Final Project Report CS 4824

Eshan Kaul

Virginia Polytechnic Institute and State University

Bachelor of Science

in

Computer Science & Economics

Professor: Lifu Huang

May 4, 2023

Blacksburg, Virginia

Keywords: Decision Tree Regressor, Gradient Boosting, Kernel Partial Least Squares

Contents

Chapter 1

Background/Motivation

India, being the third-largest global consumer of energy, has been actively exploring renewable energy resources to meet its growing energy demands. The country's rapid economic growth and increasing urbanization have contributed to a surge in energy consumption, which has led to concerns about energy security and environmental sustainability.
Promoting the use of renewable energy sources, such as solar power, can help to achieve greater energy sustainability and reduce the world's reliance on fossil fuels. Harnessing solar energy effectively, can significantly reduce greenhouse gas emissions and curtail the world's dependence on non-renewable resources.

The Indian government has recognized the potential of solar power and has recently taken on several large initiatives in the development of solar infrastructure. This has led to a rapid increase in solar power installations across the country. This project aims to further this progress by accurately predicting the AC power output of solar plants in India in the hopes that providing precise forecasts, can enable better planning and management of solar energy production, thereby maximizing the efficiency of the solar power plants.

Chapter 2

Data

For this project, the data has been sourced from Kaggle and consists of two datasets collected from two solar power plants in India over a 34-day period in 15-minute intervals. The data is divided into two files: plant generation data and plant weather sensor data.

The plant generation data is collected at the inverter level, with each inverter having multiple lines of solar panels connected to it. This dataset comprises 68,778 rows and 7 columns, providing detailed information about the energy production of the solar panels.

The plant weather sensor data, on the other hand, is collected at a plant level from a single array of sensors optimally placed within the plant. This dataset contains 3,182 rows and 6 columns, offering insights into the weather conditions affecting the solar power generation.

By analyzing and combining these datasets, we aim to develop a reliable forecasting model that can accurately predict the AC power output of solar plants in India. This will not only contribute to the country's energy sustainability efforts but also provide valuable information for all stakeholders in the renewable energy sector.

Chapter 3

Methods

3.1 Overview

As there are several features included in the data sets it is important to perform exploratory data analysis to determine/extract the most important features to include in the regression models and verify that the data does not violate any major assumptions involved in the various models that will be built out.

3.1.1 Correlation

Correlation is defined as any statistical relationship, whether causal or not, between two random variables or bivariate data. The Correlation Coefficient or Pearson correlation coefficient is commonly obtained by taking the ratio of the covariance of the two variables in question from the numerical data set, normalized to the square root of their variances. Mathematically, it is the division of the covariance (joint variability of two random variables) of the two variables by the product of their standard deviations.

$$\rho X, Y = \text{corr(X,Y)} = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y} = \frac{E[(X - \mu x)(Y - \mu y)]}{\sigma_x \sigma_y}$$

$$\text{cov}(X, Y) = E[(X - E[X])(Y - E[Y])]$$

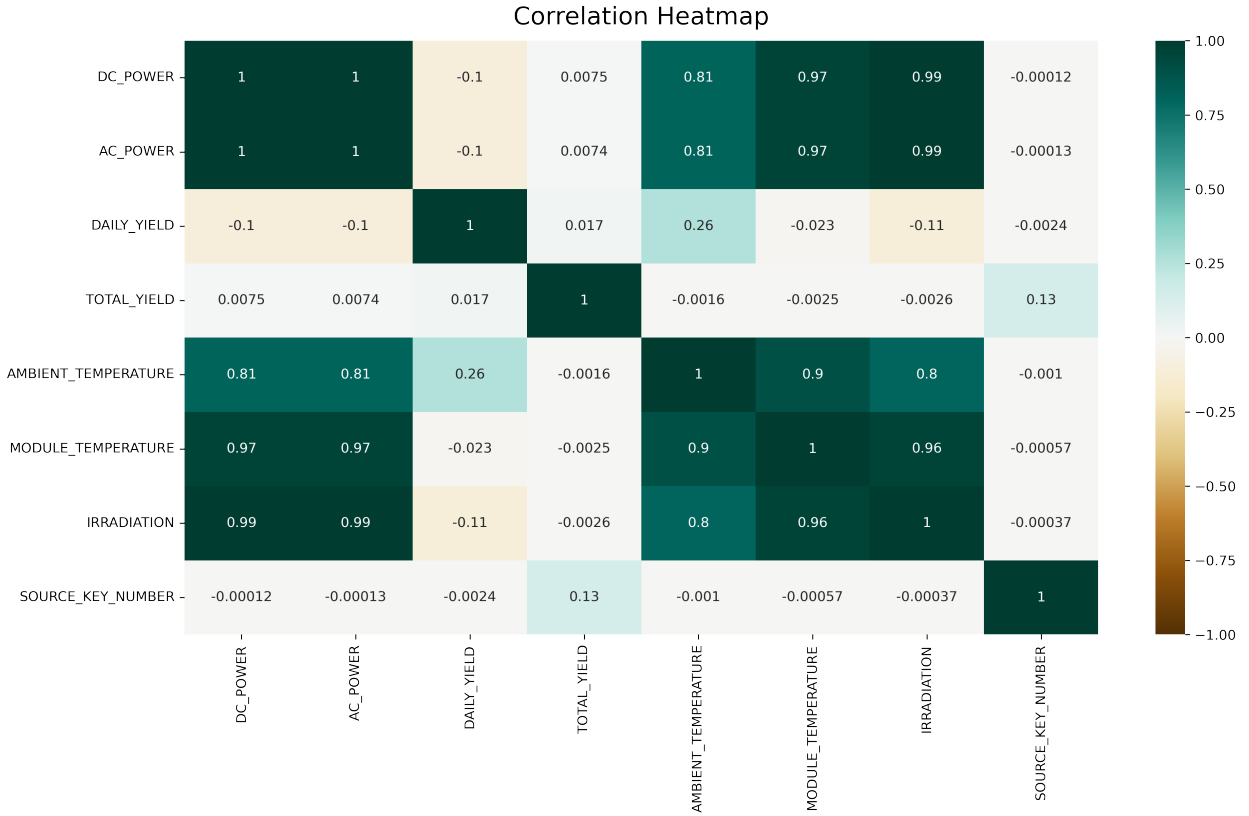

Figure 3.1: Depicts the Pearson Correlation heatmap between all the features in the plant solar generation data set.

From the correlation heatmap, we can identify which features are highly correlated with each other and should be removed from the model to reduce multicollinearity.

3.1.2 Data Distribution

The Y space consists of the AC_POWER feature. Looking at the first 4 statistical moments the distribution appears to be centered and approximately normal and has no heavy skewness however it is important to note that the data is leptokurtic with a high peak and heavy tails which is a common occurrence in time series data and the assumption of normality in terms of kurtosis can be relaxed since the data size is significantly large.

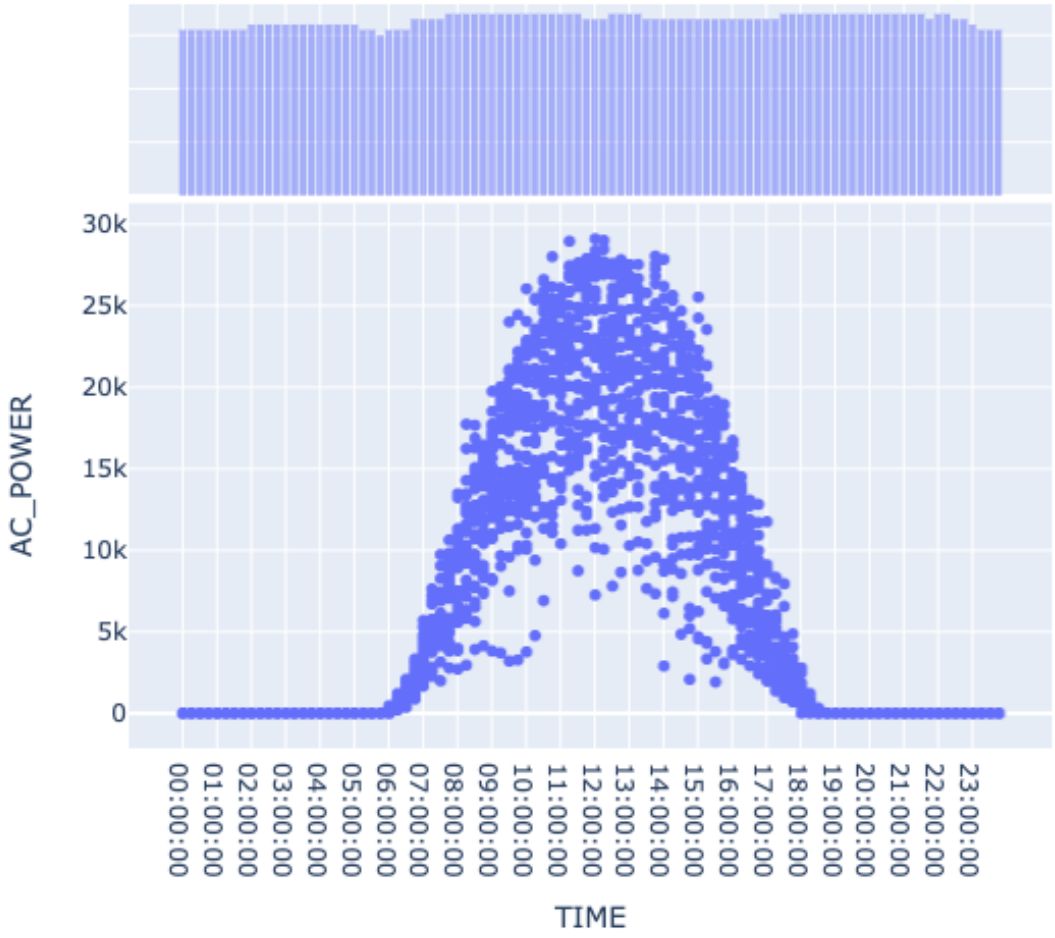

Figure 3.2: Data distribution of AC_POWER

3.2 Decision Tree Regressor

The Decision Tree Regressor is a tree-based machine learning model that aims to predict continuous target values based on input features. It recursively divides the input space into non-overlapping regions, and each region is assigned an average target value calculated from the training data. The tree is constructed by selecting the best feature and threshold at each node to minimize the mean squared error (MSE) of the target values.

3.2.1 Algorithm

1. **Initialization**: Start with the entire dataset at the root node.
2. **Node splitting**: At each node, iterate through all features and potential thresholds to find the best feature and threshold that minimizes the MSE.
3. **Recursion**: Recursively split the dataset into two subsets based on the best feature and threshold found in step 2, and create the left and right child nodes.
4. **Stopping criteria**: Stop splitting if any of the following conditions is met:
   - The maximum depth of the tree is reached.
   - The number of samples in the node is less than the minimum samples required to split.
5. **Prediction**: Assign the mean target value of the samples in a node as the prediction value for that node (leaf node).

3.2.2 Mean Squared Error (MSE)

The mean squared error (MSE) is used as a splitting criterion to minimize the impurity in the child nodes. It is calculated as follows:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{3.1}$$

where n is the number of samples in a node, y_i is the target value of the i-th sample, and $\bar{y}$ is the mean target value of all samples in the node.

The best feature and threshold to split a node are selected by minimizing the weighted sum of the MSEs in the left and right child nodes. The weighted sum of the MSEs is calculated as follows:

$$\text{weighted MSE} = \frac{n_{\text{left}}}{n}\text{MSE}_{\text{left}} + \frac{n_{\text{right}}}{n}\text{MSE}_{\text{right}} \tag{3.2}$$

where n_{left} and n_{right} are the number of samples in the left and right child nodes, respectively, and MSE_{left} and $\text{MSE}_{\text{right}}$ are the MSEs of the left and right child nodes, respectively. The feature and threshold that minimize the weighted MSE are selected for the split.

3.2.3 Benefits

1. Decision trees are easy to interpret and visualize, making them useful for explaining the model to non-technical stakeholders.

2. They can handle both continuous and categorical input features, which allows for flexibility in the type of data that can be used for training the model.

3. Decision tree models require little preprocessing, such as normalization or scaling, which can save time and effort during the data preparation phase.

3.2.4 Limitations

1. Decision trees are prone to overfitting, especially when they are deep. This can lead to poor generalization performance on new, unseen data.

2. They are sensitive to small changes in the training data, which can lead to different tree structures and affect the model's performance.

3. Decision tree models can be biased towards features with many levels, which can lead to unequal importance being placed on different features in the dataset.

3.3 Gradient Boosting Regressor

Gradient Boosting Regressor is an ensemble learning technique that builds a strong model by combining the predictions of multiple weak models, typically decision trees. It works by iteratively fitting decision trees to the residual errors of the previous model, thereby minimizing the overall loss.

3.3.1 Algorithm

1. **Initialize**: Calculate the bias term as the mean of the target variable, and initialize the predictions with this bias term.

2. **Boosting rounds**: Perform a given number of boosting rounds ($n_{\text{estimators}}$), where each round consists of the following steps:

   (a) Calculate the residuals between the target values and the current predictions.

   (b) Train a decision tree regressor on the residuals, using a specified maximum depth and minimum number of samples required to split a node.

   (c) Update the predictions by adding the weighted predictions of the trained decision tree, scaled by a learning rate (shrinkage parameter).

3. **Prediction**: Predict the target values for given input features by summing the bias term and the weighted predictions from each decision tree.

3.3.2 Learning Rate

The learning rate, or shrinkage parameter, is a positive scalar that scales the contribution of each decision tree to the overall model. A smaller learning rate results in a more conservative

model, reducing the risk of overfitting, but may require more boosting rounds to achieve optimal performance.

3.3.3 Benefits

1. Gradient Boosting Regressor can model complex nonlinear relationships between input and output variables.

2. It can handle noisy and high-dimensional data, as well as data with missing values.

3. Gradient Boosting Regressor generally performs well on a wide range of problems, often outperforming other techniques.

3.3.4 Limitations

1. Gradient Boosting Regressor can be sensitive to the choice of hyperparameters, such as the number of boosting rounds, learning rate, and tree depth.

2. It can be computationally expensive and time-consuming to train, especially for large datasets and a high number of boosting rounds.

3. The interpretation of the results is not as straightforward as in simpler models, such as linear regression.

3.4 Kernel Partial Least Squares (KPLS) Regression

Kernel Partial Least Squares (KPLS) Regression is an extension of the Partial Least Squares (PLS) Regression that utilizes kernel functions to model complex nonlinear relationships between input and output variables. It combines the feature extraction abilities of kernel

methods, such as the Support Vector Machine (SVM), with the dimensionality reduction and regression capabilities of PLS.

3.4.1 Algorithm

1. **Kernel matrix**: Compute the kernel matrix K using a chosen kernel function (e.g., RBF kernel) that maps the input features to a higher-dimensional space.

2. **Centering**: Center the kernel matrix K and target values Y.

3. **Iterative decomposition**: For a given number of components, iteratively extract the latent variables (scores) T and U, and the weight vectors P.

    - Initialize U as the centered target values Y.
    - Compute the score vector T as the product of the centered kernel matrix K and U, and normalize T.
    - Compute the weight vector P as the product of the centered kernel matrix K and T, and normalize P.
    - Update the centered kernel matrix K by subtracting the outer product of T and T times K.

4. **Regression**: Calculate the regression coefficients W by solving the linear system T * W = Y.

5. **Prediction**: Predict the target values Y_pred using the product of the scores T and the regression coefficients W.

3.4.2 Radial Basis Function (RBF) Kernel

The RBF kernel is a popular choice for nonlinear kernel methods because of its flexibility and smoothness. It is defined as:

$$K(x, x') = exp(-\gamma||x - x'||^2) \tag{3.3}$$

where x and x' are two input feature vectors, and γ is a positive parameter that controls the shape of the kernel. A larger γ results in a more flexible, localized model, while a smaller γ leads to a smoother, more global model.

3.4.3 Benefits

1. KPLS can model complex nonlinear relationships between input and output variables.
2. It combines the advantages of kernel methods with the dimensionality reduction and regression capabilities of PLS.
3. KPLS can handle collinear and noisy input features.

3.4.4 Limitations

1. KPLS requires the selection of a suitable kernel function and its parameters.
2. It can be computationally expensive for large datasets due to the need to compute and store the kernel matrix.
3. The interpretation of the results is not as straightforward as in linear PLS.

Chapter 4

Results

Model	R-squared	MSE	RMSE
Decision Tree Regressor	0.9453	8780.51	93.70
Gradient Boosting Regressor	0.8137	29927.01	172.99
Kernel PLS	0.8920	-	0.3286

Table 4.1: Comparison of Regression Model Performance

Chapter 5

Conclusions

This project, explored and compared the performance of three different regression models: Decision Tree Regressor, Gradient Boosting Regressor, and Kernel Partial Least Squares (KPLS). The study found that KPLS had the strongest performance on the solar power plant data.

It is important to note that the data has limitations and may not accurately predict power plant output given seasonal or climate changes. Additionally, ethical concerns exist regarding the potential misuse of the model by a solar plant to overcharge after predicting the optimal price point.

Future considerations should include exploring methods that are specifically designed to handle sparse data, such as Orthogonal Matching Pursuit. Additionally, ethical considerations must also be taken into account to prevent the misuse of the model's predictions. Overall, this project demonstrates the potential of machine learning models in accurately predicting solar power plant output and their contribution to promoting renewable energy sources for a sustainable future.