Technical Report: Day-Ahead Solar GHI Forecasting Using XGBoost

TL;DR

Objective: Build a machine learning model to forecast day-ahead solar irradiance (GHI) using historical data, with a strong focus on handling zero-GHI periods realistically.

Approach: I cleaned noisy, spike-heavy data using exponential smoothing and outlier removal. Wind direction was circularly encoded, and I used log transformation on GHI to improve model stability. I feature engineered for time-awareness, and filled in missing values with KNN imputation.

I trained an XGBoost Regressor on log-transformed GHI values. I then applied inverse transformation on predictions, and post-processed them by setting GHI to 0 when irradiance was too low (i.e. night time) to avoid unrealistic values.

Evaluation: Used a custom MAPE formula to avoid division-by-zero issues. The final MAPE was **20.89%**, strong performance during daylight and realistic zero predictions at night.

Key Insight: Applying EMA, log1p transformation, and irradiance filtering significantly improved model performance. This solution blends physical understanding with data-driven ML techniques and the entire pipeline is fully reproducible.

The objective of this project was to build a machine learning model capable of forecasting the next day's Global Horizontal Irradiance (GHI) using historical weather and environmental data. The evaluation metric provided was the Mean Absolute Percentage Error (MAPE), with explicit instructions to handle edge cases where the actual GHI is zero, a natural challenge in solar prediction, especially around sunrise, sunset, and night hours.

One of the first things I noticed, the dataset exhibited expected cyclic patterns driven by solar exposure, strong peaks during daylight and flatlines at night. However, the raw data was noisy, riddled with outliers, and showed inconsistencies across features. GHI values especially had strong spikes and abrupt drops, which could mislead the model and inflate error metrics if not handled carefully. It was evident that the model, if fed with such erratic behaviour, would struggle to distinguish natural daytime variations from noise.

So I started with cleaning and preprocessing the data. I began by dropping irrelevant or redundant columns such as (Unnamed:0) and (timestamp) especially because the datetime was already being converted and utilised in the form of time-based features. Columns with more than 30% missing values were also dropped, as their imputation would add more risk than value. I encoded wind direction using a circular transformation ($\cos\theta + \sin\theta$) because directions are cyclical (0° and 360° are practically the same), and I didn't want the model to treat them like they're far apart. I also dropped any rows where GHI was missing, since that's what I'm trying to predict.

To smoothen the noise (spikes) in the continuous variables, I applied an Exponential Moving Average (EMA) with a span of 24. EMA helps the model focus on recent trends while smoothing over erratic shifts. This turned out to be a key which significantly stabilised the model's learning curve and performance. I then removed the top 1% of GHI values as outliers. These spikes, while sometimes valid often created extreme loss values which did not represent typical solar patterns.

Since GHI was heavily right-skewed, I applied a log1p transformation to stabilise the variance and make the distribution more normal-like. That helps models like XGBoost perform better. The feature (hour) was extracted from the datetime index to capture the daily solar cycle this added crucial time-awareness to the model. Lastly, I used KNN imputation (with 5 neighbours) to fill in any remaining missing values in a way that was feature-aware and non-parametric.

For the model itself, I selected an XGBoost Regressor. XGBoost is well-known for handling tabular data efficiently and excels at capturing non-linear interactions, which are inevitable in solar irradiance forecasting. I used the following parameters:

```
model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=5, random_state=42)
```

After training the model on log-transformed (ghi) I performed an inverse transformation (np.expm1) on the predictions to get back actual GHI values. One challenge I kept facing was how misleading the error looked during night time. The model might predict a tiny GHI value, but if the actual was 0, the percentage error would explode. That's where my post-processing was a crucial step, whenever the (irradiance_global_reference) was below 5, I set the predicted GHI to zero. This is rooted in physical reality, when irradiance is that low, it is effectively night or just before/after it, and the sun's contribution is negligible. Ignoring this would have resulted in the model wrongly predicting light during dark hours, skewing both error metrics and real-world applicability.

Now, for evaluation, I initially calculated MAPE using sklearn, but that doesn't handle actual GHI = 0 very well (which happens a lot during the night). So I wrote a custom MAPE function. If actual GHI was zero, and the prediction was also zero, I gave it 0 error. But if actual was zero and prediction wasn't, I assigned a 100% error. This helped prevent those wild spikes in MAPE and made the evaluation more realistic with a value of **20.89%**.

The main challenge I encountered was the noisy, non-stationary nature of GHI and the highly imbalanced ratio between daytime and nighttime data. Without proper smoothing and log transformation, early model runs would oscillate wildly around low-irradiance hours. EMA, log1p, and irradiance thresholding were pivotal in handling this.

In conclusion, this solution balances physical reasoning, data-driven feature engineering, and model interpretability. Also, this entire pipeline is fully reproducible, and all dependencies are listed in the accompanying requirements file.