Emily Weed, emw232
CS M.Eng Project
Machine Learning Model to Predict Microclimate with CFD Simulation Data
Advisor: Professor Max Zhang

In this project I created machine learning models that use CFD simulation data to predict the module surface temperature of solar panels for the purpose of optimizing site design for efficiency and longevity. This can also be useful for furthering research into agrivoltaics and the importance of understanding microclimate in terms of ecological effects. The purpose of using machine learning (ML) in this application is to learn from the CFD data and predict the module surface temperature. The steps involved in this process include data processing, model creation, and evaluation as described in the following sections.

The parameters for the model consist of site design, ground cover and weather data features. The weather data was collected in 30-minute intervals, resulting in output module surface temperatures also at 30-minute intervals. Each of these input output combinations is considered one row of the data set. Thus, with 193 simulations, run with weather data from 39 distinct time periods, we get 7527 total rows of data to work with. Some simulations stopped early, resulting in missing module surface temperature data for the last few time periods. These instances were dropped from the data set. The final data set had 7381 rows to work with.
The total feature space used in the CFD models include panel height (m), row spacing (m), array width (m), array length (m), tilt angle (degrees), mounting type, orientation, total site length (m), total site width (m), evapotranspiration (mm/day), albedo, temperature (℃), wind at 2.5 m (m/s), and radiation (W/m^2). The values for row spacing, array width, array length, tilt angle, mounting type, orientation, total site length, and total site width were not changed in the simulations and thus they were not useful in the model. This left me with 6 features to be considered moving forwards. All 6 of these features are numeric, and their values were standardized to have a mean of zero and standard deviation of one prior to performing any modeling.

The 7381 data points were randomly split into training and testing data using an 80-20% split. The test set was set aside and not used for feature selection or hyperparameter tuning. To obtain a better estimate of the testing error during the tuning process, 5-fold cross validation was utilized on the training set. As the task is a regression problem (the output variable is numerical), I used the average R^2 and the normalized root mean squared error ($NRMSE$) as the evaluation metrics. These metrics are defined as following

$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$ where $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the average of the output values.

$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_i^n (y_i - \hat{y}_i)^2}}{\bar{y}}$ where $\hat{y}_i$ and $\bar{y}$ are defined similarly. The R^2 value demonstrates the variance in the output variable that is explained by the model.

I explored 4 models, Linear Regression (LR), Support Vector Regression (SVR), Random Forest Regression (RFR) and Gradient Boosted Trees for Regression (GBT). Linear Regression is the simplest model explored. It assumes there to be a linear relationship between the features and the output. Support Vector Regression stems from Support Vector Machines, a classification model in which the optimal hyperplane maximizing the margin of the data is found. This method can be adapted to apply to regression problems. Random Forest Regressor is an ensemble model that combines many decision trees that are trained on a subset of the data. Gradient Boosted Trees for Regression is another ensemble method, fitting weak decision trees to the residual of the previous tree. Python's Scikit-learn package was used for each model. During the feature selection process, each model was used to measure performance. As there were only 6 features to work with, using all 6 most likely would not have caused issues with overfitting. Nevertheless, I performed feature selection methods to assess the optimal features for the models. I ran Forward Stepwise Selection, Backwards Stepwise Selection and Best Subsets Selection. All three methods gave comparable results, as is expected when run on a small feature set such as this. Figures 1, 2 and 3 show the average validation R2 values achieved by each model. With each additional feature, the validation score increased and there was no indication that this would lead to overfitting. As we only have 6 features to work with, we will proceed modeling with all 6 features.

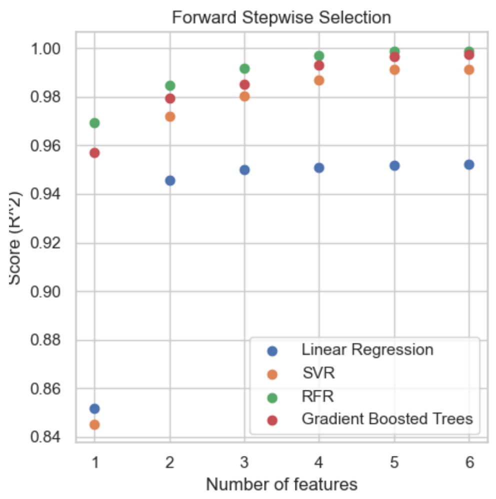


Figure 1

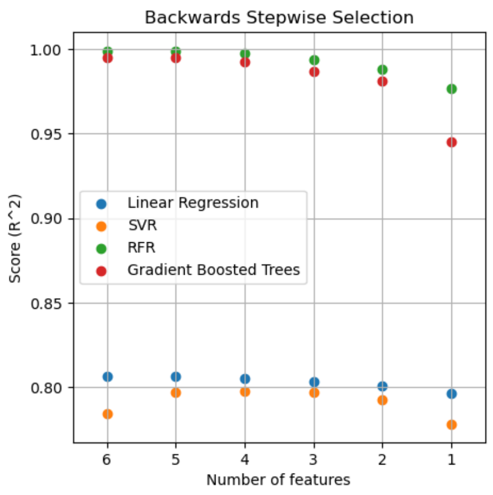


Figure 2

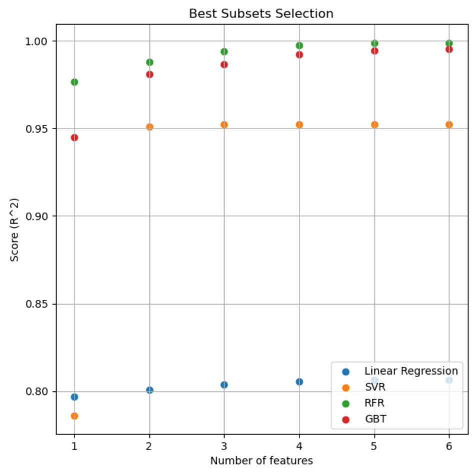


Figure 3

Hyperparameters need to be tuned for the SVR, RFR, and GBT models. Linear regression does not have any tunable parameters. To perform hyperparameter optimization for these models, grid search with 5-fold cross validation was utilized. Table 1 shows the tunable parameters, the ranges explored, and the final selection with the validation R^2 and $NRMSE$ for each model. As we can see, the GBT model achieved the highest validation R^2 value using the optimal hyperparameters found during Grid Search.

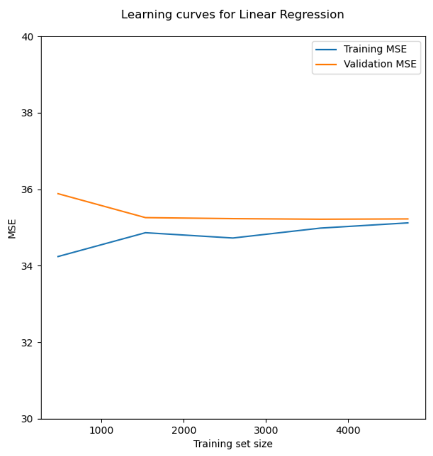
Model	Tunable Parameters	Ranges Explored	Optimal Parameters Selected	Validation R^2, $NRMSE$ with Optimal Parameters

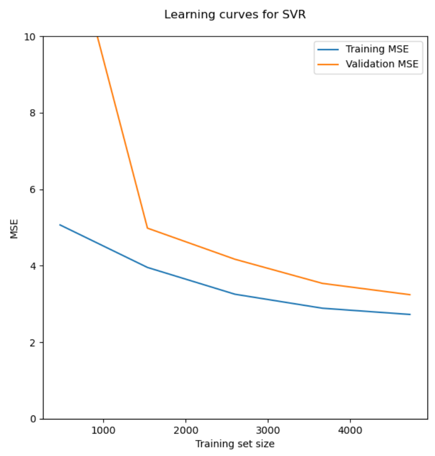
SVR	Kernel
	C
	γ
		{‘linear’, ‘poly’, ‘rbf’}
		{0.1,1,10,100}
		{0.1,0.01,0.001,0.0001}
			rbf
			100
			0.1
				0.9830617
				0.100097

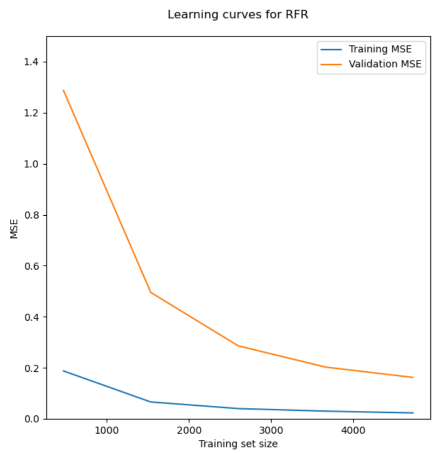
RFR	Max depth Max features Min samples split Num. estimators	[6,8,10,12,14,16] [2, 3, 4, 5, 6] [2, 4, 8, 10, 12] [100, 200, 300, 1000]	16 5 2 1000	0.9991262 0.0227147
GBT	Num. estimators Max depth Learning rate Subsample Min samples split	[250,500,1000] [3,5,7] [0.01,0.05,0.1] [0.5,0.75,1] [4,8,12,16,20]	1000 7 0.05 0.5 12	0.9997592 0.0119442

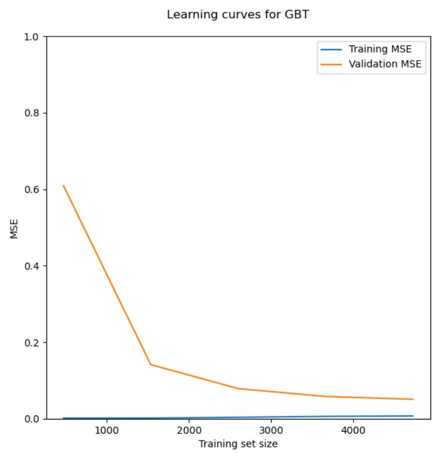
Table 1: Hyperparameter Optimization Results

As the data was generated via CFD simulations, I can explore the learning curves of the models to ensure model convergence. Figures 3, 4, 5, and 6 show the learning curves for each ML model for various numbers of training points, all drawn from the training data set. It can be observed that all 4 of the models do converge with the available training data. Linear Regression, and Gradient Boosted Tree Regression show an earlier convergence rate than Random Forest Regression and Support Vector Regression.









Running our models on our held-out tests set provides an unbiased evaluation of our models. Table 2 shows the test R^2 and $NRMSE$ values obtained by SVR, RFR, and GBT. As I expected, GBT performs the best. All three models performed a little worse than the validation values I was getting previously, this could be due to natural error, or slight overfitting in the modeling process. Utilizing more data by running more CFD simulations could be a remedy to this possible slight overfitting. As it is a small difference, I can proceed with these models for the optimization process.

	RFR	SVR	GBT
Test R^2	0.979020	0.975361	0.981946
Test $NRMSE$	0.112484	0.121901	0.104348

After training the ML models, the time to run the model to obtain a prediction is very fast. The ML model can predict a single module surface temperature based on given design parameters in a fraction of a second. This allows for design recommendations to be generated in a timely manner.

As an extension to the ML models, I explored this situation as an optimization problem. Turning the ML model into an optimization problem allows us to determine the best values for the parameters that can be adjusted based on the desired module surface temperature. In this case, the parameters that can be changed are panel height, evapotranspiration and albedo. The weather data (temperature, wind, and radiation) are inherent to the location of the solar farm and will be given to the optimization problem. I utilized the SciPy Optimization library for this objective which provides support for global minimization of an objective function subject to parameter bounds. The objective function I used was the absolute difference between the value predicted by our model and the desired target value. This ensures that by minimizing this absolute difference, we are obtaining a result as close as possible to the desired module surface temperature.

As there are many different applications in which an optimization problem would be useful in this context, I wrote multiple different wrapper functions around our base optimization problem. Utilizing the fully trained models, I can obtain optimal solar farm design parameters for panel height, evapotranspiration, and albedo based on the desired module surface temperature. The simplest application is a function which accepts the weather conditions as parameters, and optimizes over the panel height, evapotranspiration and albedo value. Utilizing this as a first step can give a good idea of the range of values suited for a certain application. Another sub-problem I explored was the application to a single crop with ideal values for evapotranspiration and albedo. It optimizes over just panel height in this scenario. Since I knew that panel height was not the most important parameter for the construction of the ML model, this optimization problem provides smaller differences in resulting module surface temperature than previous applications. An extension to this scenario I explored is to use discrete values for the panel height (0.5, 1.5, 2.5, etc.) and look at the tradeoff in module surface temperature each value gives. This could assist in analyzing the cost benefit ratio of different panel heights when designing a new solar farm. Another use case is the scenario where panel height is predetermined, and optimization over albedo and evapotranspiration is desired. Expanding the training set to include solar site designs which differ in more than just panel height can expand the optimization problem, resulting in more parameters to optimize over. These results can provide valuable insight and evidence for determining the design of future sites or modifying current ones.

This work is contained in a jupyter notebook, with detailed explanations of the steps taken. It can easily be extended to include more data or new use cases. The PhD student I worked closely with, Henry Williams, will take over wrapping up this work and applying the results to his current field of research.