DEEP LEARNING MODELS FOR SUNSHINE DURATION TIME SERIES FORECASTING (DRAFT)

by
Sheritz Keji Sogbesan

A Major Research Project

Submitted to Toronto Metropolitan University
In partial fulfillment of the requirement
For the degree of

Master of Science
In the Program of
Data Science and Analytics

Toronto, Ontario, Canada 2025

Abstract

Current and projected climate crises alike necessitate increased urgency in the exploration and adoption of renewable energy resources. Alongside other major benefits, Sunshine Duration (SD) has been found to be a key input into predictive models for the forecasting of Solar Radiation. This research paper studies the performance of two state-of-the-art models, Neural Basis Expansion Analysis for Interpretable Time Series Forecasting (N-BEATS) and Neural Hierarchical Interpolation for Time Series Forecasting (N-HiTS) in the domain of SD forecasting, and assesses it relative to various statistical and machine learning models over several forecast horizons and experimental conditions. A simple Naive model and a statistical method are selected as baseline comparators, and all models are evaluated using the RMSE and MAE error metrics. The implementation is carried out using modern machine learning tools, namely the Darts machine learning library and the Optuna hyperparameter optimization framework. Results show promise that both models can be effectively applied in this space, albeit with a stronger case made for N-HiTS.

Keywords: Sunshine duration, Time series forecasting, N-BEATS, N-HiTS, LSTM, GRU, Random Forest, XGBoost, LightGBM, Exponential Smoothing, Naive Drift, Darts, Optuna

Acknowledgements

There are a number of people to whom I would like to offer my most sincere gratitude. First, my dearest parents, who selflessly and lovingly paved the way for our various successes in life, nurtured in me a lifelong desire for learning and development, never waver in their conviction that I can accomplish anything I put my mind and efforts to, and always cheer me along every step of my journey. My Supervisor Dr. Aliaa Alnaggar, whose encouragement and guidance throughout this project were highly appreciated. Dr. Mucahit Cevik, whose great kindness and dedication to our success throughout this program will not soon be forgotten. My darling brothers, who have also believed in, and cheered me on throughout the years. My dearest friends, including my most ardent supporters to whom I owe my retained sanity and hair, Henrietta and Ghassan. My heartfelt thanks to you all.

Table of Contents

Table of Contents (Continued)

List of Figures

List of Tables

1
Introduction

Due to the limited nature of the world's energy resources in the face of growing demand, rising energy costs, and the environmental impact of greenhouse gas emissions, solar energy continues to grow in importance as a clean and renewable resource. However, global solar radiation measurement instruments such as actinographs are often lacking in reliability, and sources of more accurate measurements are expensive [1]. Sunshine Duration (SD) factors into key processes such as photovoltaic system sizing, the modeling and design of solar crop dryers, and the estimation of solar radiation, to which it is highly correlated [2]. It also comes with the benefit of greater measurement accuracy than the latter in various parts of the world, but some countries still find themselves with limited or occasionally unreliable SD database and maps. Despite this, only a limited number of studies have been carried out for the estimation of SD, relative to the wealth of literature that exists on the estimation of global solar radiation [1].

This project uses a thirty-year meteorological dataset for the town of Bordj Badji Mokhtar in Algeria to evaluate the performance of novel state-of-the-art models Neural Basis Expansion Analysis for Interpretable Time Series Forecasting (N-BEATS) and Neural Hierarchical Interpolation for Time Series Forecasting (N-HiTS), alongside a number of other deep and standard machine learning models, in the forecasting of SD. The rest of the paper is organized as follows: Section 2 provides a review of relevant literature; Section 3 covers the procurement, description, and exploratory analysis of the dataset; Section 4 describes the methodology and experiments; Section 5 provides a description and analysis of the empirical findings; and Section 6 concludes the paper with a summary, final comments, and suggestions for future work.

2

Literature Review

2.1 Time Series Forecasting

Time series forecasting is a popular area of research, with applications across various fields such as energy, business, economy, health, and the environment. The process predicts a dependent variable's future values after capturing patterns that may include seasonality, trend, and noise [3]. There are generally two classes of forecasting techniques: traditional statistical models, and artificial intelligence (AI). The statistical group includes regression analysis, moving average, exponential smoothing, stochastic time series models, etc., while machine learning, artificial neural networks, genetic algorithms, fuzzy time series, etc. are based on AI techniques [4].

Limitations in the existing literature claiming the superiority of machine learning methods to more traditional forecasting methods include the evaluation of only short-term forecasting horizons (often one-step ahead), ignoring medium and long-term ones, as well as the lack of benchmarks to compare the accuracy of ML methods to alternative ones [5]. While improvements in the accuracy of the newer ML methods have been observed, particularly where deep learning neural networks are concerned, they have yet to prove exceptional when compared to traditional statistical methods. In fact, [5] found that six of the most accurate methods reviewed were statistical, confirming dominance over the ML ones. There is therefore a need to ensure that machine learning experiments are properly contextualized before accurate conclusions can be drawn.

2.2 Sunshine Duration

Sunshine duration (SD) values not only define the time period during which direct solar radiation reaches the Earth's surface, but also provide some data on the quantity of total solar energy [6]. Given that SD is generally recognized as a key accuracy-improving input feature in the prediction of the solar radiation [7], an increasing number of studies is exploring its prediction using statistical and machine learning models, along with meteorological and other data. [8] studied the temporal and spatial variability of both monthly mean sunshine hours and solar radiation with seven widely used interpolation methods. They found that thin plate smoothing spline with UTM coordinates and elevation performed well for sunshine hours, and that the regression equations using geographical parameters and the predicted sunshine hours yielded better performance in the estimation of solar radiation. [2] developed a Radial Basis Function (RBF) neural network model to estimate sunshine hours (S) and sunshine ratios (S/S_o), with the goal of using the model to generate data for other meteorological stations with no record of daily (S); this data would then serve as the input for generating sunshine hours and ratios contour maps for Oman that would provide a reference for the monthly spatial distribution of (S) and (S/S_o), from which further estimates could be made for other locations without the use of the computationally intensive RBF. The authors developed several RBF models, and the best one yielded an RMSE of 0.75 h. In [9], the authors evaluated support vector machine's (SVM) performance in the estimation of daily SD using three different kernels: linear, polynomial, and radial basis function (RBF). They found that the SVM-RBF model yielded better simulated results, and that a combination of the maximum number of relevant atmospheric variables (maximum possible sunshine hours, cloud cover, relative humidity, maximum temperature, minimum temperature, and wind speed) was the best choice for inputs.

The authors in [1] used three different artificial neural network (ANN) models -

generalized regression neural network (GRNN), multilayer perceptron (MLP), and radial basis function Network (RBF) - to estimate monthly mean daily values of global SD for 34 stations in Turkey. MLP and GRNN displayed better performance than RBF, with observed RMSE of 0.6256, 0.7971, and 1.1279, respectively. [10] optimized a multilayer feed-forward neural network model with a learning scheme of the backpropagation of errors and the Levenberg-Marquardt algorithm for the adjustment of the connecting weights, and the model yielded an overall MAPE of 2.015% and %RMSE of 2.741%, leading to a significantly better estimation of SD relative to four other published studies using ANN models. [11] and [12] used multiple linear regression to predict SD. The former found high model performance for the months of March, April, May, October, and November; moderate performance for January, February, June, and December; and weak performance for the months of July and August. SD was also found to be inversely related with cloud cover, precipitation, and relative humidity. In [13], a remote sensing model was developed to estimate SD. Annual average, and relative interpolation errors of approximately 2.21% and 7.90%, respectively, were observed. The authors found that the use of satellite data to improve model estimation accuracy led to a significant improvement in performance.

2.3 State-of-the-Art Models

2.3.1 Neural Basis Expansion Analysis for Time Series Forecasting (N-BEATS)

N-BEATS is a deep neural architecture based on backward and forward residual links, and a very deep stack of fully connected layers that is interpretable, applicable to a wide array of target domains without modification, and fast to train. Its creators [14] tested the two configurations of the model on several well-known datasets such as M3 and M4, and were able to improve forecast accuracy by 11% over a statistical benchmark, and by 3% over the previous year's winner of the M4 competition. This was the first study to empirically demonstrate that pure deep learning without time-series components (i.e. the generic version) can outperform well-established statistical approaches on the datasets in question. Furthermore, the same hyperparameters were fixed across horizons and datasets, which was evidence of generalizability across heterogeneous time series. [15] trained an N-BEATS model on a vast dataset consisting of diverse time series data, and subsequently applied the trained model to forecast 18 months of electric power generation using zero-shot transfer learning. The M4-based N-BEATS model's performance surpassed that of all the standard deep learning models, and it exhibited faster execution time than the models that require data-specific training. [16] introduced a combined approach that used a multi-layer decomposition of an N-BEATS model to extract interpretable trend and seasonality components from traction energy consumption data, which were then used as time-varying input variables for a temporal fusion transformer model. The joint NBEATS-TFT model achieved a 0.06 RMSE and prediction error of 5%, yielding a 20% greater prediction accuracy on average compared to other models such as the standalone TFT and N-BEATS. [17] applied an LSTM-NBEATS model to hourly electricity load datasets from three European countries, and while it performed better than the baseline LSTM and N-BEATS models for two of them, the original N-BEATS model displayed the best performance for the third.

2.3.2 Neural Hierarchical Interpolation for Time Series Forecasting (N-HiTS)

The N-HiTS model is an extension of the N-BEATS architecture that was introduced by [18] to address common challenges faced by neural long-horizon forecasting, namely, volatility of predictions, and excessive computational complexity. After conducting extensive experiments on six large-scale benchmark datasets from the long horizon forecasting literature, the authors found that the model outperformed the best baseline, delivering average relative error decreases across datasets and horizons of 14% in MAE and 16% in MSE. It maintained comparable performance to other state-of-the-art methods for the shortest measured horizon, and decreased multivariate MAE and MSE by 11% and 17%, respectively, for the longest measured horizon. Furthermore, it proved 1.26% faster than the original N-BEATS, while only requiring 54% of the parameters. In [19], N-HiTS was similarly found to be more resource-efficient for handling longer input smart grid data streams relative to the Temporal Fusion Transformer (TFT); however, TFT generally outperformed N-HiTS during the training stage, and in the performance of long-term forecasting tasks. Using N-HiTS and LSTM, [20] predicted the regional standardized precipitation index (SPI) in four regions of Zacatecas, Mexico, with N-HiTS producing better results than LSTM across all regions, while consistently achieving higher correlation coefficients (0.95+) and low standard deviation values relative to the observed SPI values. [21] introduced a GRU-Temporal Fusion Transformer (GRU-TFT) model with a "DILATE" loss function to forecast solar power, and compared its performance to that of a range of traditional statistical and machine learning techniques. While their model's performance surpassed that of all other assessed models with MAE of 1.19, MSE of 2.08, and RMSE of 1.44, N-BEATS (1.997, 6.729, 2.594) and N-HiTS (2.622, 9.029, 3.005) came in second and third place, respectively.

2.4 Hyperparameter Tuning

Hyperparameters are parameters that are not learned and internally adjusted by machine learning algorithms, but are instead configured prior to the beginning of the learning process, and dictate how the model is structured [22]. As such, they wield a significant amount of influence on model performance. They are key to preventing overfitting, as well as improving the generalization of a given model, and their optimization improves forecasting accuracy while reducing model complexity [23]. Hyperparameter tuning is the fine-tuning of the model hyperparameters in order to determine the values that maximize its performance on a validation set. It has traditionally been a manual and therefore cumbersome and time-consuming process for which researchers often depend on knowledge derived from prior experience solving similar problems. Unfortunately, however, the best settings used to solve one problem may not be the same for another, as these values can vary by dataset.

Though a critical aspect of model training, the tuning process is often overlooked or simplified in certain machine learning literature. To address this, [24] compared three hyperparameter optimization algorithms in the machine learning-based urban building energy model space: Grid search, Random search, and Bayesian search. Grid search, while widely used due to its simplicity in implementation and parallelization, suffers from the curse of dimensionality, with exponential growth in number of trials to run as the number of hyperparameter increases. Although the search methods ultimately displayed similar performance, Random search stood out for its effectiveness, speed, and flexibility. Furthermore, hyperparameter tuning not only led to higher performance in most of the trained models, but also to the SVM model moving from being the lowest performer to being ranked the second best of the five tested models. [22] similarly studied those search strategies and two additional ones - Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) - in the area of machine learning classification algorithms for Arabic sentiment analysis. While the Support Vector Classifier yielded the best accuracy both prior to and after hyperparameter tuning, PSO and GA were found to have significantly enhanced the performance of the Naive Bayes classifier.

2.4.1 Optuna

Introduced in 2019, Optuna is an open-source next-generation hyperparameter optimization software with a *define-by-run* API that allows users to dynamically construct the parameter search space, efficient implementation of both searching and pruning strategies, and versatile architecture that can be deployed for purposes ranging from scalable distributed computing to light-weigh experimentation via interactive interface [25]. [26] implemented five boosting algorithms, Adaboost, Gradient boosting, XGBoost, LightGBM, and CatBoost, and their results indicated that Optuna improved prediction accuracy across all of the models. XGBoost saw the greatest improvement relative to its performance with default settings. [27] used Optuna to hypertune the temporal fusion transformer (TFT) model for hydroelectric inflow prediction, and found the tuned model's performance to be better and more stable than the other models' across the entire forecast horizon of 1 to 14 days. A comparison in [23] of two LSTM models used for wind power forecasting - one with hyperparameters determined by grid search, and the other, optimized by Optuna - showed that the latter improved on the performance of the grid search version by 1.22% to 2.65% across the six studied cases. Optuna also exhibited faster performance, spending 13.79% to 20.59% less time arriving at the optimized values.

2.5 Objectives

To the best of the author's knowledge, there has yet been no research carried out into the performance of the N-BEATS and N-HiTS models in the area of sunshine duration forecasting. The current study's objectives are therefore as follows:

1. Pioneering the use of the novel N-BEATS and N-HiTS to predict sunshine duration, and measuring their performance against various standard and deep machine learning models as well as a simple statistical benchmark model, Exponential Smoothing, and a naive baseline model selected from options offered by the Darts machine learning library;

2. Exploring the streamlining of time series forecasting using Darts for model implementation, and the Optuna framework for automated hyperparameter tuning, and comparing the latter's results to model performance using default settings.

The code and data used for this project can be found at https://github.com/sheritzs/ml-research-project.

3

Dataset Description and Exploratory Data Analysis

3.1 Dataset

For this study, a meteorological dataset for Bordj Badji Mokhtar (BBM) in Algeria was sourced from the free Historical Weather API provided by [28]. BBM is located in the southern region of the Algerian Sahara, which is one of the desert ecoregions in the northern part of Africa [29]. Using [28]'s Geocoding API, it was geolocated at a latitude of 21.3292 and a longitude of 0.94791. Spanning a period from January 1, 1994 to December 31, 2023, the dataset is comprised of three columns representing sunshine duration, relative humidity, and air temperature; this composition reflects the intent to replicate and compare results against the work in [29]. The data was originally downloaded at an hourly level of granularity, and subsequently aggregated at a daily level; the result was a decrease from 262,968 to 10,957 total rows. There was no missing data to contend with. As part of the daily aggregations, SD was transformed from seconds to an hourly representation, and the minimum, mean, maximum, and range (i.e. difference between minimum and maximum) values for temperature, as well as mean values for relative humidity were computed.

3.2 Exploratory Data Analysis

As seen in Figure 1, 72% of the observed SD is between 10.2h and 12h, while 3% is below 8.5h. BBM's SD is of a seasonal nature (Figure 2). There was an uncharacteristic dip in trend around 2017 that had yet to have fully rebounded by the end of 2023; this has had a visible impact on the residuals. There is a strong outlier presence throughout all the months of the year, which increases what would otherwise typically be a range

Figure 1: *BBM Hourly Sunshine Duration Frequency (1994-2023)*

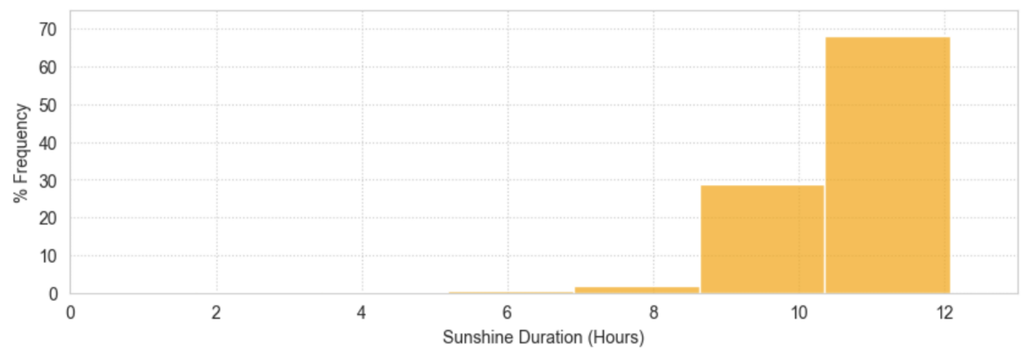
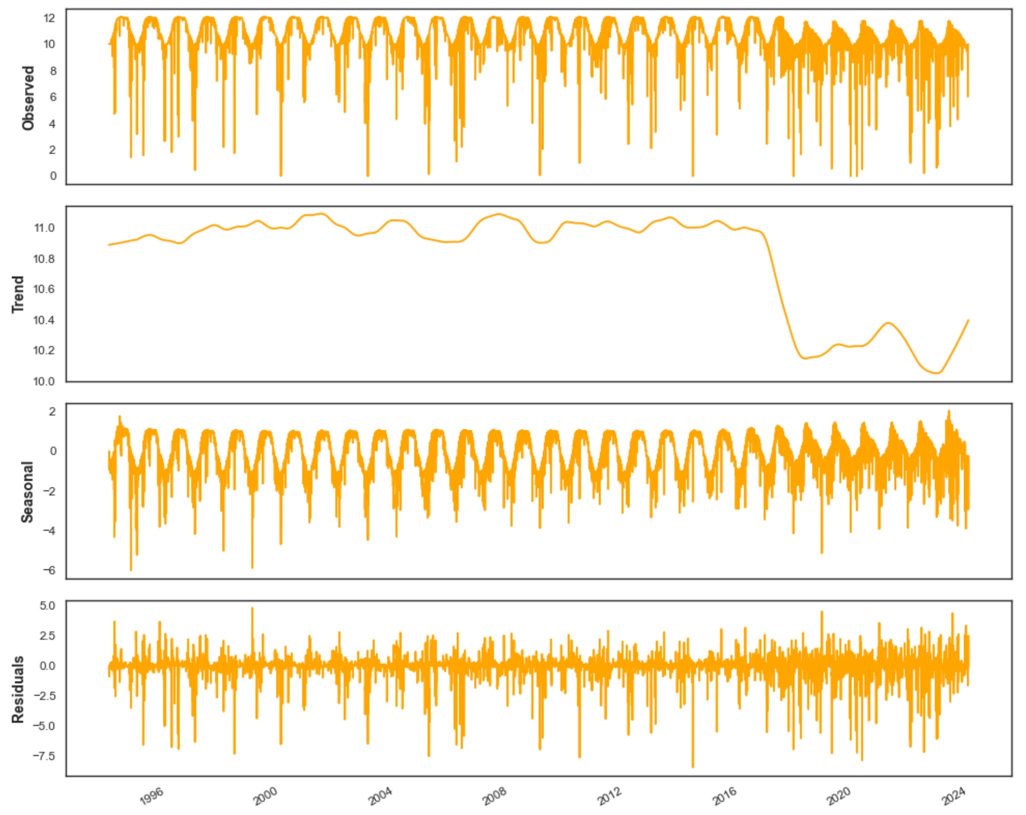


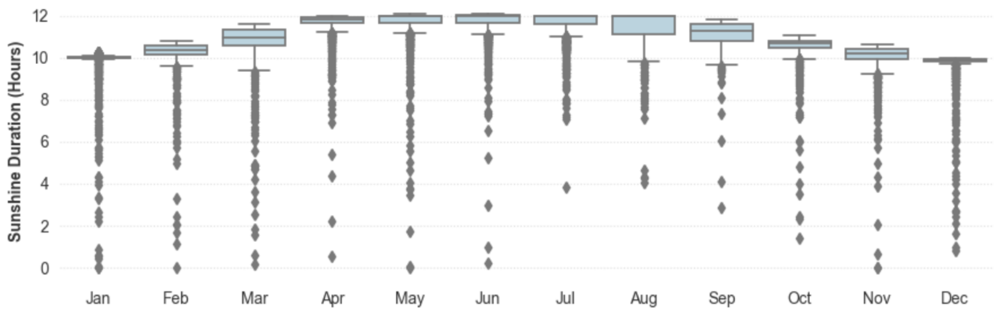
Figure 2: *BBM Sunshine Duration Seasonal Decomposition*



of roughly three hours to 12 hours (Figure 3).

Of the features, maximum and mean temperatures are the most strongly correlated with SD, with r values of 0.57 and 0.53, respectively (Figure 4); however, these values indicate a generally moderate relationship. For its part, mean humidity has a weak negative correlation with SD, at -0.33. This is because while the two display a

Figure 3: *BBM Monthly Sunshine Duration (1994-2023)*



pronounced inverse relationship at times, this dynamic is not consistent throughout the year, as seen in Figure 5. Maximum temperature and SD, meanwhile, generally follow the same trend and maintain their relative relationship throughout most of the year.

Figure 4: *Correlations*

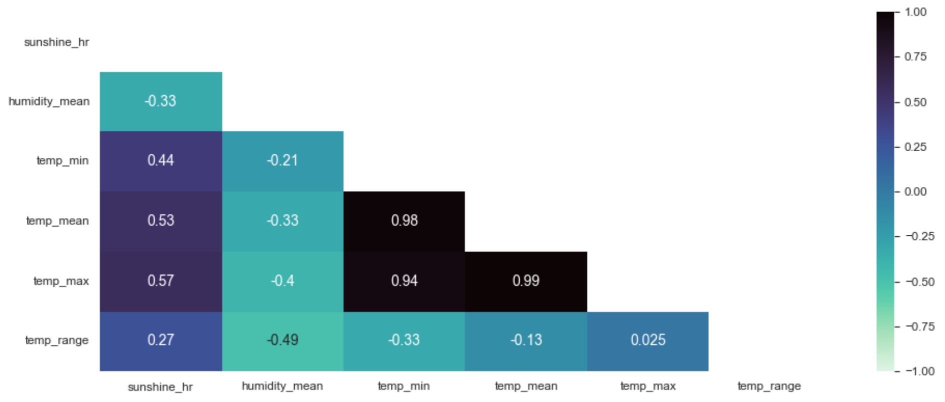
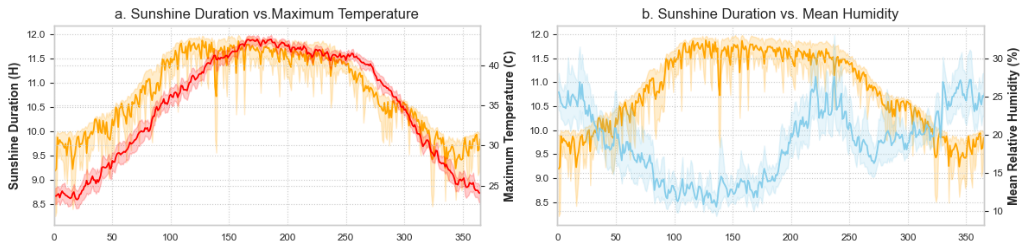


Figure 5: *Daily Trends*



4
Methodology and Experimentation

4.1 Models

4.1.1 Naive Baseline Models

The Darts library [30] provides the below collection of simple baseline benchmark models:

4.1.1.1 The **Naive Drift** model fits a line between the first and last points of the training series, and subsequently extends it into the future. This yields the following equation for a training series of length T:

$$\hat{y}_{T+h} = y_T + h\left(\frac{y_T - y_1}{T\text{ - }1}\right) \tag{1}$$

4.1.1.2 The **Naive Mean** model simply returns a prediction corresponding to the mean value of the training series.

4.1.1.3 The **Naive Moving Average** generates a forecast using an autoregressive moving average (ARMA), and it accepts an *input_chunk_length* argument as the size of the sliding window that it uses to calculate the moving average.

4.1.1.4 The **Naive Seasonal** model always predicts the value of K times ago, such that when K=1, it returns the last value of the training dataset. For a daily-level dataset and K of 365, it would return the value for the reference date in the previous year.

4.1.2 Statistical Model

4.1.2.1 Exponential Smoothing (ETS) was proposed in the late 1950s, and has been behind some of the most successful forecasting methods. These methods produce forecasts that are weighted averages of past observations, with the most recent ones carrying higher associated weights, while the past observation weights experience exponential decay [31]. The Darts library's implementation is a wrapper around statsmodel's Holt-Winter's Exponential Smoothing [30], which is an extension of Holt's method to capture seasonality. This extended method comprises the forecast equation and three smoothing equations - for one for the level l_t, one for the trend b_t, and one for the seasonal component s_t, with corresponding smoothing parameters, α, β^*, and γ. The period of seasonality, i.e. the number of seasons in a year, is denoted as m [31].

The Holt-Winters method has two variations: additive and multiplicative. The former is preferred when seasonal variations are roughly constant throughout the series, and within each year, the seasonal component will add up to approximately zero. The latter, meanwhile, is preferred when observed seasonal variations are changing proportionally to the level of the series, and the seasonal component in this case will sum up to approximately m [31]. The component form for the additive method is :

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-m(k+1)} \tag{2}$$

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{3}$$

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \tag{4}$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}, \tag{5}$$

where k is the integer part of $(h$ - $1)/m$, which guarantees that the estimates of the seasonal indices used for forecasting are obtained from the final year of the sample.

The component form for the multiplicative method is:

$$\hat{y}_{t+h|t} = (l_t + hb_t)s_{t+h-m(k+1)} \tag{6}$$

$$l_t = \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{7}$$

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \tag{8}$$

$$s_t = \gamma \frac{y_t}{(l_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m}. \tag{9}$$

4.1.3 Ensemble Machine Learning Models

4.1.3.1 Random Forest (RF) is a decision tree ensemble model that is implemented using a bagging technique, in which each decision tree is built with a randomly selected subset of the training dataset, and the estimation of the ensemble model is the average prediction of the decision trees for a given data point [32]. Though more typically used for cross-sectional data, RF can be modified for time series forecasting by taking lag features and other time-dependent properties into account [33]. The adoption of RF for time series can be described as:

$$X_t = (y_{t-p}, y_{t-p+1}, y_{t-p+2}, ..., y_{t-1}, y_t), \tag{10}$$

where X_t is the input feature at time t, and p is the number of lags determined. The inclusion of past observations as features enables the algorithm to capture temporal dependencies. The algorithm is represented by the mean feature

$$\hat{y}_t = \frac{1}{n}\sum_{i=1}^{n} f_i(X_t), \tag{11}$$

where $f_i(Xt)$ is the ith decision of the X_t input.

4.1.3.2 Extreme Gradient Boosting (XGBoost) is said to be not only the fastest, but also the best-integrated decision tree algorithm, and it has recently become popular for time series forecasting [34, 33]. Building an ensemble of regression trees in order to make predictions, it generally exhibits good performance when used on seasonal nonlinear time series. The mathematical expression of the model is [33]:

$$O_i = \sum_{t=1}^{T} L(y_t, \hat{y}_t) + \sum_{i=1}^{k} \Theta(f_i) \tag{12}$$

where $L(y_t, \hat{y}_t)$ is the loss function between the true value y_t and the predicted value $\hat{y}_t$ at time t, $\Theta(f_i)$ is the regularization function of the i-th complexity tree. $\hat{y}_t = \sum_{i=1}^{k}(f_i)(X_t)$ represents the predicted values, with $f_i(X_t)$ being the i-th tree input feature at time t. Key benefits of XGBoost include its ability to model complex relationships without any assumptions pertaining to the form of the relationship; the ability to automatically handle interactions and non-linearity, thereby capturing intricate relationships; robustness to outliers; and feature engineering capabilities that allow for the incorporation of leading indicators and external predictors in order to improve forecast accuracy [33].

4.1.3.3 Light Gradient-Boosting Machine (LightGBM) is a decision tree-based algorithm that counts time series forecasting amongst the various machine learning tasks for which it has been widely used [35]. It uses a gradient boosting framework to build multiple decision trees whose predictions it subsequently combines to make a final forecast. It splits nodes in the decision tree using a histogram-based approach, which reduces both the time and memory required for model training; this has earned it recognition for its speed and efficiency. The decision trees are trained using the gradient descent algorithm, the basic equation for which is as follows:

$$w(t+1) = w(t) \; - \; learning\_rate \; \times \; gradient(Loss(w(t))), \tag{13}$$

where $w(t)$ is the weight of the model at time t, learning_rate is the gradient algorithm's step size, and gradient(Loss($w(t)$)) is the gradient of the loss function with respect to the weight at time t [35].

4.1.4 Deep Learning Models

4.1.4.1 Long Short-Term Memory (LSTM) is a special kind of Recurrent Neural Network (RNN) that comes with additional features used to memorize a sequence of data, allowing it to remember the values from earlier stages for future use [36]. Each LSTM is a set of cells where data streams are captured and stored, with cell gates allowing data in each cell to be disposed of, filtered, or added to the next cells. These gate-based features address the generic RNN's limits of only remembering a few steps earlier in the sequence, which makes it unsuitable for longer data sequences. There are three gate structures: input gates, output gates, and forget gates; this architecture provides the ability to selectively extract useful historical information, making it well suited for time series processing [3, 37]. The gate functions are described as follows [3]:

The forget gate determines which information will be kept or discarded through:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{14}$$

where x_t is the input at time t, h_{t-1} is the previous output of the cell, and σ is the sigmoid function. An output of one (1) signifies "remember/keep everything," while zero (0) means "remember/keep nothing." Following this step, the sigmoid function creates a vector of possible new values, with the input gates subsequently deciding which values to update, and a new vector of candidate values $C_t^{'}$ being created by:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{15}$$

$$C_t^{'} = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{16}$$

The cell's old state C_{t-1} is then updated to the new cell state C_t:

$$C_t = f_t * C_{t-1} + i_t * C_t^{'} \tag{17}$$

Finally, the output of the network is decided based on the cell state. The sigmoid layer first determines which parts of the cell state will be used, and then the *tanh* function is used for the cell state and multiplied by the sigmoid layer:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{18}$$

$$h_t = o_t * tanh(C_t) \tag{19}$$

4.1.4.2 Gated Recurrent Unit (GRU) is a simplified version of LSTM that has fewer gates and no separate cell state, as it relies on a hidden state for memory transfer between recurrent units. The only two gates are the reset gate r, which determines the amount of information to be forgotten, and the update gate z, which defines the amount of information to keep [38]. The mechanics of GRU are expressed as follows [39]:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \text{ - Reset Gate} \tag{20}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \text{ - Update Gate} \tag{21}$$

$$\tilde{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \text{ - Candidate Hidden State} \tag{22}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \text{ - Final Hidden State,} \tag{23}$$

where x_t and h_t represent the input and output vectors, respectively, $\tilde{h}_t$ is the candidate activation vector, z_t is the update gate vector, r_t is the reset gate vector, and the operator $\odot$ represents the element-wise multiplication (Hadamard product). W, U, and b

represent the parameter matrices and vectors. Because it has a more simplified structure relative to LSTM, and takes fewer parameters, GRU is typically easier to train, as well as faster and more computationally efficient than LSTM. It also tends to perform better than LSTM networks on low-complexity sequences [39]. GRUs are often used to predict nonlinear, non-smooth, and strongly fluctuating time series data [40].

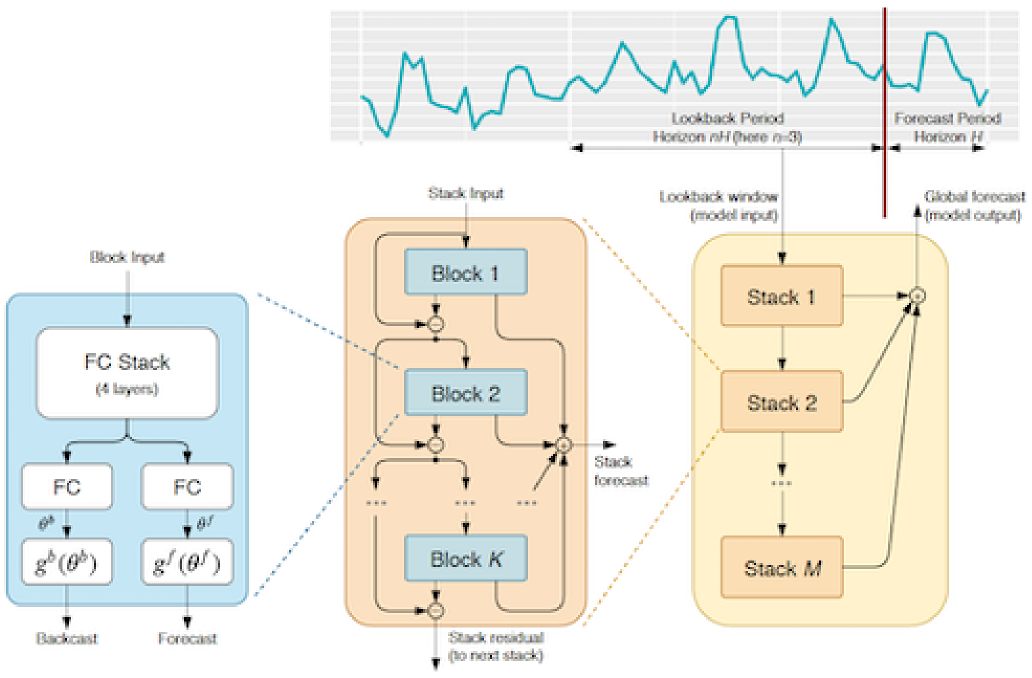
4.1.4.3 Neural Basis Expansion Analysis for Time Series Forecasting (N-BEATS) is a neural network that was designed based on the key principles that 1) the base architecture should be simple and generic, while still expressive (i.e. deep); 2) it should not rely on time-series-specific feature engineering or input scaling; and 3) it should be extendable for the purposes of making its outputs human-interpretable [14]. As shown in Figure 6, the basic building block of the architecture is a multi-layer fully-connected (FC) network with nonlinearities that predicts basis expansion coefficients both forward, θ^f, and backward, θ^b. Using a doubly residual stacking principle, blocks are organized into stacks that may have layers with shared g^b and g^f. The ℓ-th block accepts an input $\mathbf{x}_\ell$, and subsequently outputs two vectors, $\hat{\mathbf{x}}_\ell$, which is the block's best estimate of $\mathbf{x}_\ell$ (also known as the "backcast"), and $\hat{\mathbf{y}}_\ell$, which is the block's forward forecast. The first block in the model has a history lookback window of a given length ending with the last measured observation as its $\mathbf{x}_\ell$ input, while the remaining blocks have inputs $\mathbf{x}_\ell$ that are residual outputs of the preceding blocks. The following equations describe the operation of first part of the ℓ-th block [14]:

$$\mathbf{h}_{\ell,1} = FC_{\ell,1}(\mathbf{x}_\ell), \quad \mathbf{h}_{\ell,2} = FC_{\ell,2}(\mathbf{h}_{\ell,1}), \quad \mathbf{h}_{\ell,3} = FC_{\ell,3}(\mathbf{h}_{\ell,2}), \quad \mathbf{h}_{\ell,4} = FC_{\ell,4}(\mathbf{h}_{\ell,3}) \tag{24}$$

$$\theta_\ell^b = LINEAR_\ell^b(\mathbf{h}_{\ell,4}), \quad \theta_\ell^f = LINEAR_\ell^f(\mathbf{h}_{\ell,4}), \tag{25}$$

where the LINEAR layer is a linear projection layer, and the FC layer is a standard

Figure 6: *N-BEATS Architecture*



fully-connected layer with RELU non-linearity. The second part of the network's operation consists of mapping expansion coefficients θ_ℓ^f and θ_ℓ^b to outputs through basis layers, $\hat{\mathbf{y}}_\ell = g_\ell^f(\theta_\ell^f)$ and $\hat{\mathbf{x}}_\ell = g_\ell^b(\theta_\ell^b)$, and it can be described by the following equations:

$$\hat{\mathbf{y}}_\ell = \sum_{i=1}^{dim(\theta_\ell^f)} \theta_{\ell,i}^f \mathbf{v}_i^f, \quad \hat{\mathbf{x}}_\ell = \sum_{i=1}^{dim(\theta_\ell^b)} \theta_{\ell,i}^b \mathbf{v}_i^b, \tag{26}$$

where $\mathbf{v}_i^f$ and $\mathbf{v}_i^b$ are forecast and backcast basis vectors, and $\theta_{\ell,i}^f$ is the i-th element of θ_ℓ^f.

The N-BEATS architecture has two residual branches, one which runs over the backcast prediction, with the other running over the forecast branch of each layer; this operation can be described by:

$$\mathbf{x}_\ell = \mathbf{x}_{\ell-1} - \hat{\mathbf{x}}_{\ell-1}, \quad \hat{\mathbf{y}} = \sum_\ell \hat{\mathbf{y}}_\ell. \tag{27}$$

It also provides for two possible configurations. In the case of the generic architecture that does not rely on time-series-specific knowledge, g_ℓ^b and g_ℓ^f are a linear projection of the previous layer's output, such that the outputs of block ℓ are:

$$\hat{\mathbf{y}}_\ell = \mathbf{V}^f_\ell \theta^f_\ell + \mathbf{b}^f_\ell, \quad \hat{\mathbf{x}}_\ell = \mathbf{v}^b_\ell \theta^b_\ell + \mathbf{b}^b_\ell \tag{28}$$

Here, $\hat{\mathbf{y}}_\ell$ is not interpretable because no additional constraints are imposed on the form of $\mathbf{V}^f_\ell$, as a consequence of which the waveforms learned by the deep model do not have inherent structure. The interpretable architecture, on the other hand, adds structure to basis layers at the stack level, and was designed with trend and seasonality decomposition. To mimic the behaviour of a slowly varying or monotonic function for the trend model, $g^b_{s,\ell}$ and $g^f_{s,\ell}$ are constrained to be a polynomial of small degree p, a function that varies slowly across the forecast window:

$$\hat{\mathbf{y}}_{s,\ell} = \sum_{i=0}^{p} \theta^f_{s,\ell,i} \boldsymbol{t}^i, \tag{29}$$

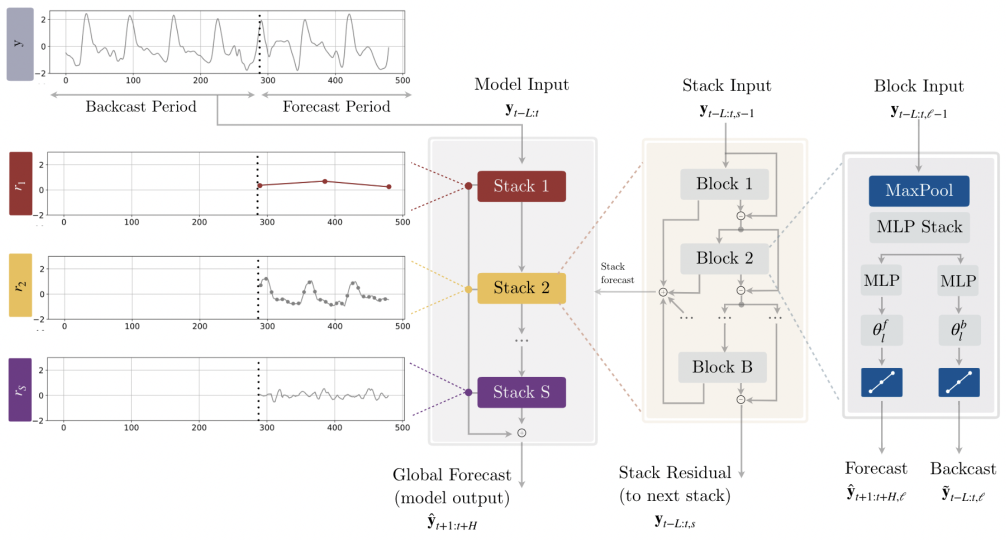
where the time vector $\mathbf{t} = [0, 1, 2, ..., H-2, H-1]^T/H$ is defined on a discrete grid running from 0 to $(H$ - 1$)/H$, forecasting H steps ahead. In order to model the regular, cyclical, recurring fluctuation that is typically characteristic of seasonality, $g^b_{s,\ell}$ and $g^f_{s,\ell}$ are constrained to belong to the class of periodic functions, and the Fourier series is used to model the periodic function:

$$\hat{\mathbf{y}}_{s,\ell} = \sum_{i=0}^{[H/2-1]} \theta^f_{s,\ell,i} cos(2\pi i t) + \theta^f_{s,\ell,i+[H/2]} sin(2\pi i t) \tag{30}$$

The interpretable architecture is made up of two stacks: the trend stack, followed by the seasonality stack. The trend component is removed from the input window $\mathbf{x}$ prior to being fed into the seasonality stack, and the partial forecasts of trend and seasonality are available as separate interpretable outputs.

4.1.4.4 Neural Hierarchical Interpolation for Time Series Forecasting (N-HiTS) is an extension of the Neural Basis Expansion Analysis approach that addresses the long-horizon forecasting challenges of prediction volatility and computational complexity by incorporating novel hierarchical interpolation and multi-rate sampling techniques which enable it to assemble predictions sequentially, emphasizing components with different frequencies and scales while decomposing the input signal and synthesizing the forecast. The focus of this section is on the novel components of the model's architecture, and additional details can be found in [18].

Figure 7: *N-HiTS Architecture*



A MaxPool layer with kernel size k_ℓ is used at each block ℓ's input to enable its focus on analyzing components of its input with a specific scale. *Multi-rate signal sampling* ensures that the multilayer perceptron (MLP) in each block faces a different effective input signal sampling rate, and the the blocks with larger pooling kernel size k_ℓ can therefore focus on analyzing the large-scale components that play a critical role in the production of consistent long-horizon forecasts. Furthermore, multi-rate processing limits the memory footprint and the amount of required computation, as well as reduces the number of learnable parameters by reducing the width of the MLP input for most blocks; this has the benefit of alleviating the effects of overfitting, while maintaining the original receptive field. Given block ℓ input $\mathbf{y}_{t-L:t,\ell}$, the operation can be represented as:

$$\mathbf{y}^{(p)}_{t-L:t,\ell} = MaxPool(\mathbf{y}_{t-L:t,\ell}, k_\ell) \tag{31}$$

After the subsampling process, block ℓ observes its input and nonlinearly regresses forward θ^f_ℓ and backward θ^b_ℓ interpolation MLP coefficients that learn the hidden vector $\mathbf{h}_\ell \in \mathbb{R}^{N_h}$, which is subsequently projected linearly:

$$\mathbf{h}_\ell = \mathbf{MLP}_\ell \left(\mathbf{y}^{(p)}_{t-L:t,\ell}\right) \tag{32}$$

$$\theta^f_\ell = \mathbf{LINEAR}^f(\mathbf{h}_\ell) \tag{33}$$

$$\theta^b_\ell = \mathbf{LINEAR}^b(\mathbf{h}_\ell) \tag{34}$$

Following this operation, the coefficients are used to synthesize backcast $\tilde{\mathbf{y}}_{t-L:t,\ell}$ and forecast $\hat{\mathbf{y}}_{t+1:t+H,\ell}$ outputs of the block through the process of *hierarchical interpolation*.

In order to address the issues of compute requirement inflation and the unnecessary explosion in model expressiveness that happen as horizon H increases in most multi-horizon forecasting models, temporal interpolation is used to recover the original sampling rate, and predict all H points in the horizon via the interpolation function g:

$$\hat{y}_{\tau,\ell} = g(\tau, \theta^f_\ell), \;\; \forall \tau \in \{t+1, ..., t+H\}, \tag{35}$$

$$\tilde{y}_{\tau,\ell} = g(\tau, \theta^b_\ell), \;\; \forall \tau \in \{t-L, ..., t\}. \tag{36}$$

The definition of the linear interpolator $g \in \mathcal{C}^1$, along with the time partition $\mathcal{T} = \{t+$

$1, t+1+1/r_\ell, ..., t+H-1/r_\ell, t+H\}$ is as follows:

$$g(\tau,\theta) = \theta[t_1] + \left(\frac{\theta[t_2]-\theta[t_1]}{t_2-t_1}\right)(\tau - t_1) \tag{37}$$

$$t_1 = arg \min_{t\in\mathcal{T}:t\leq\tau} \tau - t, \;\; t_2 = t_1 + 1/r_\ell \tag{38}$$

Expressiveness ratios are distributed across blocks in a synchronized manner with multi-rate sampling. The blocks closer to the input have smaller r_ℓ and larger k_ℓ, and the sum of the outputs of all blocks is used to assemble the resulting hierarchical forecast, $\hat{\mathbf{y}}_{t+1:t+H}$. This forecast is essentially composed out of interpolations at different time-scale hierarchy levels. The backcast residual that is formed at the previous hierarchy scale is subsequently subtracted from the input of the next hierarchy level, in order to ensure that the next level block's focus is primarily on signals outside of the band already handled by previous hierarchy members.

$$\hat{\mathbf{y}}_{t+1:t+H} = \sum_{l=1}^{L} \hat{\mathbf{y}}_{t+1:t+H,\ell} \tag{39}$$

$$\mathbf{y}_{t-L:t,\ell+1} = \mathbf{y}_{t-L:t,\ell} - \tilde{\mathbf{y}}_{t-L:t,\ell} \tag{40}$$

4.2 Evaluation Metrics

4.2.1 Mean Absolute Error (MAE)

MAE represents the average of the absolute differences between predicted values and actual observations:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{41}$$

4.2.2 Root Mean Squared Error (RMSE)

Mean Squared Error (MSE) squares the difference between actual and predicted values, which emphasizes larger errors. This can be considered appropriate for contexts such as solar prediction, in which larger errors lead to disproportionately higher costs [41]. RMSE is the result of taking the squared root of the MSE, and it is frequently used to compare the forecasting errors of different models. Lower RMSE values indicate better predictive capability in terms of a model's absolute deviation [42].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{42}$$

In (Equation 41) and (Equation 42), n represents the number of sample data points, y_i is the actual observed value for the i-th data point, and $\hat{y}_i$ is the predicted value for the i-th data point.

4.3 Implementation

4.3.1 Darts Machine Learning Library

Darts [30] is a time series forecasting-focused Python machine learning library that offers modern ML functionalities through a high-level API that was designed to be as easy to use as classical techniques, with the goal of further democratizing modern ML forecasting approaches, and promoting large-scale adoption by practitioners. Available models range from classic statistical ones such as ARIMA, to state-of-the-art deep neural networks like N-BEATS. Some of its key features are:

- An immutable TimeSeries data container type that guarantees the data's represention as a well-formed time series with the correct shape, type, and sorted time index. The TimeSeries class provides several methods to convert to/from common types such as Pandas Dataframes and Numpy arrays, and it can perform operations such as math, indexing, splitting, and time-differencing.

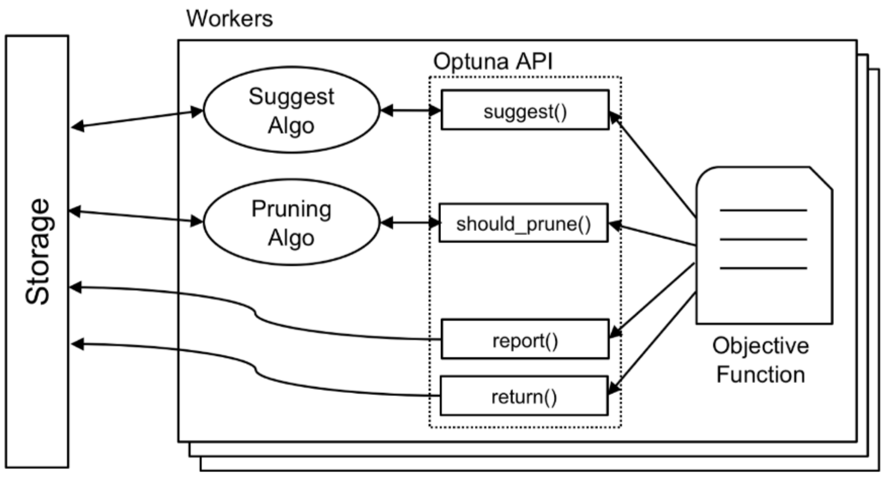
- The same basic fit() and predict() interface for all models. The neural network models are implemented using PyTorch, and there is support for training and inference on GPUs.
- Support for training an individual model on a large number of separate series, as well as user-defined/customizable slicing of training samples as necessary.
- Covariate series supported by a number of models as a way to specify external data that can be useful for forecasting the target series. Future covariates are known into the future (e.g. weather forecasts), while past covariates are only known in the past. The covariate series do not need to be aligned with the target, as Darts performs all required alignment using slicing logic based on the respective time axes.
- Transformers and pipelines for data processing, backtesting, hyperparameter search, and extensive metrics.

Although models such as N-BEATS and N-HiTS were originally created as univariate, their implementation in Darts was designed to accommodate datasets comprised of multiple time series.

4.3.2 Optuna

Optuna's next-generation hyperparameter optimization process [25] involves minimizing/maximizing an objective function that accepts a set of hyperparameters as an input and returns a validation score. The function constructs the search space in a dynamic manner that does not rely on externally defined static variables. Each optimization process is referred to as a study, and each objective function evaluation is considered a trial. Optuna has the ability to identify trial results that are informative

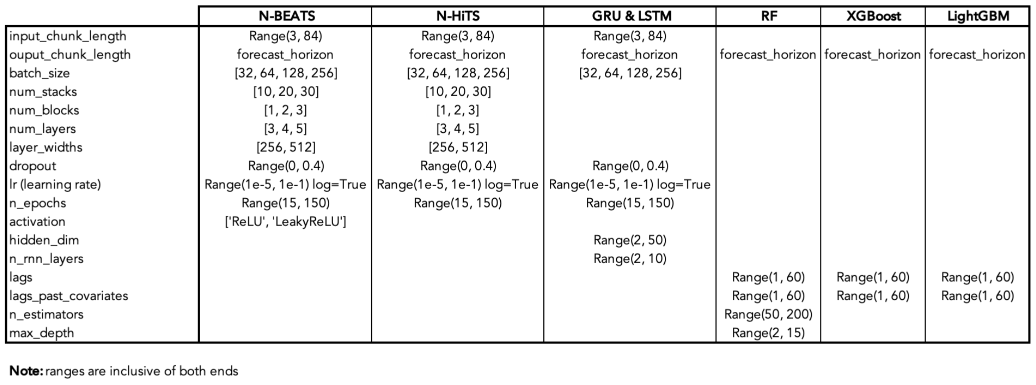
Figure 8: *Optuna System Design Overview*



about the concurrence relations among the hyperparameters. It also features a variant of a recently developed state of the art method, Asynchronous Successive Halving, in which each worker in a distributed environment is allowed to asynchronously execute aggressive early stopping based on provisional trial ranking. Each step of the iterative training yields a reported intermediate value, and The Pruner class stops unpromising trials based on the history of reported values.

Details of the hyperparameter search options used for the ML models can be found in Figure 9.

Figure 9: *Hyperparameter Search Space*



	N-BEATS	N-HiTS	GRU & LSTM	RF	XGBoost	LightGBM
input_chunk_length	Range(3, 84)	Range(3, 84)	Range(3, 84)			
ouput_chunk_length	forecast_horizon	forecast_horizon	forecast_horizon	forecast_horizon	forecast_horizon	forecast_horizon
batch_size	[32, 64, 128, 256]	[32, 64, 128, 256]	[32, 64, 128, 256]			
num_stacks	[10, 20, 30]	[10, 20, 30]				
num_blocks	[1, 2, 3]	[1, 2, 3]				
num_layers	[3, 4, 5]	[3, 4, 5]				
layer_widths	[256, 512]	[256, 512]				
dropout	Range(0, 0.4)	Range(0, 0.4)	Range(0, 0.4)			
lr (learning rate)	Range(1e-5, 1e-1) log=True	Range(1e-5, 1e-1) log=True	Range(1e-5, 1e-1) log=True			
n_epochs	Range(15, 150)	Range(15, 150)	Range(15, 150)			
activation	['ReLU', 'LeakyReLU']					
hidden_dim			Range(2, 50)			
n_rnn_layers			Range(2, 10)			
lags				Range(1, 60)	Range(1, 60)	Range(1, 60)
lags_past_covariates				Range(1, 60)	Range(1, 60)	Range(1, 60)
n_estimators				Range(50, 200)		
max_depth				Range(2, 15)		

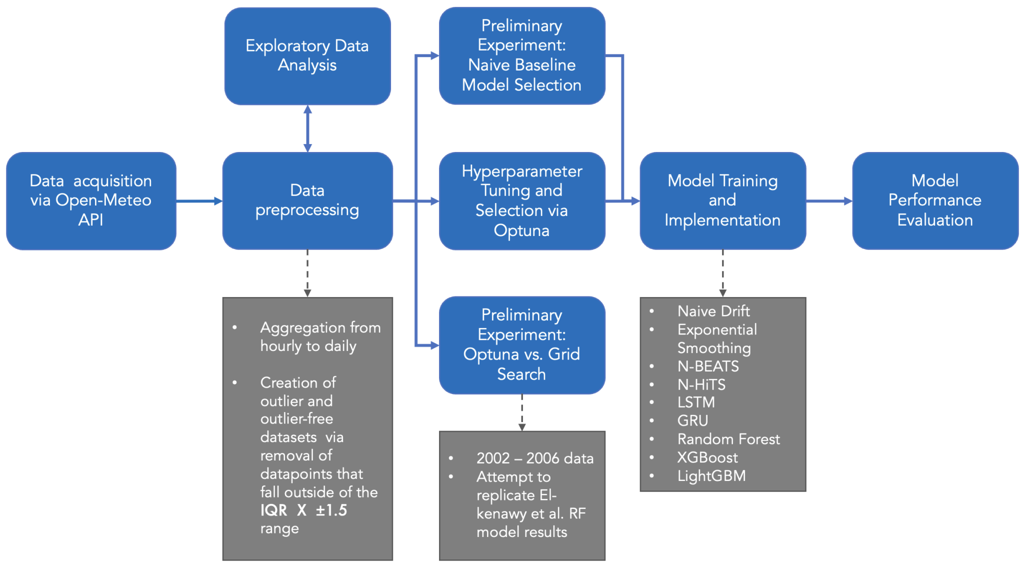
Note: ranges are inclusive of both ends

4.3.3 Computing Details

The experiments were conducted using Google Colab, which provides an online Jupyter notebook environment with a variety of free and paid runtime options, including central processing unit (CPU), graphics processing units (GPUs) and tensor processing units (TPUs). An NVIDIA Tesla T4 GPU was used as an accelerator for the deep learning models.

4.3.4 Process

Figure 10: *Research Process*



Exploratory Data Analysis
Preliminary Experiment: Naive Baseline Model Selection
Data acquisition via Open-Meteo API
Data preprocessing
Hyperparameter Tuning and Selection via Optuna
Model Training and Implementation
Model Performance Evaluation
Aggregation from hourly to daily
Creation of outlier and outlier-free datasets via removal of datapoints that fall outside of the IQR X ±1.5 range
Preliminary Experiment: Optuna vs. Grid Search
2002 – 2006 data
Attempt to replicate El-kenawy et al. RF model results
Naive Drift
Exponential Smoothing
N-BEATS
N-HiTS
LSTM
GRU
Random Forest
XGBoost
LightGBM

The overall process for this study is illustrated in Figure 10. In this section, the focus is on the steps that follow data pre-processing and exploratory data analysis.

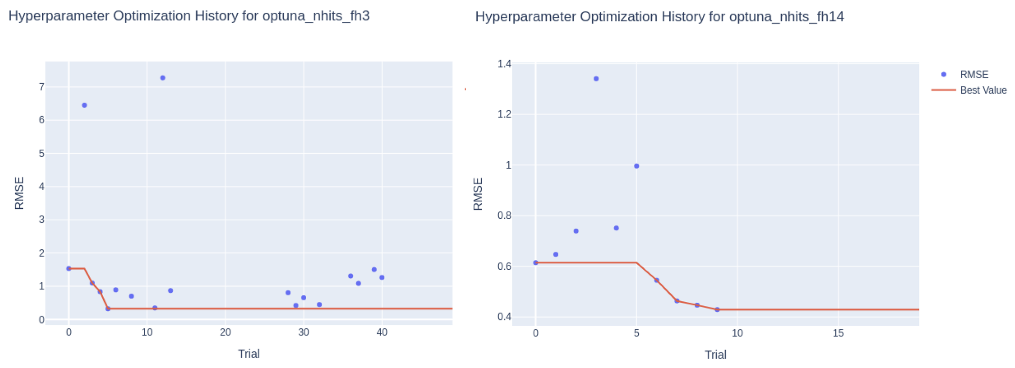
4.3.4.1 Preliminary Experiments In order to ensure proper performance contextualization for both the novel and traditional machine learning models, a preliminary experiment was conducted to select the best performing naive model to serve as a baseline benchmark. A random cut-off date, 2023-08-24, distinct from the final testing cut-off date was selected from the test year for this purpose, and the mean and median RMSE and MAE over the forecast horizons (FH) of 1, 3, 7, 14, and 28 days were computed and compared. A second preliminary experiment was conducted to validate Optuna's performance in two ways:

1. By comparing its results to a subset of [29]'s experiments where they assessed the performance of four regressors - MLP, K-nearest neighbors, support vector machine, and random forest - over four studied locations, including BBM. An RF model was built for this purpose using hyperparameters obtained from Optuna, and the assessment was carried out on the equivalent 2002-2006 subset of the BBM data, with the goal of demonstrating that Optuna could yield comparable or better results. Consistent with [29], no additional pre-processing was carried out on the data except for all the fields including the target being normalized using min-max scaling with a range of [0,1]. Additionally, the reported error metrics were normalized, or dimensionless, values. However, the current study's handling of the 60/20/20 training/validation/test split diverged from the original study's, where the split was done on a random basis. Such randomization is not recommended for time series data, as it does not preserve temporal order, and it carries the risk of data leakage. Instead, the splits were based on the sequential cutoffs that would yield the appropriate dataset lengths.

2. Using the same hyperparameter search space, a Grid Search was conducted for the optimal hyperparameters, and both the error scores obtained and computation time required were compared to Optuna's. The expectation here as well was comparable or better results from the latter for the error metrics, as well as significantly less computation time than GS.

4.3.4.2 Main Experiment A hyperparameter search was conducted for all of the machine learning models using, in turn, the training dataset containing outliers, and the one without them. The covariate training data was normalized to a range of

Figure 11: *Sample Optuna Trials for N-HiTS*



[0,1] for all but the N-BEATS model using min-max scaling, in order to test the latter's creators' assertion that such a step would not be required. The number of trials was originally set to 50 across the board; however, it was decreased to 20 for N-BEATS and N-HiTS, given the significant amount of computation time they required relative to the other models, and the fact that an initial assessment of trial data indicated that there were diminishing to no returns beyond or even before that point (See Figure 11). Once the optimal hyperparameters had been determined, two cut-off dates in the test year of 2023 were randomly selected - one for a different month from that which was used to tune the hyperparameters (2023-04-09), and one from the same month (2023-08-19). The respective models were then trained on data up until the cutoff points. Training and testing was carried out across each of the five forecast horizons for both the dataset with and the one without outliers. With the exception of the naive and statistical benchmark models, a default and a tuned version of each model was trained. The main experiment consisted of three conditions:

1. Training and testing of the data with outliers included based on hyperparameters determined using the outlier-included training set;

2. Training and testing of the outlier-free data based on hyperparameters determined using the outlier-included training set;

3. Training and testing of the outlier-free data based on hyperparameters determined

using the outlier-free training set.

These conditions will hereafter be referred to as HP-Outlier-Outlier or HPOO, HP-Outlier-Clean or HPOC, and HP-Clean-Clean or HPCC. The HPOC condition is intended to gauge how well the models adapt to the absence of outliers, in essence checking to what extent the hyperparameter selections were tied to their presence. Both this and the use of one cutoff date with the same month as validation and the other with a different month constitute a rudimentary assessment of generalizability. The HPCC condition, meanwhile, is to assess "best-case" performance, where there are no extreme values to sway the models in any given direction.

Number of epochs was included as a hyperparameter for Optuna to suggest for the deep learning models, and two versions of tuned models were trained in the HPOO and HPOC conditions - one with the default epoch number of 100, and one with the number of suggested epochs. The final model variant was selected based on whichever yielded the best results. Default model variants were appended with "-D," Tuned ones with "-T," and the Generic and Interpretable variants of N-BEATS were appended with "-G," and "-I," respectively. For the purposes of overall standing, the parent model performance was assessed across a total of 50 cases. These cases represent 3 Experiment Conditions (EC) X 5 Forecast Horizons (FH) X 2 Error metrics (EM) X 2 months - (1 EC X 5 FH X 2 EM X 1 month), with the subtraction being due to the HPCC condition only involving tests using a single cutoff date.

5

Results and Discussion

5.1 Preliminary Experiments

5.1.1 Optuna vs. Grid Search

Following testing of the optimal hyperparameters determined by the two search methods, RMSE and MAE scores of 0.0962 and 0.0377, respectively, were observed for Optuna, with Grid Search yielding fairly similar scores of 0.0970 and 0.0333. Both RMSEs represent a roughly 55% improvement over [29]'s RMSE of 0.2164, and a 77-80% improvement relative to their 0.1633 MAE. Although the methods produced comparable results, there was a drastic difference observed in computation time: while Optuna concluded its computations over the extensive search space within a mere 13.6 minutes, it took Grid Search 5.8 hours to do the same. In other words, not only was Optuna able to significantly improve upon the original study, but it was able to do so over 25 times faster than its counterpart. Of note is the fact that these results were for a locally run version of the experiment. A Colab-based test using a CPU runtime resulted in 19.9 minute total search time for Optuna vs. 837.5 minutes (14 hours) for GS, a staggering 42 times difference in speed.

5.1.2 Naive Baseline Search

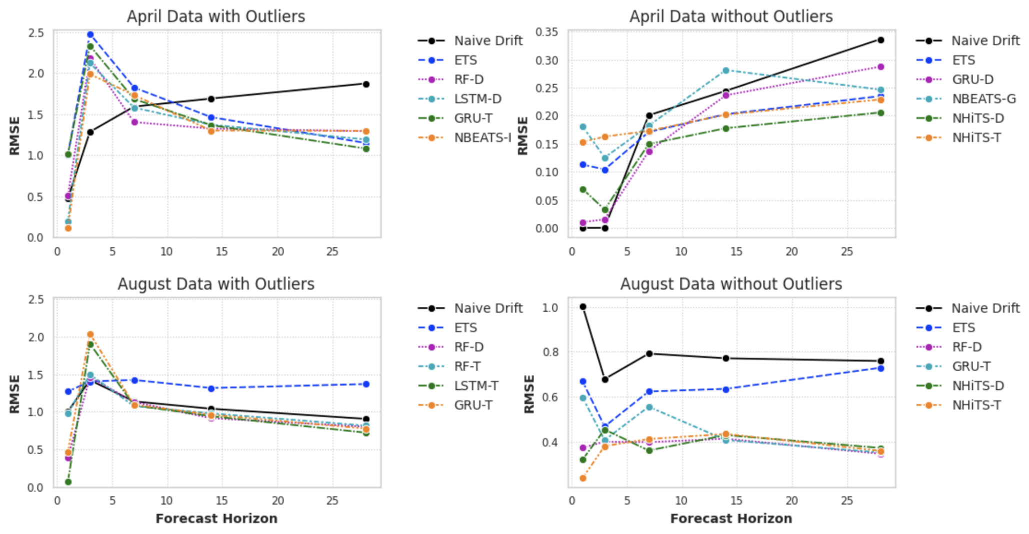
After a random training cutoff date of 2023-08-24 was selected, an assessment of Darts' four naive models was completed based on mean and median RMSE and MAE statistics across all forecast horizons, and for both the clean and outlier datasets. The results were generally consistent, with Naive Drift displaying the best performance in all cases, and Naive Seasonal, the worst. The former was therefore selected as the Naive benchmark model.

5.2 Main Experiments

Although the test datasets did not neatly fall into any given month, the two month-based sub-conditions are referred to as April and August for convenience, based on the randomly selected testing cutoff dates of 2023-04-09 and 2023-08-19. N-HiTS and N-BEATS had the best overall performance in August, appearing at the top rank just under a third, and a quarter of the time, respectively. N-BEATS' performance within the two months was the rough inverse of Naive Drift's. Across April tests, the latter was the top ranking model 47% of the time, while N-BEATS appeared the most at the bottom rank, at 30%. Conversely, Naive Drift was in last place half of the time in August. Given N-HiTS' design geared towards better performance at longer time frames, its observed results were not particularly noteworthy over the longest forecast horizon. In fact, LSTM fared better at FH-28, topping the ranks twice as often as N-HiTS. Interestingly, the latter had a better outcome with FH-1 (an example of which can be seen in Figure 12), where it was tied with Naive Drift for top performance. The only condition under which N-HiTS displayed the best performance at FH-28 was the HPOC one, where it also generally performed better than the other models. Despite not having its training data scaled, the Interpretable version of N-BEATS displayed the best performance with respect to its median RMSE of 1.300 and median MAE of 1.035 across all five forecast horizons in the HPOO Condition for April, topping the ranks at FH-14 in both error tables (Table 1, Table 5), and its median MAE of 0.626 in the HPOO Condition for August, as seen in Table 7. However, the tuned N-BEATS models had the worst performing median RMSE and MAE of 0.238 and 0.231, respectively, in the April HPCC condition (Table 9, Table 10), and worst median RMSE of 0.260 in the HPOC condition (Table 2).

Naive Drift dominated in the HPOC and HPCC April conditions with its me-

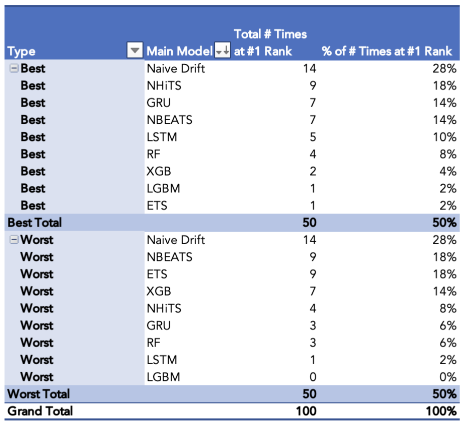
Figure 12: *Top vs. Benchmark Models based on Median RMSE - HPO Condition*



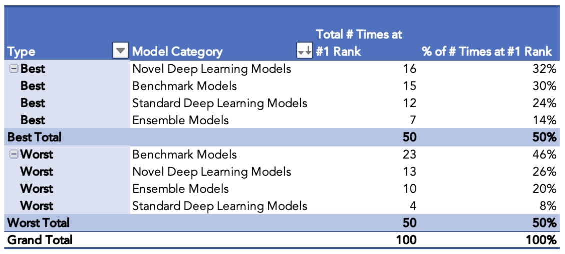
dian MAE of 0.100, besting the other models in all but the longest forecast horizon of 28 days. Exponential smoothing similarly performed well with the clean April dataset in the HPCC condition, with a median RMSE of 0.171. However, both of these models fared poorly in the August HPOC and HPOO conditions, with Naive Drift being the worst across each individual forecast horizon with respect to RMSE and MAE for the former condition (Table 8, Table 4, and Exponential Smoothing displaying the worst performance for all but the 3-day forecast horizon for both aggregated metrics in the latter condition. Their prominence in the bottom ranks in August is mostly due to the fact that the ML models, especially N-HiTS and N-BEATS, yielded much better results when faced with test data based on a cutoff date with the same month as the validation cutoff-off date. Naive Drift and ETS jointly accounted for a whopping 90% of bottom rank performance in August, with only XGB and GRU otherwise making a single appearance each. In April, meanwhile, NBEATS and N-HiTS jointly accounted for 43% of the bottom rankings, and the ML models collectively accounted for 83%. Overall, Naive Drift was simultaneously the single best and worst performing model, appearing at both the top and bottom ranks in equal measure, as seen in Figure 13. N-HiTS was the next-best performing model, while N-BEATS was tied with ETS for second-worst.

As a duo, the novel deep learning models narrowly outperformed the benchmark models.

Figure 13: *Global Model Performance*



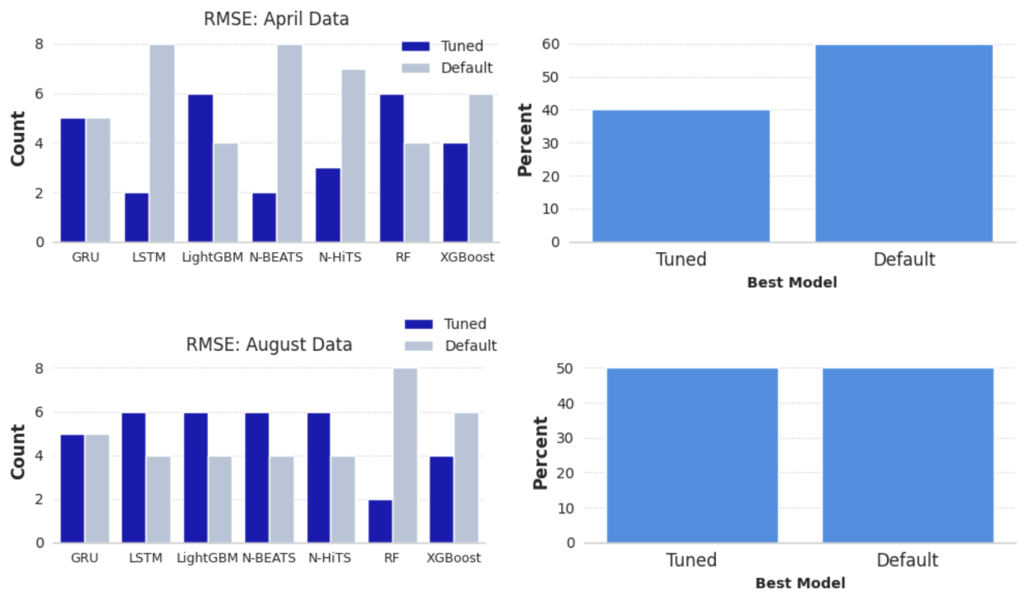
Type	Main Model	Total # Times at #1 Rank	% of # Times at #1 Rank
⊟Best	Naive Drift	14	28%
Best	NHiTS	9	18%
Best	GRU	7	14%
Best	NBEATS	7	14%
Best	LSTM	5	10%
Best	RF	4	8%
Best	XGB	2	4%
Best	LGBM	1	2%
Best	ETS	1	2%
Best Total		**50**	**50%**
⊟Worst	Naive Drift	14	28%
Worst	NBEATS	9	18%
Worst	ETS	9	18%
Worst	XGB	7	14%
Worst	NHiTS	4	8%
Worst	GRU	3	6%
Worst	RF	3	6%
Worst	LSTM	1	2%
Worst	LGBM	0	0%
Worst Total		**50**	**50%**
Grand Total		**100**	**100%**



Type	Model Category	Total # Times at #1 Rank	% of # Times at #1 Rank
⊟Best	Novel Deep Learning Models	16	32%
Best	Benchmark Models	15	30%
Best	Standard Deep Learning Models	12	24%
Best	Ensemble Models	7	14%
Best Total		**50**	**50%**
⊟Worst	Benchmark Models	23	46%
Worst	Novel Deep Learning Models	13	26%
Worst	Ensemble Models	10	20%
Worst	Standard Deep Learning Models	4	8%
Worst Total		**50**	**50%**
Grand Total		**100**	**100%**

How the tuned models performed relative to their default counterparts depended on which month their testing cutoff date corresponded to. In the HP-Outlier Condition, the default model variants produced lower RMSEs 20% more frequently than the tuned models, as seen in Figure 14. However, where the testing cutoff month was the same as the validation cutoff month (i.e. August), the tuned and default models bested each other at an identical rate. A good example of where the hyperparameter tuning had a big impact is in Table 7, where NHiTS-T was the top model at FH-1 with an MAE of 0.018, whereas its default counterpart had an MAE of 0.336. NHiTS-D ultimately had a slightly lower median MAE than NHiTS-T, however. An overall assessment of the total time it took the models to fit all the decades of training data up until the test cutoff-dates, relative to median RMSE performance across all cases is shown in Figure 15. The default model of NHiTS is 6.5 times faster than the generic version of NBEATS, taking 84.5% less time to train, with a 20% improvement in RMSE score. The Interpretable version of NBEATS yields almost identical RMSE results, while being 13.2% faster. In practical terms, however, they are equivalent. The tuned versions of the respective models are more on par in terms of computation time, but NHiTS-T's RMSE

Figure 14: *HP-Outlier Condition - Tuned vs. Default Model Performance*



is still 28% lower.

Figure 15: *RMSE vs. Training Time*

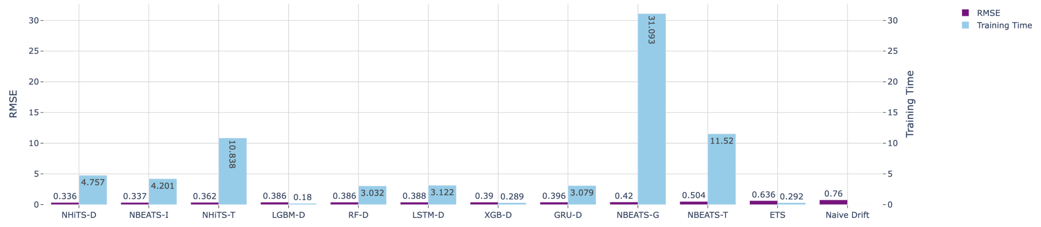


Table 1: *RMSE Summary Statistics - HP-Outlier Condition - April Outlier Dataset*

ModelName	FH-1	FH-3	FH-7	FH-14	FH-28	Median	Mean
NBEATS-I	0.112	1.987	1.731	**1.300**	1.296	**1.300**	1.288
RF-D	0.504	2.180	**1.403**	1.324	1.292	1.324	1.338
GRU-T	1.010	2.333	1.684	1.366	**1.081**	1.366	1.474
LSTM-D	0.198	2.122	1.579	1.371	1.192	1.371	1.306
RF-T	**0.014**	1.872	1.547	1.375	1.150	1.375	**1.222**
GRU-D	0.176	2.236	1.546	1.378	1.148	1.378	1.310
LGBM-D	0.248	1.847	1.637	1.391	1.246	1.391	1.293
XGB-D	0.092	1.454	1.697	1.510	1.220	1.454	1.238
LGBM-T	0.482	1.641	1.520	1.456	1.125	1.456	1.280
ETS	0.998	2.477	1.824	1.463	1.149	1.463	1.562
LSTM-T	0.910	2.466	**1.826**	1.479	1.262	1.479	1.570
NHiTS-T	0.479	**2.487**	1.649	1.509	1.200	1.509	1.472
NBEATS-T	0.592	2.326	1.786	1.559	1.427	1.559	1.542
XGB-T	**1.471**	2.219	1.581	1.715	1.311	1.581	**1.646**
Naive Drift	0.473	**1.284**	1.591	1.689	**1.872**	1.591	1.417
NBEATS-G	0.076	2.179	1.809	1.594	1.277	1.594	1.421
NHiTS-D	0.252	2.295	1.724	**1.725**	1.203	**1.724**	1.487

Table 2: *RMSE Summary Statistics - HP-Outlier Condition - April Clean Dataset*

ModelName	FH-1	FH-3	FH-7	FH-14	FH-28	Median	Mean
GRU-D	0.011	0.015	**0.137**	0.236	0.287	**0.137**	0.137
NHiTS-D	0.070	0.033	0.149	**0.178**	**0.206**	0.149	**0.131**
ETS	0.113	0.104	0.171	0.203	0.235	0.171	0.166
NHiTS-T	0.153	0.163	0.173	0.202	0.229	0.173	0.182
NBEATS-G	0.181	0.125	0.182	**0.281**	0.246	0.182	0.200
LGBM-T	0.049	0.154	0.182	0.242	0.242	0.182	0.175
GRU-T	0.005	0.128	0.183	0.217	0.228	0.183	0.157
RF-T	0.144	0.019	0.199	0.239	0.280	0.199	0.180
Naive Drift	**0.000**	**0.000**	0.201	0.244	0.336	0.201	0.164
LSTM-D	0.014	0.046	0.202	0.258	0.242	0.202	0.161
LSTM-T	0.198	0.133	0.204	0.207	0.236	0.204	0.197
RF-D	0.087	0.163	0.222	0.245	0.233	0.222	0.195
NBEATS-I	0.224	0.154	0.209	0.247	0.322	0.224	0.230
LGBM-D	0.136	0.228	0.179	0.236	0.246	0.228	0.209
XGB-T	0.103	0.244	0.174	0.268	0.257	0.244	0.215
XGB-D	0.029	**0.287**	**0.235**	0.246	0.263	0.246	0.218
NBEATS-T	**0.500**	0.121	0.231	0.260	**0.693**	**0.260**	**0.344**

Table 3: *MAE Summary Statistics - HP-Outlier Condition - April Clean Dataset*

ModelName	**FH-1**	**FH-3**	**FH-7**	**FH-14**	**FH-28**	**Median**	**Mean**
Naive Drift	**0.000**	**0.000**	**0.100**	**0.141**	0.243	**0.100**	**0.098**
NHiTS-D	0.070	0.027	0.109	0.144	**0.174**	0.109	0.105
GRU-T	0.005	0.115	0.110	0.187	0.198	0.115	0.122
GRU-D	0.011	0.014	0.119	0.206	0.229	0.119	0.116
ETS	0.113	0.104	0.132	0.175	0.212	0.132	0.145
NBEATS-I	0.224	0.153	0.133	0.161	0.244	0.161	0.179
LGBM-T	0.049	0.154	0.162	0.216	0.177	0.162	0.153
NHiTS-T	0.153	0.162	0.103	0.175	0.201	0.162	0.159
LSTM-D	0.014	0.043	0.181	0.231	0.185	0.181	0.139
NBEATS-G	0.181	0.124	0.151	0.221	0.219	0.181	0.180
RF-T	0.144	0.016	0.183	0.216	0.211	0.183	0.159
LSTM-T	0.198	0.131	0.186	0.177	0.189	0.186	0.178
NBEATS-T	**0.500**	0.111	0.179	0.190	**0.659**	0.190	**0.305**
LGBM-D	0.136	0.217	0.164	0.201	0.217	0.201	0.189
RF-D	0.087	0.158	**0.218**	0.227	0.203	0.203	0.183
XGB-D	0.029	**0.277**	0.215	0.204	0.217	0.215	0.193
XGB-T	0.103	0.233	0.155	**0.242**	0.218	**0.218**	0.195

Table 4: *MAE Summary Statistics - HP-Outlier Condition - August Clean Dataset*

ModelName	FH-1	FH-3	FH-7	FH-14	FH-28	Median	Mean
NHiTS-T	**0.240**	**0.258**	0.367	0.396	0.318	**0.318**	**0.316**
NBEATS-T	0.346	0.399	**0.315**	**0.325**	0.574	0.346	0.384
NHiTS-D	0.323	0.373	0.352	0.383	0.298	0.352	0.347
GRU-T	0.598	0.358	0.508	0.361	0.286	0.361	0.412
RF-D	0.375	0.344	0.373	0.383	0.287	0.373	0.356
LSTM-T	0.455	0.532	0.352	0.374	0.292	0.374	0.397
LSTM-D	0.470	0.375	0.423	0.356	**0.273**	0.375	0.379
LGBM-T	0.423	0.375	0.364	0.381	0.324	0.375	0.374
NBEATS-G	0.390	0.412	0.371	0.377	0.298	0.377	0.371
NBEATS-I	0.306	0.408	0.384	0.382	0.302	0.382	0.361
GRU-D	0.396	0.407	0.389	0.382	0.330	0.389	0.382
XGB-D	0.380	0.328	0.409	0.413	0.395	0.395	0.387
LGBM-D	0.396	0.430	0.462	0.401	0.331	0.401	0.403
RF-T	0.637	0.384	0.420	0.417	0.287	0.417	0.427
XGB-T	0.591	0.517	0.463	0.487	0.385	0.487	0.488
ETS	0.670	0.440	0.551	0.573	0.660	0.573	0.578
Naive Drift	**1.004**	**0.547**	**0.679**	**0.658**	**0.682**	**0.679**	**0.708**

6

Conclusions and Future Work

6.1 Summary and Conclusions

This study encompassed the first attempt to apply the state-of-the-art deep learning models N-BEATS and N-HiTS towards the task of sunshine duration forecasting. The Optuna hyperparameter optimization framework and Darts machine learning library were used for automated hyperparameter tuning and model implementation, respectively. Preliminary experiments confirmed Optuna's significantly superior efficiency relative to Grid Search, as well as its ability to surpass results in an existing work, and identified Naive Drift as a top candidate for naive benchmark model. The latter was paired with Exponential Smoothing, which served as the statistical benchmark model. With the inclusion of these comparators, and assessments across various forecast horizons, the current research addressed limitations of existing literature pertaining to the lack of simple benchmark comparisons in studies on the accuracy of machine learning models, as well as an oft-narrow focus on short-term forecasting horizons such as one-step ahead. Testing was done using default and tuned variants of a variety of ensemble and deep learning models, in addition to the baseline ones. Model performance was assessed across various dimensions, namely top-line conditions such as Hyperparameter-Outlier and Hyperparameter-Clean, five forecast horizons, two distinct testing cutoff dates, and error metrics RMSE and MAE.

N-BEATS and N-HiTS showed strong promise in this area of research, with a few surprises. They performed best in the test month corresponding to the validation month, despite the presence of outliers. In fact, the models did not fare particularly well in the HPCC condition, where there were no outliers present at any point in the process. On the contrary, N-BEATS' performance actually suffered, suggesting that it may require a greater degree of complexity in the data. For its part, N-HiTS did not exhibit stand-out performance over the longest range forecast horizon, instead performing well at FH-1. Overall, N-HiTS appeared to be the better suited for the task of SD forecasting for BBM. The two models ultimately topped the rankings more than the other model categories; however, the fact that they narrowly beat out the benchmarks in that regard underscores the importance of the inclusion of these simpler models to ground any conclusions drawn with regards to the level of advancement and performance of the more sophisticated models. This will ensure that practitioners do not waste valuable effort, time, and computing resources, when simpler solutions could provide equivalent or better results, with the added benefit of producing them near-instantaneously. At the very least, they would be empowered to properly quantify key trade-offs.

Finally, in comparing tuned model performance to default models', the former were found to be better than the latter at most half of the time. It is therefore not a foregone conclusion that hyperparameter tuning will yield superior results to default settings. An area of improvement in the current experimental design would be the withdrawal of number of epochs as a tuned hyperparameter, as it significantly interfered with Optuna's ability to take advantage of the very things that drive its efficiency by allowing it to engage in aggressive early stopping of unsuccessful trials, instead forcing them to continue on to completion despite their futility.

6.2 Future Work

Whereas this study focused on a combination of fields that proved effective in existing literature, future studies can include and assess additional features such as cloud cover, wind speed, and precipitation, as well as time-series components that could prove beneficial in terms of increased signal and information for the models. And while the current research did not take advantage of N-BEATS-I's interpretability benefits, it could be used to highlight the most important of those features, and contribute towards feature engineering efforts.

Additionally, it would be worth seeking out SD datasets with a greater level of complexity that is not necessarily the result of an abundance of outliers, be they legitimate data points or otherwise. Furthermore, the study design can be expanded to assess performance throughout all the different months and seasons of a given test year, to better assess the strengths and weaknesses of the models under consideration.

Although Darts was user-friendly and effective as advertised, another possible area of research would be to carry out this same or a similar study using one or more of the established machine libraries, in order to compare their results. For example, the decision tree models have fewer hyperparameter options available for tuning in Darts, as compared to Scikit-learn. It is possible that the additional available settings for XGBoost in the latter would have at least mitigated some of the poor performance exhibited by the tuned variants relative to their default counterparts.

A

Remaining Error Tables

Table 5: *Hyperparam-Outlier Condition - Outlier Dataset: April MAE*

ModelName	**FH-1**	**FH-3**	**FH-7**	**FH-14**	**FH-28**	**Median**	**Mean**
NBEATS-I	0.112	1.707	1.462	**1.035**	0.953	**1.035**	1.051
GRU-T	1.010	2.138	1.450	1.104	0.837	1.104	1.274
NBEATS-G	0.076	1.937	1.485	1.126	0.894	1.126	1.107
LGBM-T	0.482	1.342	1.365	1.135	0.937	1.135	1.066
ETS	0.998	2.291	**1.559**	1.139	0.907	1.139	1.339
GRU-D	0.176	1.925	**1.323**	1.146	0.979	1.146	1.116
LSTM-D	0.198	1.812	1.351	1.147	**0.799**	1.147	1.076
LGBM-D	0.248	1.493	1.447	1.172	1.029	1.172	1.094
LSTM-T	0.910	2.259	1.504	1.185	0.824	1.185	1.311
RF-D	0.504	2.125	1.338	1.220	1.144	1.220	1.258
XGB-D	0.092	1.385	1.382	1.229	0.971	1.229	**1.048**
NHiTS-T	0.479	**2.296**	1.436	1.236	0.858	1.236	1.257
RF-T	**0.014**	1.596	1.388	1.243	0.967	1.243	1.075
NBEATS-T	0.592	2.090	1.439	1.290	1.270	1.290	1.329
Naive Drift	0.473	**1.155**	1.421	**1.560**	**1.756**	1.421	1.298
XGB-T	**1.471**	1.924	1.356	1.445	1.077	1.445	**1.453**
NHiTS-D	0.252	2.092	1.458	1.459	0.922	**1.458**	1.273

Table 6: *Hyperparam-Outlier Condition - Outlier Dataset: August RMSE*

ModelName	**FH-1**	**FH-3**	**FH-7**	**FH-14**	**FH-28**	**Median**	**Mean**
RF-D	0.387	1.462	1.131	**0.914**	0.802	**0.914**	0.935
LSTM-T	0.065	1.908	1.086	0.942	**0.723**	0.942	0.944
GRU-T	0.467	**2.041**	1.095	0.961	0.777	0.961	1.050
RF-T	0.976	1.498	1.077	0.979	0.816	0.979	1.054
NHiTS-T	**0.018**	1.479	1.089	0.988	0.787	0.988	**0.891**
GRU-D	0.543	1.446	**1.002**	1.001	0.764	1.001	0.959
LGBM-D	0.229	1.409	1.203	1.036	0.852	1.036	0.961
Naive Drift	1.004	1.422	1.139	1.041	0.906	1.041	1.092
LSTM-D	0.388	1.598	1.048	1.056	0.738	1.048	0.979
NHiTS-D	0.336	1.421	1.089	1.050	0.780	1.050	0.954
XGB-T	1.032	1.774	1.290	1.026	1.063	1.063	1.208
NBEATS-G	0.807	1.495	1.205	1.070	0.764	1.070	1.069
NBEATS-I	0.223	1.952	1.157	1.077	0.736	1.077	1.037
LGBM-T	0.903	1.575	1.104	1.086	0.841	1.086	1.099
NBEATS-T	0.621	1.676	1.162	0.986	1.204	1.162	1.135
XGB-D	1.170	**1.354**	1.406	1.152	1.005	1.170	1.210
ETS	**1.271**	1.405	**1.425**	**1.318**	**1.370**	**1.370**	**1.360**

Table 7: *Hyperparam-Outlier Condition - Outlier Dataset: August MAE*

ModelName	**FH-1**	**FH-3**	**FH-7**	**FH-14**	**FH-28**	**Median**	**Mean**
NBEATS-I	0.223	1.416	0.718	**0.626**	**0.431**	**0.626**	0.673
GRU-D	0.543	1.293	**0.661**	0.703	0.600	0.661	0.743
NHiTS-D	0.336	1.091	0.792	0.664	0.475	0.664	**0.670**
LSTM-D	0.388	1.061	0.762	0.698	0.551	0.698	0.693
NHiTS-T	**0.018**	1.271	0.848	0.703	0.498	0.703	0.674
LSTM-T	0.065	1.364	0.704	0.742	0.491	0.704	0.678
GRU-T	0.467	1.446	0.880	0.713	0.576	0.713	0.799
NBEATS-G	0.807	1.007	0.868	0.784	0.530	0.807	0.801
RF-D	0.387	1.143	0.886	0.814	0.636	0.814	0.780
RF-T	0.976	1.155	0.790	0.832	0.617	0.832	0.867
LGBM-D	0.229	1.170	0.907	0.835	0.696	0.835	0.779
LGBM-T	0.903	1.150	0.744	0.858	0.650	0.858	0.861
NBEATS-T	0.621	1.098	0.919	0.753	1.153	0.919	0.910
XGB-D	1.170	**0.947**	1.083	0.926	0.788	0.947	0.977
Naive Drift	1.004	1.261	0.986	0.953	0.830	0.986	1.003
XGB-T	1.032	**1.537**	1.046	0.731	0.821	1.032	1.033
ETS	**1.271**	1.354	**1.355**	**1.246**	**1.312**	**1.312**	**1.308**

Table 8: *Hyperparam-Outlier Condition - Clean Dataset: August RMSE*

ModelName	**FH-1**	**FH-3**	**FH-7**	**FH-14**	**FH-28**	**Median**	**Mean**
NHiTS-D	0.323	0.456	0.363	0.431	0.375	**0.375**	0.387
NHiTS-T	**0.240**	**0.383**	0.414	0.436	0.362	0.383	**0.370**
RF-D	0.375	0.402	0.400	0.415	0.349	0.400	0.390
GRU-T	0.598	0.409	0.557	**0.409**	0.354	0.409	0.456
LGBM-T	0.423	0.433	0.403	0.417	0.363	0.417	0.409
NBEATS-I	0.306	0.434	0.420	0.420	0.352	0.420	0.392
LSTM-T	0.455	0.646	0.411	0.422	0.344	0.422	0.450
GRU-D	0.396	0.470	0.422	0.425	0.369	0.422	0.417
NBEATS-G	0.390	0.534	0.423	0.489	0.420	0.423	0.447
NBEATS-T	0.346	0.504	**0.345**	0.423	0.625	0.423	0.444
LGBM-D	0.396	0.495	0.509	0.440	0.386	0.440	0.444
RF-T	0.637	0.425	0.447	0.442	0.347	0.442	0.457
XGB-D	0.380	0.390	0.449	0.470	0.465	0.449	0.434
LSTM-D	0.470	0.451	0.454	0.423	**0.341**	0.451	0.432
XGB-T	0.591	0.535	0.500	0.520	0.447	0.520	0.519
ETS	0.670	0.469	0.625	0.636	0.730	0.636	0.628
Naive Drift	**1.004**	**0.680**	**0.792**	**0.772**	**0.760**	**0.772**	**0.796**

Table 9: *Hyperparam-Clean Condition - Clean Dataset: April RMSE*

ModelName	**FH-1**	**FH-3**	**FH-7**	**FH-14**	**FH-28**	**Median**	**Mean**
LSTM-D	0.028	0.057	**0.133**	0.249	0.235	**0.133**	**0.139**
ETS	0.113	0.104	0.171	**0.203**	0.235	0.171	0.166
Naive Drift	**0.000**	**0.000**	0.201	0.244	**0.336**	0.201	0.164
GRU-D	0.023	0.022	0.205	0.270	**0.219**	0.205	0.157
NHiTS-T	0.088	0.056	0.205	0.233	0.225	0.205	0.169
NBEATS-G	0.182	0.195	0.207	0.254	0.268	0.207	0.219
LSTM-T	0.009	0.339	0.191	0.219	0.223	0.219	0.200
RF-D	0.107	0.169	0.235	0.245	0.222	0.222	0.200
LGBM-T	0.131	0.103	0.231	0.223	0.241	0.223	0.192
XGB-T	0.226	0.150	0.154	0.236	0.246	0.226	0.206
LGBM-D	0.136	0.228	0.179	0.236	0.246	0.228	0.209
GRU-T	0.180	**0.642**	0.177	0.231	0.251	0.231	0.285
RF-T	0.191	0.253	**0.291**	0.229	0.233	0.233	0.238
NBEATS-I	**0.337**	0.191	0.244	0.213	0.329	0.244	0.260
XGB-D	0.029	0.287	0.235	0.246	0.263	0.246	0.218
NHiTS-D	0.009	0.305	0.249	0.254	0.320	0.254	0.232
NBEATS-T	0.148	0.344	0.265	**0.457**	0.297	**0.297**	**0.301**

Table 10: *Hyperparam-Clean Condition - Clean Dataset: April MAE*

ModelName	FH-1	FH-3	FH-7	FH-14	FH-28	Median	Mean
Naive Drift	**0.000**	**0.000**	**0.100**	**0.141**	0.243	**0.100**	**0.098**
LSTM-D	0.028	0.057	0.106	0.219	0.209	0.106	0.121
GRU-D	0.023	0.021	0.108	0.232	0.201	0.108	0.116
ETS	0.113	0.104	0.132	0.175	0.212	0.132	0.145
LGBM-T	0.131	0.099	0.211	0.199	**0.178**	0.178	0.166
NHiTS-T	0.088	0.056	0.194	0.182	0.190	0.182	0.149
GRU-T	0.180	**0.639**	0.171	0.186	0.218	0.186	**0.263**
LSTM-T	0.009	0.338	0.186	0.168	0.207	0.186	0.182
NBEATS-I	**0.337**	0.181	0.182	0.188	0.250	0.188	0.221
NBEATS-G	0.182	0.192	0.138	0.215	0.206	0.192	0.188
RF-D	0.107	0.167	0.228	0.223	0.196	0.196	0.186
XGB-T	0.226	0.147	0.143	0.201	0.214	0.201	0.189
LGBM-D	0.136	0.217	0.164	0.201	0.217	0.201	0.189
RF-T	0.191	0.253	**0.281**	0.208	0.202	0.208	0.224
XGB-D	0.029	0.277	0.215	0.204	0.217	0.215	0.193
NHiTS-D	0.009	0.303	0.240	0.212	**0.263**	0.240	0.211
NBEATS-T	0.148	0.332	0.180	**0.400**	0.254	**0.254**	0.262

References

[1] H. M. Kandirmaz, K. Kaba, and M. Avci, “Estimation of Monthly Sunshine Duration in Turkey Using Artificial Neural Networks,” en, *International Journal of Photoenergy*, vol. 2014, pp. 1–9, 2014, ISSN: 1110-662X, 1687-529X. DOI: 10.1155/2014/680596. [Online]. Available: http://www.hindawi.com/journals/ijp/2014/680596/ (visited on 05/25/2024).

[2] J. A. Jervase, A. Al-Lawati, and A. S. S. Dorvlo, “Contour maps for sunshine ratio for Oman using radial basis function generated data,” en, *Renewable Energy*, 2003.

[3] S. Arslan, “A hybrid forecasting model using LSTM and Prophet for energy consumption with decomposition of time series data,” en, *PeerJ Computer Science*, vol. 8, e1001, Jun. 2022, ISSN: 2376-5992. DOI: 10.7717/peerj-cs.1001. [Online]. Available: https://peerj.com/articles/cs-1001 (visited on 06/07/2024).

[4] A. Guenoupkati, A. A. Salami, M. K. Kodjo, and K. Napo, “Short-Term Electricity Generation Forecasting Using Machine Learning Algorithms: A Case Study of the Benin Electricity Community (C.E.B),” en, *TH Wildau Engineering and Natural Sciences Proceedings*, vol. 1, Jun. 2021, ISSN: 2748-8829. DOI: 10.52825/thwildauensp.v1i.25. [Online]. Available: https://www.tib-op.org/ojs/index.php/th-wildau-ensp/article/view/25 (visited on 07/07/2024).

[5] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and Machine Learning forecasting methods: Concerns and ways forward,” en, *PLOS ONE*, vol. 13, no. 3, A. R. Hernandez Montoya, Ed., e0194889, Mar. 2018, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0194889. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0194889 (visited on 10/02/2024).

[6] D. Matuszko and S. Węglarczyk, “Relationship between sunshine duration and air temperature and contemporary global warming: RELATIONSHIP BETWEEN SUNSHINE DURATION AND AIR TEMPERATURE,” en, *International Journal of Climatology*, vol. 35, no. 12, pp. 3640–3653, Oct. 2015, ISSN: 08998418. DOI: 10.1002/joc.4238. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/joc.4238 (visited on 05/24/2024).

[7] J.-L. Chen, G.-S. Li, and S.-J. Wu, “Assessing the potential of support vector machine for estimating daily solar radiation using sunshine duration,” en, *Energy Conversion and Management*, vol. 75, pp. 311–318, Nov. 2013, ISSN: 01968904. DOI: 10.1016/j.enconman.2013.06.034. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0196890413003464 (visited on 05/21/2024).

[8] W. Wu, X.-P. Tang, C. Yang, N.-J. Guo, and H.-B. Liu, “Spatial estimation of monthly mean daily sunshine hours and solar radiation across mainland China,” en, *Renewable Energy*, vol. 57, pp. 546–553, Sep. 2013, ISSN: 09601481. DOI: 10.1016/j.renene.2013.02.027. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0960148113001365 (visited on 05/21/2024).

[9] K. Kaba, H. M. Kandirmaz, and M. Avci, “Estimation of daily sunshine duration using support vector machines,” en, *International Journal of Green Energy*, vol. 14, no. 4, pp. 430–441, Mar. 2017, ISSN: 1543-5075, 1543-5083. DOI: 10.1080/15435075.2016.1265971. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/15435075.2016.1265971 (visited on 06/11/2024).

[10] M. Laidi, S. Hanini, and A. El Hadj Abdallah, “Novel Approach for Estimating Monthly Sunshine Duration Using Artificial Neural Networks: A Case Study,” en, *Journal of Sustainable Development of Energy, Water and Environment Systems*, vol. 6, no. 3, pp. 405–414, Sep. 2018, ISSN: 18489257. DOI: 10.13044/j.sdewes.d6.0226. [Online]. Available: http://www.sdewes.org/jsdewes/pid6.0226 (visited on 05/26/2024).

[11] M. T. Zateroglu, “Statistical Models For Sunshine Duration Related To Precipitation And Relative Humidity,” en, *European Journal of Science and Technology*, Dec. 2021, ISSN: 2148-2683. DOI: 10.31590/ejosat.1022962. [Online]. Available: https://dergipark.org.tr/en/doi/10.31590/ejosat.1022962 (visited on 05/25/2024).

[12] M. T. Zateroglu, “Estimating the sunshine duration using multiple linear regression in Kocaeli, Turkey,” en, *Időjárás*, vol. 127, no. 3, pp. 285–298, 2023, ISSN: 03246329, 2677187X. DOI: 10.28974/idojaras.2023.3.2. [Online]. Available: https://www.met.hu/ismeret-tar/kiadvanyok/idojaras/index.php?no=2023.3.2 (visited on 05/25/2024).

[13] X. Zhu, X. Qiu, Y. Zeng, J. Gao, and Y. He, “A remote sensing model to estimate sunshine duration in the Ningxia Hui Autonomous Region, China,” en, *Journal of Meteorological Research*, vol. 29, no. 1, pp. 144–154, Feb. 2015, ISSN: 2095-6037, 2198-0934. DOI: 10.1007/s13351-015-4059-1. [Online]. Available: http://link.springer.com/10.1007/s13351-015-4059-1 (visited on 05/26/2024).

[14] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, *N-BEATS: Neural basis expansion analysis for interpretable time series forecasting*, en, arXiv:1905.10437 [cs, stat], Feb. 2020. [Online]. Available: http://arxiv.org/abs/1905.10437 (visited on 09/10/2024).

[15] F. Kamalov, H. Sulieman, S. Moussa, J. Avante Reyes, and M. Safaraliev, “Powering Electricity Forecasting with Transfer Learning,” en, *Energies*, vol. 17, no. 3,

p. 626, Jan. 2024, ISSN: 1996-1073. DOI: 10.3390/en17030626. [Online]. Available: https://www.mdpi.com/1996-1073/17/3/626 (visited on 10/21/2024).

[16] J. Zhihui, J. Xun, C. Yin, Z. Guo, and G. Tian, "Interpretable Forecasting of Traction Energy Consumption Based on Nbeats and Temporal Fusion Transformers," en, in *2024 IEEE 13th Data Driven Control and Learning Systems Conference (DDCLS)*, Kaifeng, China: IEEE, May 2024, pp. 1836–1843, ISBN: 9798350361674. DOI: 10.1109/DDCLS61622.2024.10606795. [Online]. Available: https://ieeexplore.ieee.org/document/10606795/ (visited on 08/30/2024).

[17] S. Huang, D. Zhang, T. Zheng, G. Tong, J. Xu, and F. Jia, "Studies of short-term load forecasting model based on LSTM-NBEATS," en, in *2022 China Automation Congress (CAC)*, Xiamen, China: IEEE, Nov. 2022, pp. 3284–3288, ISBN: 978-1-66546-533-5. DOI: 10.1109/CAC57257.2022.10055018. [Online]. Available: https://ieeexplore.ieee.org/document/10055018/ (visited on 10/21/2024).

[18] C. Challu, K. G. Olivares, B. N. Oreshkin, F. Garza, M. Mergenthaler-Canseco, and A. Dubrawski, *N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting*, en, arXiv:2201.12886 [cs], Nov. 2022. [Online]. Available: http://arxiv.org/abs/2201.12886 (visited on 09/02/2024).

[19] H. Liang, W. Yu, C. Qian, D. W. Griffith, and N. Golmie, "Assessing Deep Learning Performance in Power Demand Forecasting for Smart Grid," en, *International Journal of Sensor Networks*, vol. 44, no. 1, p. 10 061 818, 2024, ISSN: 1748-1279, 1748-1287. DOI: 10.1504/IJSNET.2024.10061818. [Online]. Available: http://www.inderscience.com/link.php?id=10061818 (visited on 09/02/2024).

[20] R. Magallanes-Quintanar, C. E. Galván-Tejada, J. I. Galván-Tejada, H. Gamboa-Rosales, S. D. J. Méndez-Gallegos, and A. García-Domínguez, "Neural Hierarchical Interpolation for Standardized Precipitation Index Forecasting," en, *Atmosphere*, vol. 15, no. 8, p. 912, Jul. 2024, ISSN: 2073-4433. DOI: 10.3390/atmos15080912. [Online]. Available: https://www.mdpi.com/2073-4433/15/8/912 (visited on 11/24/2024).

[21] F. M. A. Mazen, Y. Shaker, and R. A. Abul Seoud, "Forecasting of Solar Power Using GRU–Temporal Fusion Transformer Model and DILATE Loss Function," en, *Energies*, vol. 16, no. 24, p. 8105, Dec. 2023, ISSN: 1996-1073. DOI: 10.3390/en16248105. [Online]. Available: https://www.mdpi.com/1996-1073/16/24/8105 (visited on 05/26/2024).

[22] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, "Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis," en, *Informatics*, vol. 8, no. 4, p. 79, Nov. 2021, ISSN: 2227-9709. DOI: 10.3390/informatics8040079. [Online]. Available: https://www.mdpi.com/2227-9709/8/4/79 (visited on 10/06/2024).

[23] S. Hanifi, S. Lotfian, H. Zare-Behtash, and A. Cammarano, “Offshore Wind Power Forecasting—A New Hyperparameter Optimisation Algorithm for Deep Learning Models,” en, *Energies*, vol. 15, no. 19, p. 6919, Sep. 2022, ISSN: 1996-1073. DOI: 10.3390/en15196919. [Online]. Available: https://www.mdpi.com/1996-1073/15/19/6919 (visited on 10/21/2024).

[24] S. J. Quan, “Comparing hyperparameter tuning methods in machine learning based urban building energy modeling: A study in Chicago,” en, *Energy and Buildings*, vol. 317, p. 114 353, Aug. 2024, ISSN: 03787788. DOI: 10.1016/j.enbuild.2024.114353. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378778824004699 (visited on 10/06/2024).

[25] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, *Optuna: A Next-generation Hyperparameter Optimization Framework*, en, arXiv:1907.10902 [cs, stat], Jul. 2019. [Online]. Available: http://arxiv.org/abs/1907.10902 (visited on 10/21/2024).

[26] M. Hassanali, M. Soltanaghaei, T. Javdani Gandomani, and F. Zamani Boroujeni, “Software development effort estimation using boosting algorithms and automatic tuning of hyperparameters with Optuna,” en, *Journal of Software: Evolution and Process*, vol. 36, no. 9, e2665, Sep. 2024, ISSN: 2047-7473, 2047-7481. DOI: 10.1002/smr.2665. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/smr.2665 (visited on 10/20/2024).

[27] E. C. Da Silva, E. C. Finardi, and S. F. Stefenon, “Enhancing hydroelectric inflow prediction in the Brazilian power system: A comparative analysis of machine learning models and hyperparameter optimization for decision support,” en, *Electric Power Systems Research*, vol. 230, p. 110 275, May 2024, ISSN: 03787796. DOI: 10.1016/j.epsr.2024.110275. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378779624001639 (visited on 10/21/2024).

[28] P. Zippenfenig, *Open-meteo.com weather api*, 2023. DOI: 10.5281/zenodo.7970649. [Online]. Available: https://open-meteo.com/.

[29] E.-S. M. El-kenawy *et al.*, “Sunshine duration measurements and predictions in Saharan Algeria region: An improved ensemble learning approach,” en, *Theoretical and Applied Climatology*, vol. 147, no. 3-4, pp. 1015–1031, Feb. 2022, ISSN: 0177-798X, 1434-4483. DOI: 10.1007/s00704-021-03843-2. [Online]. Available: https://link.springer.com/10.1007/s00704-021-03843-2 (visited on 06/02/2024).

[30] J. Herzen *et al.*, “Darts: User-friendly modern machine learning for time series,” *Journal of Machine Learning Research*, vol. 23, no. 124, pp. 1–6, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-1177.html.

[31] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, English, 2nd. Australia: OTexts, 2018.

[32] R. Alsharif, M. Arashpour, E. Golafshani, M. Bazli, and S. R. Mohandes, "Ensemble machine learning framework for daylight modelling of various building layouts," en, *Building Simulation*, vol. 16, no. 11, pp. 2049–2061, Nov. 2023, ISSN: 1996-3599, 1996-8744. DOI: 10.1007/s12273-023-1045-x. [Online]. Available: https://link.springer.com/10.1007/s12273-023-1045-x (visited on 09/09/2024).

[33] P. Mishra *et al.*, "Modeling and forecasting rainfall patterns in India: A time series analysis with XGBoost algorithm," en, *Environmental Earth Sciences*, vol. 83, no. 6, p. 163, Mar. 2024, ISSN: 1866-6280, 1866-6299. DOI: 10.1007/s12665-024-11481-w. [Online]. Available: https://link.springer.com/10.1007/s12665-024-11481-w (visited on 07/06/2024).

[34] O. Bamisile, C. J. Ejiyi, E. Osei-Mensah, I. A. Chikwendu, J. Li, and Q. Huang, "Long-Term Prediction of Solar Radiation Using XGboost, LSTM, and Machine Learning Algorithms," en, in *2022 4th Asia Energy and Electrical Engineering Symposium (AEEES)*, Chengdu, China: IEEE, Mar. 2022, pp. 214–218, ISBN: 978-1-66547-914-1. DOI: 10.1109/AEEES54426.2022.9759719. [Online]. Available: https://ieeexplore.ieee.org/document/9759719/ (visited on 07/06/2024).

[35] A. Rangga, D. S. Sihabudin Sahid, and Y. Dewi Lulu Widyasari, "A Comparative Analysis of Multivariate and Univariate Time-Series Forecasting for Oil Production," en, in *2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, Kuala Lumpur, Malaysia: IEEE, Dec. 2023, pp. 1–6, ISBN: 9798350315516. DOI: 10.1109/ICRAIE59459.2023.10468530. [Online]. Available: https://ieeexplore.ieee.org/document/10468530/ (visited on 09/08/2024).

[36] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," en, in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, FL: IEEE, Dec. 2018, pp. 1394–1401, ISBN: 978-1-5386-6805-4. DOI: 10.1109/ICMLA.2018.00227. [Online]. Available: https://ieeexplore.ieee.org/document/8614252/ (visited on 07/06/2024).

[37] G. Feng, L. Zhang, F. Ai, Y. Zhang, and Y. Hou, "An Improved Temporal Fusion Transformers Model for Predicting Supply Air Temperature in High-Speed Railway Carriages," en, *Entropy*, vol. 24, no. 8, p. 1111, Aug. 2022, ISSN: 1099-4300. DOI: 10.3390/e24081111. [Online]. Available: https://www.mdpi.com/1099-4300/24/8/1111 (visited on 06/09/2024).

[38] A. A. Pierre, S. A. Akim, A. K. Semenyo, and B. Babiga, "Peak Electrical Energy Consumption Prediction by ARIMA, LSTM, GRU, ARIMA-LSTM and ARIMA-GRU Approaches," en, *Energies*, vol. 16, no. 12, p. 4739, Jun. 2023, ISSN: 1996-1073. DOI: 10.3390/en16124739. [Online]. Available: https://www.mdpi.com/1996-1073/16/12/4739 (visited on 06/21/2024).

[39] Y. Si, S. Nadarajah, Z. Zhang, and C. Xu, "Modeling opening price spread of Shanghai Composite Index based on ARIMA-GRU/LSTM hybrid model," en, *PLOS ONE*, vol. 19, no. 3, M. U. Tariq, Ed., e0299164, Mar. 2024, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0299164. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0299164 (visited on 06/08/2024).

[40] S. Zhou, C. Song, J. Zhang, W. Chang, W. Hou, and L. Yang, "A Hybrid Prediction Framework for Water Quality with Integrated W-ARIMA-GRU and LightGBM Methods," en, *Water*, vol. 14, no. 9, p. 1322, Apr. 2022, ISSN: 2073-4441. DOI: 10.3390/w14091322. [Online]. Available: https://www.mdpi.com/2073-4441/14/9/1322 (visited on 06/08/2024).

[41] P. Bendiek, A. Taha, Q. H. Abbasi, and B. Barakat, "Solar Irradiance Forecasting Using a Data-Driven Algorithm and Contextual Optimisation," en, *Applied Sciences*, vol. 12, no. 1, p. 134, Dec. 2021, ISSN: 2076-3417. DOI: 10.3390/app12010134. [Online]. Available: https://www.mdpi.com/2076-3417/12/1/134 (visited on 06/07/2024).

[42] M. Despotovic, V. Nedic, D. Despotovic, and S. Cvetanovic, "Review and statistical analysis of different global solar radiation sunshine models," en, *Renewable and Sustainable Energy Reviews*, vol. 52, pp. 1869–1880, Dec. 2015, ISSN: 13640321. DOI: 10.1016/j.rser.2015.08.035. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1364032115008953 (visited on 05/20/2024).