

DATA MINING PROJECT REPORT

Submitted by



**Bala Surya Teja Nandamuri ( WV7708)**

**Jani Pasha Mohammed ( XW1139 )**

**Vedantini Bogawat (EH9515 )**

**Punya Reddy Katpally ( BJ8603 )**

**Sadvilas Buddiga ( JX8549 )**

**Project Summary**

Our project focuses on using weather and environmental data to predict solar energy production accurately. Our aim was to develop a model that could forecast solar energy production anywhere and at any time. We obtained our data set from a reliable source.

To achieve our goal, we use various data mining techniques such as neural networks, linear regression, decision trees, and random forests. We preprocessed the raw data to remove unnecessary values and identify the optimal characteristics for our prediction model. We selected these algorithms based on their ability to manage large datasets and accurately predict continuous values.

We assessed the efficacy of our prediction model using a variety of measures, including Mean absolute percentage error (MAPE), root mean squared error (RMSE), and R-squared value. Additionally, the greatest features, those that are most crucial for forecasting solar power generation were discovered.

Our data mining findings show that we were able to create a dependable predictive model that predicts solar power generation properly depending on the numerous criteria considered. When deciding how much electricity can be produced at a specific location and time, our model can be helpful for people and organizations interested in solar energy generation. Our model can be useful for individuals and organizations interested in solar energy generation to make informed decisions about the amount of power that can be generated at any given location and time.

**Introduction**

Our project uses data mining to accurately predict solar power generation based on numerous factors such as date, time, location, weather, and environmental conditions. Predicting solar power output is important for efficient energy management and optimal use of renewable resources. By analyzing historical data patterns and relationships, we can make real-time predictions that assist in balancing electricity supply and demand, leading to a more stable and reliable power grid.

However, predicting solar power output is challenging due to the complexity of underlying factors such as cloud cover and intermittent weather patterns. Accurate predictions require high-quality and available data and proper mining algorithms. Despite these limitations, data mining insights can lead to informed predictions that help energy providers, policymakers, and consumers optimize their energy usage and harness the full potential of solar energy resources.

Our data mining project aims to improve the accuracy of solar power predictions, optimize energy utilization, and contribute to a more sustainable and efficient energy ecosystem.

**Main Chapter**

**Understanding the Purpose of the Project.**

Our project's goal is to forecast solar energy production utilizing a variety of environmental and meteorological variables. The project's goal is to show that these variables may be used to predict solar power generation accurately without the requirement for irradiance data, which can be difficult to estimate precisely. The research intends to give a more effective and precise technique to anticipate solar power output, which is crucial for integrating solar photovoltaics into conventional electrical grid systems, by depending purely on location and meteorological data. The analysis focuses on forecasting power output from horizontal photovoltaics and uses publicly available data from twelve distinct locations over 14 months. The initiative seeks to maximize the production and distribution of renewable energy, particularly solar energy and enhances the integration of solar photovoltaics into conventional electrical grid systems and is one of the most important renewable energy sources in the world.

**Obtain Data for Analysis**

In our dataset, there are records from twelve separate locations over 14 months. In our data set, there are total records of 21045 rows and seventeen columns. These are the records of our raw dataset. There are a total of sixteen predictors and one outcome variable.

.A screenshot of a computer code

Description automatically generated with low confidence

**Explore, Clean and Preprocess Data.**

Below are the names of columns in our dataset.

A picture containing text, screenshot, font, line

Description automatically generated

In our dataset all the data types are either int64 or float64, except for Location, and Season which are object data types. We will now be dropping undesired variables which look not especially useful for our process and then converting the other object data types.

A screenshot of a computer

Description automatically generated with medium confidence

**Changing of Data Types**

In our dataset we will be changing Season datatype which is Object data type into category datatype for our analysis purpose.

A screenshot of a computer code

Description automatically generated with low confidence

**Dropping of Undesired Columns**

We drop a few columns which are undesired or add significance to our analysis.

A screenshot of a computer

Description automatically generated with medium confidence

**Dimensions of Data frame after dropping of undesired variables**

A picture containing text, screenshot, font, number

Description automatically generated

**Conversion of Titles to One-Worded Titles**

A screenshot of a computer program

Description automatically generated with low confidence

**Checking of Null Values**

A screenshot of a computer

Description automatically generated with medium confidence

**Datatypes after conversions**

A screenshot of a computer

Description automatically generated with medium confidence

**Determining of Data Mining Task**

In this case, predictive modeling refers to the process of using data mining to forecast solar power generation. To do this accurately, the project utilizes a dataset that includes environmental and weather factors. To ensure accuracy, it is necessary to create and evaluate predictive models.

**Partition of Data.**

To avoid overfitting, the dataset is divided into two parts: a training partition and a validation partition. The training partition is used to develop and train the classification model, while new data is utilized to evaluate the model's effectiveness. To ensure sufficient data for training and proper testing of predictive power, the dataset is split into a 70% training partition and a 30% validation partition for this project.

**Using Data Mining Techniques.**

* **Multiple Linear Regression**

The multiple linear regression model is a helpful tool used in predictive analytics to explain and predict a quantitative outcome variable (also known as the response or predicted variable) based on a group of independent variables (also known as predictors, input variables, or regressions). By applying a linear equation to the data, this model establishes the relationship between the independent variables and the outcome variable, identifying the most effective coefficients for predicting the outcome variable. It is a useful tool for predicting outcomes and finding significant predictors.

**Building of Linear Regression with All the Predictors.**

A screenshot of a computer code

Description automatically generated with medium confidence

**The output of the Above Model**

A screenshot of a computer

Description automatically generated with medium confidence

**Model Equation for above model**

**Energy Production / Polypwr =** 1278.34 - 00.0 Date - 0.02 Time – 0.015 Latitude + 0.04 Longitude -0.00 Altitude -0.31 Month + 2.51 Hour – 0.06 Humidity 0.19 AmbientTemp + 0.07 Wind.Speed + 0.06 Visibility -0.03 Pressure +0.00 Cloud.Ceiling + 2.81 Season\_Spring + 2.36 Season\_Summer -1.66 Season\_Winter

Here is an equation for a multiple linear regression model that predicts the price of a product based on fifteen varied factors. The constant term in the equation is 1278.34, which is the predicted energy when all factors are at zero. Each factor has a coefficient that indicates how much it impacts the price and in what direction. A positive coefficient means the factor increases energy production, while a negative coefficient means it does not.

**Actual vs Predicted Values from Linear Regression Model**

A screenshot of a computer program

Description automatically generated with low confidence

**Finding of Performance Measures for Training and Validation Data Set**A screenshot of a computer program

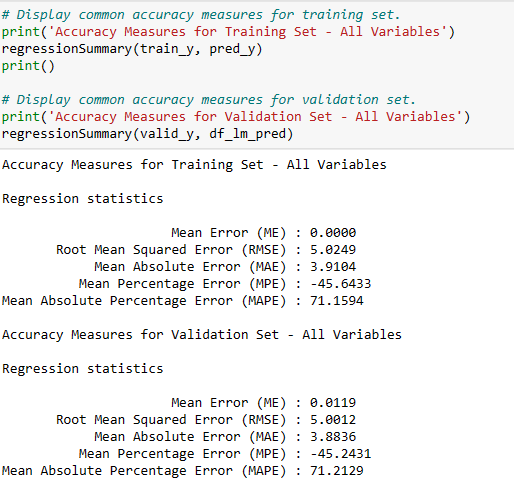
Description automatically generated with low confidence

A picture containing text, screenshot, font, number

Description automatically generated

The R2 and Adjusted R2 for the validation set are better than that of the Training Data Set which are 0.511 and 0.509 while training has 0.501 and 0.5, which shows that there is no sign of overfitting.

Also, the AIC and BIC scores for validation partitions are much lower than that of the training model which shows that model performs better on validation period and is considered as a good fit model for our prediction.



The above are accuracy measures for Linear Regression Model. From the sniper we can observe that there is no overfitting of data as there is extraordinarily little difference between RMSE and MAPE of training and validation period.

A screenshot of a computer

Description automatically generated with medium confidence

**Using Different Algorithms for Models**

Use Python’s sci-kit-learn package and, specifically, the Linear Regression algorithm (function) to predict energy production for training data.

* **Backward Elimination Algorithm**



The output displayed above represents the backward elimination algorithm, which is a method used in linear regression models to select features. The algorithm starts with all variables in the model and then eliminates the least important variable iteratively until a stopping criterion is met. Initially, none of the variables are removed as the starting score is 89404.34.

The algorithm then selects the best variables, including "Date," "Time," "Latitude," "Longitude," "Altitude," Month," Hour," Humidity," AmbientTemp," and Wind.Speed, Visibility, Pressure, "Cloud.Ceiling," "Season\_Spring," "Season\_Summer," and "Season\_Winter" are chosen based on their importance in predicting the outcome variable. Overall, the backward elimination algorithm improves accuracy and simplifies the model by removing unnecessary variables.

**Linear Regression Model with Backward Elimination Method**

A screenshot of a computer program

Description automatically generated with medium confidence

**Model Equation**

**Energy Production / Polypwr =** = 1267.28 - 00.0 Date - 0.03 Time – 0.015 Latitude + 0.04 Longitude -0.01 Altitude -0.31 Month + 2.57 Hour – 0.06 Humidity + 0.19 AmbientTemp + 0.07 Wind.Speed -0.03 Pressure +0.00 Cloud.Ceiling + 2.80 Season\_Spring + 2.35 Season\_Summer -1.67 Season\_Winter

**Prediction and Accuracy Measures of Linear Regression (Backward Elimination)**

A screenshot of a computer program

Description automatically generated with medium confidence

**Forward Selection Algorithm**

A screenshot of a computer program

Description automatically generated with low confidence

The output shows the results of a forward selection algorithm for a multiple linear regression model. The algorithm starts with a constant and gradually adds the most important variable until no further improvement is observed. The output lists the performance metrics for each iteration, along with the order in which variables were added to the model and their corresponding scores. The best variables are listed first in the final model list, which is displayed at the end of the output. In this situation, the best variables are 'AmbientTemp', 'Cloud.Ceiling', 'Season\_Spring', 'Altitude', 'Season\_Summer', 'Latitude', 'Humidity', 'Longitude', 'Wind.Speed', 'Month', 'Season\_Winter', 'Date', 'Pressure', 'Visibility' are the subsequent terms.

**Linear Regression with Forward Selection**

A screenshot of a computer program

Description automatically generated with medium confidence

**Model Equation**

**Energy Production / Polypwr** = 1246.93 - 00.18 AmbientTemp + 0.01 Cloud.Ceiling + 2.79 Season\_Spring – 0.00 Altitude + 2.37 Season\_Summer -0.16 Latitude -0.06 Humidity +0.04 Longitude + 0.07 Wind.Speed -0.31 Month – 1.68 Season\_Winter -0.03 Pressure -0.00 Date + 0.08 Visibility

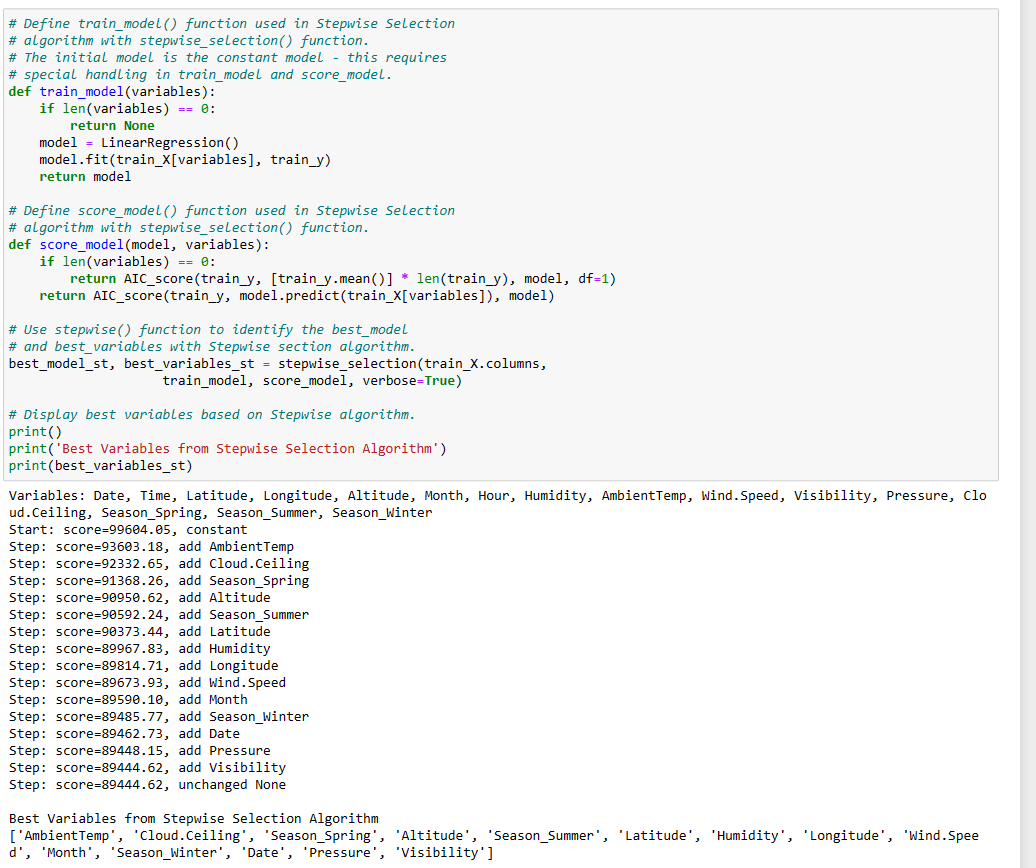
**Prediction and Accuracy Measures of Linear Regression (Forward Selection)**

A screenshot of a computer program

Description automatically generated with low confidence

* **Stepwise Algorithm**

The output displays the variables that were chosen by a stepwise selection algorithm, which began with a constant and either added or removed variables in each step. The algorithm chose the variable that improved the model the most based on a specific criterion (in this case, the score). In this instance, the algorithm started with a constant and gradually added variables until no further improvements could be made. The output shows the score, which is a measure of model fit, at each step, and the final chosen variables are listed at the end.



The stepwise selection algorithm utilized here combines forward and backward selection. It begins by adding variables one by one (forward selection) and then removes variables that do not enhance the model (backward elimination) until the best model is discovered. The chosen variables are: "AmbientTemp", "Cloud.Ceiling", "Season\_Spring", "Altitude", "Season\_Summer", "Latitude", "Humidity", "Longitude", "Wind.Speed", "Month", "Season\_Winter", "Date", "Pressure", and "Visibility". These variables were determined to be the most predictive of the outcome variable in the data.

**Linear Regression with Stepwise Selection**

A screenshot of a computer program

Description automatically generated with medium confidence

**Model Equation**

**Energy Production / Polypwr** = 1246.93 + 0.18 AmbientTemp + 0.01 Cloud.Ceiling +2.79 Season\_Spring -0.00 Altitude + 2.37 Season\_Summer – 0.16 Latitude -0.06 Humidity + 0.04 Longitude +0.07 Wind.Speed -0.31 Month -1.68 Season\_Winter -0.03 Pressure -0.00 Date +0.08 Visibility

**Prediction and Accuracy Measures of Linear Regression (Stepwise)**

A screenshot of a computer program

Description automatically generated with medium confidence

**Decision Trees**

Decision trees, also known as trees or CART (Classification and Regression Trees), are a highly effective method for classification and prediction in data mining. This well-liked machine learning technique is easy for analysts to use, and customers can easily understand it. Decision trees allow for both classification and prediction, with regression trees used for prediction and classification trees for categorizing or forecasting outcomes based on a set of predictors.

The decision tree algorithm creates a set of rules that are easy to understand and interpret through tree diagrams. It does this by iteratively dividing the data into subsets to determine the most important predictor. This process is repeated recursively until a stopping criterion, such as the maximum depth of the tree or the minimum number of observations in each terminal node, is reached. Decision trees are versatile in handling both categorical and continuous predictors, as well as missing data, while also being resistant to data noise and outliers. However, overfitting can be an issue, resulting in deficient performance with new data.

In our scenario, we must predict numeric variables, so we will be using a **regression tree.**

Regression trees are a type of decision tree that predicts numerical outcomes based on a set of pre-determined predictors. Unlike classification trees, they calculate the average value of the numerical target variable instead of using a majority vote to make predictions. The impurity of a node is determined by the total squared deviations from the mean of the target variable. To evaluate regression trees, the Root Mean Squared Error (RMSE) is commonly used instead of the accuracy proportion from the confusion matrix used for classification trees.

**One can create regression trees using the DecisionTreeRegressor () function in Python.**

**Creation of Decision Tree**

A screenshot of a computer code

Description automatically generated with low confidence

The best parameters for our decision tree would be Max Depth of nine and Minimum Impurity of 0.004 and Min Sample Split is twenty-eight.

**Plotting of Decision Tree**

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing screenshot

Description automatically generated

**The number of Nodes in our Decision Tree is 489.**

A screenshot of a computer code

Description automatically generated with low confidence

**Accuracy Measures of our Decision Tree**

A screenshot of a computer screen

Description automatically generated with medium confidence

From accuracy measures we can observe that the RMSE and MAPE values of Regression Tree for Validation partition is slightly higher than of Training Partition, which shows the model performs on Training Partition, Since the difference is small, we can still use the model for our prediction.

**Prediction for Above Model**

A screenshot of a computer

Description automatically generated with medium confidence

**KNN**

A nonparametric method called the k-Nearest Neighbors method can be used to categorize results or forecast numerical results. In contrast to parametric methods, k-NN does not presuppose a specific relationship between the predictor variables and the result. It is data-driven and avoids estimating the parameters of a predetermined function form. K-NN is a widely used method in machine learning because it is easy to use and highly automated.

We perform Data Normalization in the KNN method and later in Neural Nets. For Data Normalization

To use the k-NN classifier, you must first standardize (normalize) the original dataset and its partitions using the StandardScaler () function from the sci-kit-learn (sklearn) library.

For any numeric value xj in column j, the standard score or scaled value Zj is calculated as follows:

**Zj = (xj – Uj)/Sj.**

Here, Uj represents the meaning of values in column j.

Sj represents the standard deviation of values in column j.

**Generation of Normalized Values using StandardScaler ()**

A screenshot of a computer code

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

**Using of KNN method for Numerical outcome**

To determine the class, it is recommended to use the average of response values instead of relying on the majority vote. This average can be weighted, with decreasing weight as distance increases.

The best k can be identified using the RMSE (root mean squared error) measure. To implement this, use the KNeighborsRegressor () function from the sci-kit-learn (sklearn. neighbors) library in Python.

Identifying of the best value of k for our Model

A screenshot of a computer program

Description automatically generated with medium confidence

A picture containing text, screenshot, font, typography

Description automatically generated

In the above snippet we can see that the best k for our model would be k=11, despite the lower RSME values can be observed at another k, this can be explained from the below elbow chart.

**Plotting of the Elbow Chart**

A screenshot of a computer code

Description automatically generated with low confidence

A blue line on a white background

Description automatically generated with medium confidence

In the above chart we can see that value after k-11 started to increase, based on the chart we can determine that the best value can be k=11

A picture containing text, screenshot, font, line

Description automatically generated

**Accuracy Measures of Training and Validation Method for KNN Model**

A screenshot of a computer

Description automatically generated with medium confidence

In the above model we can observe that the values of the RMSE and MAPE for training partition are lower than that of the Validation Partition, it shows the model works better on the training partition than on the validation, but since the difference of values are not very significant, but we consider it for our predictions.

**Predicted Data using KNN method.**

A screenshot of a computer

Description automatically generated with medium confidence

**Neural Networks**

Models for classification and prediction include neural networks, also known as artificial neural networks. They draw their inspiration from the brain's biological processes, where neurons are connected and pick up added information. To learn and generalize from specific details, neural networks mimic how humans do. Contrary to other predictive models, they can capture intricate relationships between predictors and responses. The high predictive performance of neural networks is their primary strength, making them a preferred option for data analysis.

Neural network models are used to capture the complex relationships between input variables and results. Typically, these models have multiple layers: an input layer, a hidden layer, and an output layer. During training, the model coefficients and parameters are continuously adjusted based on the network's interim performance. The wonderful thing about neural networks is that they can learn these relationships on their own, without the user having to specify them. The hidden layers between the input and output layers accept input values while succeeding layers receive inputs from earlier layers. This allows non-linear relationships between predictors and results to be captured, while still achieving high predictive performance.

If you need to predict numeric outcomes using neural networks, the scikit-learn library has MLPRegressor () that can help. This model has parameters and default values like MLPClassifier (), including max\_iter, hidden\_layer\_sizes, learning rate, and learning\_rate\_init. To achieve accurate results, it is suggested to use one node for the output layer and apply the function regressionSummary () to track accuracy measures. Since numeric predictors can have a wide range of values, it is recommended to use StandardScaler () from the sci-kit-learn library to normalize the scaling of both training and validation data. By scaling the training predictors, you can improve the accuracy of the predictions.

**Scaling of Values using StandardScaler ()**

A screenshot of a computer

Description automatically generated

**Implementation of Neural Net**

Neural networks can be implemented in two ways: Multilayer Perceptrons (MLP). For categorical outcomes, sci-kit-learn's MLPClassifier () is used, while MLPRegressor () is used for numerical outcome prediction. Python provides TensorFlow, Keras, and PyTorch for more complex deep-learning networks. MLPs are a classical type of neural network.

**Using of GridSearchCV for the best training of Neural Net**

A screenshot of a computer code

Description automatically generated with medium confidence

**Implementing of Neural Net**

A screenshot of a computer program

Description automatically generated with medium confidence

**Final Intercepts of Our Neural Net Model**

A screenshot of a computer code

Description automatically generated with low confidence

The final intercepts in the power generator neural network model are the bias terms of the neurons in the output layer. The first array of intercepts corresponds to the bias terms of the twelve nodes (neurons) in the hidden layer, while the second array corresponds to the bias term of the one node (neuron) in the output layer.

These bias terms are added to the weighted sum of inputs to a neuron before passing it through an activation function. They help the network to model more complex relationships between inputs and outputs by adjusting the output of the neuron based on the input.

The final intercepts represent the learned bias values that the neural network has determined are necessary to make accurate predictions for power generation based on the input features.

**Below are the network weights used.**

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer code

Description automatically generated with low confidence

The first array consists of network weights of sixteen nodes in the input layer to the nodes of twelve nodes hidden layer, The second array consists of weights twelve nodes in hidden layer to the one node in output layer.

**Predictions of Neural Net Model**

A screenshot of a computer

Description automatically generated

**Accuracy Measures of Neural Net Model**

A screenshot of a computer

Description automatically generated with medium confidence

The above snippet is accuracy measures of the Neural Network Model. We can observe the MAPE and RMSE values that there is no overfitting, as there is no significant difference between the RMSE and MAPE of the training and Validation partition.

**Comparison of Accuracy Measures and Choosing Best Model**

**Accuracy Measures of Linear Regression Model**

A screenshot of a computer

Description automatically generated with medium confidence

**Accuracy Measures using Backward Elimination**

A screenshot of a computer error

Description automatically generated with low confidence

**Accuracy Measures using Forward Selection**

A screenshot of a computer error

Description automatically generated with low confidence

**Accuracy Measures using Stepwise Selection**

A picture containing text, screenshot, font

Description automatically generated

**Accuracy Measures of Regression Tree Model**

A screenshot of a computer error

Description automatically generated with low confidence

Accuracy Measures of KNN

A screenshot of a computer error

Description automatically generated with low confidence

**Accuracy Measures of Neural Network**

A screenshot of a computer error

Description automatically generated with low confidence

|  |  |  |
| --- | --- | --- |
| **Model** | **RMSE** | **MAPE** |
| **Linear Regression** | **5.0012** | **71.2129** |
| **Backward Elimination** | **5.0004** | **71.1141** |
| **Forward Selection** | **5.0039** | **71.2022** |
| **Stepwise Selection** | **5.0039** | **71.2022** |
| **Regression Tree** | **4.6020** | **54.9150** |
| **KNN Model** | **4.4133** | **56.5267** |
| **Neural Network** | **4.4221** | **56.2825** |

From the above model we can see that RMSE is lowest for KNN Model which has the best RMSE of 4.4133 among all the models, while the MAPE is best for the Regression Tree which is 54.9150. The RMSE for Neural Net and KNN model are close that of 56.2825 and 56.5267 respectively, even the RMSE and is similar among KNN and Neural Net Model, while the models have higher RMSE and MAPE than the Regression Tree, KNN and MAPE

When assessing a model's performance, it is important to consider both RMSE and MAPE. RMSE measures the average deviation between predicted and actual values, while MAPE measures the percentage deviation. The choice between the two measures depends on the specific context and requirements of the problem. RMSE is more suitable when minimizing the magnitudes of errors is crucial.

Here we can say that KNN Model is best Model as it as it got the Best RMSE of 4.4133 among all the model compared, since we are measuring values are based on magnitude, also it has similar MAPE of Neural Net Model which has second best MAPE after Regression Tree. We can consider Neural Net Model followed by Regression Tree as it has the second best RMSE and MAPE values.

**Ranking of Models**

1. KNN Model
2. Neural Net Model
3. Regression Model
4. Backward Elimination
5. Linear Regression
6. Stepwise Selection = Forward Selection

**Conclusion**

After analyzing the implemented models, it was found that the KNN model performed the best, followed by the neural network and regression tree models. These models have shown to be successful in predicting the target variable with low errors. The KNN model demonstrated the lowest RMSE and MAPE values, indicating its reliability for this project's target variable prediction.

Although the Neural Network and Regression Tree models produced promising results, they were outperformed by the KNN model. Therefore, it is recommended to use the KNN model for future target variable forecasting due to its high accuracy. However, it is important to keep in mind that the performance of the model may be affected by the specific dataset and issue at hand. It is always advisable to evaluate the model's performance on fresh data before deploying it in a production environment.

The use of data mining methods and algorithms has been effective in predicting the target variable in this project. These models provide valuable insights into the relationship between the predictors and the target variable, enabling informed decision-making.

Nevertheless, it is important to acknowledge the limitations of these models. Since they rely on historical data, they may not accurately predict how the data will change in the future. Furthermore, the quality of the training data plays a vital role in the models' performance. Therefore, it is crucial to ensure that the data used to train the models is relevant and of high quality to address the current problem.

To conclude The KNN model has proven to be the most accurate in predicting the target variable, and data mining techniques and algorithms have been successfully implemented in this project. Data mining can offer insightful insights and support informed decision-making, but it is critical to be aware of the restrictions and guarantee data quality.

**Limitations**

Using data mining techniques, algorithms, and methods in this project requires accurate and pertinent data. The performance of the models can be significantly impacted by the quality and quantity of data available for analysis. It is important to carefully assess the data and select appropriate models based on specific presumptions. However, not all models may be suitable for all kinds of data and issues.

Another drawback of data mining techniques is the potential for overfitting the models to the training data. This can result in subpar performance when used with new and unseen information. To prevent overfitting, it is essential to employ methods like cross-validation and regularization to ensure that the models are generalized and work well on fresh data.

**Bibliography**

* [**https://towardsdatascience.com/predicting-solar-power-output-using-machine-learning-techniques-56e7959acb1f**](https://towardsdatascience.com/predicting-solar-power-output-using-machine-learning-techniques-56e7959acb1f)
* [**https://www.kaggle.com/datasets/saurabhshahane/northern-hemisphere-horizontal-photovoltaic**](https://www.kaggle.com/datasets/saurabhshahane/northern-hemisphere-horizontal-photovoltaic)
* [**https://data.mendeley.com/datasets/hfhwmn8w24/5**](https://data.mendeley.com/datasets/hfhwmn8w24/5)
* [**https://kanoki.org/2020/05/13/decision-tree-in-sklearn/**](https://kanoki.org/2020/05/13/decision-tree-in-sklearn/)