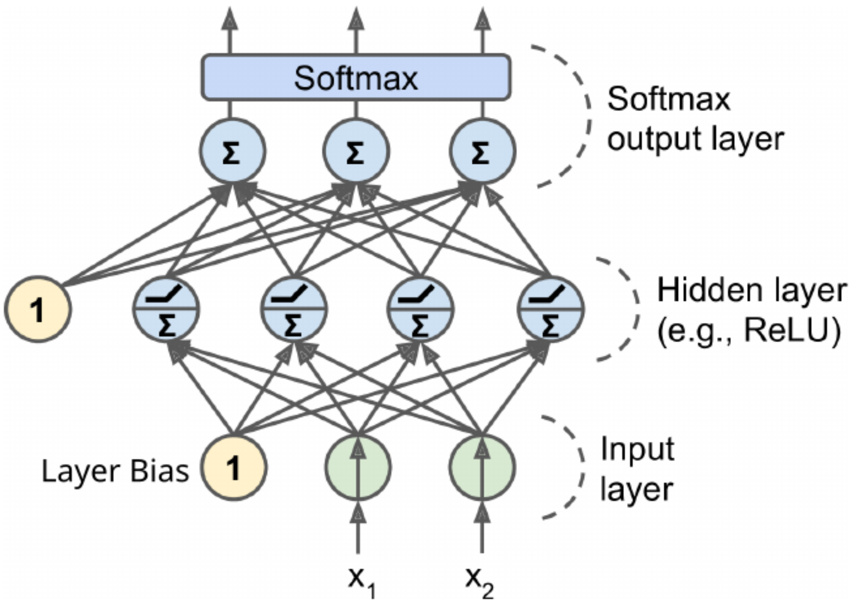


Figure 2.8.: Simple MLP for classification of three classes with bias nodes and activation function [19]

Hereby two inputs x_1 x_2 and a bias factor is feeding a *Hidden Layer* via weighted connections. These layers consist of a defined amount of nodes. Formula 2.6 shows the resulting equation for a neuron in the hidden-layer. The activation function in this example, 2.8 is the rectivied linear units (ReLu), formula 2.7. But it can be selected from different functions (Tanh, Step, Logit, Softmax, ReLu) [19] with respect to the given task.

$$n = \mathrm{ReLu}(w_1 * bias + w_2 * X_1 + w_3 * X_2) \tag{2.6}$$

$$ReLu(x) = \begin{cases} 0, \; if \; x < 0 \\ x, \; if \; x >= 0 \end{cases} \tag{2.7}$$

With such circumstances a complex relation of input parameters to an output can be created. Depending on the task, this output can be a classification result, a value for a regressions or multiple values like for an upper and lower bound estimation [33]
The training of a network in this ML-method is the optimization of all weights in-between the neurons in order to evolve a relation of input features to the output. This can be achieved by using the backpropagation algorithm [44].
Ergo after a random initialization of all weights, a first prediction iteration is processed. Later by the resulting prediction error, the backpropagation algorithm calculates the influence of each neuron on the prediction error. This is accomplished by iteratively processing the error contribution for each layer, until reaching the input layer [19]. Consequently the error gradient across all connection weights can be measured with the condition that all activation functions are continuous and differentiable [19]. The finalization of the training step is an update of all weights with respect to the error gradient.

3. State of Research

3.1. Deterministic Irradiance Forecasting

3.1.1. ML Model Comparison

As the basic topics and fundamentals for irradiance forecasting are exposited, this chapter gives an insight on the state of publications with respect to the project. Therefore the paper of Dávid Markovics and Martin János Mayer [32] presents an excellent overview on different ML approaches for 15-minute-ahead-prediction on 16 power plants. Several sets of engineered input features have been, as displayed in figure 3.1. Additionally performance of all models are evaluated with function-default and tuned hyperparameters [32].

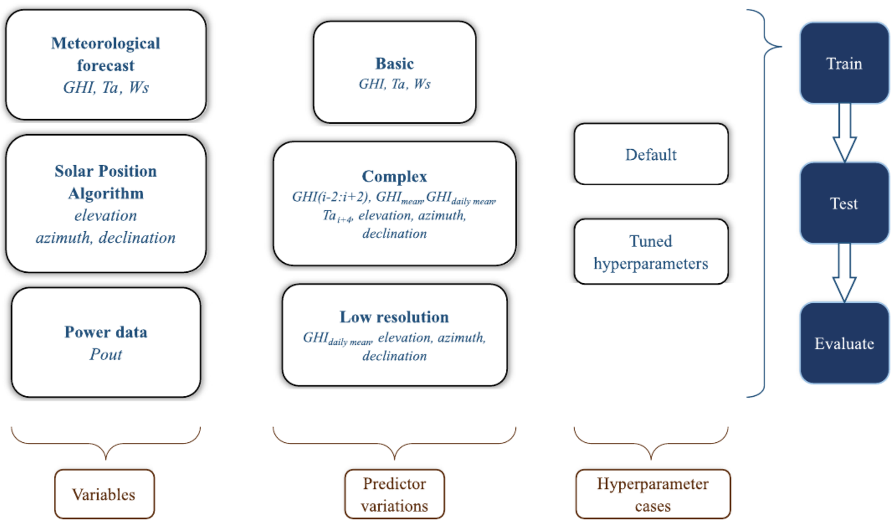


Figure 3.1.: Different input-feature-sets [1] and training options [32].

These results of all individual test cases are highly valuable, as a selection of appropriate ML models can be made. Since the outcome for all approaches and used training sets are listed, including the performance, see appendix A . The evaluated metrics include the root-mean-squared-error (RMSE), which, compared to the MAE, shows possible trends to outliers in the predictions. Both give an idea about the prediction error of the model in the targets unit. Additionally the Pearson-correlation-coefficient is a measure for the correlation of the target and prediction data. As a reference to the performance of a

[1]Ta: Ambient temperature, Ws: Wind-speed

smart-persistence model, the skill-score shows wether a ML-model outperforming the smart-persistence and is considerable or not.

Moreover, as these models are trained on the data of different power plants, the paper [32] provides histograms of best performing hyperparameters on all models. This serves as a useful reference on which hyperparameters can focus on, while training one of these model for irradiance prediction.

3.1.2. Persistence Forecast Data as ML Input Feature

Generating deterministic forecasts is also the major topic in the publication of Huertas and Brito [21]. They discuss the influence of a physic-based prediction method as input parameter of a ML-model. Hereby the physic-based prediction is a SSP approach which feeds a RF model.

The environmental input feature set includes measured CSI values K_t, their mean and standard deviation is calculated by the last hour of measurement. Thus, the mean value $\tilde{K}_t$ for the calculation of the standard deviation $\sigma_{Kt}(t)$ is a rolling window value, where the calculation is shifted with the step i for the last 60 minutes as w, see formula 3.1

$$\sigma_{Kt}(t) = \sqrt{\frac{1}{w}\sum_{i=0}^{w}(K_t(t-i) - \tilde{K}_t(t-i))^2} \tag{3.1}$$

This is extended by the SSP forecast data, where the SSP is calculated with respect to the forecast horizon. Similar to the environmental values, a mean and standard deviation is calculated for the SSP result.

By training of the RF model, a maximum number of trees is set to 500. Based on this generated forecasting approach, an evaluation on different forecast horizons is made. Hence in figure 3.2 all used feature sets are evaluated across the forecast horizon. In that regard the model with only *measured* features performs worse, followed by the model with *predicted* features except the SSP prediction. A superiority to these approaches generates the RF models, feed with *aggregated* features, including measurement mean and standard deviation as well as SSP mean and standard deviation. Which outperformed by the model with features from *instantaneous* data, which include measurements and SSP prediction. The best results are generated by using *all* features, what provides the most data.

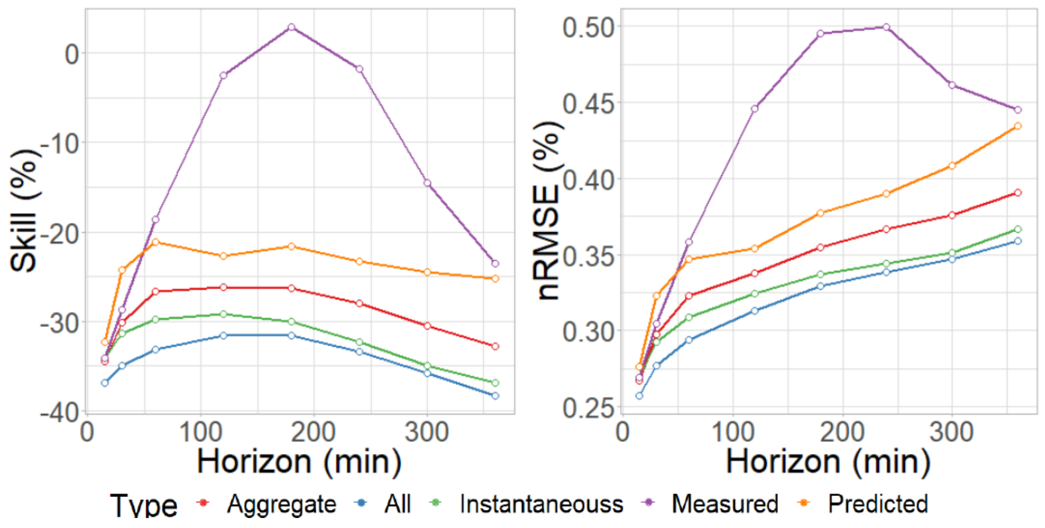


Figure 3.2.: Skill-score with and normalized RMSE by the observed target data, with respect to the forecast horizon [21]

Concluding the results of Huertas and Brito [21] forecasting data of an additional model can improve the prediction results, as well as combining predictions and measurements enhanced the estimation process [21]. A second learning is the error behavior per forecast horizon, which shows a steep inclination within about an hour. After this, the error slope almost stagnates.

3.2. Probabilistic Irradiance Prediction

3.2.1. Multi-model Ensemble Result Distribution

In order to estimate confidence intervals, the step from a deterministic ensemble to a probabilistic one, is crucial. This can be done by accumulating the individual prediction of several ML-models, that is shown in the publication of Mohammed and Yaqub [34], as part of the Global Energy Forecasting Competition (GEFCOM) [20].

During this competition hourly data points are provided, one week ahead for three time intervals within one year. After 16 weeks the whole dataset consists of 56.953 data points between April 2012 until May 2014. Available features for training and prediction are listed in table 3.1. As already mentioned, the time distance between two data points is one hour, all grouped on a hourly base. Defined by properties of all features, an impact on the solar power is generated by each [34], so no further engineering is performed.

Table 3.1.: Features of GEFCOM [20] competition dataset

Description	**Unit**
Total column liquid water	kg/m^2
Total column ice water	kg/m^2
Surface pressure	Pa
Relative humidity at 1000 mbar	%
Total cloud cover	$0 - 1$
10 metre U wind component	m/s
10 metre V wind component	m/s
2 metre temperature	K
Surface solar rad down	J/m^2
Surface thermal rad down	J/m^2
Top net solar rad	J/m^2
Total precipitation	m

By using the provided data, seven individual models are trained. Analyzed ML-models within the ensemble approach are [34]:

- Decision Tree Regressor
- RF Regressor
- KNN Regressor with uniform weighting
- KNN Regressor with distance weighting
- Ridge Regression
- Lasso Regression
- Gradient Boosting Regressor

To form a probabilistic result, three methods are chosen, where a *Naive model* [34] calculates a cumulative probability distribution on the results. The second method is an assumption of *Normal distribution* with mean and standard deviation of the deterministic predictions, to calculate 99 quantiles. As an adoption of this method, the third approach uses additional data, by performing within two different initial settings with the models and adding the respective month as an feature.
By evaluation of these three approaches, the superiority of the third method with different initial settings is declared.
Also a fluctuation of the loss over the year is mentioned within the publication. Therefore Mohammed and Yaqub argue that the main reason is the cloud cover, which results

in lower offset by more clear-sky occurrence in summer as lower irradiance fluctuation results [34].

3.2.2. Natural Gradient Boosting Method

As a different approach on generation of probabilistic predictions, the publication of Mitrentsis and Lens [33] uses a NGB approach to predict data within an interval of 15 minutes to 36 hours. The offset between the single predictions is set to 15 minutes, like the dataset resolution. All measurements of the dataset are supplied by two PV plants in Southern Germany, covering an time-interval from February 2018 until October 2019. As not relevant for irradiance prediction, night time is excluded.
By the reason that each data point only supplies information on the actual environmental state, no knowledge of the past is included. To tackle this issue, historic information by shifted timestamps is appended. So three past measurements are added with ($t-15$, $t-30$, and $t-45$). Also data with a past offset of 24 hours has been added, but neglected as no increase of prediction accuracy is traceable. However the dataset is appended by information on the specific part of the year. Hence two continuous values as circle positions are added, see formulas 3.2 and 3.3.

$$month_{sin} = sin(2\pi * \frac{month}{12}) \tag{3.2}$$

$$month_{cos} = cos(2\pi * \frac{month}{12}) \tag{3.3}$$

In order to evaluation, two further methods are implemented as well. These are a Gaussian-Process [54] and a Lower Upper Bound Estimator (LUBE). For this benchmark, the used metrics are for deterministic predictions MAE, RMSE and mean-bias-error (MBE), presented in section 6.1. For probabilistic forecasts interval coverage probability (ICP) and normalized mean interval width (MIW), see section 6.1, are chosen. As the paper argues on development of high accurate forecasting, an important influence are the input features. To understand the feature importance within the NGB-model, for each feature the shapley additive explanation (SHAP) score [30] is calculated. Hereby the model performance is observed with and without a specific feature to estimate the importance within the prediction process. Thus with respect to the models and forecast horizon, feature impacts are displayed in figure 3.3.

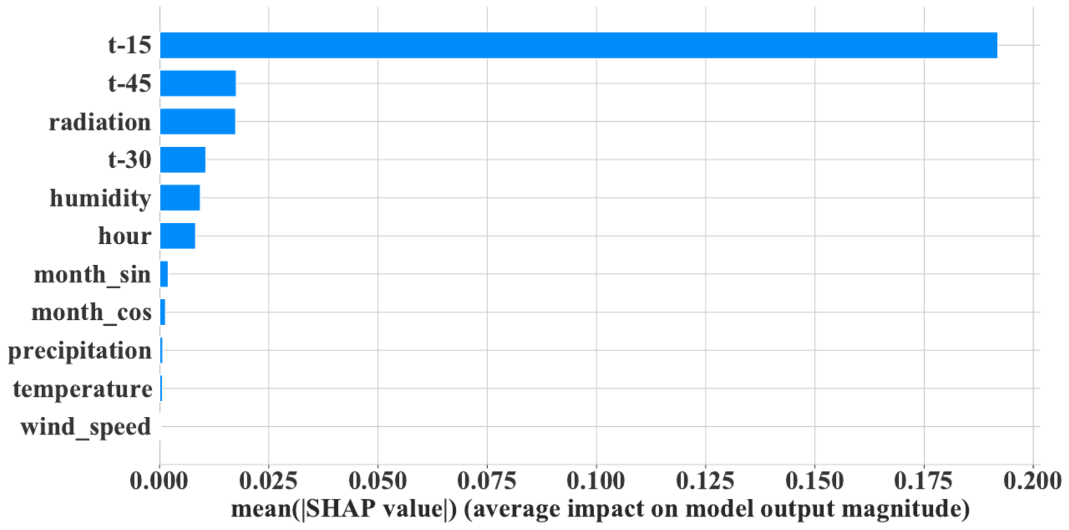


Figure 3.3.: Feature impact to prediction [33]

By the information provided of the feature impact, a subset of input is taken for NGB training as well as the full set. As a result for deterministic and probabilistic prediction an improvement by use of the subset it observed. Therefore the corresponding results by the two training-approaches are shown in table 3.4. Mentionable is here the improvement for the feature subset where the RMSE decreases by 6% and the CRPS by roughly 10%. That shows the outperforming accuracy by using the reduced set. Mitrentsis and Lens argue this result, by a local optimum training solution due to the higher dimensionality when using all features [33].

Figure 3.4.: Averaged metrics of the two training approaches with all features and a selected subset [33]

	NGBoost (all features)	NGBoost (reduced features)
MAE (pu)	0.0366	0.0326
RMSE (pu)	0.0617	0.0579
MBE (pu)	-0.0009	-0.0002
PICP (-)	0.8615	0.8720
PINAW (pu)	0.1658	0.1785
CRPS (pu)	0.0275	0.0247

Concluding the findings, the optimal feature selection is highly depending on the specific model and prediction target, more than on the dataset itself. Furthermore, additional features lead to a higher complexity and extended training times [33].

4. Data Preparation and preparatory Work

4.1. Dataset Introduction

The dataset, which has been chosen for this project is provided by the *DLR-Almería*, figure 4.1, Energy Meteorology team of Dr. Stefan Wilbert [53].
The time resolution of the measurements is one minute, covering a time frame from the 1st of January 2015 til the 31th of December 2019 with 1.18 million measurements.

Figure 4.1.: Site of measurement at Plataforma Solar de Almería [53]

For each datapoint available measurements are:

- Irradiance-data
  - GHI in W/m^2
  - DNI in W/m^2
  - DHI in W/m^2
- Environmental-data
  - Temperature in Celsius
  - Relative humidity in percent
  - Pressure in millibars
  - Wind-speed in km/h
  - Wind-direction in degree
  - Sun-position in degree azimuth and elevation
  - Linke-turbidity-factor in percent

4.2. Data Preparation and additional Features

In order to use the dataset for the whole project, all measurements prior to sunrise and after sunset are excluded, as they are considered irrelevant for irradiation measurement. Additional data-points with failed measurement, identified by zero or not a number (NaN) are excluded as well. Hence the remaining measurements are taken for feature-engineering.

Hereby additional features for backward information and value variance are calculated with respect to the publication of Pedro and Larson [40].

- **Backward average**, formula 4.1, gives information on the recent irradiance in steps of 5 minutes til 30 minutes in the past [40].
- **Lagged average**, formula 4.2, is calculated similar to the backward average, formula 4.1, but with a gap by one time offset value. This formula provides information on the past state without recent measurement [40].
- **Variability**, consists of formula 4.4, as the difference of past measurements in steps of five minutes. The root of these squared differences, normalized by the number of measurements, formula 4.3, provides information on how the given input value of irradiance varies on certain time spans [40].

$$\text{Backward average}_i(t) = \frac{1}{N} \sum_{t \in [t-i*5,\ t]} k_t(t)\ , \quad for\ i = \{1,2,...,6\} \tag{4.1}$$

$$\text{Lagged average}_i(t) = \frac{1}{N} \sum_{t \in [t-i*5, t-(i-1)*5]} k_t(t)\ , \quad for\ i = \{1,2,..,6\} \tag{4.2}$$

$$\text{Variability} = \frac{1}{N} \sqrt{\sum_{t \in]t-i*5,t]} \Delta k_t(t)^2}\ , \quad for\ i = \{1,2,...,6\} \tag{4.3}$$

$$\text{with } \Delta k_t(t) = k_t(t) - k_t(t-5*t) \tag{4.4}$$

These features depend on a specific amount of prior time data-points. Therefore the timestamps, consisting of insufficient prior data and failing in calculation are excluded. The remaining dataset is divided into a training set for the individual model-training, a validation set to evaluate model performance after training, a historic set for the similarity lookup and a performance set to test the method.

The usage of subsets, randomly chosen of the dataset without replacement, are defined as follows: The training-set is determined to train all individual deterministic models in the project, where the validation-set serves for estimation the prediction error on unseen data. As for selection of similar conditions for each prediction is necessary, the historic-set is taken for data-lookup. Additionally as there is no direct contact to the models for probabilistic estimation, it is also used for training these models. Finally the performance-set is proposed on benchmarking all individual ML models, as well as evaluating the whole project.

As last step on data preparation, standardization and data-scaling is performed. So by formula 4.5, with feature-mean and feature-standard-deviation, the data is transformed to a Gaussian-distribution. The scaling then is accomplished by formula 4.6, using the feature-extreme-values and spread on each feature. Thereby finally all data is distributed between an interval of zero and one as necessary for processing with ML-techniques [23].

$$x_{i\ normed} = \frac{x_i - \tilde{x}}{\sigma_x} \tag{4.5}$$

$$x_{i\ scaled} = \frac{x_{i\ normed} - min(x)}{max(x) - min(x)} \tag{4.6}$$

4.3. Reference Models

4.3.1. Smart Persistence Model

By completed definition of the dataset and subsets, this section presents the considered reference models for benchmarking the project-modules.
Therefore all individually used deterministic ML-functions are evaluated to the performance of SSP predictions, presented in section 2.1.4. The simple model was chosen, as a typically benchmark model, explained in section 2.1.4. A second reason is the lack of additional environmental data for a more accurate, physic based model as for example the Kumler *Smart Persistence* [28]. Thus the target value for all prediction serves the CSI to avoid a seasonal bias as in irradiance, see section 5.2. The CSI is calculated with respect to the CSIR-model of Ineichen and Perez [22].

Resulting to this adaption, the SSP by formula 2.2 is simplified to formula 4.7 with the estimated correction factor of 1.05. The factor is an approximation by comparison of CSIR-predictions to clear sky condition days within the dataset. These clear sky conditions are identified by a undisturbed GHI curve, qualitatively compared to the estimation generated by the SSP.

$$SP(t,h) = 1.05 * CSI(t) * I_{cs}(t+h) \tag{4.7}$$

4.3.2. Probabilistic Prediction Reference Model

As the generation of probabilistic intervals is the goal of this project, there is another reference model to benchmark the method of probabilistic prediction. This model consists of three parts. One part is the resulting combination of selected forecasts to an analog ensemble [2] The analog-ensemble (ANEN) is estimated by formula 4.8, where the distance of a data point t to historic measurements T is evaluated. In that regard all features are normalized by σ_{fi} and compared to historic features $F_i - A_t$ by weight vector W_i. The sum of these root-squared distances defines the resulting similarity score of the data point to a historic one, figure 4.2. With respect to the historical timestamp and the forecast horizon, then the respective target value is added to the ensemble.
Limited by the available time horizon on the project, the weight-vector W_i is not estimated by approaching the lowest prediction error, but set to one for all values.

$$||F_t, A_T|| = \sum_{i=1}^{N} \frac{W_i}{\sigma_{fi}} \sqrt{\sum_{j=-\tilde{t}}^{\tilde{t}} (F_{i,t+j} - A_{i,T+j})^2} \tag{4.8}$$

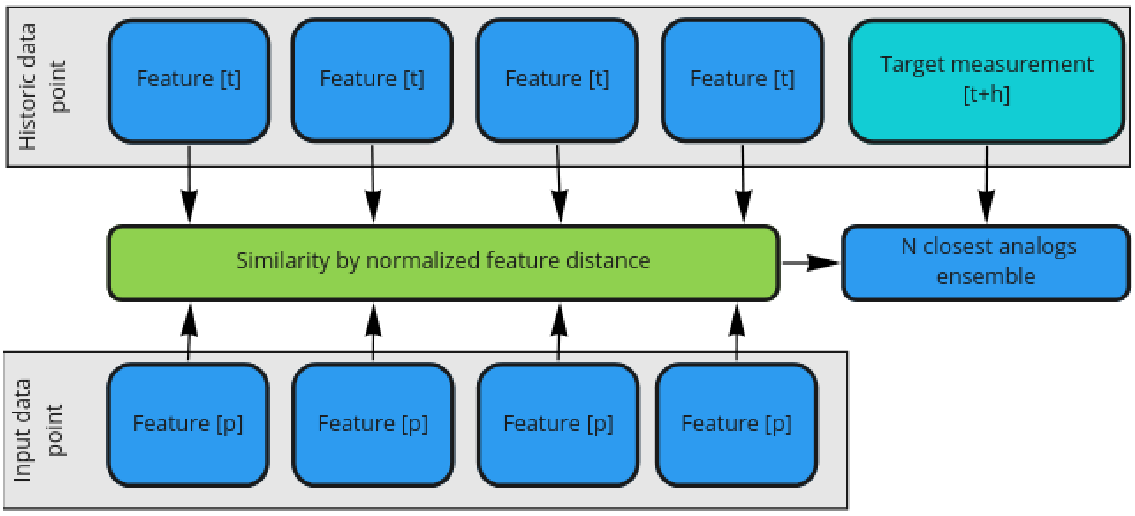


Figure 4.2.: Comparison of features for similarity measurement and forecast extraction [2] *process in green, data in blue, time-offset-data in light blue*

As second part a prediction ensemble is generated by a specific number of best performing individual ML models. These m models are selected by performance on k historical

data-points, evaluated in use of the dynamic selection method [11]. By the evaluation of Santos and Domingos [11] for three of four sites the values $m = 3$ and $k = 20$ are producing the best results, out of which a prediction ensemble is generated.
After the resulting analog-ensemble and prediction-ensemble, a third combined-ensemble is generated by combining both ensemble results. For each of these groups of deterministic predictions, probabilistic predictions are generated as benchmark-method to this project. The confidence interval for each ensemble, is estimated based on the assumption, of normal-distributed predictions. [34], displayed in figure 4.3.

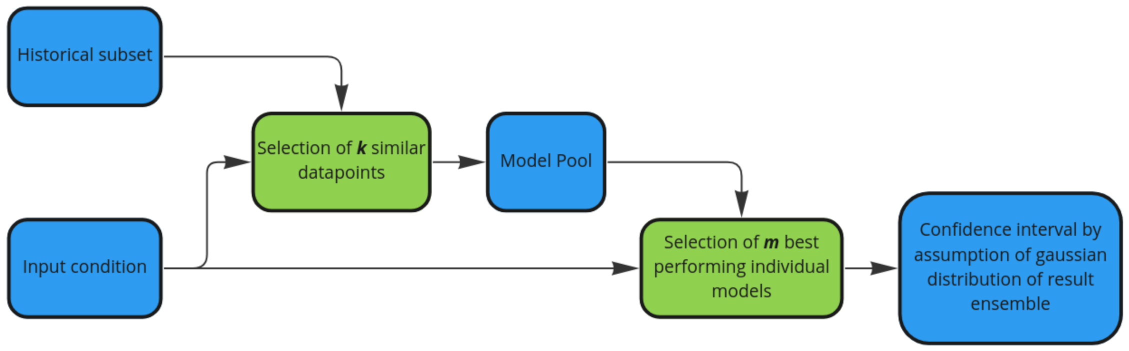

Historical subset
Selection of ***k*** similar datapoints
Model Pool
Input condition
Selection of ***m*** best performing individual models
Confidence interval by assumption of gaussian distribution of result ensemble

Figure 4.3.: Flowchart of generated reference model function, with respect to dynamic selection process [11], for probabilistic prediction generation by an Gaussian confidence interval *process in green, data in blue*

5. Implementation of the Probabilistic Dynamic Ensemble

5.1. Concept of the Dynamic-Probabilistic-Model

After introducing the relevant past work, the concept of the proposed method is declared within this chapter. The basic idea is to take an advantage of the knowledge from past situations and combine it with the predictions of the best performing models on those cases, displayed in the flow chart 5.1.

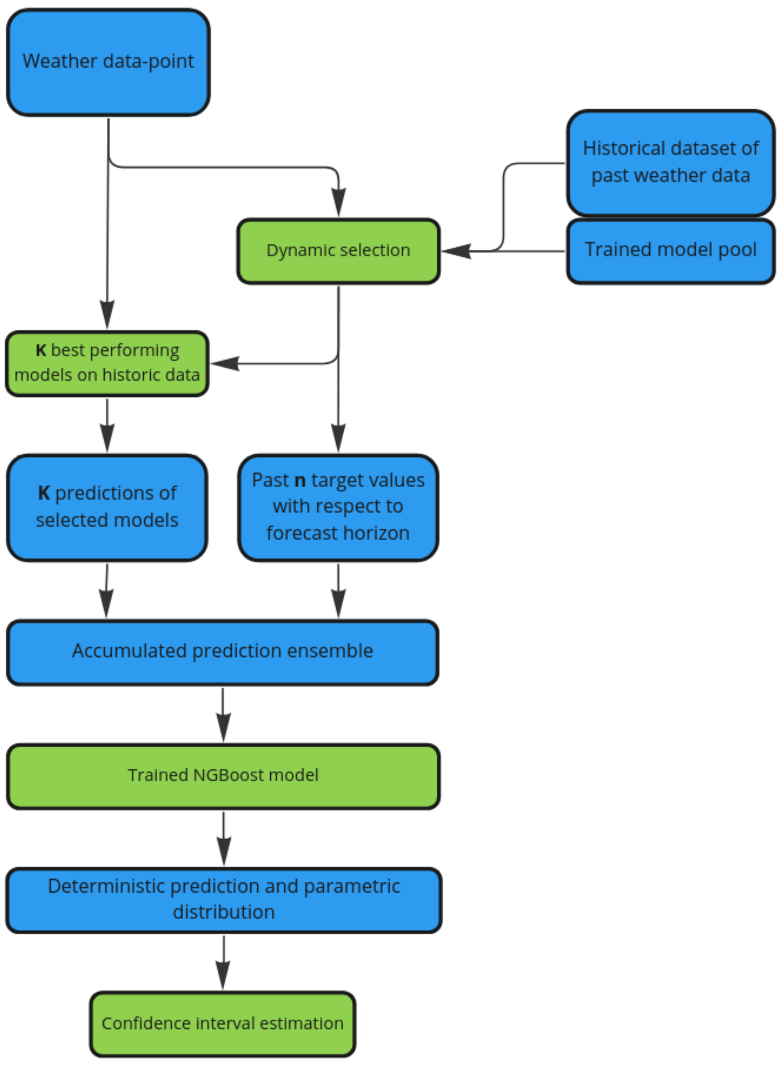

Weather data-point
Historical dataset of past weather data
Dynamic selection
Trained model pool
K best performing models on historic data
K predictions of selected models
Past n target values with respect to forecast horizon
Accumulated prediction ensemble
Trained NGBoost model
Deterministic prediction and parametric distribution
Confidence interval estimation

Figure 5.1.: Process of a prediction by use of the proposed method

Nevertheless wrong information can still be within this accumulated ensemble, by wrong predictions or not correlating past situation.
To get the best feasible results, all data is post-processed by an NGB model. This model is trained by the ensemble result and target-value, to learn the importance of certain measurements compared to the result. An advantage of the NGB approach [13] is the resulting information of the most certain deterministic prediction, as well as the selected parametric distribution.
In order to create a probabilistic prediction to a given data-point, a historic dataset from the same origin as the input data is needed. Additionally a pool of trained models and prediction results of all historic data is needed in prior as well. For the model pool, a most promising selection by the result of Markovics [32] is taken. As these models are trained with different training approaches, the algorithm can select the best models within this pool, figure 5.2.

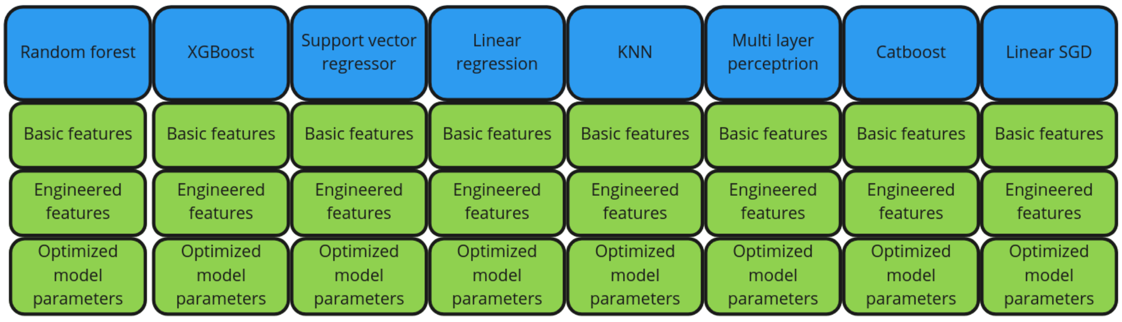


Figure 5.2.: Available models *blue* by selected training approaches *green*

5.2. Dataset Preparation

This chapter explains the implementation steps of the probabilistic approach, written in Python [16] 3. All used libraries are declared in appendix G.
For the task of data preparation, figure 5.3, the provided data is read from a hierarchical data format (HDF) [18] file to a *Pandas DataFrame* [42]. This data type provides various features and functions useful for handling big sets of data.
As it is common to use the CSI for PV prediction tasks [57] [40] the GHI value is converted to the CSI.

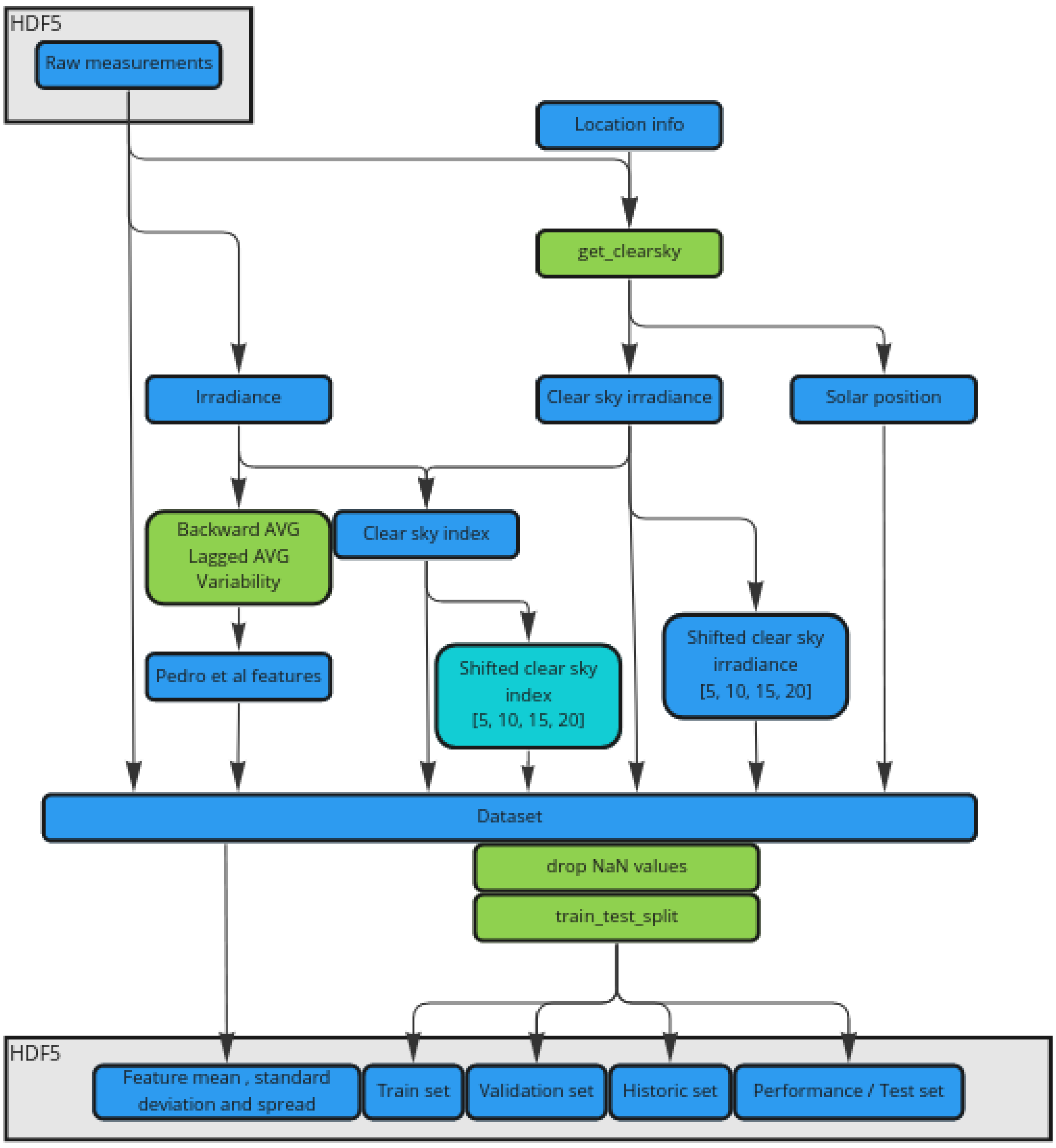


Figure 5.3.: Process of raw data preprocessing with functions in green, data in blue and light blue marked target value

The main reason is that the CSI does not inherit high seasonal bias, in reference to the raw GHI displayed in figure 5.4. Necessary to mention is that the a bias also for the CSI persists, by the seasonal weather conditions and cloud coverage [52].
Consequently CSI is calculated by formula 2.1 with respect to the CSIR of the given data.

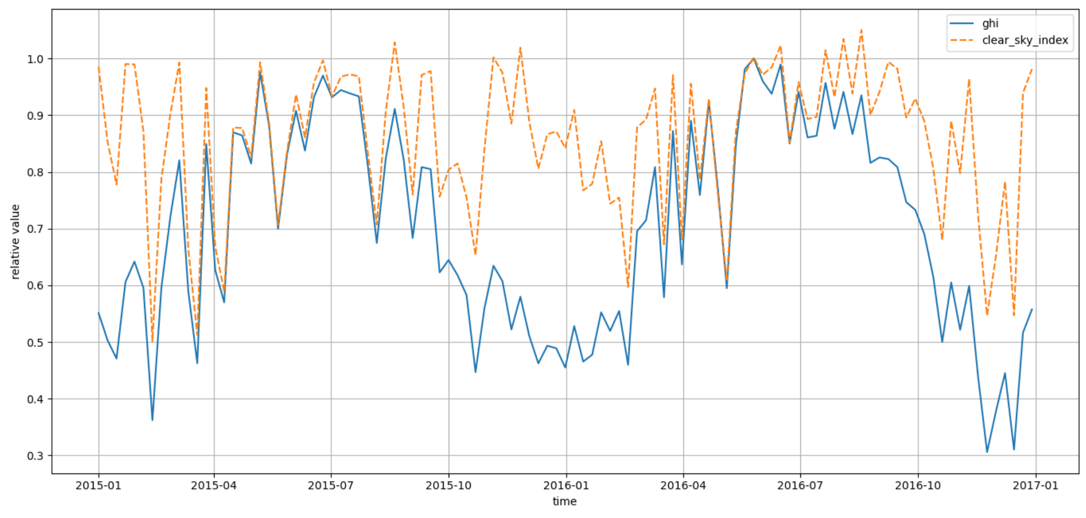


Figure 5.4.: High biased normalized GHI value in blue compared to CSI_GHI in orange with low seasonal bias, data-points averaged by seven days

To estimate those CSIR values, the function *get_clearsky* of the pvlib-library [49] is chosen. The calculation is based on the location of measurement: latitude = 37.093040, longitude = -2.2353150 and altitude = 500 meter sea level to calculate the respective solar movement by a given timestamp. As this information is valuable by its relation to the irradiance, the solar elevation, azimuth and zenith are appended to the dataset.
The estimation of the CSIR itself is calculated by the model of Ineichen and Perez [22], see section 2.1.2. For the calculation, air-pressure, by altitude, and the linke-turbidity are provided as important parts to estimate the influence of air-mass. As in figure 5.5 displayed, during clear-sky-condition, the estimated values overlap the real measurement within a small offset. To avoid information of nighttime measurements, all sections of the dataset, containing information before and after sunset, are excluded by their time-index.

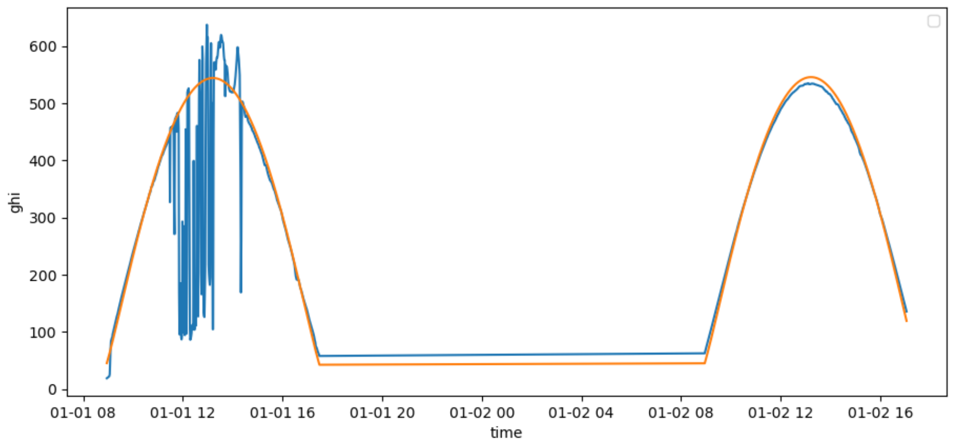


Figure 5.5.: CSIR (orange) compared to real GHI (blue) for a overcast day 1st Jan 2015 on the left and clear day 2nd Jan 2015 on the right hand side

Additional to the CSI as input feature and reference to the cloud coverage, future CSI values are added to the data points within an interval of 5, 10, 15 and 20 minutes. These values are used for model training and evaluation, as target value to the given forecast horizon.
To provide an information of past condition for each data point as well, the backward-feature-generation-method from Pedro et. al [40], section 4.2, is selected. These formulas set past information by the given times step, which is not provided within data points in the first rows or ones including faulty or missing measurements. Respective timestamps are neglected for the final dataset. At last the subsets needed for model training and evaluation, section 4.2 are randomized and divided by the function *train_test_split* of the scikit-learn (sklearn) library [39]. These are listed in table 5.1.

Table 5.1.: Composition of datasets and share according to total usable data

Dataset	**Share**	**Data points**
Train	48 %	569,339
Validation	12 %	142,336
Historic	20 %	237,224
Performance	20 %	237,224
Total	100 %	1,168,123

The distribution for the CSI of those subsets is presented in figure 5.6. As all graphs almost perfectly align, a bias is expected to be minimal. As mentioned in section 4.2 all features are standardized by their mean and standard deviation and scaled to an interval from zero to one.

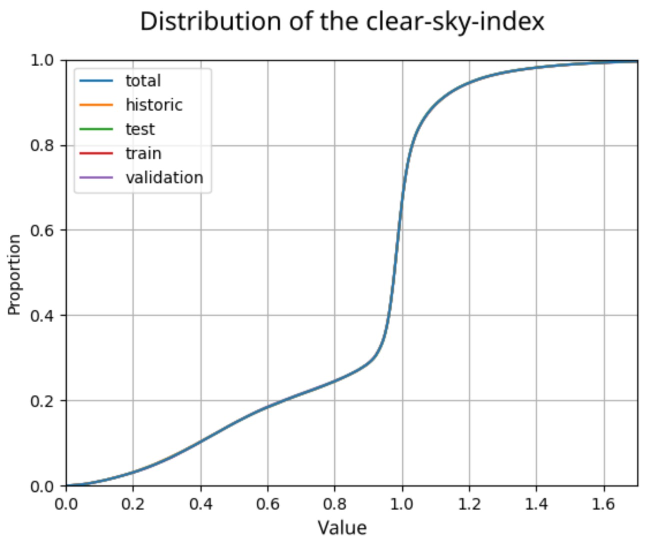


Figure 5.6.: CSI cumulative distribution of all datasets

5.3. Model Training

5.3.1. Training Sections

The following step after data preparation is the training of all individual ML-models, figure 5.7. Therefore predefined model-implementations from sklearn for (RF, SVM, LR, KNN, MLP and Linear-stochastic gradient descent (SGD)) are used. For extreme gradient boost (XGBoost) the correspondent library [8] is integrated, as well as for CatBoost [41].
To evaluate all named individual models by feature importance and hyperparameter optimization, the training-process is subdivided to three sections, table 5.2.

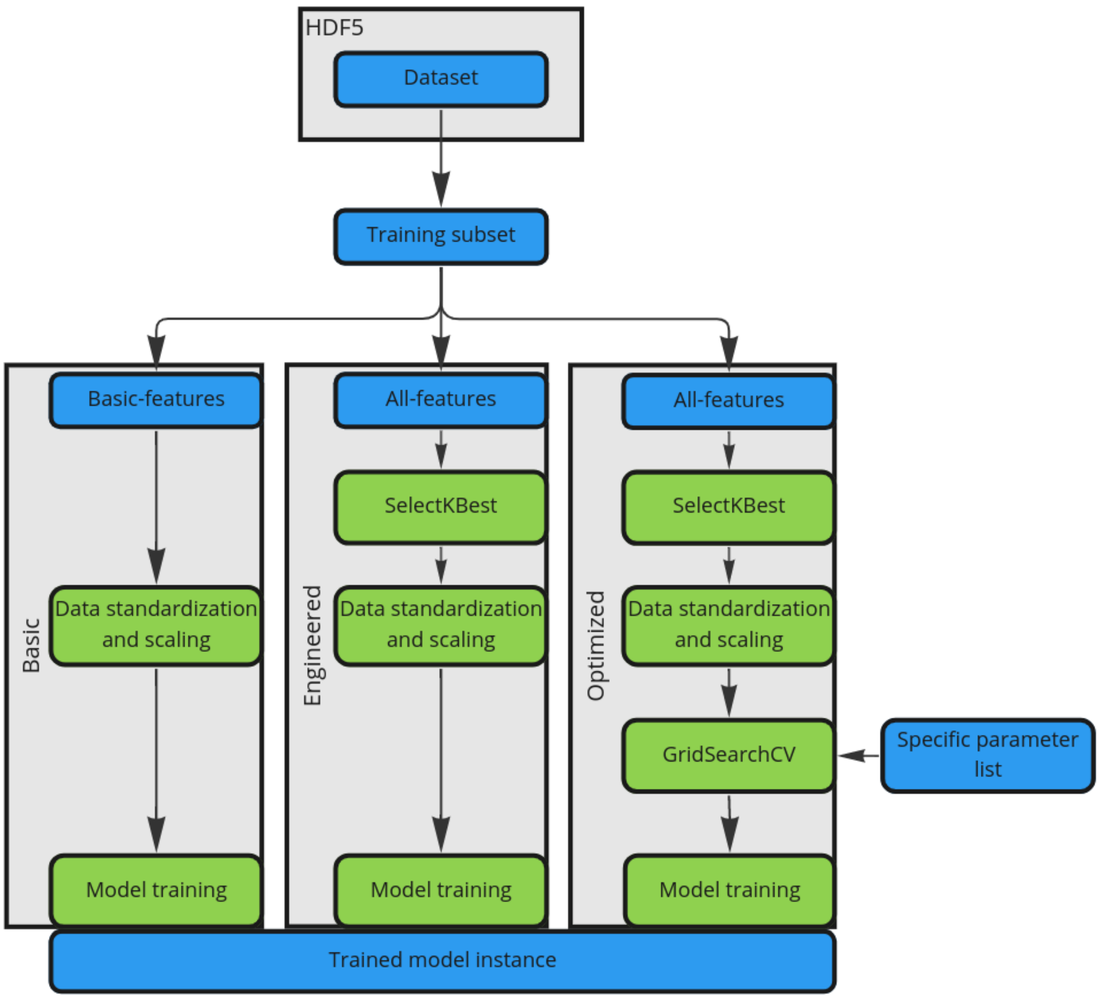


Figure 5.7.: Flow chart of the training process and result generation

- **Basic feature set** considering only raw measurement data.
- **Enigneered feature set** by feature evaluation with all available measurement-data, including the backward information, section 4.2, introduced by Pedro Et al. [40]
- **Optimized model set** with optimized model parameters. The notation *standard parameters* refers to the implemented values of the used libraries [7] [41] [8]. By *optimized parameters* the best performing options, see appendix B, by a parameter-evaluation with *GridSearchCV* of the sklearn module are selected.

Table 5.2.: Feature sets for model training and parameter estimation

Section Notation	Features
Basic features $[standard\ parameters]$	CSI, temperature, humidity, wind speed, sun elevation and azimuth
Engineered features $[standard\ parameters]$	Basic features with linke-turbidity Plain irradiance types: GHI, DNI, DHI, GHI-CSIR Backward engineered features for (5,10,...30 minutes) [40]: GHI-backward average , GHI-lagged average , GHI-variance
Optimized models $[optimized\ parameters]$	Basic features with linke-turbidity Plain irradiance types: GHI, DNI, DHI, GHI-CSIR Backward engineered features for (5,10,...30 minutes) [40]: GHI-backward average , GHI-lagged average , GHI-variance

5.3.1.1. Feature Engineering

For feature-evaluation, every feature importance is measured by the correlation to its target value, to estimate the influence in the prediction process.
This scoring is achieved by the *SelectKBest* function from the sklearn module [7]. The optimization metric for the function is set to *f_regression*. This score calculates the cross-correlation, formula 5.1, with input X from the features to the target value Y.
By the information of feature-influence, a subset of 18 features is selected, displayed in figure 5.8.

$$\text{Cross-correlation-score } = \frac{x - \tilde{x} * \tilde{y}}{\sigma_x * \sigma_y} \tag{5.1}$$

For better comparability, all scores are normalized for figure 5.8. As the score shows a second significance step within the logarithmic scale after lagged-GHI-30, the first 17 features are selected, marked by the green area in figure 5.8.
The feature naming consists of *l* for lagged, *b* for backward, *v* for variability and *cs* for clear-sky-irradiance. By a comparison of the importance score for all forecast horizons, only small significance deviations are remarkable for DNI. Hence the difference for the azimuth score within the forecast horizons may be neglected as the score is at the lower significance-end of the evaluated field.

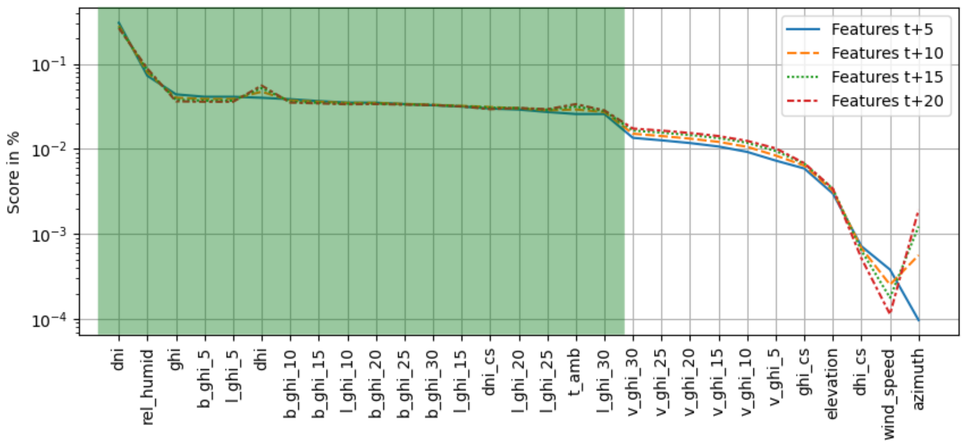


Figure 5.8.: Logarithmic score of available features with respect to target

5.3.1.2. Optimized Model Training

The in this way generated subset of input features are pitched for model optimization as well. To set the available parameters of each ML-model to an appropriate number, the same options as in the publication of Markovics [32] are taken into account, 5.3.
To gather the best parameters, the selection function *GridSearchCV* of the sklearn module [7] is utilized. This method trains a model on subsets of the training-set with a generated grid of given parameters. The so generated models-instances are evaluated to a circumstance specific scoring function. Hence the best parameters for the mentioned metric can be estimated.
For the hyperparameter tuning the negative-RMSE is taken with three folds for cross-validation, chosen due to computational resources.

Table 5.3.: Markovics model parameters [32] used in *GridSearchCV* [7]

Model	Parameters	Options
LR	Normalize:	True, False
SVM	Kernel:	rbf
	Epsilon:	[0.5, 0.25, 0.23, 0.2, 0.15, 0.12, 0.1, 0.08]
	C:	[0.001, 0.01, 0.1, 0.5, 0.9, 1, 1.15]
MLP	hidden layer sizes	: [(5,), (10,), (50,), (5,5), (10,10), (5,5,5), (100,)],
	alpha:	[0.0001, 0.001, 0.01, 0.1, 1]
KNN	n neighbors:	[5, 10, 15, 20, 30, 50, 70, 100, 200, 300, 500]
RF	n estimators:	[50, 100, 250, 300]
	max depth:	[None, 5, 8]
	min samples leaf:	[100, 10, 1, 0.01]
	max features:	['auto', 'sqrt']
XGBoost	objective:	[logistic, squarederror]
	learning rate:	[0.01, 0.05, 0.1, 0.3]
	max depth:	[3, 4, 5, 6]
	min child weight:	[1, 2 ,3, 4]
	n estimators:	[50, 100, 200]
	reg lambda:	[0.1, 0.4, 1]
CatBoost	learning rate:	[0.03, 0.1]
	max depth:	[-1, 3, 5, 6, 8, 10, 15, 25, 31, 50]
	l2 leaf reg:	[3, 10, 50, 100, 200, 500, 1000]
	iterations:	[50, 100, 150]
Linear-SGD	loss:	[squared error,huber,epsilon insensitive
free chosen		,squared epsilon insensitive]
parameter set	alpha:	[0.0001, 0.001, 0.01, 0.1, 1]
	penalty:	[l2, l2, elasticnet]
	max iter:	[500,1000,5000,10000]

5.3.1.3. Prediction Storage

Each mentioned training approach, section 5.3.1, generates prediction results which are normalized and scaled values, such as the targets passed for training purpose in section 4.2. Therefore the reverse calculation with respect to formula 4.5 and 4.6 is processed. This results back to CSI values. To generate an outcome with an unit, better assessable, the CSIR is chosen to receive a irradiance prediction.
For later use all these generated predictions are stored with specific naming (e.g. random_forest_optimized_model_5) containing the model, the training approach and the forecast horizon. As the storage type,HDF [18] is used by the method *DataFrame.to_hdf* [42].

This file-format can inherit different datasets along with informative attributes and offers an uncomplicated handling [18]. In the file all predictions are sorted by respective keys for the dataset on which the predictions are generated.

5.3.2. NGB Training

In order to generate probabilistic predictions, a NGB-model for each forecast horizon is trained. Accordingly the in python3 [16] implemented NGB library [13] is included. The process of training the network is done in use of the validation-set. This data is considered to create the training-input-data with the dynamic-selection-process [11], as the set is not utilized within other training or testing processes.
The first step is similar to the approach for analog-ensemble-generation presented in section 4.3.2, which is denoted as *Calculate similarity score* in figure 5.9. Followed by the dynamic-selection as proposed by Santos and Domingso [11].

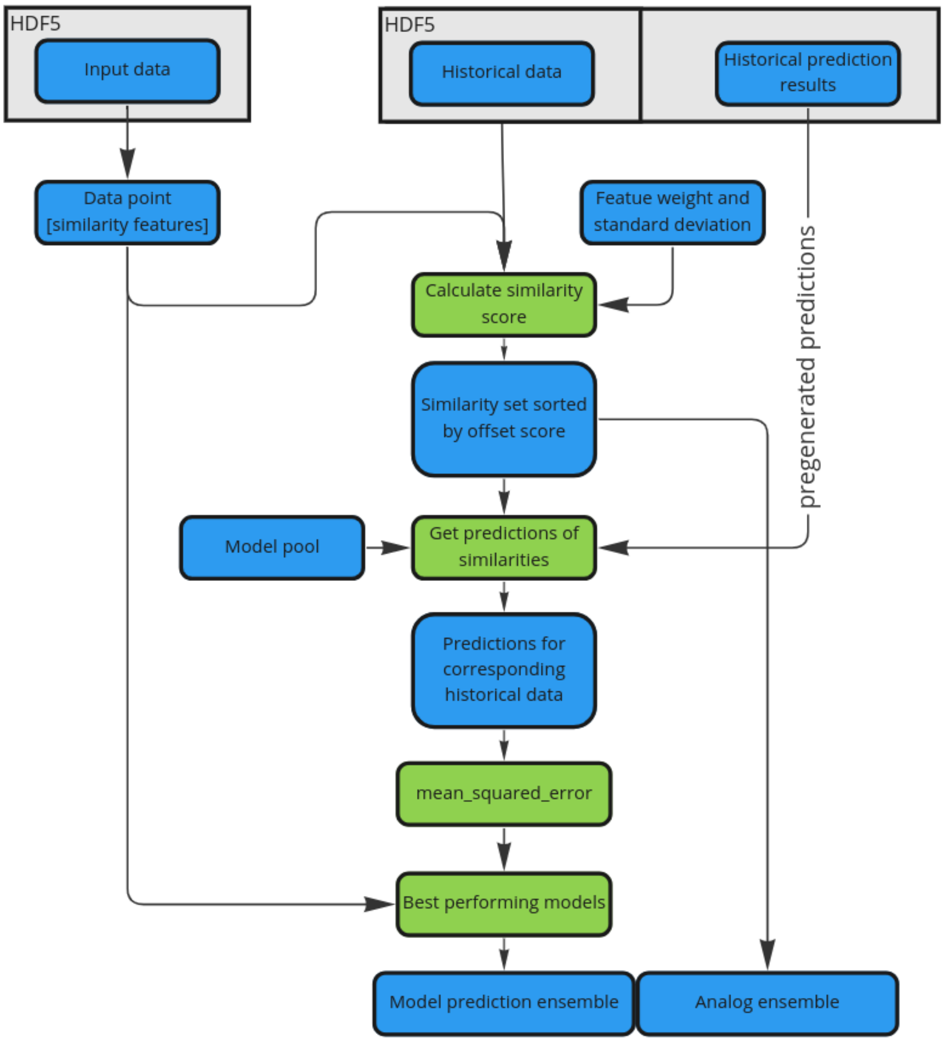

HDF5
Input data
HDF5
Historical data
Historical prediction results
Data point [similarity features]
Featue weight and standard deviation
pregenerated predictions
Calculate similarity score
Similarity set sorted by offset score
Model pool
Get predictions of similarities
Predictions for corresponding historical data
mean_squared_error
Best performing models
Model prediction ensemble
Analog ensemble

Figure 5.9.: Flow chart of dynamic model selection [11] by evaluation of similar data points with data in blue and processes in green

For this reason the data and predictions of the historic-dataset is extracted from the result-HDF. The selection of similar data points is calculated by the euclidean distance to a given situation.
The features to be considered for this comparison are the features used for model training. As the different measurements spread by value, they are normalized by their standard deviation. The sum of these differences is the similarity score, where the smallest score represents the highest similarity.
In consequence the dataset is sorted with decreasing similarity-value. Being mentioned in section 4.3.2, best results are generated by considering 20 data points. Their future CSI values, with respect to the forecast horizon, accumulates to the analog-ensemble.
The amount of selected data-points is crucial, as a low number represents a lack of information diversity. In contrary a high value exposes the risk of generating a high variance within the data subset.
Therefore the selection of 20 samples and three best performing models are evaluated by Santos and Neto [11] as best approach. In this project, a total number of ten mode-predictions is chosen to provide additional information to the probabilistic prediction of the NGB-model.
Looking-up the prediction of all individual ML-models is achieved by a functionality of the pandas *DataFrame* [42]. So with the index of the similarity set, the corresponding predictions and their target values can be selected.
These results in a dataset of predictions and measured target values for occurrences are comparable to the input data point. Where the case that no prediction data is available, the model-pool is loaded and predictions are generated by the selected historical data-points.

The next step is to evaluate the prediction performance of the data set, with the purpose that the RMSE of all models by the included target value is calculated with the corresponding function of sklearn [7]. These metric results in information on the model performance, where the ten best performing are chosen to generate prediction for the input situation. All prediction of those models for the given input data-point are accumulated to the model-prediction-ensemble. Concatenating the dataset of analog-ensembles and model-prediction-ensembles, the result is used for training the NGB-model as presented in figure 5.10.

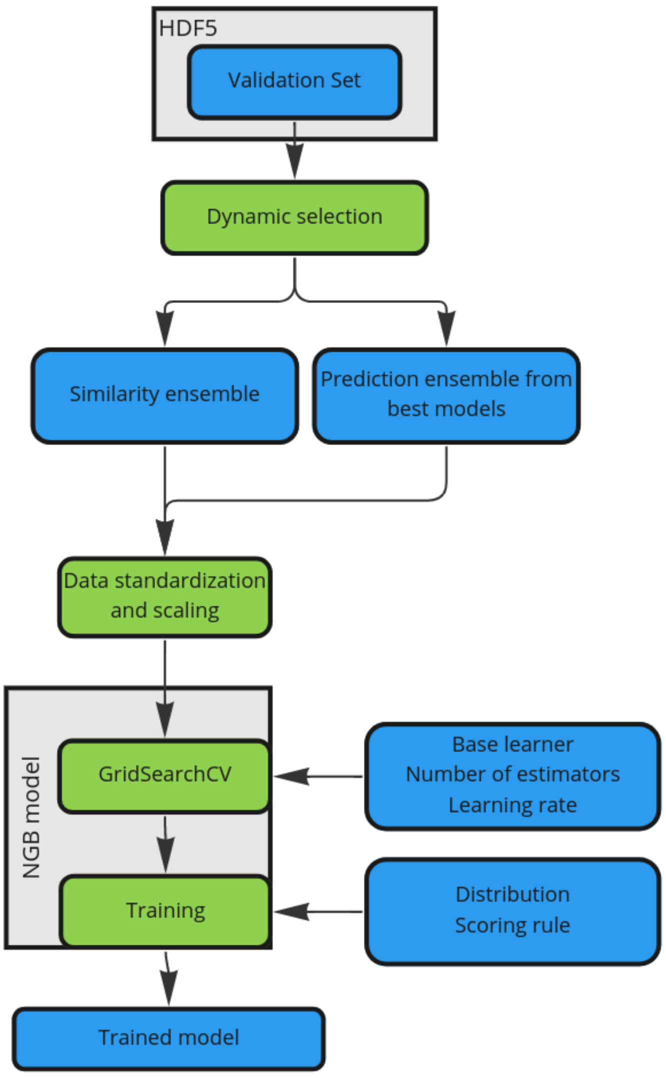


Figure 5.10.: Flow chart of NGB training procedure with functions in green and data in blue

Hereby the ensemble-dataset is normalized and scaled as mentioned in section 4.2. While training the NGB, the parameter selection function *GridSearchCV* [7] is utilized as with the deterministic models in section 5.3.1.2. The used parameters are listed in table 5.4. They include the options as considered in the publication of Duan and Anand [13] with additional options by deviation parameters.

Table 5.4.: Used parameters for the gridsearch method

Parameter	**Options**
Base learner	Decision tree with squared error criterion max depth [3, 4] MLP with 100 hidden layers, ReLu activation
Number of estimators	[100, 200, 400, 600, 800]
Natural gradient	[True, False]
Learning rate	[0.0001, 0.001, 0.01, 0.1]

The selected parametric distribution within the NGB module is the Normal-distribution as proposed by Duan [13], along with the CRPS as scoring metric.

5.3.3. Probabilistic Prediction

Consequently to the training approach, the data with prior normalization and scaling is processed. These deterministic predictions are feed with the input data to the NGB-model, figure 5.11. The resulting probabilistic predictions are de-normalized and scaled back by irradiance-value properties. Since the Gaussian distribution is chosen for the probabilistic prediction, a confidence interval is calculated by an interval function of the NGB library [13], considering the respective mean and standard deviation.
Within the reference function, the interval is calculated by the *t.interval* function out of the *stats* module from the *scipy* library [51].

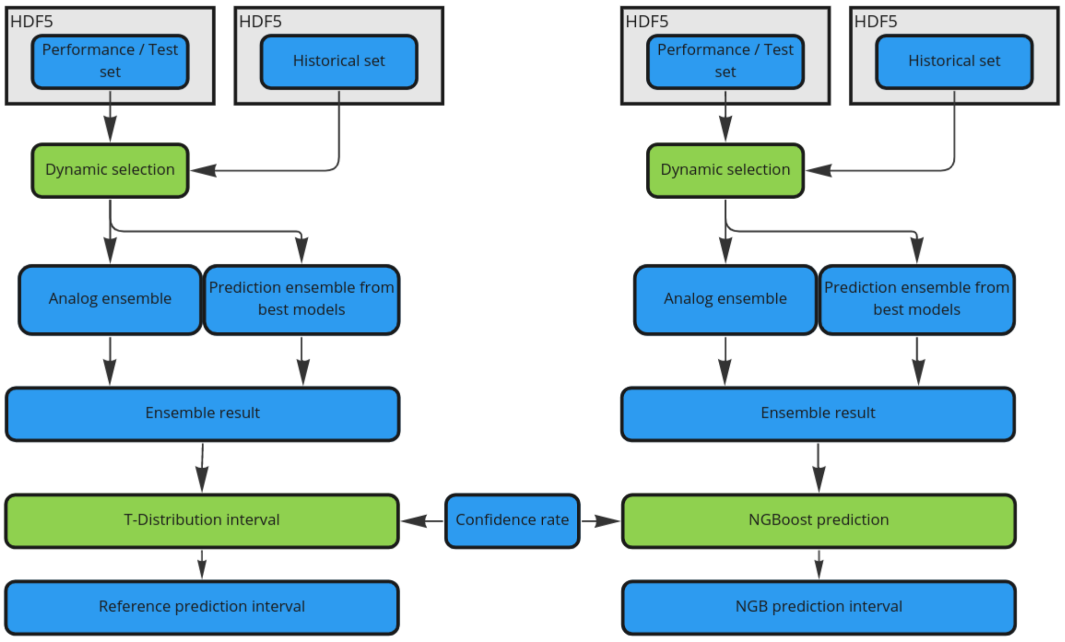


Figure 5.11.: Generation of probabilistic predictions by mixture of deterministic models and input data

6. Method evaluation

6.1. Quality Measures and Scores

6.1.1. Deterministic Prediction Metrics

To evaluate all presented parts of the method, chapter 5, different scoring metrics are necessary for deterministic and probabilistic results. Hence for deterministic results of the individual models, section 5.3.1, chosen evaluation metrics are MAE , MSE, RMSEand Skill-score. So MAE exploits the average prediction error with equal weighting of all individual errors, formula 6.1 with $\hat{y}$ **as prediction**, y **as the target** and n **the number of data points** . All metrics are calculated after decomposing the CSI back to GHI by the specific CSIR, as the SI-based unit $\frac{W}{m^2}$ of an irradiance is more accessible.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} y_i - \hat{y}_i \tag{6.1}$$

Compared to this equal weighting, the MSE penalizes high errors by the squared-value, formula 6.3. Therefore models, which tend to outliers can be identified by this metric. A downside of this evaluation score is that resulting values are not directly comparable to the target unit.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6.2}$$

To have an outlier highlighting value, sharing the targets unit, the RMSE is calculated, formula 6.3.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \tag{6.3}$$

As a difference to these error based metrics, the skill-score compares a model performance to a benchmark method. A common used comparison metric within the skill-score is the RMSE [56]. For irradiance predictions, a reasonable benchmark method is the smart persistence [56] or in this case by lack of data, the SSP, section 2.1.4. Hence by formula 6.4 the factor of majority to the SSP method is calculated.

$$\text{Skill-Score} = 1 - \frac{RMSE_{model}}{RMSE_{benchmark}} \tag{6.4}$$

6.1.2. Probabilistic Prediction Metrics

As these metrics are not practicable with confidence interval results, therefore various are chosen: The ICP, MIW and the CRPS [55].

First there is the ICP, which is a score of how many target values fall within the estimated interval, formula 6.5. So the hit-index c_i, formula 6.6 is summed up for all predictions, where it is set to one in case the true value y_i is within the interval lower bound $L(x_i)$ and upper bound $U(x_i)$. To receive a relative score, the sum of hit-points is then divided by the number fo measurements N.

$$\text{ICP} = \frac{1}{N}\sum_{i=1}^{N} c_i \tag{6.5}$$

$$\text{with } \ c_i = \begin{cases} 1, \text{if } y_i \in [L(x_i), U(x_i)] \\ 0, \text{else} \end{cases} \tag{6.6}$$

Due to the reasons, that predictions whit a high uncertainty result in an wide interval and thus can generate a high ICP by a poor prediction. To tackle that issue, the ICP should go along with the MIW. The MIW is a metric to provide information about the interval uncertainty and therefore its width. Hence in formula 6.7 sums up all interval-spreads by $U(x_i) - L(x_i)$ where c_i indicates a hit of the target value. By this factor, only successful predictions are evaluated to generate a relation of both metrics to another. Resulting to this adaption, all failed predictions are set to zero, consequently the sum off interval-width is normalized by the count of hits $\hat{c}$ in formula 6.8.

$$\text{MIW} = \frac{1}{\hat{c}}\sum_{i=1}^{N} (U(x_i) - L(x_i)) * c_i \tag{6.7}$$

$$\hat{c} = \sum_{i=1}^{N} c_i \tag{6.8}$$

So by considering ICP and MIW, the performance of a probabilistic prediction can be evaluated by success and precision. The remaining downside is, that these metrics are only feasible for fix interval bounds. In comparison, to evaluate an interval by its distribution, the CRPS is chosen. As it computes the integral of squared differences between the distributions cumulative density function $F(y)$ and the observation $F_o(y)$, formula6.9 [17].

$$\text{CRPS} = \int_{-\infty}^{\infty} [F(y) - F_o(y)]^2 dy \tag{6.9}$$

6.2. Individual Deterministic Predictions

After all metric in use are declared, the performance of individual deterministic models in operation, see section 5.1, are listed. Firstly the smallest forecast-horizon and basic input features as in section 5.3.1.1 mentioned are evaluated. Consequently figure 6.1 compares all models on each metric. So by MAE, CatBoost, MLP and Random Forest perform better than the reference model SSP. But with respect to MSE and RMSE only CatBoost and MLP deliver better results than SSP what results to a positive Skill-score. For Random Forest, the results implies that the overall prediction is better than the reference model, but also produces an higher amount of outliers with big prediction offsets. Opposite behavior shows LR compared to KNN. So LR performs in average worse than KNN, but with less high offset errors. For LR and Linear-SGD is to be mentioned, that the model performance is worse than SSP. Therefore those models struggle with the input parameters or features. The worst performance is shown from the SVM, which produces in general bad results.

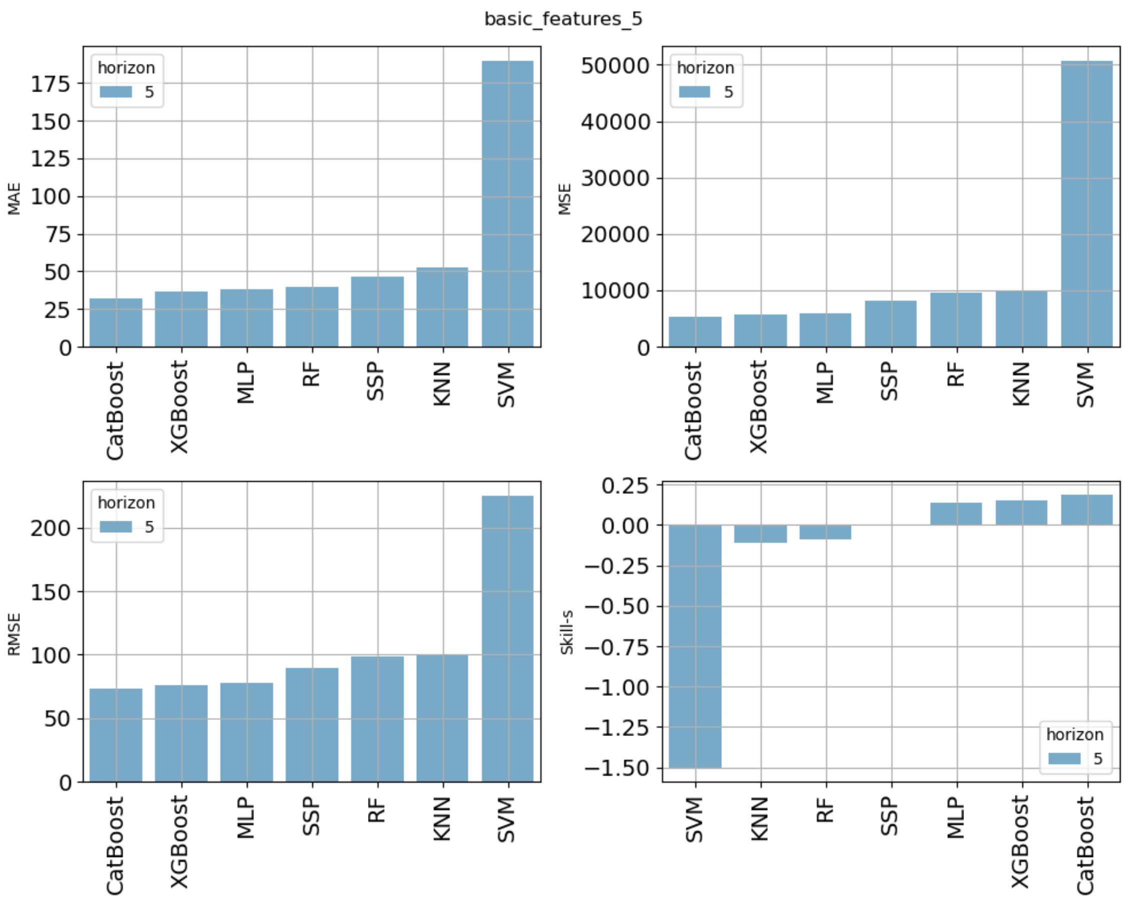


Figure 6.1.: Model performances with a forecast horizon of 5 minutes and basic features

The following training section, where engineered features, section 5.3.1.1, are used with default model parameters, generates results as seen in figure 6.2. Here, the most improvement with additional features and use of an feature selection algorithm can be observed within the KNN model. By exclusion of features with low target correlation, influence of more meaningful neighbors raises, as low impact features have no more influence. In comparison tho KNN, the MLP , LR and linear SGD metrics almost stagnates. Reason for that is the training process of these models, where low importance features receive small influence on the result generation.

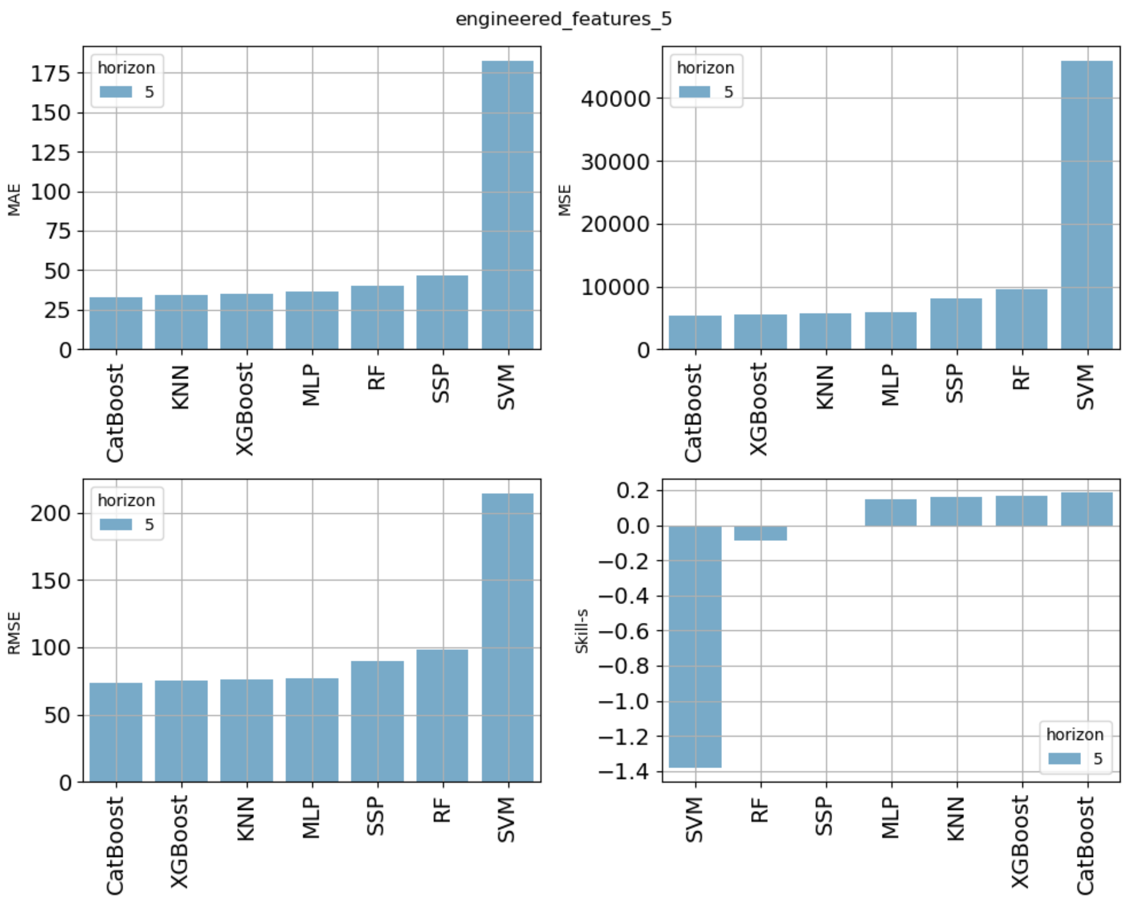


Figure 6.2.: Model performances with 5 minute forecast horizon and engineered features

The last training section involves optimized models by an searching algorithm, section 5.3.1, and given parameters to chose from table 5.3. With that regard a enhancement on all model, except for KNN, linear-SGD and MLP can be observed in figure 6.3. Hereby KNN stagnates, because the default number of neighbors from initialization has already produced best results. For linear-SGD and MLP the MAE and therefore average performance worsens.
But the RMSE, MSE and consequently Skill-score improves as the optimization algorithm uses the MSE as measure of improvement. Reason of this behavior is the hyperparamter optimization, where the RMSE is chosen as metric to penalize high errors .
Opposite to these models, the random forest performance is highly enhanced by optimized model parameters. So within the contained decision trees, the model is capable of assigning a low importance on low influence features, but the dimension and dept allows the training process to generate better structure for the task.

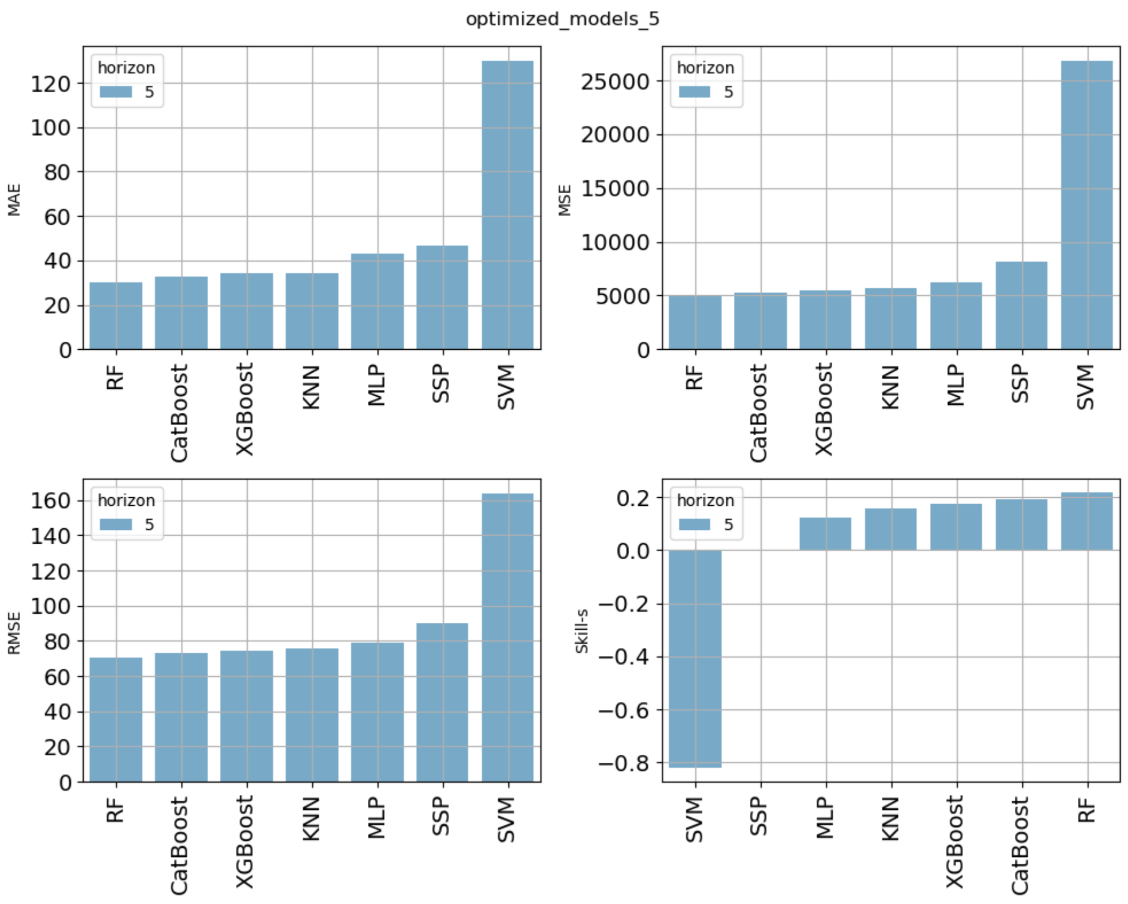


Figure 6.3.: Model performances with 5 minutes forecast horizon, engineered features and optimized model parameters

As the specific goal on individual models is to outperform the reference SSP method, all model results by forecast horizon are presented within this section. Hence for a forecast horizon of 5 minutes all model sections reach at least in one approach a better result than the SSP, except the Linear-SGD model and the SVM.
In consequence by comparing all four charts (6.4, 6.5, 6.6, 6.7), it is clearly visible that by rising error of the SSP prediction the score of all models increases.

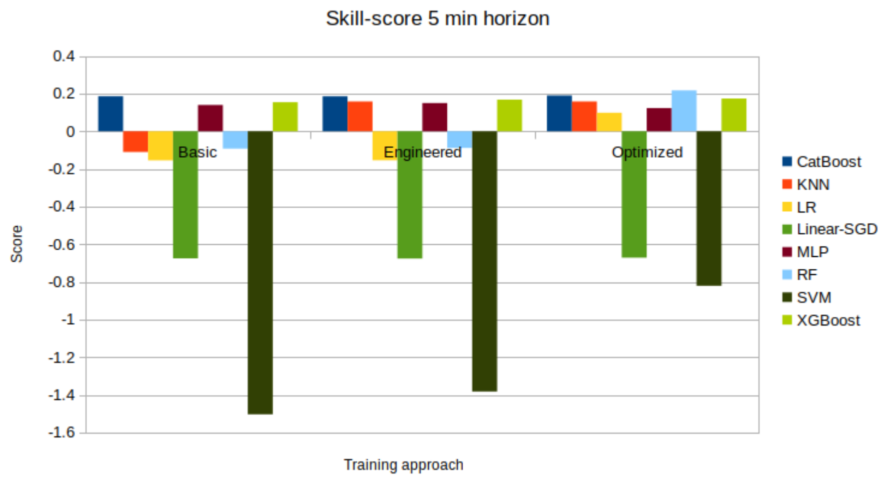


Figure 6.4.: Model Skill-score for each training approach to 5 minutes horizon

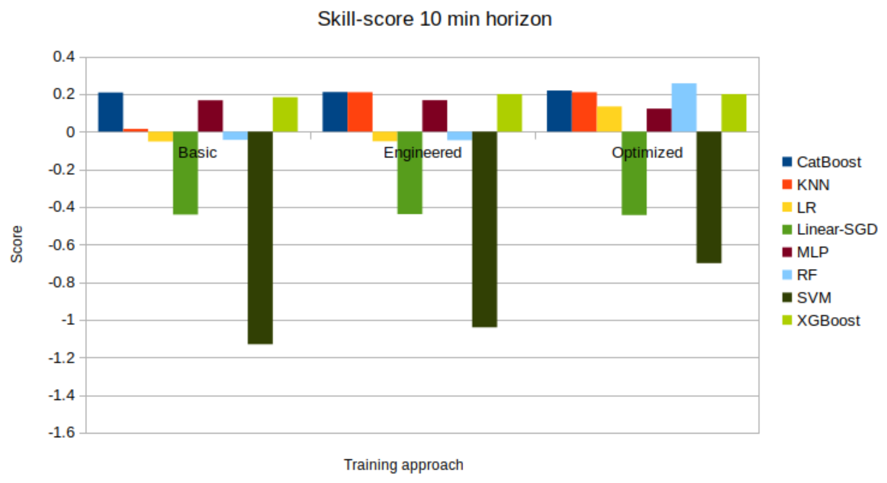


Figure 6.5.: Model Skill-score for each training approach to 10 minutes horizon

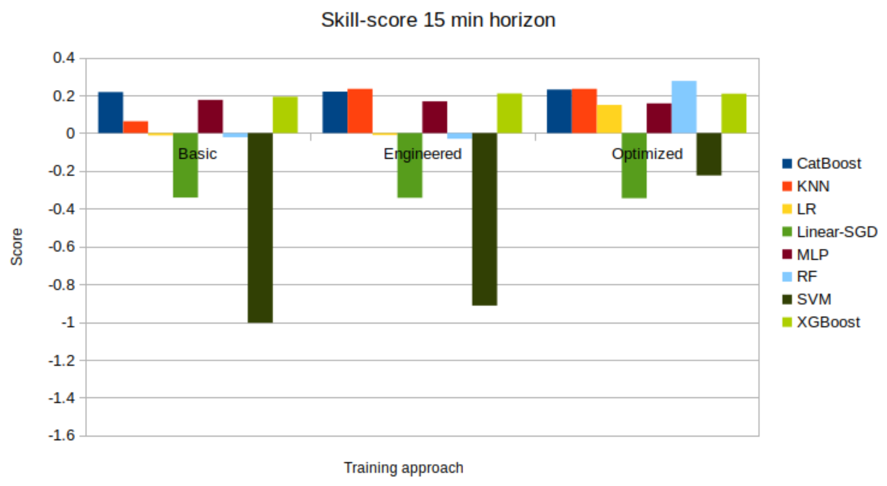


Figure 6.6.: Model Skill-score for each training approach to 15 minutes horizon

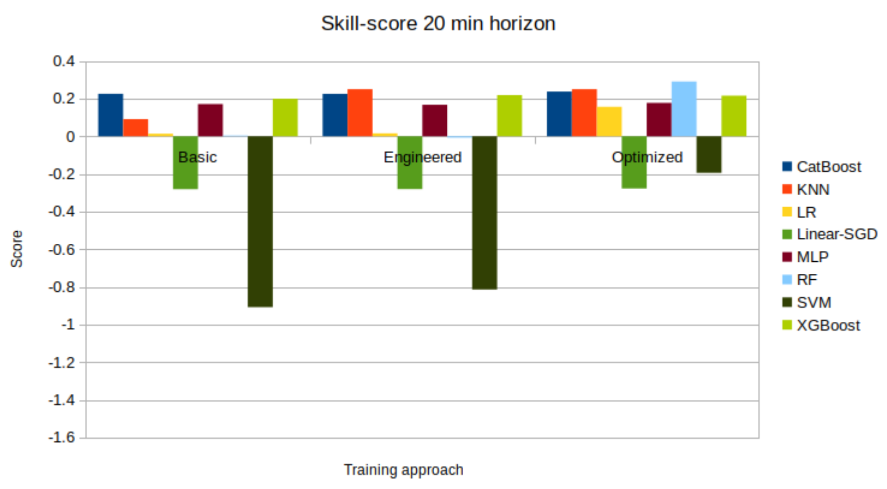


Figure 6.7.: Model Skill-score for each training approach to 20 minutes horizon

6.3. Probabilistic Predictions

After an overview of the individual models results is presented, the ensembles generated by the dynamic selection, see section 5.3.2, are evaluated. To achieve a more representative comparison, this evaluation is based on an analog ensemble of 20 members and a prediction ensemble with 10 best performing members to be chosen from a total of 24 available models. To underline the results of the individual models, a histogram of most selected models as the three best for a 5 minute horizon is appended to the document, see appendix D.

According tho the probabilistic results, developed by the three reference ensembles and the NGB-approach, are presented within this section. Therefore in table 6.1 the results for ICP, MIW and averaged CRPS are presented to an 5 minute forecast horizon. Hereby the interval created by the analog ensemble shows a high ICP combined with a wide MIW what implicates a good average similarity of most ensemble members with also a remarkable number of outliers.

Compared to the analog ensemble, the prediction ensemble of the best performing ML-models, proofs a higher prediction precision, by a lower MAE, but suffers on a lower coverage probability as the interval width is narrow. Better results than analog ensemble and prediction ensemble are delivered by both accumulated to an combined approach, as well as the NGB-method. The coverage probability hereby is over 86% by a mean width smaller than 108 $\frac{W}{m^2}$.

Table 6.1.: Interval metrics with t-distribution for 5 min horizon

Method	ICP	MIW	ACRPS	MAE
Interval 5% - 95%				
Analog ensemble	0.95	139.90	21.74	42.87
ML ensemble	0.60	30.81	26.84	31.19
Combined ensemble	0.94	113.90	21.40	39.19
NGB distribution	0.86	108.00	19.02	25.19

To provide additional information on the interval hit and miss behavior, all interval-hits, interval-misses and interval-widths are distributed by a daily interval between 2 AM and 10 PM. So figure 6.8 to figure 6.11 show these results for all interval generation approaches. To gather a better understanding, at which time of the day a interval is likely to give right or wrong estimations, the distribution is normalized by the amount of hits and misses. Therefore in the left histogram for the analog ensemble with 5 minutes forecast horizon in figure 6.8 the 5% of wrong predicted intervals is distributed mostly within the time of 8:30 AM, 3:30PM, 5:30PM and 7:30PM.

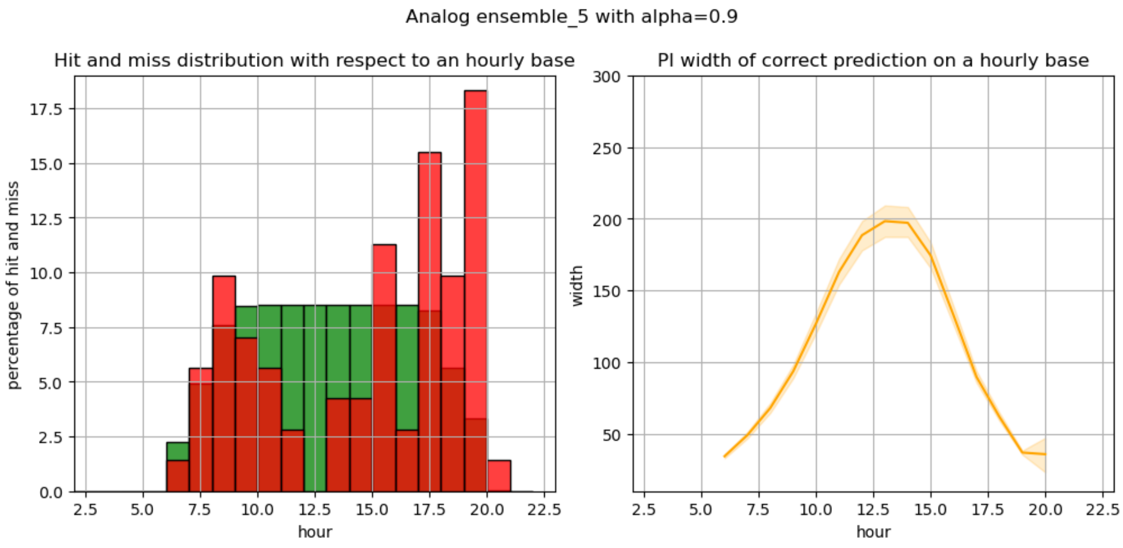


Figure 6.8.: GHI-interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

Considering the distribution of interval width on the right hand side of graph 6.9, the high hit rate of all intervals approaches around noon is reasoned by the highest interval width at this time. Hence the low coverage probability of the ML-ensemble is by considering the low interval width compensated.

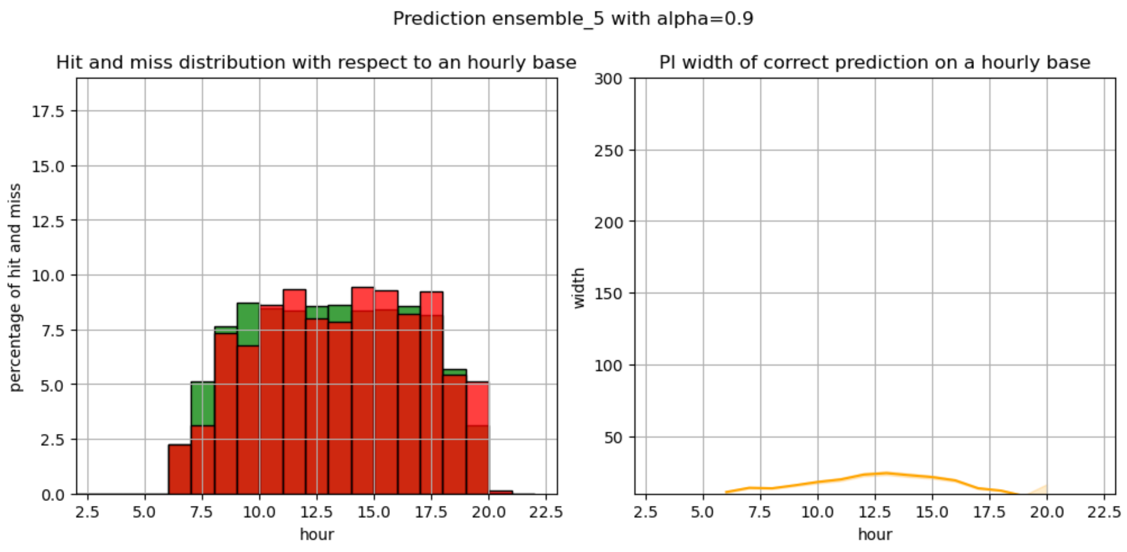


Figure 6.9.: GHI-interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

The third interval estimation approach, which combines the results of historic similarities and best model predictions shows in figure 6.10 the advantage in comparison to the source ensembles. Thus for the distribution of target hitting or missing the interval, the high miss-rates in the late afternoon of the similarities is compensated by the sharp ensemble of the model predictions. A similar effect can be observed for the interval width. There, the overall width on the whole day is smaller, as well as its spread close to noon.

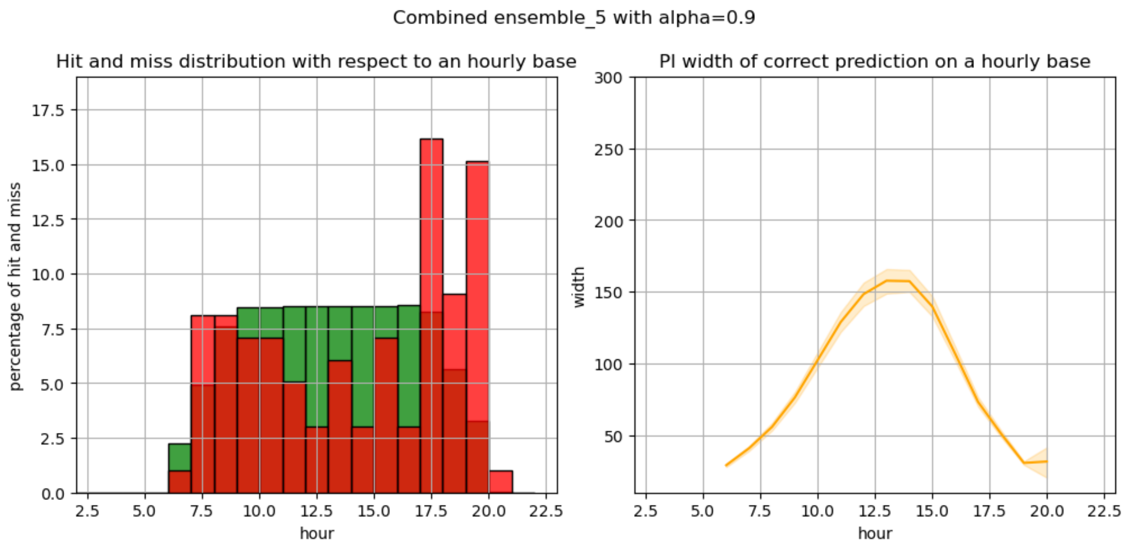


Figure 6.10.: GHI-interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

Comparing the hit and miss distribution of the combined ensemble interval to the NGB interval distribution in figure 6.11, a similar behavior of the missed predictions across the day is to be mentioned. Here both figures show a decreasing number of missed predictions about 2:30PM, where the combined distribution suffers from the high occurrences of the analog ensemble.

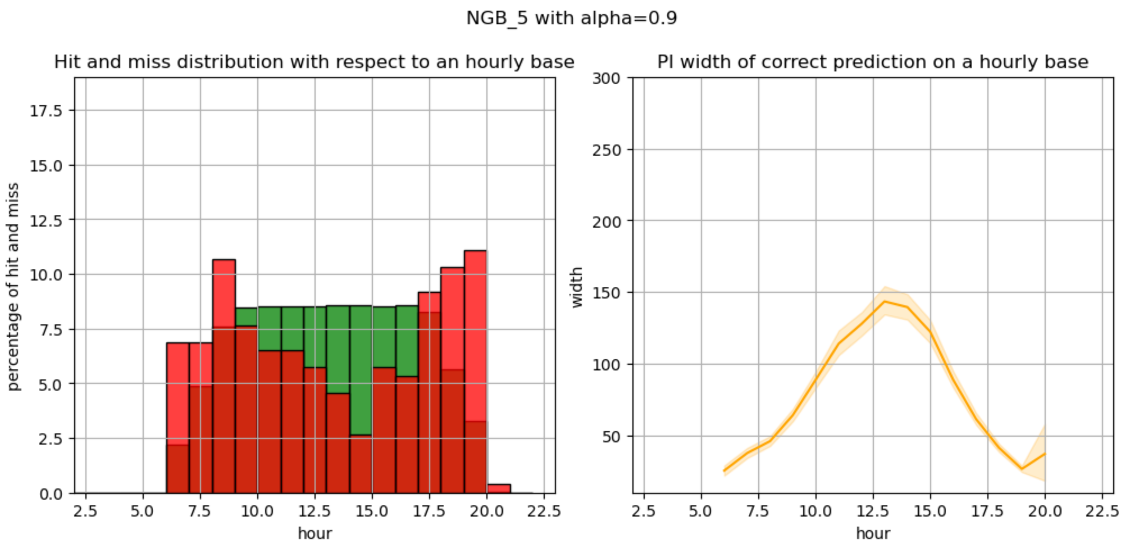


Figure 6.11.: GHI-interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

All interval distribution graphs for a forecast horizon of 10, 15, and 20 minutes are listed within appendix E.

Hence the analog ensemble covers the most target values by the created interval, with the largest MIW. In comparison, the interval calculated by the ML ensemble shows the least coverage by the smallest MIW.
Consequently by using all results in a combined ensemble, a better result for the averaged CRPS can be reached. This combination profits of the more accurate prediction from the model predictions, as well as the higher variability of the analog ensemble.
Since the performance of a single forecast horizon is analyzed, the behavior of the approaches on higher horizons is displayed in figure 6.12. Hereby all metrics are shown, where numeric values are available in appendix F.

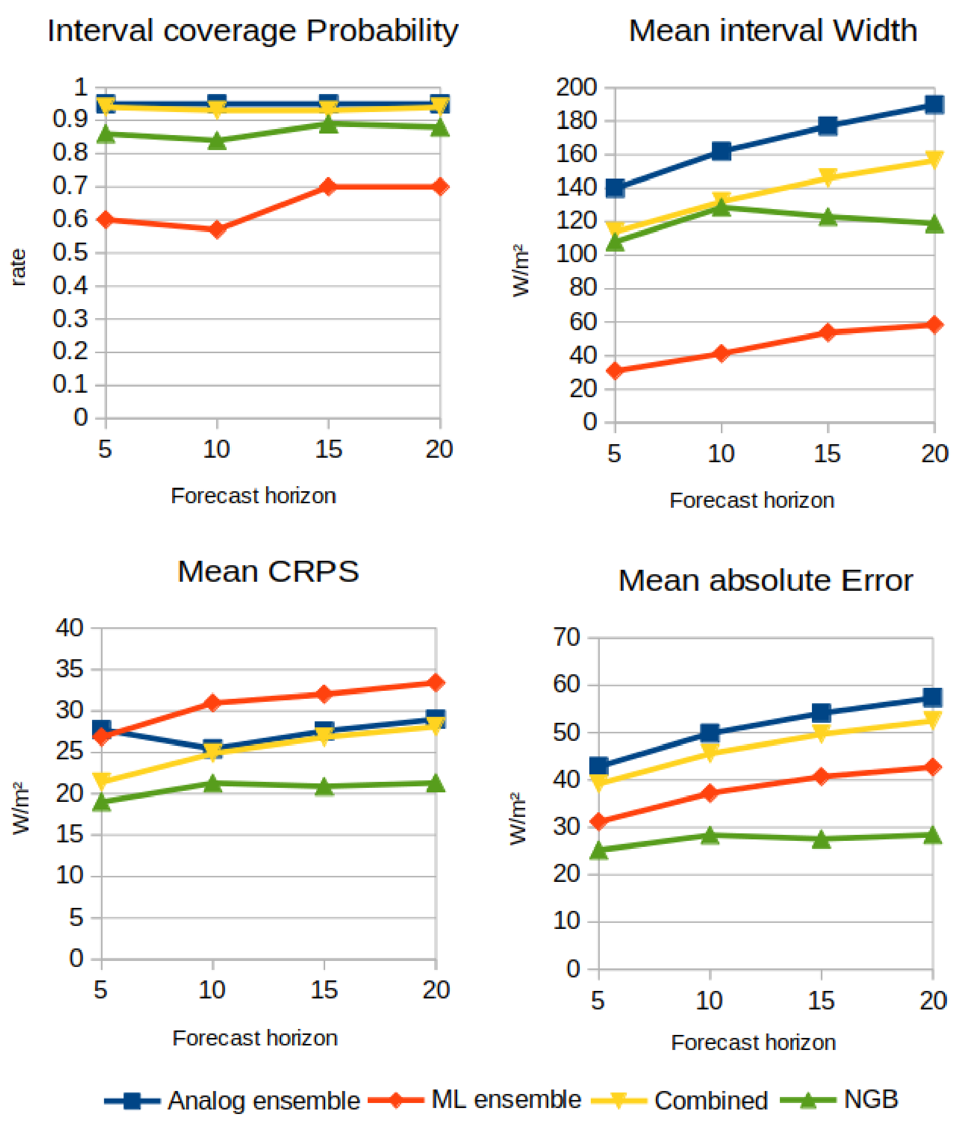


Figure 6.12.: Probabilistic result metrics by forecast horizon on all approaches

So the metric behavior of the combined ensemble show the expected decrease of performance by higher forecast horizons. Accordingly, the width of correct interval predictions raises by a stagnating coverage probability as higher prediction steps are more difficult to predict. The same behavior appears within the MAE.

It is worth mentioning that the increased coverage probability of the ML ensemble, where only a small rise at the interval width can be seen.

In a benchmark, the composition method with additional NGB model, the green colored data provide evidence that within the interval of 5 to 10 minutes the metrics are quite comparable to the combined ensemble. However by a forecast horizon of 15 and 20 minutes, the coverage probability raises accompanied by a decreasing interval width, what is a strong indicator for rising prediction quality. Consequently this finding is confirmed by a stagnating CRPS and MAE, where the other approaches show aggravate metric values.

6.4. Process Performance

After all modules of the proposed approach are classified regarding the selected metrics, the average time consumption for each predictions is evaluated.

For intra-hour forecast, a model should perform within a second. By that time resolution, a prediction can be corrected on every time-step and therefore new measured data. The performance evaluation is performed on a laptop computer with properties as shown in table 6.2.

Table 6.2.: Hardware setup of the personal computer used for prediction time measurement

Component	**Details**
Model	HP Z-Book Fury
Harddrive	1.5 Terabyte(TB) Solid state drive
RAM card	31.1 gibibyte (GiB)
Graphic card	Mesa Intel HD Graphics 4600
Central processor unit	Intel Core i7-11850H @ 2.50 Gigahertz x 16

Since the time occupation for an prediction iteration depends on the selected individual models, a high spread to the mean of 27.5 seconds can be observed in figure 6.13.

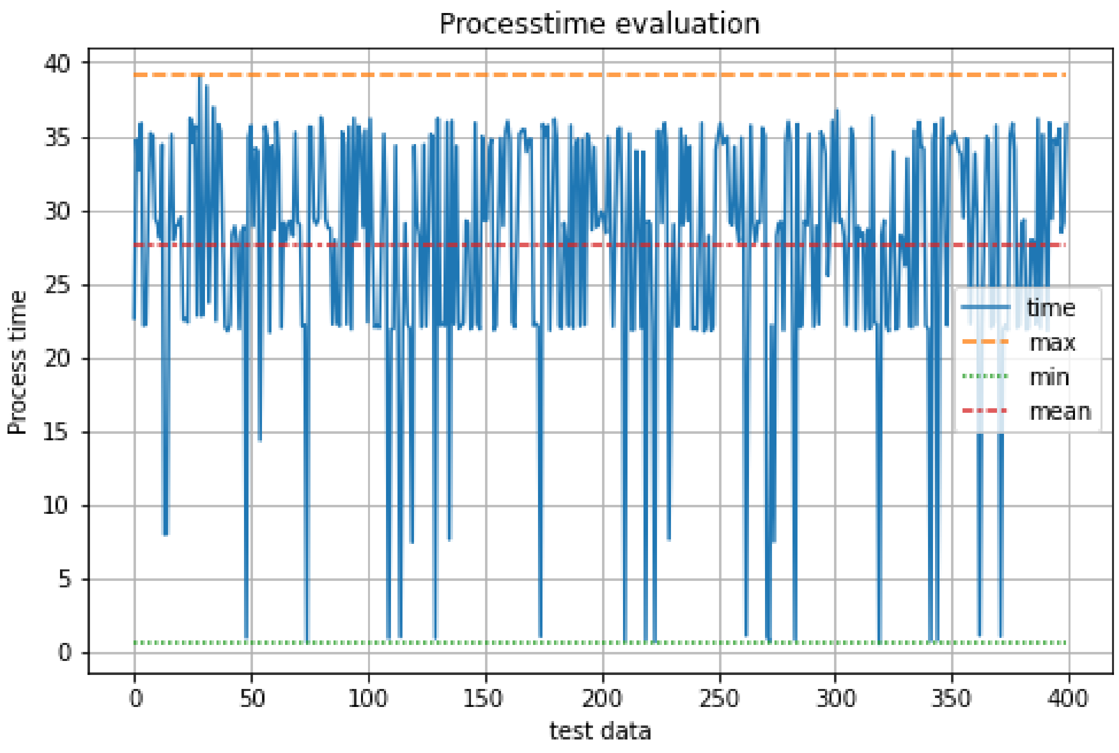


Figure 6.13.: Process time evaluation on 400 prediction iterations

However even by that spread the maximal occupied processing time reaches 39.1 seconds. This value lays still within the upper limit for processing time.

6.5. Evaluation Result

The final section of this chapter is the conclusion of all benchmarked modules in context to the goals of section 1.2. So all specific ML models except the linear-SGD and the SVM reach at least in one training approach a better result on all forecast horizons, than the reference SSP model. Benchmarking the probabilistic NGB approach to the combined ensemble as reference, within a forecast of 5 and 10 minutes only a minor superiority of the new approach is reached. But on forecast horizons for 15 and 20 minutes, the NGB prediction clearly outperforms the combined ensemble, as well as its components.
To give a résumé, all specific parts are listed with dedicated results as follows:

As fist goal, the maximal process time for one full prediction is set to one minute, so by incoming measurements with an one minute interval, results are available before the next measurement. Hence, the maximal needed processing time within a sample of 400 data points was at 39.1 seconds with a mean prediction time of 27.5 seconds, the process hits that goal with the used computational resources.

The second goal is a better RMSE for each individual model compared to the SSP prediction
Since six of eight models with optimized parameters create superior results on all forecast horizons with respect to the SSP result, the goal is partly reached.

Closing the evaluation chapter with the final goal on the method:
Consequently the new NGB prediction approach is challenged to outperform all input ensembles, as well as their combined version by the metrics of CRPS and MAE. Hereby for all evaluated forecast horizons the NGB approach reaches superior results than its input components. The performance is even increasing by forecast horizon. Hence the goal is fully reached.

7. Conclusion and future prospects

7.1. Project Résumé

This chapter sums up all completed work and also discusses the achieved results of the various modules with respect to the set goals within section 1.2. Additionally a final conclusion is presented along with some future prospects on further improvement of the method.

Hence this thesis presents an approach of different ensemble model combinations to create probabilistic results by feeding the resulting ensemble to a NGB model. The goal of this setup is to create reliable probabilistic predictions to forecast future irradiance income by an interval of most likely occurrence.
Considering recent publications on that topic, there are well performing ML models for intra hour prediction existing. Additionally it is discussed, that by providing prediction data of several models to a superordinate model, the overall result can be enhanced.
This finding is within the project included to an probabilistic perdition approach, the NGB. Consequently this approach is chosen to estimate a probabilistic prediction by a superposition of individual ML models.

These probabilistic predictions are supposed to help estimation of future power outcome of PV plants to ease the power-network stabilization. Hence the method should produce better results than the reference model and perform a whole prediction cycle within one minute.

As mentioned, the selected ensemble model consists of two separate parts, introduced in section 5.3.2. The first part hereby is accumulated by irradiance values of historic occurrences with similar environmental and irradiance data. As of these 20 historic measurements, the future irradiance is known, those values are used within the ensemble.

The second part is based on ML models, trained on a dataset of the same geographical location as the data to predict to. On these models a subset is created by 10 individuals which produce best results on the historic similar environmental conditions of the ensembles first part. By the three different training settings, most tested ML models create by at least one training approach better predictions than the reference method for every forecast horizon. Except the linear-SGD and the SVM show significant worse results.

For each trained model predictions for the historic lookup dataset are generated, so depending on the chosen similarities it is known which models perform best. With this information, these model perform predictions for a given data point. This accumulated data serves as input features to the NGB model, which is capable of estimating parametric distributions, in this project a normal distribution.

While benchmarking this new probabilistic method to the two ensemble parts, which it consists of, a superiority to both individuals is found. Hereby the NGB prediction profits from information of the historic analogs as reference to the possible outcomes on a given input situation. Additionally the sharp predictions of the individual ML models provide information to the NGB of the most likely deterministic irradiance according to the forecast horizon.

7.2. Result Conclusion

Considering all results of this project a method is developed which combines two ensemble approaches to estimate a more precise outcome with respect to a resulting confidence interval width and hit rate of the target. Comparing the accumulated ensemble to the additional NGB prediction, the improvement of the ensemble by the NGB is mostly by a better CRPS and MAE.

7.3. Future Prospects

As this project is limited by a certain time and computational resources, only a proof of concept is made. By further tweaking of the ML parameters and evaluating more different models, better results might be reached, as well as for the NGB regressor.
Also for the analog ensemble, a survey on the chosen similarity search features and their weights would be a good approach on better ensemble generation. As the superposition ensemble of the two approaches is also weighted equally, this can also create a point of improvement, to have the results weighted according to their similarity score or model performance.
Mentioning these results, the number of 20 similarities and 10 prediction results were selected to provide sufficient data to the NGB algorithm. These numbers might also inherit potential of improvement by different values or with additional features.

Bibliography

[1] Hervé Abdi et al. The method of least squares. Encyclopedia of measurement and statistics, 1:530–532, 2007.

[2] Stefano Alessandrini, L Delle Monache, S Sperati, and G Cervone. An analog ensemble for short-term probabilistic solar power forecast. Applied energy, 157:95–110, 2015.

[3] Shun-Ichi Amari and Scott C Douglas. Why natural gradient? In Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181), volume 2, pages 1213–1216. IEEE, 1998.

[4] Niklas Benedikt Blum, Stefan Wilbert, Bijan Nouri, Jonas Stührenberg, Jorge Enrique Lezaca Galeano, Thomas Schmidt, Detlev Heinemann, Thomas Vogt, Andreas Kazantzidis, and Robert Pitz-Paal. Analyzing spatial variations of cloud attenuation by a network of all-sky imagers. Remote Sensing, 14(22):5685, 2022.

[5] Andreas Brader. Cloud-stalking 2.0.

[6] Jason Brownlee. Statistical methods for machine learning: Discover how to transform data into knowledge with Python. Machine Learning Mastery, 2018.

[7] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pages 108–122, 2013.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794, 2016.

[9] Carol Lynn Curchoe. All models are wrong, but some are useful. Journal of Assisted Reproduction and Genetics, 37(10):2389–2391, 2020.

[10] Utpal Kumar Das, Kok Soon Tey, Mehdi Seyedmahmoudian, Saad Mekhilef, Moh Yamani Idna Idris, Willem Van Deventer, Bend Horan, and Alex Stojcevski.

Forecasting of photovoltaic power generation and model optimization: A review. Renewable and Sustainable Energy Reviews, 81:912–928, 2018.

[11] Domingos S de O. Santos Jr, Paulo SG de Mattos Neto, João FL de Oliveira, Hugo Valadares Siqueira, Tathiana Mikamura Barchi, Aranildo R Lima, Francisco Madeiro, Douglas AP Dantas, Attilio Converti, Alex C Pereira, et al. Solar irradiance forecasting using dynamic ensemble selection. Applied Sciences, 12(7):3510, 2022.

[12] MJ Mariska de Wild-Scholten. Energy payback time and carbon footprint of commercial photovoltaic systems. Solar Energy Materials and Solar Cells, 119:296–305, 2013.

[13] Tony Duan, Avati Anand, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Ng, and Alejandro Schuler. Ngboost: Natural gradient boosting for probabilistic prediction. In International Conference on Machine Learning, pages 2690–2700. PMLR, 2020.

[14] Energy-Charts. Solar-anteil an stromerzeugung. https://www.energy-charts.info/charts/renewable_share/chart.htm?l=en&c=DE&share=solar_share_public&interval=year [abgerufen 07.08.2022].

[15] Energy-Charts. Solar-anteil installiert. https://www.energy-charts.info/charts/installed_power/chart.htm?l=en&c=DE&stacking=single&chartColumnSorting=default&year=-1&legendItems=00000000000001&download-format=image%2Fjpeg [abgerufen 07.08.2022].

[16] Johannes Ernesti and Peter Kaiser. Python 3. Rheinwerk, Bonn, 2017.

[17] Christopher AT Ferro, David S Richardson, and Andreas P Weigel. On the effect of ensemble size on the discrete and continuous ranked probability scores. Meteorological Applications: A journal of forecasting, practical applications, training techniques and modelling, 15(1):19–24, 2008.

[18] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the hdf5 technology suite and its applications. In Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, pages 36–47, 2011.

[19] A Geron. Hands-on machine learning with scikit-learn & tensorflow o'reilly media, inc. O'Reilly Media, Inc, 1005:564, 2017.

[20] Tao Hong, Pierre Pinson, and Shu Fan. Global energy forecasting competition 2012, 2014.

[21] Javier Huertas Tato and Miguel Centeno Brito. Using smart persistence and random forests to predict photovoltaic energy production. Energies, 12(1):100, 2018.

[22] Pierre Ineichen and Richard Perez. A new airmass independent formulation for the linke turbidity coefficient. Solar Energy, 73(3):151–157, 2002.

[23] Alexander Jung. Machine Learning: The Basics. Springer Nature, 2022.

[24] Marek Kejna, Joanna Uscka-Kowalkowska, and Paweł Kejna. The influence of cloudiness and atmospheric circulation on radiation balance and its components. Theoretical and Applied Climatology, 144(3):823–838, 2021.

[25] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. Comprehensive review of neural network-based prediction intervals and new advances. IEEE Transactions on neural networks, 22(9):1341–1356, 2011.

[26] kintech engineering. Pyranometer. https://www.kintech-engineering.com/pt-br/catalogue/solar-pt/sr20/ [abgerufen 04.07.2021].

[27] Roger Koenker, Victor Chernozhukov, Xuming He, and Limin Peng. Handbook of quantile regression. 2017.

[28] Andrew Kumler, Yu Xie, and Yingchen Zhang. A physics-based smart persistence model for intra-hour forecasting of solar radiation (pspi) using ghi measurements and a cloud retrieval technique. Solar Energy, 177:494–500, 2019.

[29] Francisco JL Lima, Fernando R Martins, Enio B Pereira, Elke Lorenz, and Detlev Heinemann. Forecast for surface solar irradiance at the brazilian northeastern region using nwp model and artificial neural networks. Renewable Energy, 87:807–818, 2016.

[30] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. Advances in neural information processing systems, 30, 2017.

[31] Peter Lynch. The origins of computer weather prediction and climate modeling. Journal of computational physics, 227(7):3431–3444, 2008.

[32] Dávid Markovics and Martin János Mayer. Comparison of machine learning methods for photovoltaic power forecasting based on numerical weather prediction. Renewable and Sustainable Energy Reviews, 161:112364, 2022.

[33] Georgios Mitrentsis and Hendrik Lens. An interpretable probabilistic model for short-term solar power forecasting using natural gradient boosting. Applied Energy, 309:118473, 2022.

[34] Azhar Ahmed Mohammed, Waheeb Yaqub, and Zeyar Aung. Probabilistic forecasting of solar power: An ensemble learning approach. In International Conference on Intelligent Decision Technologies, pages 449–458. Springer, 2017.

[35] Arash Moradzadeh, Amin Mansour Saatloo, Behnam Mohammadi-ivatloo, and Amjad Anvari-Moghaddam. Performance evaluation of two machine learning techniques in heating and cooling loads forecasting of residential buildings. Applied Sciences, 10, 05 2020.

[36] Qiang Ni, Shengxian Zhuang, Hanming Sheng, Gaoqiang Kang, and Jian Xiao. An ensemble prediction intervals approach for short-term pv power forecasting. Solar Energy, 155:1072–1083, 2017.

[37] J Casa Nova, José Boaventura Cunha, and PB de Moura Oliveira. Solar irradiation forecast model using time series analysis and sky images. In Proceedings of the 5th Conference of the European Federation for Information Technology in Agriculture, Food and Environment, pages 1408–1415, 2005.

[38] GW Paltridge and D Proctor. Monthly mean solar radiation statistics for australia. Solar Energy, 18(3):235–243, 1976.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, 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. Journal of Machine Learning Research, 12:2825–2830, 2011.

[40] Hugo TC Pedro, David P Larson, and Carlos FM Coimbra. A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods. Journal of Renewable and Sustainable Energy, 11(3):036102, 2019.

[41] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. Advances in neural information processing systems, 31, 2018.

[42] Jeff Reback, Wes McKinney, Joris Van Den Bossche, Tom Augspurger, Phillip Cloud, Adam Klein, Simon Hawkins, Matthew Roeschke, Jeff Tratner, Chang She, et al. pandas-dev/pandas: Pandas 1.0. 5. Zenodo, 2020.

[43] D Reinert, F Prill, H Frank, M Denhard, M Baldauf, C Schraff, C Gebhardt, C Marsigli, and G Zängl. Dwd database reference for the global and regional icon and icon-eps forecasting system. Technical report, Technical Report Version 2.1. 1, 2020.

[44] David E Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation: The basic theory. Backpropagation: Theory, architectures and applications, pages 1–34, 1995.

[45] Stefan Saatmann and Sandra Maeding. Energiewende und regulierung–wie werden sonne und wind im stromnetz integriert und reguliert? Leipzig, Germany, 23-24 September 2013, page 13, 2013.

[46] Manajit Sengupta, Aron Habte, Stefan Wilbert, Christian Gueymard, and Jan Remund. Best practices handbook for the collection and use of solar resource data for solar energy applications. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2021.

[47] Durga L Shrestha and Dimitri P Solomatine. Machine learning approaches for estimation of prediction interval for the model output. Neural networks, 19(2):225–235, 2006.

[48] Joshua S Stein, Clifford W Hansen, and Matthew J Reno. Global horizontal irradiance clear sky models: implementation and analysis. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA . . . , 2012.

[49] Joshua S Stein, William F Holmgren, Jessica Forbess, and Clifford W Hansen. Pvlib: Open source photovoltaic performance modeling functions for matlab and python. In 2016 ieee 43rd photovoltaic specialists conference (pvsc), pages 3425–3430. IEEE, 2016.

[50] Brandon Stafford *pingswept*. Pysolar https://github.com/pingswept/pysolar. https://pysolar.readthedocs.io/en/latest/, Version 0.9 - Commit 2d4b89c.

[51] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. Nature methods, 17(3):261–272, 2020.

[52] Stephen G Warren, Ryan M Eastman, and Carole J Hahn. A survey of changes in cloud cover and cloud types over land from surface observations, 1971–96. Journal of climate, 20(4):717–738, 2007.

[53] Stefan Wilbert. psa-hp-dataset, 12 2019.

[54] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for machine learning, volume 2. MIT press Cambridge, MA, 2006.

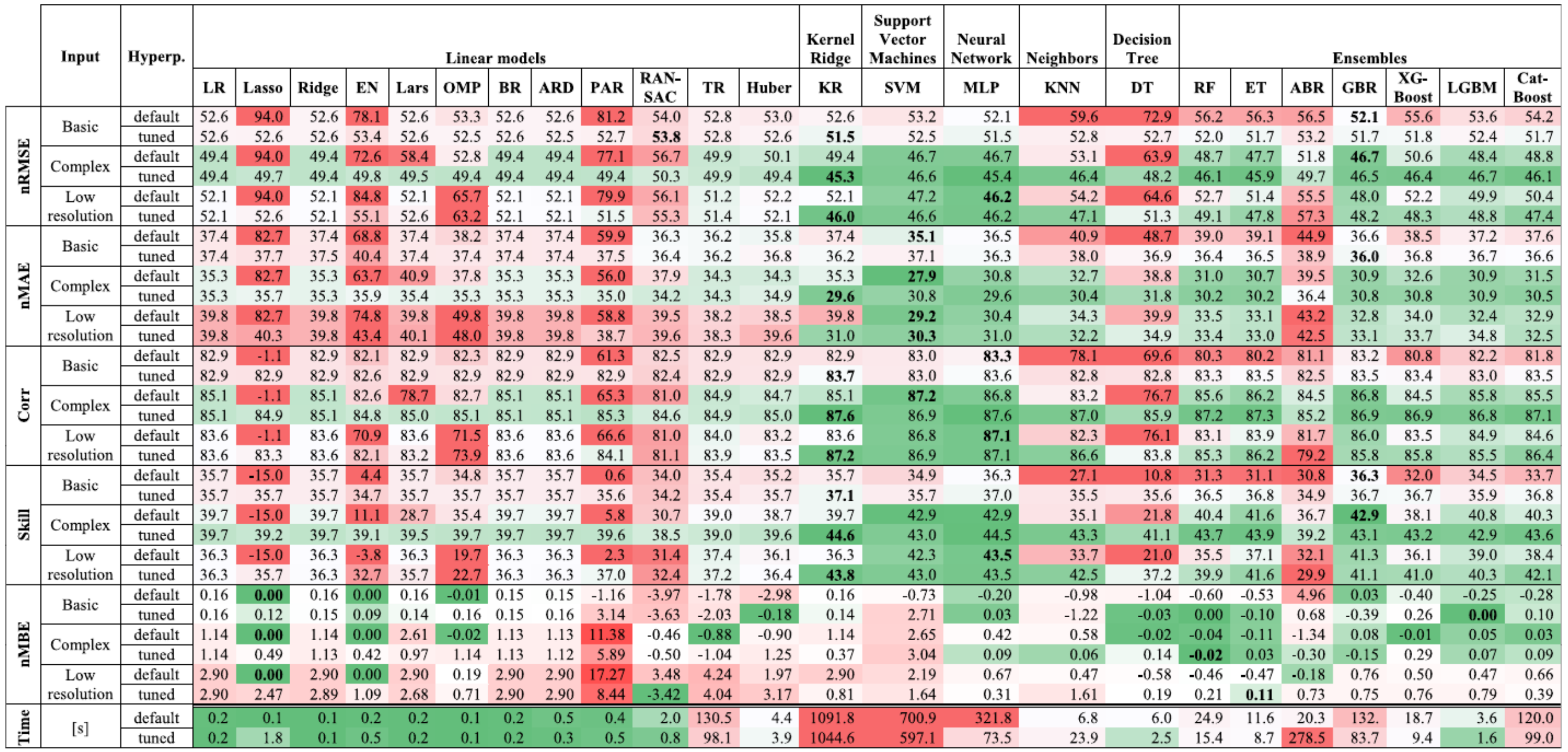
[55] Hiroki Yamamoto, Junji Kondoh, and Daisuke Kodaira. Assessing the impact of features on probabilistic forecasting of photovoltaic power generation. 2022.

[56] Dazhi Yang. A guideline to solar forecasting research practice: Reproducible, operational, probabilistic or physically-based, ensemble, and skill (ropes). Journal of Renewable and Sustainable Energy, 11(2):022701, 2019.

[57] Dazhi Yang. A universal benchmarking method for probabilistic solar irradiance forecasting. Solar Energy, 184:410–416, 2019.

Appendix

A. Results of Markovics Model Comparison



	Input	Hyperp.	Linear models												Kernel Ridge	Support Vector Machines	Neural Network	Neighbors	Decision Tree	Ensembles						
			LR	Lasso	Ridge	EN	Lars	OMP	BR	ARD	PAR	RAN-SAC	TR	Huber	KR	SVM	MLP	KNN	DT	RF	ET	ABR	GBR	XG-Boost	LGBM	Cat-Boost
nRMSE	Basic	default	52.6	94.0	52.6	78.1	52.6	53.3	52.6	52.6	81.2	54.0	52.8	53.0	52.6	53.2	52.1	59.6	72.9	56.2	56.3	56.5	**52.1**	55.6	53.6	54.2
		tuned	52.6	52.6	52.6	53.4	52.6	52.5	52.6	52.5	52.7	**53.8**	52.8	52.6	**51.5**	52.5	51.5	52.8	52.7	52.0	51.7	53.2	51.7	51.8	52.4	51.7
	Complex	default	49.4	94.0	49.4	72.6	58.4	52.8	49.4	49.4	77.1	56.7	49.9	50.1	49.4	46.7	46.7	53.1	63.9	48.7	47.7	51.8	**46.7**	50.6	48.4	48.8
		tuned	49.4	49.7	49.4	49.8	49.5	49.4	49.4	49.4	49.4	50.3	49.9	49.4	**45.3**	46.6	45.4	46.4	48.2	46.1	45.9	49.7	46.5	46.4	46.7	46.1
	Low resolution	default	52.1	94.0	52.1	84.8	52.1	65.7	52.1	52.1	79.9	56.1	51.2	52.2	52.1	47.2	**46.2**	54.2	64.6	52.7	51.4	55.5	48.0	52.2	49.9	50.4
		tuned	52.1	52.6	52.1	55.1	52.6	63.2	52.1	52.1	51.5	55.3	51.4	52.1	**46.0**	46.6	46.2	47.1	51.3	49.1	47.8	57.3	48.2	48.3	48.8	47.4
nMAE	Basic	default	37.4	82.7	37.4	68.8	37.4	38.2	37.4	37.4	59.9	36.3	36.2	35.8	37.4	**35.1**	36.5	40.9	48.7	39.0	39.1	44.9	36.6	38.5	37.2	37.6
		tuned	37.4	37.7	37.5	40.4	37.4	37.4	37.4	37.4	37.5	36.4	36.2	36.8	36.2	37.1	36.3	38.0	36.9	36.4	36.5	38.9	**36.0**	36.8	36.7	36.6
	Complex	default	35.3	82.7	35.3	63.7	40.9	37.8	35.3	35.3	56.0	37.9	34.3	34.3	35.3	**27.9**	30.8	32.7	38.8	31.0	30.7	39.5	30.9	32.6	30.9	31.5
		tuned	35.3	35.7	35.3	35.9	35.4	35.3	35.3	35.3	35.0	34.2	34.3	34.9	**29.6**	30.8	29.6	30.4	31.8	30.2	30.2	36.4	30.8	30.8	30.9	30.5
	Low resolution	default	39.8	82.7	39.8	74.8	39.8	49.8	39.8	39.8	58.8	39.5	38.2	38.5	39.8	**29.2**	30.4	34.3	39.9	33.5	33.1	43.2	32.8	34.0	32.4	32.9
		tuned	39.8	40.3	39.8	43.4	40.1	48.0	39.8	39.8	38.7	39.6	38.3	39.6	31.0	**30.3**	31.0	32.2	34.9	33.4	33.0	42.5	33.1	33.7	34.8	32.5
Corr	Basic	default	82.9	-1.1	82.9	82.1	82.9	82.3	82.9	82.9	61.3	82.5	82.9	82.9	82.9	83.0	**83.3**	78.1	69.6	80.3	80.2	81.1	83.2	80.8	82.2	81.8
		tuned	82.9	82.9	82.9	82.6	82.9	82.9	82.9	82.9	82.9	82.4	82.9	82.9	**83.7**	83.0	83.6	82.8	82.8	83.3	83.5	82.5	83.5	83.4	83.0	83.5
	Complex	default	85.1	-1.1	85.1	82.6	78.7	82.7	85.1	85.1	65.3	81.0	84.9	84.7	85.1	**87.2**	86.8	83.2	76.7	85.6	86.2	84.5	86.8	84.5	85.8	85.5
		tuned	85.1	84.9	85.1	84.8	85.0	85.1	85.1	85.1	85.3	84.6	84.9	85.0	**87.6**	86.9	87.6	87.0	85.9	87.2	87.3	85.2	86.9	86.9	86.8	87.1
	Low resolution	default	83.6	-1.1	83.6	70.9	83.6	71.5	83.6	83.6	66.6	81.0	84.0	83.2	83.6	86.8	**87.1**	82.3	76.1	83.1	83.9	81.7	86.0	83.5	84.9	84.6
		tuned	83.6	83.3	83.6	82.1	83.2	73.9	83.6	83.6	84.1	81.1	83.9	83.5	**87.2**	86.9	87.1	86.6	83.8	85.3	86.2	79.2	85.8	85.8	85.5	86.4
Skill	Basic	default	35.7	-15.0	35.7	4.4	35.7	34.8	35.7	35.7	0.6	34.0	35.4	35.2	35.7	34.9	36.3	27.1	10.8	31.3	31.1	30.8	**36.3**	32.0	34.5	33.7
		tuned	35.7	35.7	35.7	34.7	35.7	35.7	35.7	35.7	35.6	34.2	35.4	35.7	**37.1**	35.7	37.0	35.5	35.6	36.5	36.8	34.9	36.7	36.7	35.9	36.8
	Complex	default	39.7	-15.0	39.7	11.1	28.7	35.4	39.7	39.7	5.8	30.7	39.0	38.7	39.7	42.9	42.9	35.1	21.8	40.4	41.6	36.7	**42.9**	38.1	40.8	40.3
		tuned	39.7	39.2	39.7	39.1	39.5	39.7	39.7	39.7	39.6	38.5	39.0	39.6	**44.6**	43.0	44.5	43.3	41.1	43.7	43.9	39.2	43.1	43.2	42.9	43.6
	Low resolution	default	36.3	-15.0	36.3	-3.8	36.3	19.7	36.3	36.3	2.3	31.4	37.4	36.1	36.3	42.3	**43.5**	33.7	21.0	35.5	37.1	32.1	41.3	36.1	39.0	38.4
		tuned	36.3	35.7	36.3	32.7	35.7	22.7	36.3	36.3	37.0	32.4	37.2	36.4	**43.8**	43.0	43.5	42.5	37.2	39.9	41.6	29.9	41.1	41.0	40.3	42.1
nMBE	Basic	default	0.16	**0.00**	0.16	0.00	0.16	-0.01	0.15	0.15	-1.16	-3.97	-1.78	-2.98	0.16	-0.73	-0.20	-0.98	-1.04	-0.60	-0.53	4.96	0.03	-0.40	-0.25	-0.28
		tuned	0.16	0.12	0.15	0.09	0.14	0.16	0.15	0.16	3.14	-3.63	-2.03	-0.18	0.14	2.71	0.03	-1.22	-0.03	0.00	-0.10	0.68	-0.39	0.26	**0.00**	0.10
	Complex	default	1.14	**0.00**	1.14	0.00	2.61	-0.02	1.13	1.13	11.38	-0.46	-0.88	-0.90	1.14	2.65	0.42	0.58	-0.02	-0.04	-0.11	-1.34	0.08	-0.01	0.05	0.03
		tuned	1.14	0.49	1.13	0.42	0.97	1.14	1.13	1.12	5.89	-0.50	-1.04	1.25	0.37	3.04	0.09	0.06	0.14	**-0.02**	0.03	-0.30	-0.15	0.29	0.07	0.09
	Low resolution	default	2.90	**0.00**	2.90	0.00	2.90	0.19	2.90	2.90	17.27	3.48	4.24	1.97	2.90	2.19	0.67	0.47	-0.58	-0.46	-0.47	-0.18	0.76	0.50	0.47	0.66
		tuned	2.90	2.47	2.89	1.09	2.68	0.71	2.90	2.90	8.44	-3.42	4.04	3.17	0.81	1.64	0.31	1.61	0.19	0.21	**0.11**	0.73	0.75	0.76	0.79	0.39
Time	[s]	default	0.2	0.1	0.1	0.2	0.2	0.1	0.2	0.5	0.4	2.0	130.5	4.4	1091.8	700.9	321.8	6.8	6.0	24.9	11.6	20.3	132.	18.7	3.6	120.0
		tuned	0.2	1.8	0.1	0.5	0.2	0.1	0.2	0.3	0.5	0.8	98.1	3.9	1044.6	597.1	73.5	23.9	2.5	15.4	8.7	278.5	83.7	9.4	1.6	99.0

Figure 1.: Metric results of all test-cases and ML models [32]

B. Model Parameters

Optimized model parameters with respect to forecast horizon
optimized_models_5

- random_forest 'bootstrap': False, 'ccp_alpha': 0.0, 'criterion': 'squared_error', 'max_depth': None, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 300, 'n_jobs': 4, 'oob_score': False, 'random_state': None, 'verbose': False, 'warm_start': False
- xgboost 'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'colsample_bylevel': 1, 'colsample_bynode': 1, 'colsample_bytree': 0.3, 'enable_categorical': False, 'gamma': 0, 'gpu_id': -1, 'importance_type': None, 'interaction_constraints': ", 'learning_rate': 0.3, 'max_delta_step': 0, 'max_depth': 6, 'min_child_weight': 4, 'missing': nan, 'monotone_constraints': '()', 'n_estimators': 200, 'n_jobs': 16, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 10, 'reg_lambda': 0.1, 'scale_pos_weight': 1, 'subsample': 1, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None, 'alpha': 10
- linear_regression 'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': 'deprecated', 'positive': False
- cat_boost 'iterations': 150, 'learning_rate': 0.1, 'l2_leaf_reg': 3, 'loss_function': 'RMSE', 'verbose': False, 'max_depth': 15, 'gpu_cat_features_storage': True
- k-nearest 'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'
- multy-layer-perceptron 'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto', 'beta_1': 0.9, 'beta_2': 0.999, 'early_stopping': False, 'epsilon': 1e-08, 'hidden_layer_sizes': (100,), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_fun': 15000, 'max_iter': 200, 'momentum': 0.9, 'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5, 'random_state': None, 'shuffle': False, 'solver': 'adam', 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': False, 'warm_start': False
- linear_sgd 'alpha': 0.0001, 'average': False, 'early_stopping': False, 'epsilon': 0.1, 'eta0': 0.01, 'fit_intercept': True, 'l1_ratio': 0.15, 'learning_rate': 'invscaling', 'loss': 'squared_error', 'max_iter': 500, 'n_iter_no_change': 5, 'penalty': 'l2', 'power_t': 0.25, 'random_state': None, 'shuffle': True, 'tol': 0.001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False

optimized_models_10

- random_forest 'bootstrap': False, 'ccp_alpha': 0.0, 'criterion': 'squared_error', 'max_depth': None, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 300, 'n_jobs': 4, 'oob_score': False, 'random_state': None, 'verbose': False, 'warm_start': False

- xgboost 'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'colsample_bylevel': 1, 'colsample_bynode': 1, 'colsample_bytree': 0.3, 'enable_categorical': False, 'gamma': 0, 'gpu_id': -1, 'importance_type': None, 'interaction_constraints': '', 'learning_rate': 0.3, 'max_delta_step': 0, 'max_depth': 6, 'min_child_weight': 1, 'missing': nan, 'monotone_constraints': '()', 'n_estimators': 200, 'n_jobs': 16, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 10, 'reg_lambda': 0.4, 'scale_pos_weight': 1, 'subsample': 1, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None, 'alpha': 10

- linear_regression 'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': 'deprecated', 'positive': False

- cat_boost 'iterations': 150, 'learning_rate': 0.1, 'l2_leaf_reg': 3, 'loss_function': 'RMSE', 'verbose': False, 'max_depth': 15, 'gpu_cat_features_storage': True

- k-nearest 'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'

- multy-layer-perceptron 'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto', 'beta_1': 0.9, 'beta_2': 0.999, 'early_stopping': False, 'epsilon': 1e-08, 'hidden_layer_sizes': (100,), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_fun': 15000, 'max_iter': 200, 'momentum': 0.9, 'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5, 'random_state': None, 'shuffle': False, 'solver': 'adam', 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': False, 'warm_start': False

- linear_sgd 'alpha': 0.0001, 'average': False, 'early_stopping': False, 'epsilon': 0.1, 'eta0': 0.01, 'fit_intercept': True, 'l1_ratio': 0.15, 'learning_rate': 'invscaling', 'loss': 'squared_error', 'max_iter': 10000, 'n_iter_no_change': 5, 'penalty': 'l2', 'power_t': 0.25, 'random_state': None, 'shuffle': True, 'tol': 0.001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False

optimized_models_15

- random_forest 'bootstrap': False, 'ccp_alpha': 0.0, 'criterion': 'squared_error', 'max_depth': None, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split':

2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 300, 'n_jobs': 4, 'oob_score': False, 'random_state': None, 'verbose': False, 'warm_start': False

- xgboost 'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'colsample_bylevel': 1, 'colsample_bynode': 1, 'colsample_bytree': 0.3, 'enable_categorical': False, 'gamma': 0, 'gpu_id': -1, 'importance_type': None, 'interaction_constraints': '', 'learning_rate': 0.3, 'max_delta_step': 0, 'max_depth': 6, 'min_child_weight': 3, 'missing': nan, 'monotone_constraints': '()', 'n_estimators': 200, 'n_jobs': 16, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 10, 'reg_lambda': 1, 'scale_pos_weight': 1, 'subsample': 1, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None, 'alpha': 10

- linear_regression 'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': 'deprecated', 'positive': False

- cat_boost 'iterations': 150, 'learning_rate': 0.1, 'l2_leaf_reg': 3, 'loss_function': 'RMSE', 'verbose': False, 'max_depth': 15, 'gpu_cat_features_storage': True

- k-nearest 'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'

- multy-layer-perceptron 'activation': 'relu', 'alpha': 0.001, 'batch_size': 'auto', 'beta_1': 0.9, 'beta_2': 0.999, 'early_stopping': False, 'epsilon': 1e-08, 'hidden_layer_sizes': (10, 10), 'learning_rate': 'constant', 'learning_rate_init': 0.001, 'max_fun': 15000, 'max_iter': 200, 'momentum': 0.9, 'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5, 'random_state': None, 'shuffle': False, 'solver': 'adam', 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': False, 'warm_start': False

- linear_sgd 'alpha': 0.0001, 'average': False, 'early_stopping': False, 'epsilon': 0.1, 'eta0': 0.01, 'fit_intercept': True, 'l1_ratio': 0.15, 'learning_rate': 'invscaling', 'loss': 'squared_error', 'max_iter': 1000, 'n_iter_no_change': 5, 'penalty': 'l2', 'power_t': 0.25, 'random_state': None, 'shuffle': True, 'tol': 0.001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False

optimized_models_20

- random_forest 'bootstrap': False, 'ccp_alpha': 0.0, 'criterion': 'squared_error', 'max_depth': None, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples': None,'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 300, 'n_jobs': 4, 'oob_score': False, 'random_state': None, 'verbose': False, 'warm_start': False

- xgboost 'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'colsample_bylevel': 1, 'colsample_bynode': 1, 'colsample_bytree': 0.3, 'enable_categorical': False, 'gamma': 0, 'gpu_id': -1, 'importance_type': None, 'interaction_constraints': '', 'learning_rate': 0.3, 'max_delta_step': 0, 'max_depth': 6, 'min_child_weight': 4, 'missing': nan, 'monotone_constraints': '()', 'n_estimators': 200, 'n_jobs': 16, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 10, 'reg_lambda': 1, 'scale_pos_weight': 1, 'subsample': 1, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None, 'alpha': 10
- linear_regression 'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': 'deprecated', 'positive': False
- cat_boost 'iterations': 150, 'learning_rate': 0.1, 'l2_leaf_reg': 3, 'loss_function': 'RMSE', 'verbose': False, 'max_depth': 15, 'gpu_cat_features_storage': True
- k-nearest 'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'
- multy-layer-perceptron 'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto', 'beta_1': 0.9, 'beta_2': 0.999, 'early_stopping': False, 'epsilon': 1e-08, 'hidden_layer_sizes': (100,), 'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'max_fun': 15000, 'max_iter': 200, 'momentum': 0.9, 'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5, 'random_state': None, 'shuffle': False, 'solver': 'adam', 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': False, 'warm_start': False
- linear_sgd 'alpha': 0.0001, 'average': False, 'early_stopping': False, 'epsilon': 0.1, 'eta0': 0.01, 'fit_intercept': True, 'l1_ratio': 0.15, 'learning_rate': 'invscaling', 'loss': 'squared_error', 'max_iter': 10000, 'n_iter_no_change': 5, 'penalty': 'l2', 'power_t': 0.25, 'random_state': None, 'shuffle': True, 'tol': 0.001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False

C. Individual Model Results

Table 1.: Results of individual model with a forecast horizon of **5 minutes**

Model-case	MAE	MSE	RMSE	Skill-S
SSP				
-	46.61	8098	89.99	0
random forest				
basic	39.54	9661	98.29	-0.0922
engineered	40.12	9599	97.98	-0.0887
optimized	**29.89**	**4965**	**70.47**	**0.2170**
xgboost				
basic	36.44	5793	76.11	0.1543
engineered	34.89	5609	74.89	0.1678
optimized	34.02	5526	74.34	0.1739
support vector				
basic	190.00	50769	225.32	-1.5038
engineered	182.58	45973	214.41	-1.3826
optimized	129.98	26859	163.89	-0.8211
LR				
basic	66.57	10796	103.90	-0.1546
engineered	66.56	10787	103.86	-0.1541
optimized	42.13	6586	81.16	0.0982
cat boost				
basic	32.39	5368	73.26	0.1859
engineered	32.75	5370	73.28	0.1857
optimized	32.85	5313	72.89	0.1900
KNN				
basic	52.72	9979	99.89	-0.1100
engineered	34.39	5740	75.76	0.1581
optimized	34.39	5740	75.76	0.1581
MLP				
basic	38.11	5994	77.42	0.1397
engineered	36.31	5855	76.52	0.1497
optimized	42.94	6225	78.90	0.1232
linear sgd				
basic	98.30	22732	150.77	-0.6754
engineered	97.54	22761	150.87	-0.6765
optimized	99.62	22631	150.44	-0.6717

Table 2.: Results of individual models with a forecast horizon of **10 minutes**

Model-case	MAE	MSE	RMSE	Skill-S
SSP				
-	54.53	10906	104.43	0
random forest				
basic	45.51	11886	109.02	-0.0439
engineered	46.12	11927	109.21	-0.0458
optimized	**34.52**	**6026**	**77.63**	**0.2567**
xgboost				
basic	42.09	7278	85.31	0.1831
engineered	40.24	6987	83.59	0.1996
optimized	39.83	6989	83.60	0.1995
support vector				
basic	187.72	49516	222.52	-1.1308
engineered	181.49	45423	213.13	-1.0408
optimized	150.05	31505	177.50	-0.6996
Linear Regression				
basic	68.74	12083	109.92	-0.0526
engineered	68.62	12058	109.81	-0.0515
optimized	47.52	8182	90.45	0.1338
cat boost				
basic	38.95	6840	82.71	0.2080
engineered	38.75	6795	82.43	0.2107
optimized	38.51	6659	81.60	0.2186
KNN				
basic	54.67	10590	102.91	0.0146
engineered	38.17	6813	82.54	0.2096
optimized	38.17	6813	82.54	0.2096
MLP				
basic	43.54	7575	87.04	0.1666
engineered	44.91	7567	86.99	0.1670
optimized	50.07	8403	91.67	0.1222
linear sgd				
basic	99.19	22655	150.52	-0.4413
engineered	100.34	22579	150.26	-0.4388
optimized	97.78	22729	150.76	-0.4436

Table 3.: Results of individual models with a forecast horizon of 15 minutes

Model-case	**MAE**	**MSE**	**RMSE**	**Skill-score**
SSP				
-	59.24	12526	111.92	0
random forest				
basic	48.50	13089	114.41	-0.0222
engineered	49.67	13293	115.29	-0.0301
optimized	**37.15**	**6573**	**81.07**	**0.2756**
xgboost				
basic	45.52	8178	90.43	0.1920
engineered	43.65	7829	88.48	0.2094
optimized	43.40	7853	88.62	0.2082
support vector				
basic	189.09	50297	224.27	-1.0038
engineered	182.24	45833	214.09	-0.9128
optimized	109.13	18774	137.02	-0.2243
linear regression				
basic	70.05	12843	113.33	-0.0126
engineered	69.80	12798	113.13	-0.0108
optimized	50.83	9072	95.25	0.1490
cat boost				
basic	42.59	7679	87.63	0.2170
engineered	42.35	7636	87.38	0.2192
optimized	41.84	7423	86.16	0.2302
KNN				
basic	56.02	11007	104.91	0.0626
engineered	40.37	7352	85.74	0.2339
optimized	40.37	7352	85.74	0.2339
MLP				
basic	46.98	8515	92.28	0.1755
engineered	48.90	8680	93.17	0.1676
optimized	52.09	8902	94.35	0.1570
linear sgd				
basic	100.62	22530	150.10	-0.3411
engineered	99.13	22582	150.27	-0.3427
optimized	98.08	22645	150.48	-0.3446

Table 4.: Results of individual models with a forecast horizon of 20 minutes

Model-case	**MAE**	**MSE**	**RMSE**	**Skill-score**
SSP				
-	62.89	13799	117.47	0
random forest				
basic	49.95	13697	117.03	0.0037
engineered	51.35	14000	118.32	-0.0073
optimized	**38.85**	**6940**	**83.31**	**0.2908**
xgboost				
basic	48.01	8851	94.08	0.1991
engineered	45.89	8424	91.78	0.2187
optimized	45.88	8490	92.14	0.2156
support vector				
basic	188.92	50228	224.12	-0.9079
engineered	181.41	45421	213.12	-0.8143
optimized	110.72	19676	140.27	-0.1941
linear regression				
basic	71.18	13427	115.87	0.0136
engineered	70.81	13376	115.65	0.0154
optimized	53.40	9818	99.08	0.1565
cat boost				
basic	45.01	8278	90.99	0.2254
engineered	45.04	8281	91.00	0.2253
optimized	44.31	8025	89.58	0.2374
KNN				
basic	57.21	11404	106.79	0.0909
engineered	41.97	7760	88.09	0.2501
optimized	41.97	7760	88.09	0.2501
MLP				
basic	53.77	9487	97.40	0.1708
engineered	52.05	9572	97.84	0.1671
optimized	51.20	9341	96.65	0.1772
linear sgd				
basic	98.20	22643	150.48	-0.2810
engineered	98.36	22626	150.42	-0.2805
optimized	100.26	22524	150.08	-0.2776

D. Most used Models within Dynamic Selection

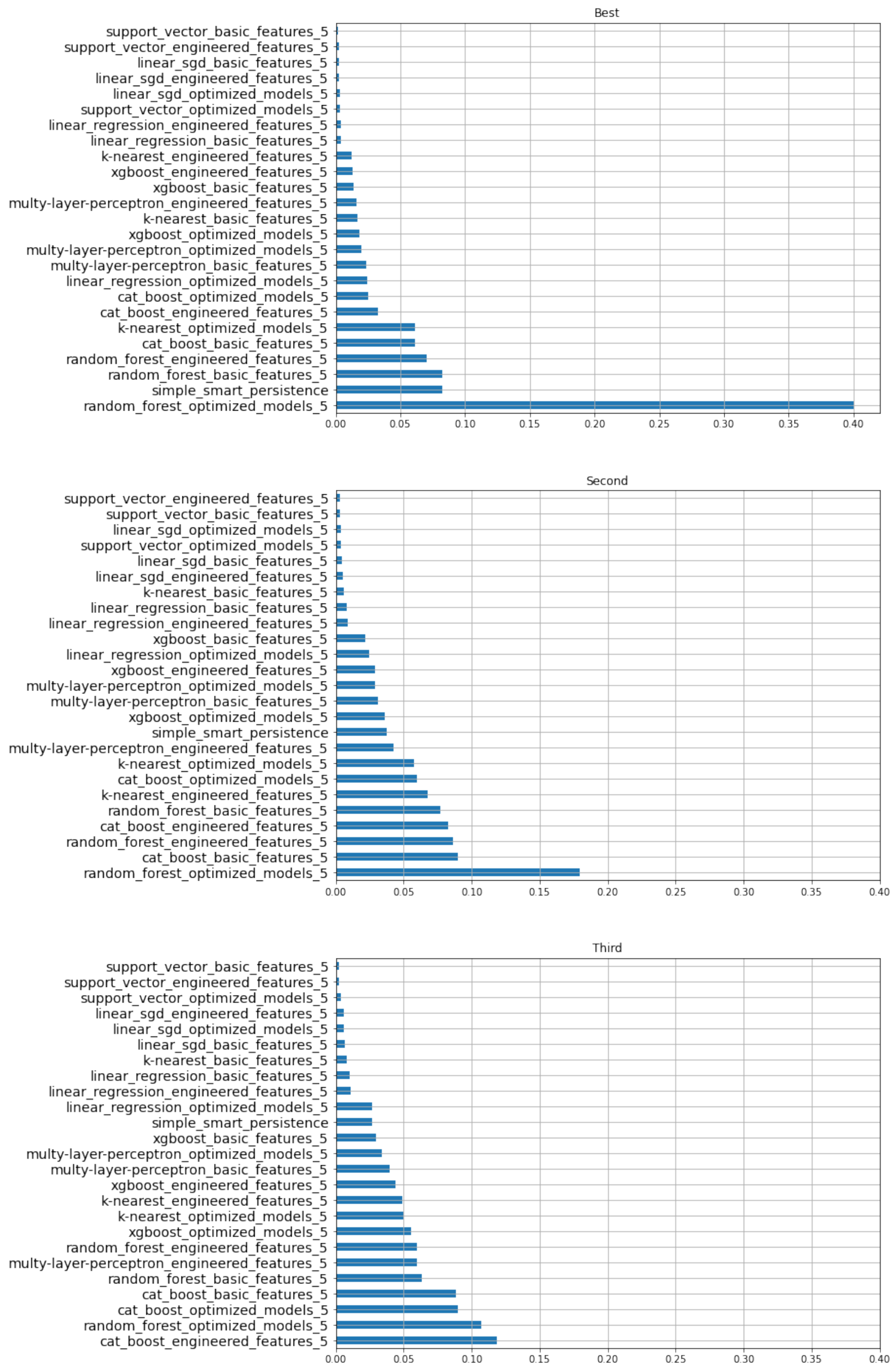


Figure 2.: Best and second best model histogram for 5 minutes horizon

E. Interval Coverage and Width Distribution

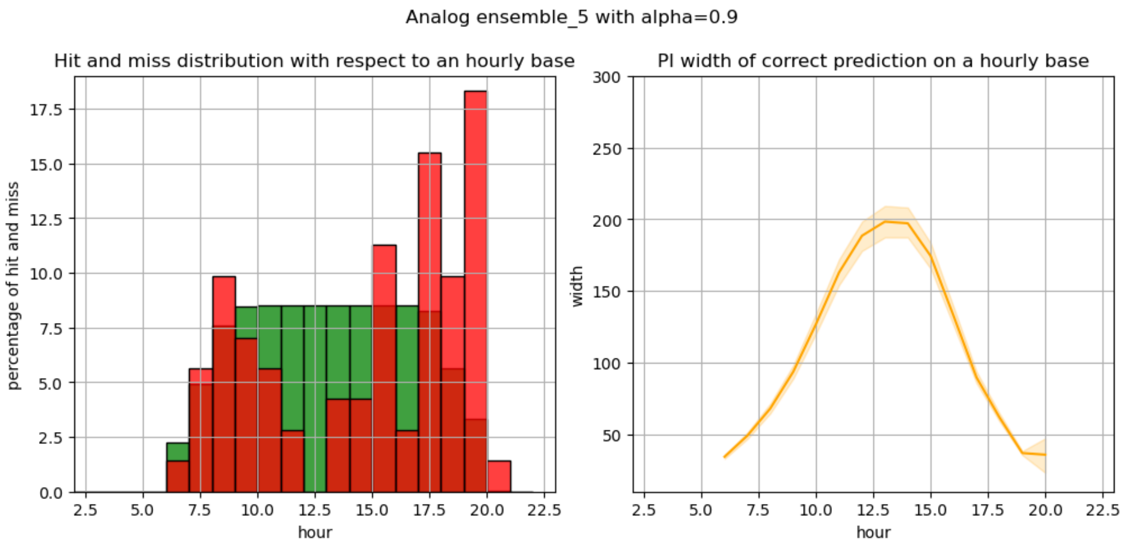


Figure 3.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

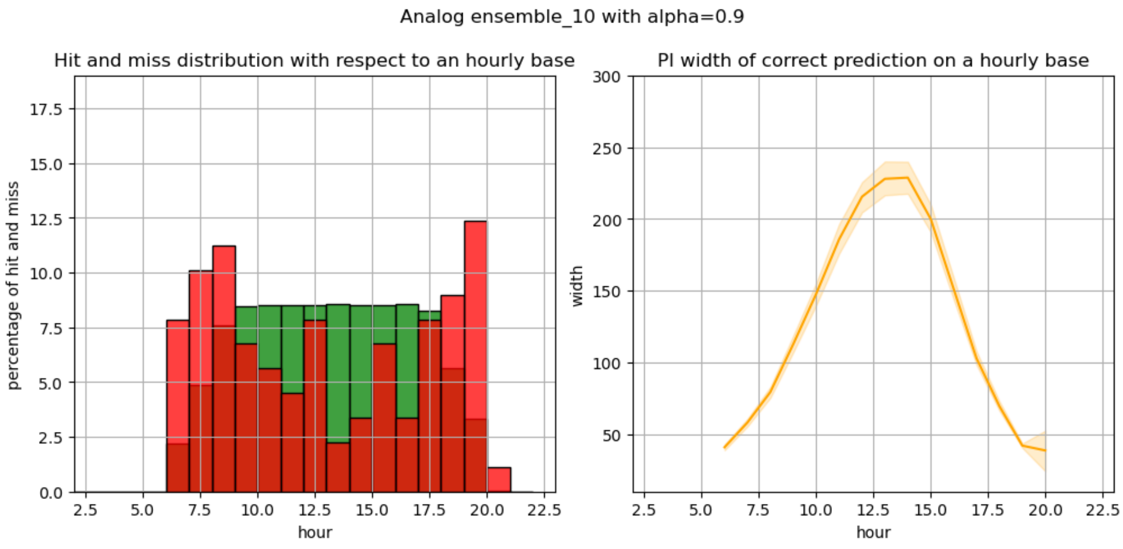


Figure 4.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

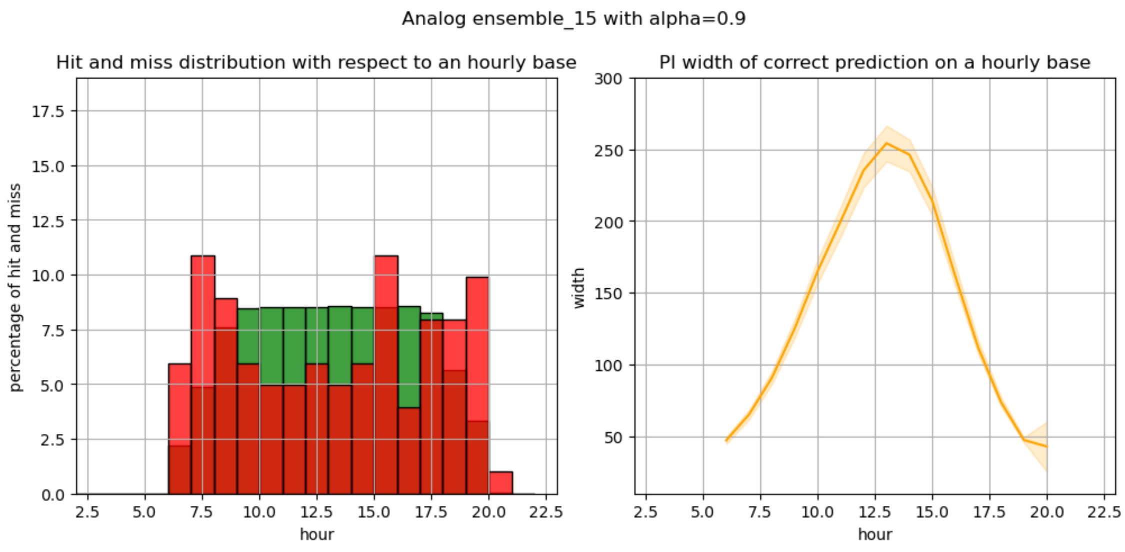


Figure 5.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

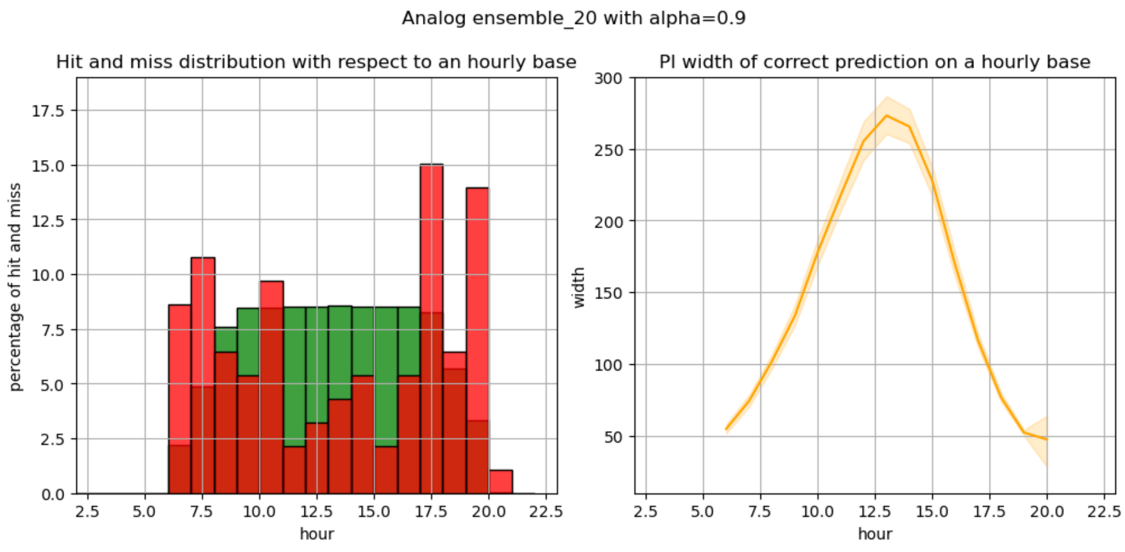


Figure 6.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

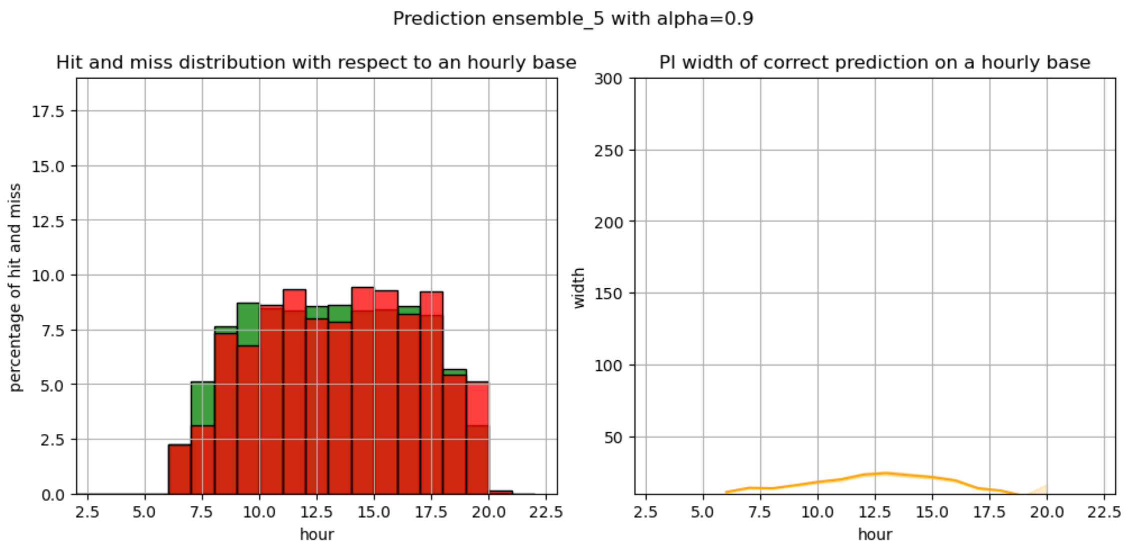


Figure 7.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

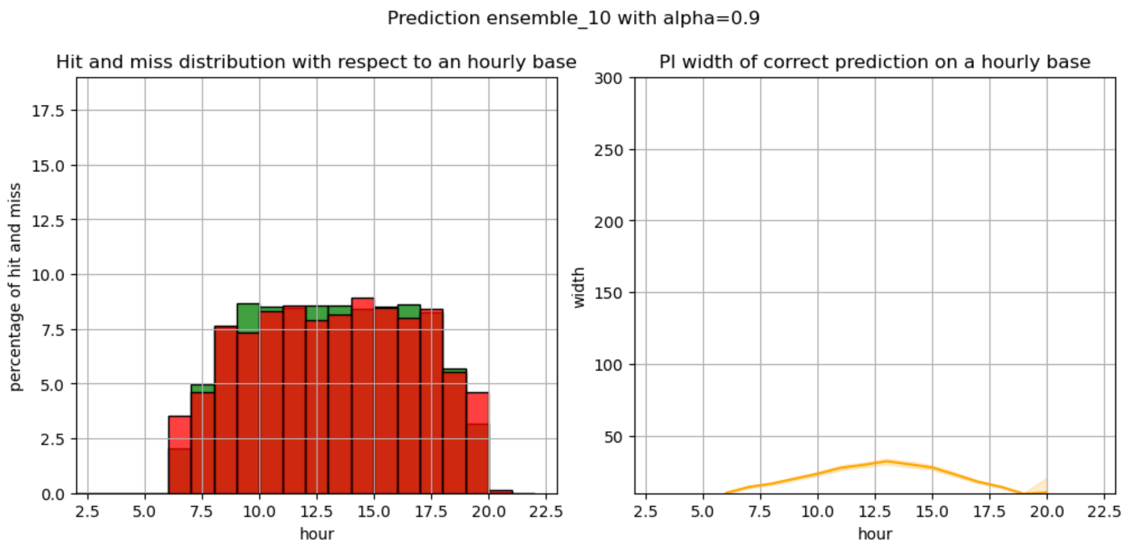


Figure 8.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

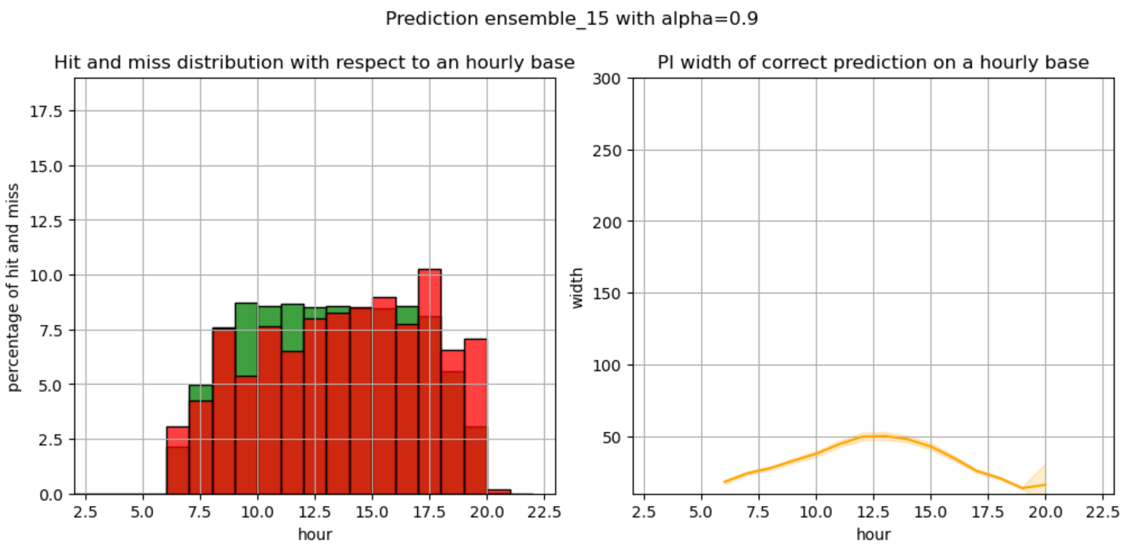


Figure 9.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

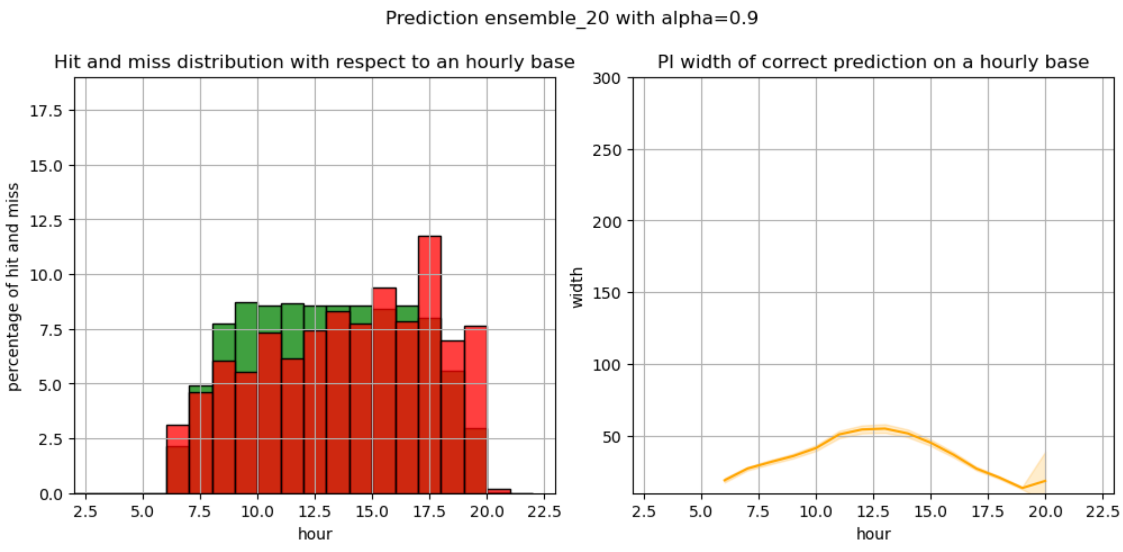


Figure 10.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

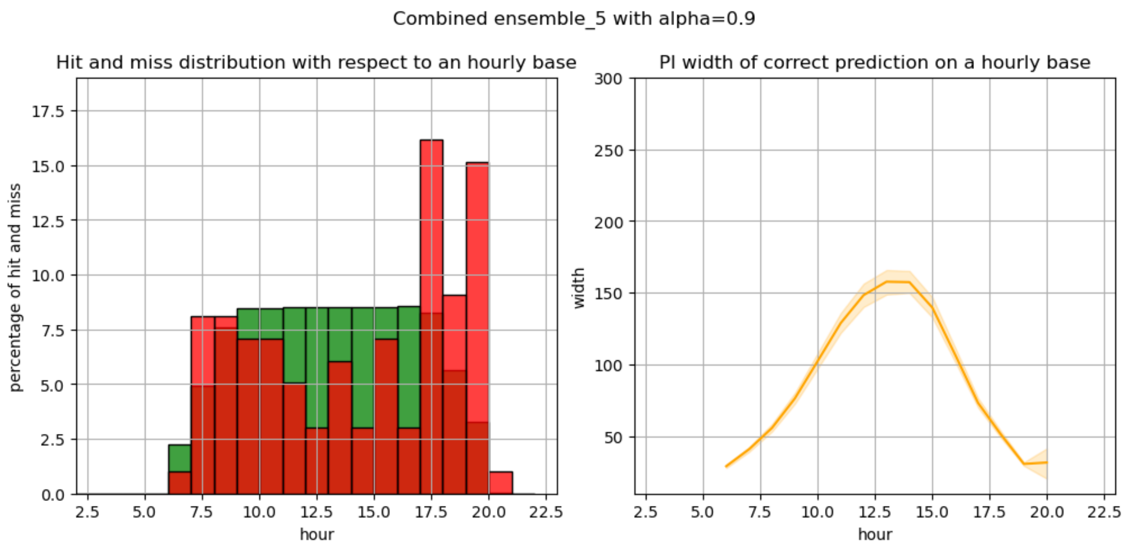


Figure 11.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

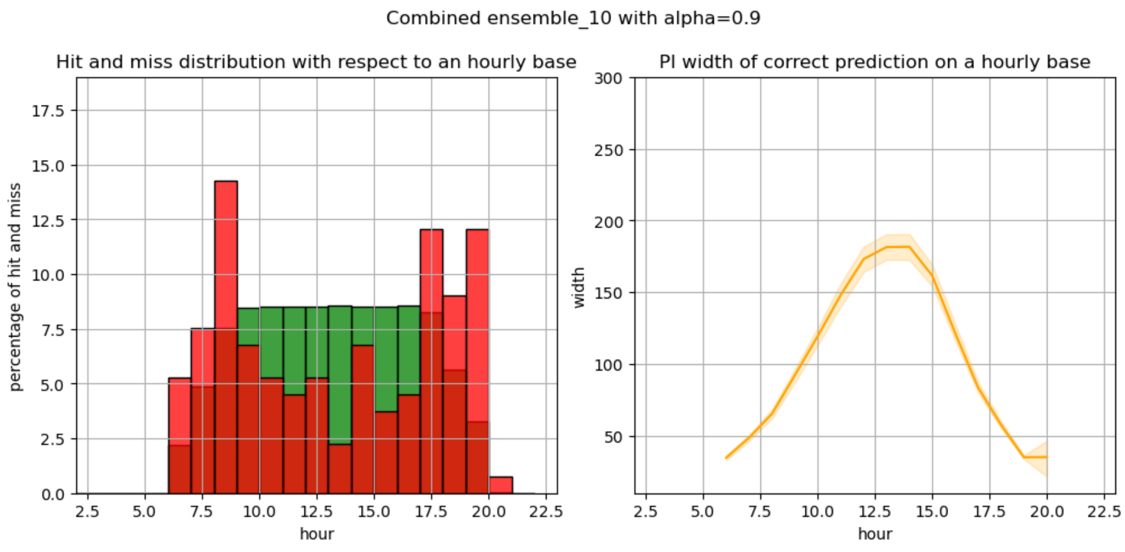


Figure 12.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

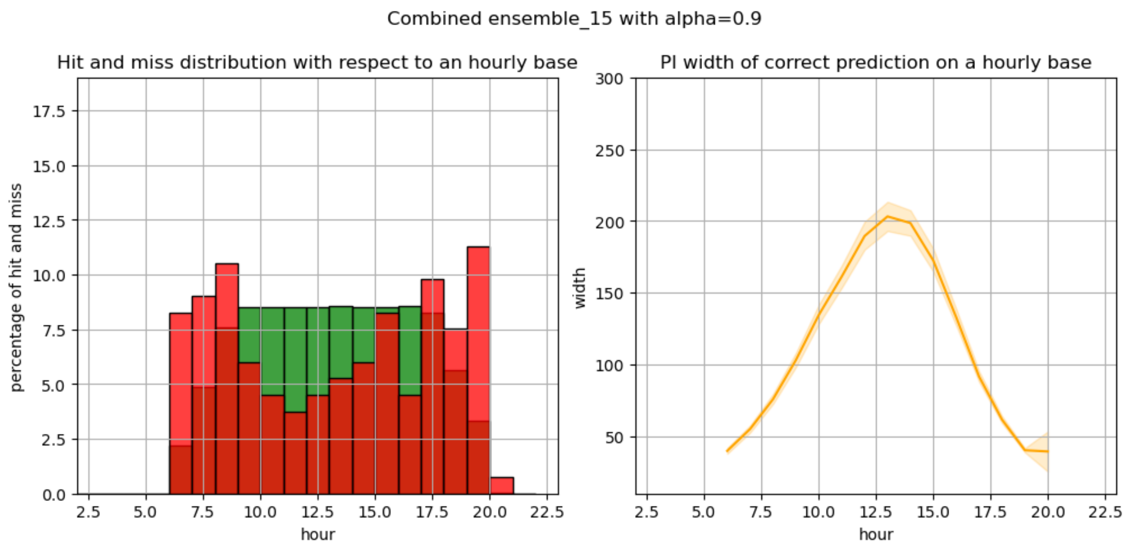


Figure 13.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

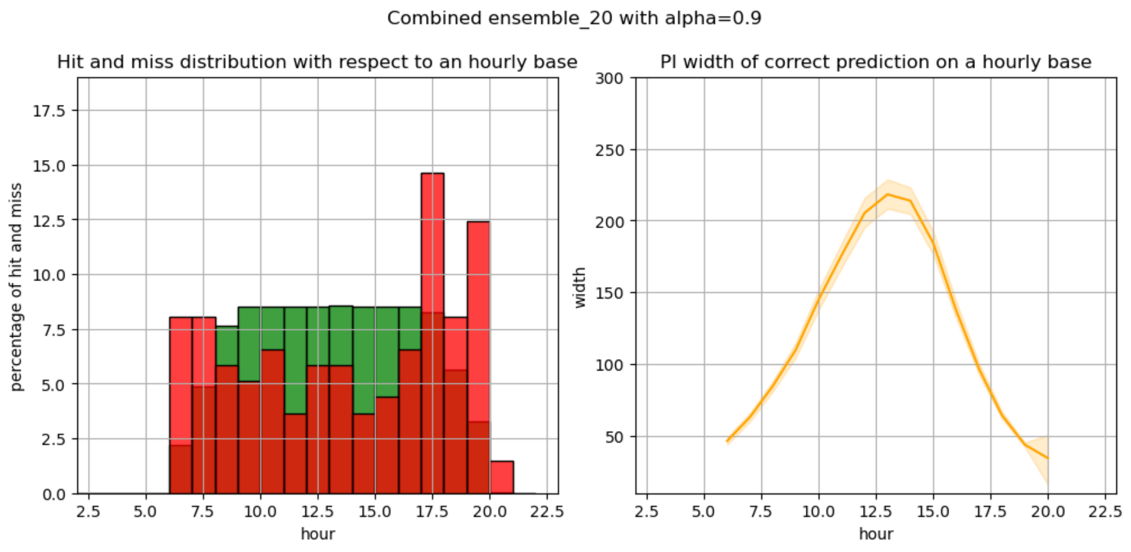


Figure 14.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

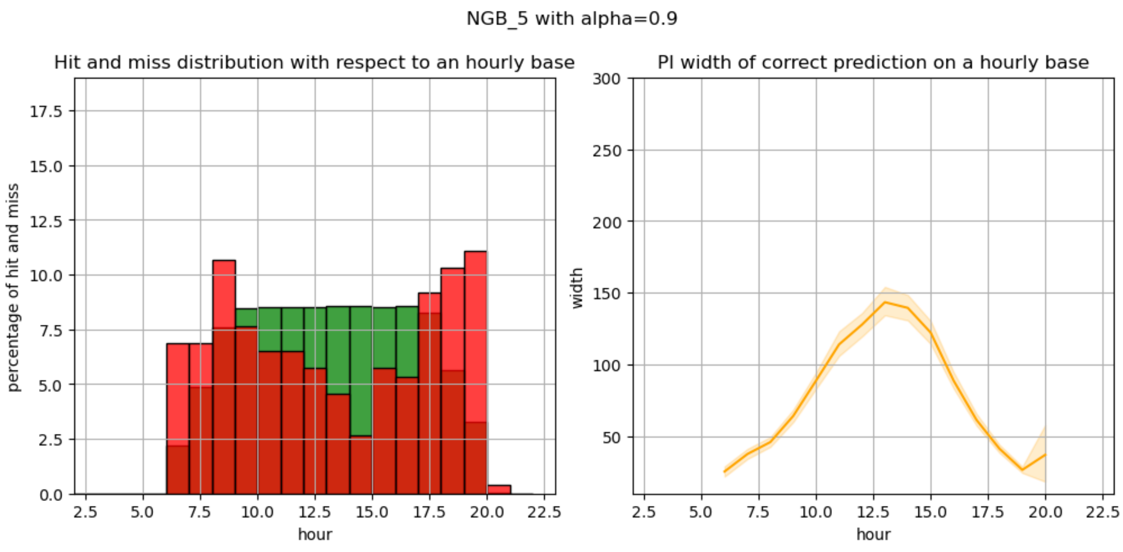


Figure 15.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

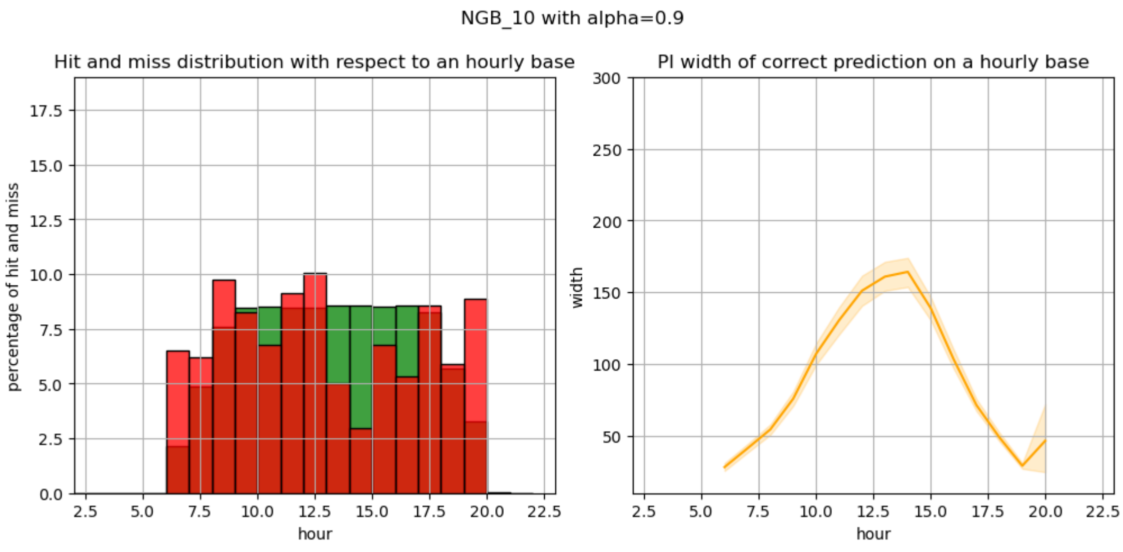


Figure 16.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

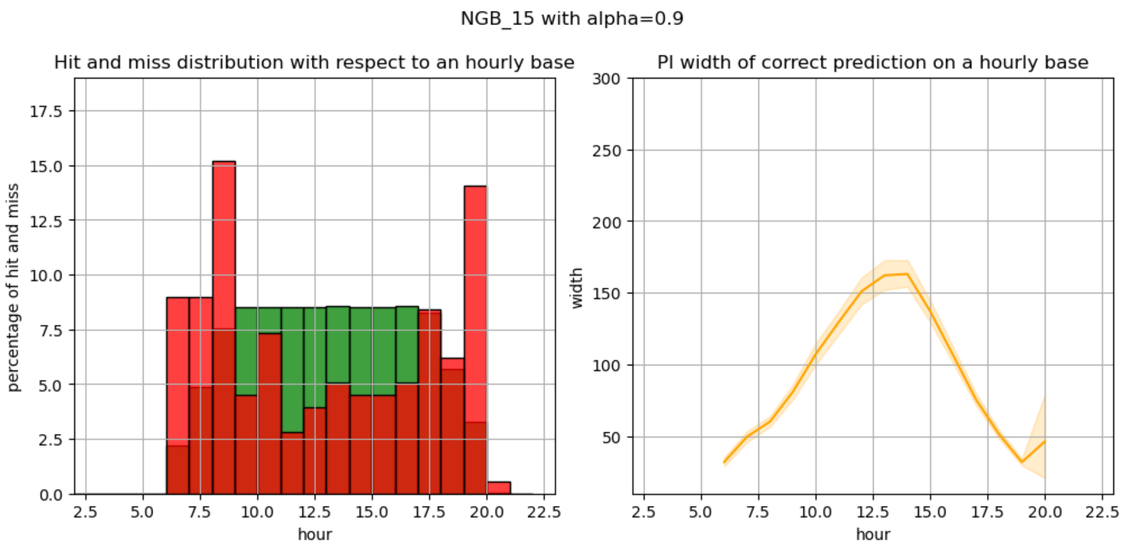


Figure 17.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

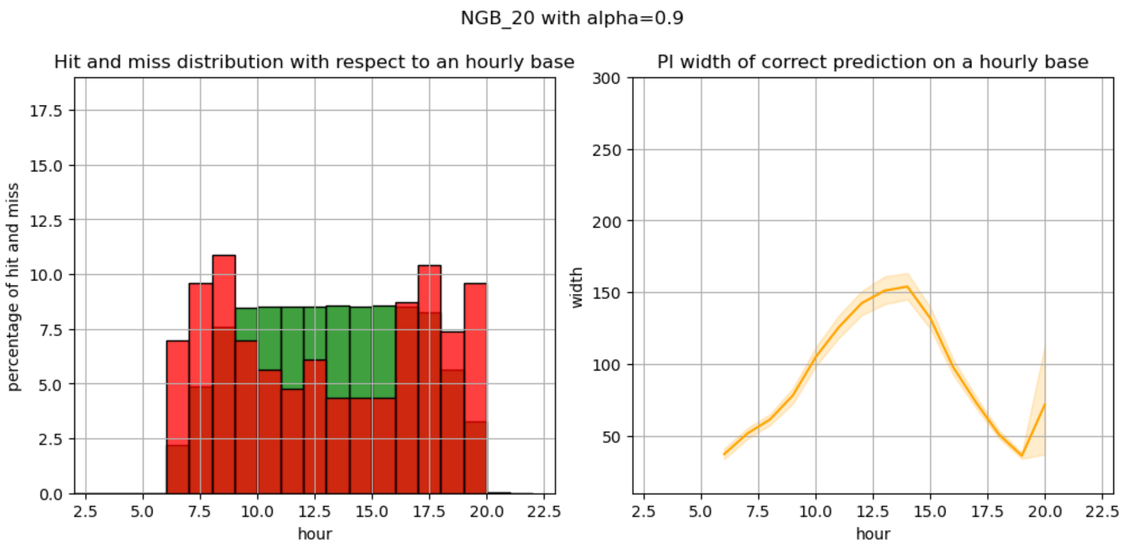


Figure 18.: GHI interval hits in *green* , misses in *red* and MIW in *orange*, distributed from 2AM to 10PM

F. Probabilistic Prediction Results of all Approaches and Forecast Horizons

Table 5.: Interval metrics with t-distribution for 5 min horizon

Method	**ICP**	**MIW**	**ACRPS**	**MAE**
Interval 5% - 95%				
Analog ensemble	0.95	139.90	21.74	42.87
ML ensemble	0.60	30.81	26.84	31.19
Combined ensemble	0.94	113.90	21.40	39.19
NGB distribution	0.86	108.00	19.02	25.19

Table 6.: Interval metrics with t-distribution for 10 min horizon

Method	**ICP**	**MIW**	**ACRPS**	**MAE**
Interval 5% - 95%				
Analog ensemble	0.95	162.01	25.42	49.82
ML ensemble	0.57	41.30	30.98	37.22
Combined ensemble	0.93	132.	24.89	45.60
NGB distribution	0.84	128.55	21.28	28.34

Table 7.: Interval metrics with t-distribution for 15 min horizon

Method	**ICP**	**MIW**	**ACRPS**	**MAE**
Interval 5% - 95%				
Analog ensemble	0.95	177.04	27.57	54.16
ML ensemble	0.70	53.78	32.03	40.71
Combined ensemble	0.93	145.97	26.82	49.68
NGB distribution	0.89	122.89	20.88	27.75

Table 8.: Interval metrics with t-distribution for 20 min horizon

Method	**ICP**	**MIW**	**ACRPS**	**MAE**
Interval 5% - 95%				
Analog ensemble	0.95	189.81	29.01	57.41
ML ensemble	0.70	58.32	33.40	42.75
Combined ensemble	0.94	156.49	28.10	52.52
NGB distribution	0.88	118.87	21.30	28.41

G. Python Environment

used packages:

- catboost=0.26.1=py38h06a4308_0
- h5py=3.6.0=nompi_py38hfbb2109_100
- hdf4=4.2.15=h10796ff_3
- hdf5=1.12.1=nompi_h2386368_104
- libxgboost=1.5.0=h6a678d5_2
- pandas=1.4.0=py38h43a58ef_0
- pickleshare=0.7.5=py_1003
- pillow=9.0.1=py38he2f12e7_1
- pip=22.0.3=pyhd8ed1ab_0
- py-xgboost=1.5.0=py38h06a4308_2
- python=3.8.12=ha38a3c6_3_cpython
- python_abi=3.8=2_cp38
- scikit-learn=1.0.2=py38h1561384_0
- scipy=1.8.0=py38h56a6a73_1
- xgboost=1.5.0=py38h06a4308_2
- yaml=0.2.5=h7f98852_2
- crps==2.0.1
- ngboost==0.3.12
- properscoring==0.1