FORECASTING PHOTOVOLTAIC PRODUCTION WITH NEURAL NETWORKS AND WEATHER FEATURES

Stéphane Goutte[b], Klemens Klotzner[e], Hoang-Viet Le[a,b,*] and Hans-Jörg von Mettenheim[c,a,d]

[a]Keynum Investments, France
[b]Université Paris Saclay, UMI SOURCE, IRD, UVSQ, France
[c]IPAG Business School, France
[d]Oxford-Man Institute of Quantitative Finance, United Kingdom
[e]European Energy Market Makers, Luxembourg

Abstract

Accurate solar energy forecasting is crucial for optimizing energy generation and distribution. In this study, we investigate the potential use of weather forecast data and entity embedding techniques as inputs for machine learning models to generate solar energy forecasts up to 2 days ahead. We apply several machine learning methods to 2 years of hourly solar energy production data (2020-2021) from 16 power plants located in different regions of Northern Italy, along with corresponding weather forecast data. Our results show that weather variables significantly impact solar energy production, and incorporating weather forecast data and entity embedding techniques into machine learning models improves the accuracy of solar energy forecasting up to 2 days ahead. We find that using entity embedding on sky descriptor variables and each power plant in the dataset improves the performance of the models. Our forecasting results are compared to a local forecasting provider, and our models show significantly better performance in solar energy forecasting. This study demonstrates the potential of machine learning techniques, specifically entity embedding, to improve the accuracy of solar energy forecasting for power plants. The improved forecasting accuracy can benefit both the power plant operators and the electricity grid operators by providing more reliable information on energy production and distribution.

Keywords: Solar Energy, Time Series Forecasting, Machine Learning, Neural Networks, Entity Embedding

*Corresponding author: Hoang-Viet Le, viet.le@keynum.fr

1 INTRODUCTION

Machine learning has revolutionized how we analyze data and make predictions based on patterns and trends. This technology has found applications in various industries, including energy production, where accurate forecasting is essential for efficient and cost-effective energy generation. Solar energy, in particular, has gained significant attention due to its potential to contribute to sustainable energy generation.

One of the biggest challenges in solar energy production is the variability of weather conditions, which directly impacts the amount of energy generated. Machine learning models can help forecast solar energy production by analyzing historical data and weather patterns to predict future solar irradiance. By leveraging a vast amount of data, machine learning models can improve the accuracy of solar energy forecasting and help energy producers make more informed decisions. The benefits of accurate solar energy forecasting are far-reaching. It can help energy producers optimize their energy generation and distribution systems, reduce operational costs, and improve the integration of renewable energy sources into the power grid. Additionally, solar energy forecasting can assist energy consumers in making informed decisions about energy consumption, storage, and pricing.

Several machine learning techniques have been applied to solar energy forecasting, including artificial neural networks, support vector machines, and decision trees (Abuella and Chowdhury, 2015; Rodríguez et al., 2018; Xiang, Xiaoyan et al., 2021; Jebli et al., 2021; Lim et al., 2022). These models can take into account various factors such as cloud cover, temperature, humidity, and wind speed to predict solar irradiance. Moreover, ensemble methods, such as combining multiple machine learning models, have shown promising results in improving the accuracy of solar energy forecasting. However, there is still a need for more sophisticated machine-learning techniques that can handle complex data structures such as categorical variables

Entity embedding is a technique that can address the challenges associated with categorical variables. Entity embedding represents categorical variables as continuous vectors in a low-dimensional space, where each dimension represents a meaningful feature of the variable(Guo and Berkhahn, 2016). However, there is no literature about the usage of entity embedding in energy production yet except for the work of Wagner et al. (2022) where it was used to predict the electricity price instead. But it is only about electricity prices, not energy production. There is also the work of Rosato et al. (2016) where embedding is used for only time series features but not weather forecast categorical features.

In this study, we investigate the potential use of weather forecast data as inputs, including categorical weather data, for machine learning models to generate solar energy forecasts up to 2 days ahead. We apply several machine learning techniques to 2 years of hourly solar energy production data from 16 power plants in Northern Italy, along with corresponding weather forecast data from the same regions. Our results show that weather variables significantly impact solar energy production, and incorporating weather forecast data into machine learning models improves the accuracy of solar energy forecasting. This study demonstrates the potential of machine learning techniques to enhance the accuracy of solar energy forecasting for power plants in Northern Italy and contributes to the growing body of research on the integration of renewable energy sources into the power grid.

1.1 Data Source

1.1.1 Solar Energy Production Data

The solar energy production data used in this study consists of hourly production measurements from 16 photovoltaic plants located in different regions of Northern Italy. The data span a period of two years from 2020 to 2021 and is provided directly by a solar production company from Italy.

1.1.2 Weather Forecast Data

The weather forecast data used in this study corresponds to the same regions as the 16 photovoltaic plants. The weather forecast data is obtained from publicly available sources and includes hourly forecasts of temperature, humidity, wind speed, and cloud cover. The weather forecast data is provided in a separate set of files and is also organized by month and region.

Given the geographical proximity of the 16 photovoltaic plants, we consider the possibility of using all the available solar energy production data together as inputs to machine learning models. The inclusion of weather forecast data in the models is expected to improve the accuracy of solar energy forecasting.

1.2 Data Preprocessing

1.2.1 Data Construction

Data preprocessing is a crucial step in the process of data analysis, which involves cleaning and organizing data in a way that makes it usable for analysis. In this study, the first step of data preprocessing was to gather all the necessary data values from different data sources into a single collection of databases. As there were two different sources of data - one reflecting the data production of each specific plant and the other containing weather forecasts, mapping between the two datasets based on the time stamps were required. Since each day's forecasting needed to be made by 11 am of the previous day, the weather forecasting information nearest before the forecasting took place was used. After the mapping, our unified dataset will have the following form:

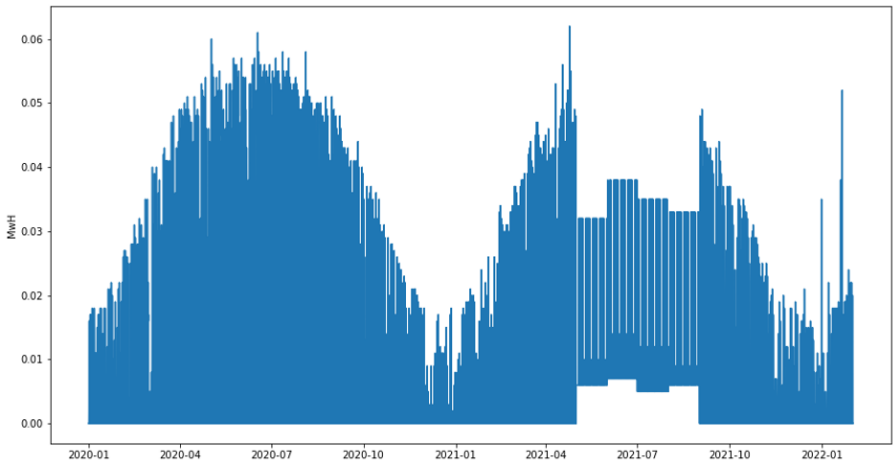
Table 1. The Preprocessed Trade data

	Field name	Type	Description
1	timestamp	datetime	The hour that energy production was recorded.
2	PlantID	integer	The integer that represents each energy plant.
3	Day-Night	string	The daylight status
4	Sky descriptor	integer	The integer that represents each type of sky condition
5	Temperature	float	The forecasted temperature in Celcius
6	Wind speed	float	The speed of wind in km/h
7	Wind direction	integer	The direction of the wind by compass degrees (0-359, with 0 equals to North)
8	Humidity	integer	The forecasted humidity level in percentage

1.2.2 Data Imputation

The next step is to find and process possible missing values. Sometimes, there can be missing values for some features in the dataset. Some specific machine learning models can ignore or even learn from missing data. However, several of our models such as the neural networks can not handle missing data. As a result, it is essential to be able to detect and fill these values so that there will not be any errors during the training as well as not losing some information. Several approaches can be used in this step. The two approaches that are more commonly used are to fill the missing values with average, median, zero, or the value of the previous observation or to use an interpolation technique, such as linear, time, quadratic, or cubic interpolation. In this case, the energy production dataset that we received contains three months from May 2021 to August 2021 where there were problems with the production recording procedure of the energy company. Therefore, the production data we received during this period was daily average instead of hourly, which is not suitable for our hourly forecasting model.

Figure 1. Candle stick example



To deal with this problem, we decided to remove this period from our study and the dataset that followed the incident are used as the test dataset for our study. The period before the incident is used as the training dataset. In short, after data cleaning, we have 178,056 samples for the training set which ranges from January 2020 to April 2021, and 59,376 samples for the test set from August 2021 to January 2022. To ensure that our models are well-tuned and not overfitting the training data, we further divided the training set into a validation set, which contains data from January 2021 to April 2021. This way, the training set can span a full year from January 2020 to December 2020. The validation set will be used during the training phase to evaluate the performance of the models and adjust the hyperparameters accordingly. The final models will be evaluated using the test set, which contains data from August 2021 to January 2022.

2 METHODOLOGY

2.1 Feature Engineering

2.1.1 Cyclical Data

As our study focuses on time series analysis, it is crucial to incorporate the time aspect of the data into our features. However, time-related data is often represented in the ”datetime” format (YYYY-

MM-DD HH:MM:SS), which makes it difficult to extract information beyond the ascending order of data points. This format does not reveal cyclical patterns such as hours of the day, days of the week, months, seasons, etc., which are important for our analysis.

Several approaches have been proposed to address this issue. The simplest approach is to use the number of minutes, hours, months, or weekdays as features. However, this method fails to account for the cyclical nature of the data, such as the difference between 24:00 and 01:00 hours. Another approach is to use dummy variables for each hour, but this increases the number of variables and neglects the aspect of consecutive hours.

To address both of these problems, a popular method is to apply sine and cosine transformations to the cyclical data. This method preserves the cyclical nature of the data and accounts for the difference between hours. In our study, we will apply this transformation to the month and hour features as follows:

$$\begin{aligned} month\_cos &= \cos(2\pi \frac{month}{12}) \\ month\_sin &= \sin(2\pi \frac{month}{12}) \end{aligned} \tag{1}$$

$$\begin{aligned} hour\_cos &= \cos(2\pi \frac{hour}{24}) \\ hour\_sin &= \sin(2\pi \frac{hour}{24}) \end{aligned} \tag{2}$$

Here, equation 1 was used to add information regarding the months of a year, while equation 2 provided knowledge regarding the hours of a day.

2.1.2 Categorical Data

Categorical data is a common type of data that can be found in many datasets, including the one used in our study. It is a type of data that includes variables with discrete values that represent different categories or groups. This type of data can be further categorized into ordinal and nominal data. Ordinal data is data that has an inherent order, while nominal data does not. In our study, both of our categorical data (the plant id and sky descriptors) are nominal where their label does not imply any order. Machine learning models cannot directly work with categorical data, as they require numerical inputs. Therefore, it is crucial to transform the categorical data into a numerical format that can be used as input for the models.

One of the most common ways to transform categorical data is to use dummy variables (or one hot encoding), where each category is represented by a binary variable indicating whether the observation falls into that category or not. However, this approach has several drawbacks. It leads to a high number of input features, especially when the number of categories is large, which can cause overfitting and slow down the training process. Furthermore, dummy variables do not capture any meaningful relationship or similarity between the categories.

One alternative to using dummy variables is to apply entity embedding, which is an advanced technique commonly used for handling categorical data in machine learning. Entity embedding maps each categorical variable to a low-dimensional vector space, where the distances between vectors capture the relationships between the categories. Entity embedding has been used in various applications, including computer vision and natural language processing. For example, in natural language processing, words can be represented as entity embeddings, whereas words with similar meanings are represented by similar vectors. These vectors can then be used as input to a neural network to perform various tasks, such as sentiment analysis or language translation. This technique can handle

both nominal and ordinal data and has been shown to have several advantages over the use of dummy variables, including the ability to capture nonlinear relationships between categories, and the ability to reduce the number of variables.

In our study, we will use both dummy variables and entity embedding to transform the categorical data. We will compare the performance of these two methods to see which one performs better for our specific dataset and model architecture. By using both methods, we can explore the trade-offs between simplicity and expressiveness, and determine the best approach for our particular problem.

2.2 Model Specifications

In this paper, five popular machine learning algorithms were used which is simple Linear regression, two decision tree-based gradient boosting models (XGBoost and LightGBM), and two feed-forward neural network models with one using the entity embedding for both categorical features and the other using dummy variables.

For the Linear regression model, we choose the logistic regression algorithm provided by the Scikit-learn library.

For both XGBoost and LightGBM, we used their official Python libraries. For these two algorithms, their best hyperparameters are tuned using a randomized search on the validation dataset.

Regarding neural network models, in this study, all of our models are built and trained using the Python library Keras running on top of the TensorFlow framework. The architectures of our neural networks which only use the dummy variables are optimized using grid searches on the number of layers, units, and dropout values are as follows:

Table 2. MLP no embedding architecture

#	Layer	Units	Activation Function	Dropout
0	Input	-	-	-
1	Dense	70	LeakyReLU	0.4
2	Dense	70	LeakyReLU	0.4
3	Dense	1	Linear	-

For the neural network models that use embedding, we used up to two embedding layers (depending on if weather features are used or not). If weather features are not used, there is only one embedding layer for plantID whereas an additional layer for sky descriptor is applied otherwise. The optimal specification for this network is as follows:

Table 3. MLP embedding architecture

#	Layer	Units	Activation Function	Dropout	Connected from
0	Input(float)	-	-	-	-
1	Input(plantID)	-	-	-	-
2	Input(sky desc)	-	-	-	-
3	Embedding(plantID)	2	-	-	Input(plantID)
4	Embedding(sky desc)	2	-	-	Input(sky desc)
5	Flatten(plantID)	-	-	-	Embedding(plantID)
6	Flatten(sky desc)	-	-	-	Embedding(sky desc)
7	Concatenate	-	-	-	Input(float) Flatten(plantID) Flatten(sky desc)
8	Dense	80	LeakyReLU	0.4	Concatenate
9	Dense	80	LeakyReLU	0.4	Dense 8
10	Dense	1	Linear	-	Dense 9

All of the neural network models are trained in 750 training epochs with batch size equaling 1024. Early stopping and model checkpoints based on the validation loss are also used to prevent overfitting so that we can get the best-performing model on the validation set.

2.3 Model Evaluation

2.3.1 Root Mean Squared Error

Root Mean Squared Error (RMSE) is one of the most commonly used metrics to evaluate forecasting models. It measures the average deviation between the predicted and actual values, with larger errors being penalized more heavily than smaller ones due to the squaring. The RMSE is calculated by taking the square root of the average of the squared differences between the predicted and actual values. RMSE is sensitive to outliers, which can greatly affect the overall score.

2.3.2 Mean Absolute Error

Mean Absolute Error (MAE) is another widely used metric that measures the average absolute difference between the predicted and actual values. Unlike RMSE, MAE does not penalize larger errors more heavily, which can make it a better choice when outliers are present. MAE is calculated by taking the average of the absolute differences between the predicted and actual values.

2.3.3 R-squared

R-squared (R2) is a metric that measures the proportion of variance in the dependent variable (i.e., the energy production) that can be explained by the independent variables (i.e., weather and time-related features). R2 ranges from 0 to 1, with higher values indicating that the model is a better fit for the data. R2 is useful for comparing different models, but it does not provide information on the absolute error of the predictions.

2.3.4 Benchmarks

In addition to evaluating our models using the metrics described above, we will also use two benchmarks to assess the performance of our models. The first benchmark is the prediction from an unnamed forecasting company that currently provides services to the energy company that contributed our data. This benchmark will give us an idea of how well our models perform compared to the current forecasting methods in practice. The second benchmark is the persistence model, which is a simple forecasting model that assumes the future values will be the same as the last observed value (in this case the value 24 hours ago). The persistence model is often used in energy forecasting because it provides a baseline performance that can be used to evaluate the effectiveness of more complex models.

While all three evaluation metrics (RMSE, MAE, and R-squared) are important in assessing the performance of our models, we will use the root mean squared error (RMSE) as the main metric to select the best-performing model. This is because RMSE puts more weight on larger errors, which is important in energy forecasting where large errors can have significant economic and environmental impacts. We will also use the other two metrics as secondary measures to gain a more complete understanding of the performance of our models.

3 Result and Discussion

3.1 General Discussion

As discussed in the previous section, the goal of our study is not only to find out the potential application of machine learning in predicting energy production but also to find out whether or not the usage of weather forecasts is necessary for the improvement of the models. Therefore, we applied a few subsets of features compared to each other to compare their usefulness.

The first dimension for the set of features is the usage of weather forecasting as inputs for the models and the second dimension is about the usage of embedding. However, as the two gradient boosting algorithms are capable of dealing with categorical data on their own, the usage of embedding only applies to the multilayer perceptron models (MLP). For linear regression, only dummy variables are used. In the end, we will have four different MLP models, two models for each of the two gradient boosting models as well as linear regression. In combination with two benchmark models, there are 12 different models in total for comparison. Their results are shown in Table 4.

Table 4. Performance of All Models

Model	Weather	RMSE	MAE	R2
MLP with embedding	Yes	**0.00459**	0.00225	**84.06%**
MLP with dummy	Yes	0.00466	0.00234	83.56%
LightGBM	Yes	0.00499	**0.00224**	81.16%
XGBoost	Yes	0.00509	0.00256	80.40%
MLP with embedding	No	0.00538	0.00267	78.14%
MLP with dummy	No	0.00540	0.00270	77.98%
XGBoost	No	0.00565	0.00281	75.86%
LightGBM	No	0.00585	0.00270	74.13%
Linear Regression	Yes	0.00597	0.00326	73.09%
Linear Regression	No	0.00630	0.00329	69.95%
Benchmark	-	0.00652	0.00322	55.54%
Persistence	-	0.00707	0.00286	62.25%

Table 4 show the list of all models with their performance metrics calculated on the test set. The order in which they are listed is based on their RMSE on the test set, from the lowest to the highest, which also means the best to worst. From just a quick view of the table, we can see that the top models in RMSE are the MLP models. Next comes the two gradient-boosting models. The top 8 performing ones are all combinations of those models in general. Both two versions of the linear regression model come next. However, we can observe that all of our models no matter what specification surpassed the two benchmark models. The persistence model is the worst one and then the predictions from the forecasting company are the second worst. The pattern is quite similar regarding the R2 metrics where the ranking order is almost the same except for the two benchmark predictions. Here, the persistence model has the R2 of 62.25% whereas the benchmark prediction is only 55.54%. Regarding the MAE metrics, the ranking order is a bit different where the performance of linear regression seems to be the worst one and the best one is actually from the LightGBM. However, the MAE of the Embedding MLP is quite close to that of the LightGBM with 0.00225 compared to 0.00224.

We can conclude that the best-performing model here in the test set is the MLP model with embedding where it has the best RMSE and R2, and the second-best MAE. Its performance reduces the RMSE by 35.07% compared to the persistence model and 29.6% compared to the benchmark prediction and the MAE by 21.32% and 30.01% respectively. The R2 also increases from the range of 60% or less to up to 84.06%. The performance of this strategy is shown in figure 2 where it is compared with the true energy production for the first 1,000 data points of the test set.

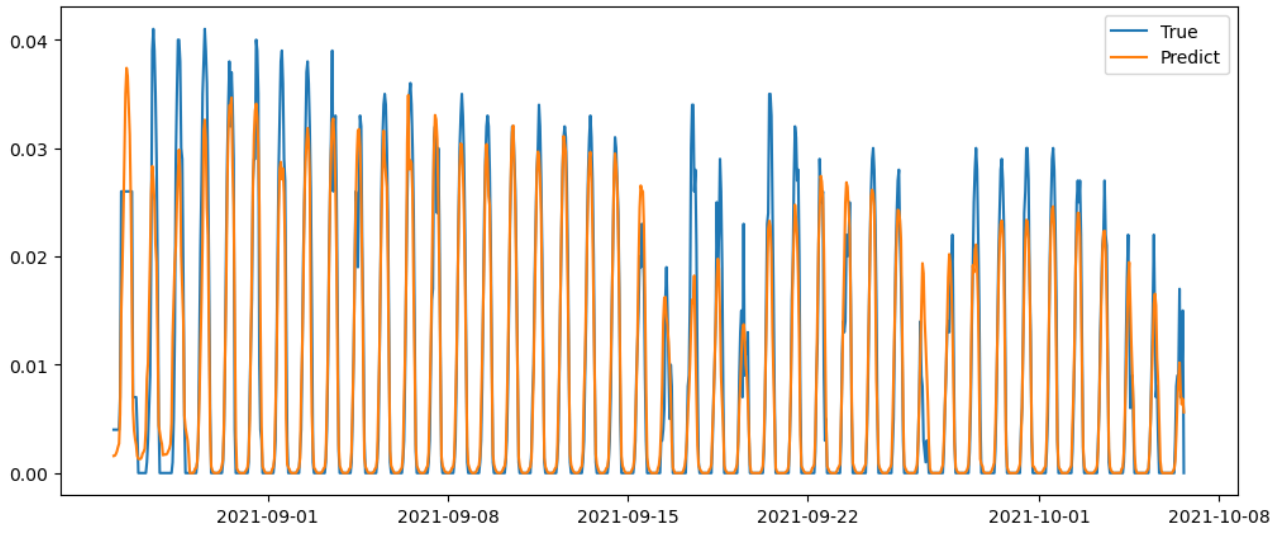


Figure 2. Model Performance in Test set

3.2 The Necessity of Weather Forecast Data

Table 5. Comparison between 2 candlestick representation methods

Model	RMSE	MAE	R2
With Weather	0.00506	0.00253	80.45%
Without Weather	0.00572	0.00283	75.21%

Table 5 presents the average performance metrics of all machine learning models using either the weather features or not. From the table, we can see that the inclusion of the weather forecast data increases more than 5% of R2. It also reduces the RMSE by 11.53% and MAE by 10.6%. We can conclude that weather forecast data is necessary to improve the performance of energy forecasting even if they are more than 24 hours old.

3.3 Models Comparison

Finally, we would like to compare the general performance of every machine learning model that we used. Table 6 shows the average performance metrics of all models with or without the inclusion of the weather forecasting data.

Table 6. Average Performance of all models

Model	RMSE	MAE	R2
MLP with embedding	**0.00499**	**0.00246**	**81.10%**
MLP with dummy	0.00503	0.00252	80.77%
XGBoost	0.00537	0.00269	78.13%
LightGBM	0.00542	0.00247	77.64%
Linear Regression	0.00614	0.00328	71.52%
Benchmark	0.00652	0.00322	55.54%
Persistence Model	0.00707	0.00286	62.25%

From the table, it is easy to observe that the neural networks (MLP) are better in all metrics for this specific regression task. They are ranked from the highest to the lowest in the RMSE calculated on the test set. Among them, the MLP model with entity embedding is better in almost every metric compared to other models. The MLP model with dummy variables comes second with slightly worse results. Both of them, however, are a few percent better compared to the gradient boosting algorithms (6.33% in RMSE). The performance of the two gradient boosting algorithms (XGBoost and LightGBM) are also quite close too. Still, the performance of those algorithms is closer to the MLP than the Linear Regression and our two benchmarks as well.

Regarding the difference between the MLP with and without embedding, we observe a larger difference with the inclusion of weather forecasting data (1.5% reduction in RMSE compared to around 0.3%). This is reasonable considering the inclusion of the weather forecast data also comes with the incorporation of another embedding layer for the sky descriptors data. One of the advantages of using entity embedding is that we can extract the embedding vector for each categorical feature and interpret them to see if they make sense. The embedding vector represents a high-dimensional space where each dimension corresponds to a feature's weight in the embedding. By analyzing the embedding vectors, we can gain insights into the categorical features and their relationships with the target variable.

Figure 3. Sky Descriptor Embedding

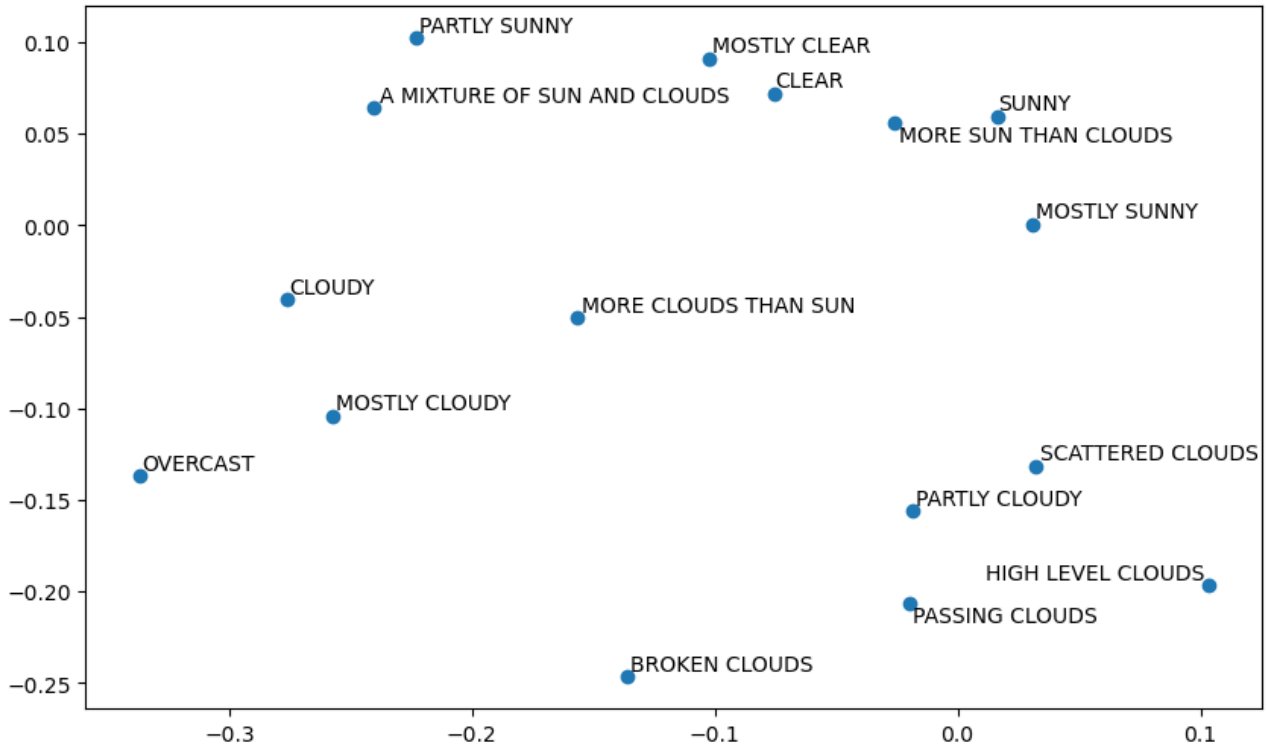
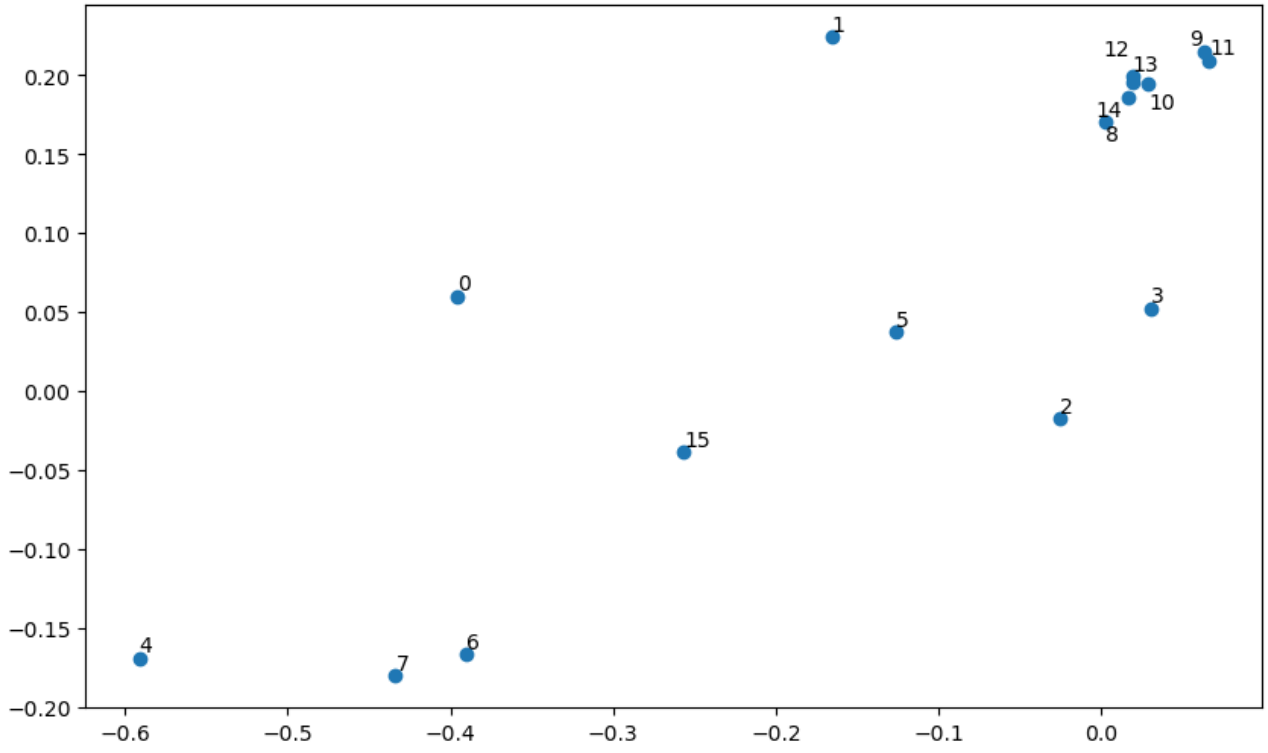


Figure 3 visualizes the relationships between different categories of sky conditions in two dimensions. The value of the 2-d vectors of each sky condition is optimized for solar energy prediction. Of course, we cannot confirm exactly the representation of the two dimensions, however, we can interpret them based on common sense to see if they are reasonable or not. From the figure, we can see that most of the sunny sky conditions stay near each other and are located on the top right of the

figure. The sky conditions from the upper part are mostly related to the sunny weather. Regarding the horizontal axis, we can see that on the far left is the overcasting weather which is typically low-level clouds that are thick and often appear gray or white, covering most of the sky. Overcast clouds are often associated with rainy or stormy weather conditions. On the far right, you have high-level clouds which are typically thin and associated with fair weather. From this visualization, we can see that the embedding of those sky descriptors shows rational meaning and can also help us discover new relationships between them and the target variable.

Figure 4. PlantID Embedding



The other embedding about each plantID is shown in figure 4. What we observed from the figure is that plants from number 8 to number 14 are quite near to each other. However, as more detail about each plant's characteristics is not revealed to us (their geographic locations, their specification, etc.), the interpretation of the embedding is not obvious in this case. We can only interpret that the vertical axis may represent the production capacity of each plant as their vertical order is the same as the order of the average production of each plant in the training dataset.

4 Conclusion

The findings of our study have demonstrated the efficacy of popular machine learning methods such as neural networks and gradient boosting trees in enhancing the accuracy of solar energy prediction as opposed to traditional forecasting methods, including linear regression, persistence model, and prediction from a local forecasting company. The results revealed that the neural networks exhibited superior performance in comparison to other models, particularly when entity embedding was employed for categorical features. This highlights the importance of utilizing entity embedding as an

effective method for handling categorical data in machine-learning models. Additionally, our study reaffirms the recommendation from previous literature regarding the utilization of weather features as inputs in forecasting systems. Our results indicated that weather features significantly enhance forecasting accuracy, even when only weather forecast data are used.

Furthermore, our study contributes to the understanding of the interpretability of entity embedding vectors. The embedding vectors can be examined and interpreted to assess whether they align with our domain knowledge or intuition. The ability to extract these vectors allows us to gain insights into the relationship between different categorical variables and their impact on the forecasted outcome. Overall, our study provides valuable insights into the use of machine learning methods in solar energy forecasting, which can inform future research and practical applications in the field.

REFERENCES

Abuella, M. and Chowdhury, B. (2015). Solar power forecasting using artificial neural networks.

Guo, C. and Berkhahn, F. (2016). Entity embeddings of categorical variables.

Jebli, I., Belouadha, F.-Z., Kabbaj, M. I., and Amine, T. (2021). Deep learning based models for solar energy prediction. *Advances in Science, Technology and Engineering Systems Journal*, 6:349–355.

Lim, S.-C., Huh, J.-H., Hong, S.-H., Park, C.-Y., and Kim, J.-C. (2022). Solar power forecasting using cnn-lstm hybrid model. *Energies*, 15:8233.

Rodríguez, F., Fleetwood, A., Galarza, A., and Fontán, L. (2018). Predicting solar energy generation through artificial neural networks using weather forecasts for microgrid control. *Renewable Energy*, 126:855–864.

Rosato, A., Rosa, A., Araneo, R., and Panella, M. (2016). Embedding of time series for the prediction in photovoltaic power plants. pages 1–4.

Wagner, A., Ramentol, E., Schirra, F., and Michaeli, H. (2022). Short- and long-term forecasting of electricity prices using embedding of calendar information in neural networks. *Journal of Commodity Markets*, 28:100246.

Xiang, Xiaoyan, Sun, Yao, and Deng, Xiaofei (2021). Short time solar power forecasting using persistence extreme learning machine approach. *E3S Web Conf.*, 294:01002.