SOLAR RADIATION PREDICTION

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

MARRI BALASATISH

Regd. No.: 20B91A05H2

Under Supervision of Mr. Gundala Nagaraju

Henotic Technology Pvt Ltd, Hyderabad

(Duration: 7th July, 2022 to 6th September, 2022)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(An Autonomous Institution)

Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada

CHINNA AMIRAM, BHIMAVARAM,

ANDHRA PRADESH

Table of Contents

Abstract

Solar radiation is the Earth's primary source of energy and has an important role in the surface radiation balance,hydrological cycles,vegetation photosynthesis, and weather and climate extremes.The accurate prediction of solar radiation is therefore very important in solar industry.

Forecasting the output power of solar systems is required for the good operation of the power grid or for the optimal management of the energy fluxes occurring into the solar system. Before forecasting the solar systems output, it is essential to focus the prediction on the solar irradiance. The global solar radiation forecasting can be performed by several methods; the two big categories are the cloud imagery combined with physical models, and the machine learning models. In this context, the objective of this paper is to give an overview of forecasting methods of solar radiation using machine learning approaches. Although, a lot of papers describes methodologies like neural networks or support vector regression, it will be shown that other methods (regression tree, random forest, gradient boosting and many others) begin to be used in this context of prediction. The performance ranking of such methods is complicated due to the diversity of the data set, time step, forecasting horizon, set up Sand performance indicators. Overall, the error of prediction is quite equivalent. To improve the prediction performance some authors proposed the use of hybrid models or to use an ensemble forecast approach.

1.0 Introduction

An electrical operator should ensure a precise balance between the electricity production and consumption at any moment. This is often very difficult to maintain with conventional and controllable energy production system, mainly in small or not interconnected (isolated) electrical grid(as found in islands). Many countries nowadays consider using renewable energy sources into the electricity grid. This creates even more problems as the resource (solar radiation, wind, etc.) is not steady. It is therefore very important to be able to predict the solar radiation effectively especially in case of high energy integration.

One of the most important challenge for the near future global energy supply will be the large integration of renewable energy sources (particularly non-predictable ones as wind and solar) into existing or future energy supply structure. An electrical operator should ensure a precise balance between the electricity production and consumption at any moment. As a matter of fact, the operator has often some difficulties to maintain this balance with conventional and controllable energy production system, mainly in small or not interconnected (isolated) electrical grid (as found in islands). The reliability of the electrical system then become dependent on the ability of the system to accommodate expected and unexpected changes (in production and consumption) and disturbances, while maintaining quality and continuity of service to the customers. Then, the energy supplier must manage the system with various temporal horizons.

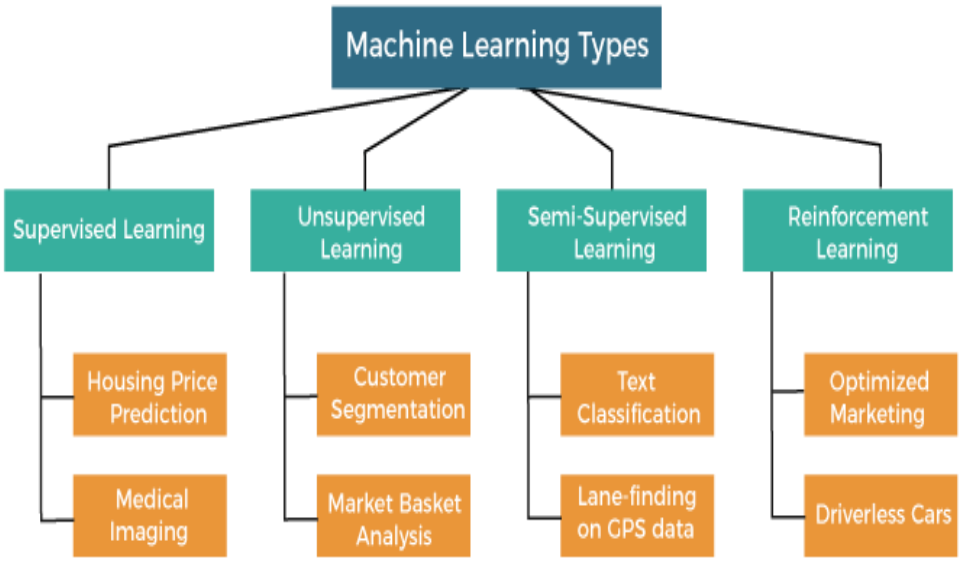
1.1 What are the different types of Machine Learning?

Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and on the basis of training, they build the model & perform a specific task.

These ML algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations, etc.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning



1. Supervised Machine Learning

As its name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

- **Classification**
- **Regression**

a. Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "**Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

b. Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

2. Unsupervised Machine Learning

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the

machine is trained using the unlabelled dataset, and the machine predicts the output without any supervision.

Categories of Unsupervised Machine Learning

Unsupervised Learning can be further classified into two types, which are given below:

- Clustering
- Association

1. Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behaviour.

2. Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in Market Basket analysis, Web usage mining, continuous production, etc.

3. Semi-Supervised Learning

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabelled datasets during the training period.

4. Reinforcement Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance.

1.2 Benefits of Using Machine Learning in Solar Industry

The high availability of data in the energy sector makes it a great environment for machine learning and data science solutions. let's make a fair balance by describing the benefits.

1.Smart Infrasturcture of Solar Energy System

A variety of data sources can be integrated into this decision process: historical weather data, power production data collected from other energy grids, and even simulated load demand data. Such data sources make it possible to integrate advanced ML applications when designing the infrastructure of solar energy grid systems.

ML has already been successfully used to tackle some infrastructure design decisions: from improving solar energy storage to identifying the optimal layout of solar panels.

2.Intelligent maintenance of solar energy plants

The maintenance of solar energy plants is particularly challenging given the properties of renewable energy sources: they are fluctuating and are highly influenced by weather conditions. ML solutions enable energy providers to be proactive and to keep a close eye on the performance and availability of their solar energy plants.

3.Solar energy production forecasting

"For example, to help a utility company prepare for the after-effects of a major storm, the team could mine and model historical data of what kind of damage was caused to power lines or telephone poles, and why. By coupling that with a hyper-local forecast, IBM could help a company plan for how many repair crews would be needed, and where."

This example is not isolated and clearly shows how investing in software and custom machine learning and analytics services can have an impact on the performance and maintenance of expensive hardware (such as solar plants or power networks). By leveraging

forecasted energy production information, grid operators can cut down on operational costs and make informed decisions, based on real-time performance data.

4.optimized transmission & distribution networks

ML algorithms integrate consumption patterns and can power applications dealing with the health of distribution networks. They power preventive maintenance solutions and also make it possible to identify anomalous behaviour (such as theft). Such capabilities allow energy suppliers to maximize the usage of renewable energy and to act on quality and congestion issues before they happen.

5.Understanding the solar energy market

The traditional use of ML algorithms is focused on forecasting market-clearing prices. However, innovative solutions for solar energy generators integrate market data into daily operational and maintenance decisions: technical parameters, demand fluctuations, and grid performance can all be balanced by means of machine learning and real-time data analytics. Such capabilities offer huge untapped potential for both established companies, as well as for startups and energy traders.

1.3 About Industry

Solar power is the energy from the sun that is converted into thermal or electrical energy. Solar energy is the cleanest and most abundant renewable energy source available, and the U.S has some of the richest solar resources in the world. Solar technologies can harness this Energy for a variety of uses, including generating electricity, providing light or a comfortable interior environment, and heating water for domesti , commercial or industrial use.

1.3.1 AI / ML Role in Solar Industry

Artificial intelligence (AI) and machine learning (ML) have become important technology solutions as the industry is constantly looking for ways to cater to the rapidly increasing demand for clean, cheap and reliable energy. These advanced technologies have the potential

to analyze the past , optimize the present, and predict the future .This means that AI and ML have the potential to solve most of the challenges that currently prevail.

With technology making rapid advancements, the renewable energy sector has made significant progress in the last decade. However, there are a few challenges that still prevail that can be addressed with the help of AI and ML.

The demand for renewable energy will increase in the near future which makes it more important for the investment in emerging technologies such as AI, ML, and IOT to improve productivity and overcome the shortfalls.

2.0 Claims

Your institution might have solar panels on-site and use the electrical output to power its facilities, but that does not necessarily mean it can claim to use solar power. The ability to claim “use” of solar electricity from the on-site solar system is contingent on your ownership or exclusive rights to the associated RECs. The requirement to own RECs to substantiate your use of solar energy is true of electricity generation from either a self-owned or third-party owned system, such as through a power purchase agreement (PPA) or solar lease. To make a solar project financially feasible or improve its return on investment, the project’s RECs are often not conveyed to the electricity consumer, but are sold by the project owner or project developer into the open market. Although selling the associated project RECs brings down the delivered cost of electricity to the consumer the consumer cannot in the absence of owning the RECs claim to be using solar power. In these cases, it is the eventual buyer of the project’s RECs that can make the claim of using renewable electricity from the project.

2.1 Internship Project-Data-Link

The data is related with the Solar Radiation Prediction

Data set has 11 predicator variables and 32.6K rows.

The main factors for term Radiation are Temperature, Pressure, Humidity, Speed, WindDirection(Degrees) etc.

The internship project data has taken from Kaggle and the link is:

https://www.kaggle.com/datasets/dronio/SolarEnergy

3.0 AI / ML Modelling and Results

3.1 Your Problem of Statement

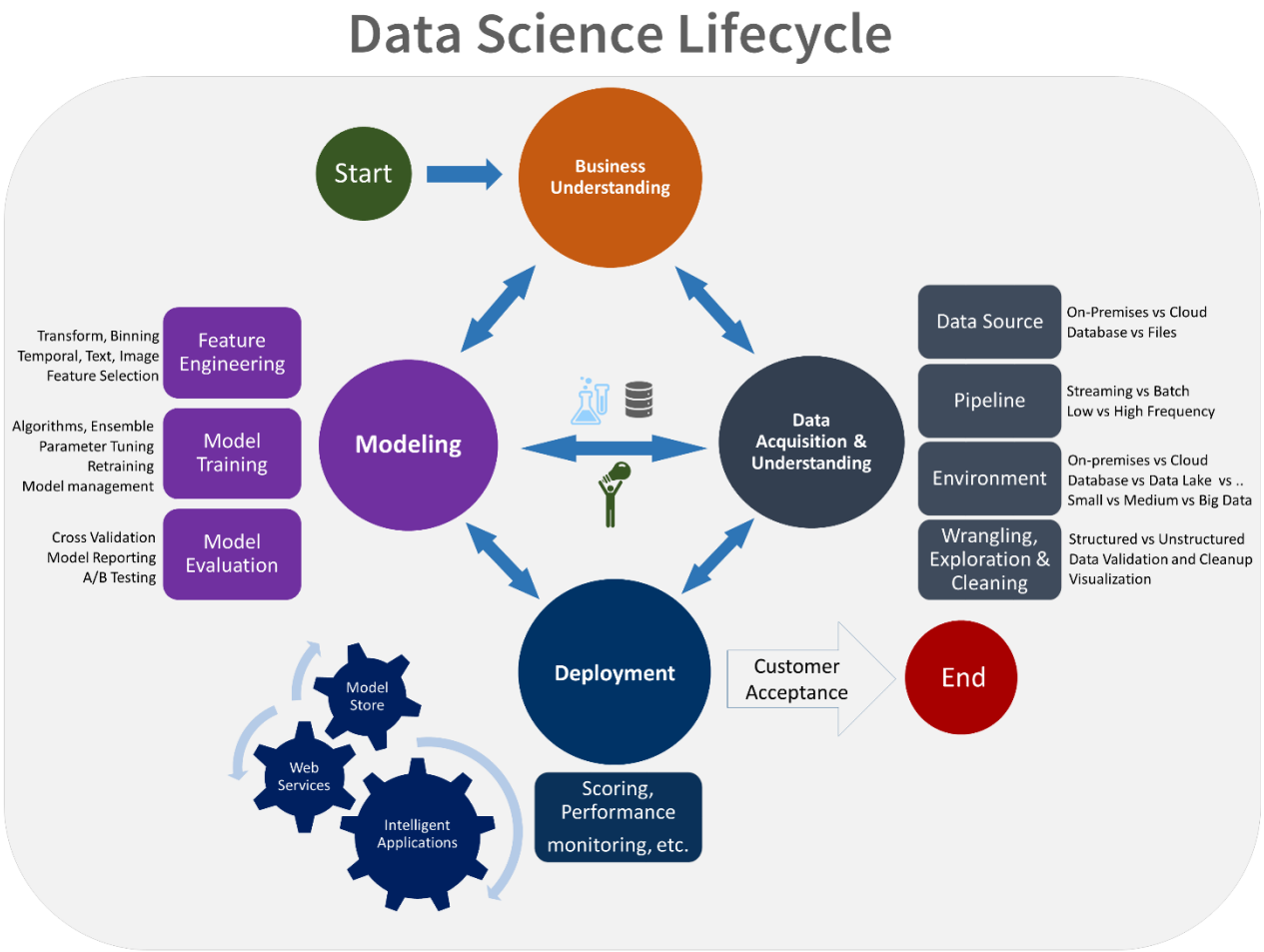
Predictive models are most effective when they are constructed using acompany's own historical claims data since this allows the model to recognize the specific nature of a company's exposure as well as its claims practices. The construction of the model also involves input from the company throughout the process, as well as consideration of industry leading claims practices and benchmarks.

Predictive modelling can be used to quantify the impact to the claims department resulting from the failure to meet or exceed claim service leading practices. It can also be used to identify the root cause of claim leakage. Proper use of predictive modelling will allow for potential savings across two dimensions:

- Early identification of claims with the potential for high leakage, thereby allowing for the proactive management of the claim
- Recognition of practices that are unnecessarily increasing claims settlement payments.

3.2 Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data



3.2.1 Data Exploratory Analysis

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact or target variable.

The exploratory analysis is done for Auto Quote / Policy Conversion with different parameters and all the charts are presented in **Appendices 6.2 - List of charts**

3.2.2 Data Pre-processing

We removed variables which does not affect our target variable(Claimed Target)as they may add noise and also increase our computation time, we checked the data for anomalous data points and outliers. We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

3.2.2.1 Check the Duplicate and low variation data

As companies collect more and more data about their customers, an increased amount of duplicate information starts appearing in the data as well, causing a lot of confusion among internal teams.Having duplicate data means that the model will be trained more on that type of data and thus will be biased. These duplicates add weight to the fit of those specific observations (cases). Whether this effect is big or small is hard to tell from the information we have provided. So that we checked the duplicates in our dataset as they are not required and also increase our computation time.

3.2.2.2 Identify and address the missing variables

The real-world data often has a lot of missing values. Missing values are representative of the messiness of real world data. There can be a multitude of reasons why they occur — ranging from human errors during data entry, incorrect sensor readings, to software bugs in the data processing pipeline. The cause of missing values can be data corruption or failure to record data. The handling of missing data is very important during the pre-processing of the dataset as many machine learning algorithms do not support missing values. We usually use the simple imputer and KNN imputer to identify the missing values.

3.2.2.3 Handling of Outliers

An outlier is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution error. The analysis of outlier data is referred to as outlier analysis or outlier mining. There is no one method to detect outliers because of the facts at the center of each dataset. One dataset is different from the other. A rule-of-the-thumb could be that you, the domain expert, can inspect the unfiltered, basic observations and decide whether a value is an outlier or not. These are different outlier methods for outlier analysis: Box plot, Scatter plot, Mathematical Function. There are different types of outliers: Global Outliers, Contextual Outliers, Collective Outliers.

3.2.2.4 Categorical data and Encoding Techniques

Machine learning models require all input and output variables to be numeric. This means that if your data contains categorical data, you must encode it to numbers before you can fit and evaluate a model. The two most popular techniques are an **Ordinal Encoding and a One-Hot Encoding**. They are 2 types of techniques we are using in this dataset one is label encoder and other is label Binarizer. Label encoder is used for ordinal data and label Binarizer is used for numerical data. The other technique is manual which was used sometimes.

Feature Scaling

Feature scaling is the process of normalising the range of features in a dataset. There are three different types of feature scaling : Min Max Scaler , Standard Scaler, Robust Scaler.

Min Max scaler : Normalisation, also known as min-max scaling, is a scaling technique whereby the values in a column are shifted so that they are bounded between a fixed range of 0 and 1.

Standardization : On the other hand, standardisation or Z-score normalisation is another scaling technique whereby the values in a column are rescaled so that they demonstrate the properties of a standard Gaussian distribution, that is mean = 0 and variance = 1.

Robust scaler : Robust scaler scales features using statistics that are robust to outliers. More specifically, Robust scaler removes the median and scales the data according to the interquartile range, thus making it less susceptible to outliers in the data.

3.2.3 Selection of Dependent and Independent variables

The variable that are not affected by the other variables are called independent variables. The variables which depend on other variables or factors. We expect these variables to change when the independent variables, upon whom they depend, undergo a change. The dependent or target variable here isClaimed Target which tells us a particular policy holder has filed a claim or not the target variable is selected based on our business problem and what we are trying to predict.

In our data set ‘y’ which is nothing but a claim is the dependent variable and remaining all are independent variables.

3.2.4 Data Sampling Methods

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. We used three Sampling methods.

3.2.4.1 Stratified sampling

Stratified sampling randomly selects data points from majority class so they will be equal to the data points in the minority class. So, after the sampling both the class will have same no of observations.

It can be performed using strata function from the library sampling.

3.2.4.2 Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset.

We used this method to split the dataset into train dataset which contains 70% of the total data and test dataset with the remaining 30% of the data.

3.2.5 Models Used for Development

We built our predictive models by using the following eight algorithms.

3.2.5.1 Model 01(Linear Regeression)

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

3.2.5.2 Model 02(Decision Tree Regression)

A regression tree is a type of decision tree. analysis to predict values of the target field. The predictions are based on combinations of values in the input fields. A regression tree calculates a predicted mean value for each node in the tree.

3.2.5.3 Model 03(Random Forest Regression)

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

3.2.5.4 Model 04(K-Nearest Neighbours)

K-nearest neighbours (KNN) is a type of supervised learning algorithm used for both regression and classification. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closet to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of ‘K’ training data and class holds the highest probability will be selected. In the case of regression, the value is the mean of the ‘K’ selected training points.

3.2.5.5 Model 05(XGBoost Regression)

XGBoost is a powerful approach for building supervised regression models.These are some key members of XGBoost models, each plays an important role.

RMSE: It is the square root of mean squared error (MSE).

MAE: It is an absolute sum of actual and predicted differences, but it lacks mathematically, that's why it is rarely used, as compared to other metrics.

3.2.5.6 Model 06(Extra Trees Regression)

Extra Trees (Extremely Randomized Trees) the ensemble learning algorithms. It constructs the set of decision trees. During tree construction the decision rule is randomly selected. This algorithm is very similar to Random Forest except random selection of split values.

3.2.5.7 Model 07(Gradient Boosting regression)

Gradient boosting Regression calculates the difference between the current prediction and the known correct target value.

This difference is called residual. After that Gradient boosting Regression trains a weak model that maps features to that residual. This residual predicted by a weak model is added to the existing model input and thus this process nudges the model towards the correct target. Repeating this step again and again improves the overall model prediction.

3.2.5.8 Model 08(Support Vector Regression)

Support Vector Regression is a supervised learning algorithm that is used to predict discrete values.Support Vector Regression uses the same principle as the SVMs.The basic idea behind SVR is to find the best fit line.In SVR, the best fit line is the hyperplane that has the maximum number of points.

3.3 AI / ML Models Analysis and Final Results

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models.

We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given auto dataset of size ~ 272252 policies.

3.3.1 Different Model codes:

The python code for different models is as follows:

3.3.1.1 Extra Tree Regression

```
from sklearn.ensemble import ExtraTreesRegressor
modelETR = ExtraTreesRegressor()
modelGBR = GradientBoostingRegressor(loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0,
                                     criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
                                     min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0,
                                      init=None, random_state=None, max_features=None,
                                     alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False,
                                     validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0)
# Fit the model with train data

modelETR.fit(x_train, y_train)

# Predict the model with test data

y_pred = modelETR.predict(x_test)

# Print the model name

print('Model Name: ', modelETR)

# Evaluation metrics for Regression analysis

from sklearn import metrics

print('Mean Absolute Error (MAE):', round(metrics.mean_absolute_error(y_test, y_pred),3))
print('Mean Squared Error (MSE):', round(metrics.mean_squared_error(y_test, y_pred),3))
print('Root Mean Squared Error (RMSE):', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),3))
print('R2_score:', round(metrics.r2_score(y_test, y_pred),6))
print('Root Mean Squared Log Error (RMSLE):', round(np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),3))

# Define the function to calculate the MAPE - Mean Absolute Percentage Error

def MAPE (y_test, y_pred):
    y_test, y_pred = np.array(y_test), np.array(y_pred)
    return np.mean(np.abs((y_test - y_pred) / y_test)) * 100
# Evaluation of MAPE

result = MAPE(y_test, y_pred)
print('Mean Absolute Percentage Error (MAPE):', round(result, 2), '%')

# Calculate Adjusted R squared values

r_squared = round(metrics.r2_score(y_test, y_pred),6)
adjusted_r_squared = round(1 - (1-r_squared)*(len(y)-1)/(len(y)-x.shape[1]-1),6)
print('Adj R Square: ', adjusted_r_squared)
print('-------------------------------------------------------------------------------------------------------------------')
#-------------------------------------------------------------------------------
```

```
new_row = {'Model Name' : modelETR,
           'Mean_Absolute_Error_MAE' : metrics.mean_absolute_error(y_test, y_pred),
           'Adj_R_Square' : adjusted_r_squared,
           'Root_Mean_Squared_Error_RMSE' : np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
           'Mean_Absolute_Percentage_Error_MAPE' : result,
           'Mean_Squared_Error_MSE' : metrics.mean_squared_error(y_test, y_pred),
           'Root_Mean_Squared_Log_Error_RMSLE': np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),
           'R2_score' : metrics.r2_score(y_test, y_pred)}
```

3.3.1.2 Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
modelrfr = RandomForestRegressor()
modelGBR = GradientBoostingRegressor(loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0,
                                     criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
                                     min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0,
                                      init=None, random_state=None, max_features=None,
                                     alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False,
                                     validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0)
# Fit the model with train data

modelrfr.fit(x_train, y_train)

# Predict the model with test data

y_pred = modelrfr.predict(x_test)

# Print the model name

print('Model Name: ', modelrfr)

# Evaluation metrics for Regression analysis

from sklearn import metrics

print('Mean Absolute Error (MAE):', round(metrics.mean_absolute_error(y_test, y_pred),3))
print('Mean Squared Error (MSE):', round(metrics.mean_squared_error(y_test, y_pred),3))
print('Root Mean Squared Error (RMSE):', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),3))
print('R2_score:', round(metrics.r2_score(y_test, y_pred),6))
print('Root Mean Squared Log Error (RMSLE):', round(np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),3))

# Define the function to calculate the MAPE - Mean Absolute Percentage Error

def MAPE (y_test, y_pred):
    y_test, y_pred = np.array(y_test), np.array(y_pred)
    return np.mean(np.abs((y_test - y_pred) / y_test)) * 100
# Evaluation of MAPE

result = MAPE(y_test, y_pred)
print('Mean Absolute Percentage Error (MAPE):', round(result, 2), '%')

# Calculate Adjusted R squared values

r_squared = round(metrics.r2_score(y_test, y_pred),6)
adjusted_r_squared = round(1 - (1-r_squared)*(len(y)-1)/(len(y)-x.shape[1]-1),6)
print('Adj R Square: ', adjusted_r_squared)
print('-----------------------------------------------------------------------------------------------------------')
#-----------------------------------------------------------------------------
```

```
new_row = {'Model Name' : modelrfr,
            'Mean_Absolute_Error_MAE' : metrics.mean_absolute_error(y_test, y_pred),
            'Adj_R_Square' : adjusted_r_squared,
            'Root_Mean_Squared_Error_RMSE' : np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
            'Mean_Absolute_Percentage_Error_MAPE' : result,
            'Mean_Squared_Error_MSE' : metrics.mean_squared_error(y_test, y_pred),
            'Root_Mean_Squared_Log_Error_RMSLE': np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),
            'R2_score' : metrics.r2_score(y_test, y_pred)}
```

3.3.1.3 Different Models Python Code

```
#LOOP ALL REGRESSOR ALGORITHMS FOR REGREESION MODEL IN SUPERVISED LEARNING



# Build the Regression / Regressor models


from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xgb
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor

# Create objects of Regression / Regressor models with default hyper-parameters

modelmlg = LinearRegression()
modeldcr = DecisionTreeRegressor()
modelrfr = RandomForestRegressor()
modelSVR = SVR()
modelXGR = xgb.XGBRegressor()
modelETR = ExtraTreesRegressor()
modelKNN = KNeighborsRegressor(n_neighbors=5)
modelGBR = GradientBoostingRegressor(loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0,
                                     criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
                                     min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0,
                                      init=None, random_state=None, max_features=None,
                                     alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False,
                                     validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0)

# Evalution matrix for all the algorithms

MM = [modelmlg, modeldcr,modelrfr, modelSVR, modelXGR,modelETR,modelKNN, modelGBR]
```

```
for models in MM:

    # Fit the model with train data

    models.fit(x_train, y_train)

    # Predict the model with test data

    y_pred = models.predict(x_test)

    # Print the model name

    print('Model Name: ', models)

    # Evaluation metrics for Regression analysis

    from sklearn import metrics

    print('Mean Absolute Error (MAE):', round(metrics.mean_absolute_error(y_test, y_pred),3))
    print('Mean Squared Error (MSE):', round(metrics.mean_squared_error(y_test, y_pred),3))
    print('Root Mean Squared Error (RMSE):', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)),3))
    print('R2_score:', round(metrics.r2_score(y_test, y_pred),6))
    print('Root Mean Squared Log Error (RMSLE):', round(np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),3))

    # Define the function to calculate the MAPE - Mean Absolute Percentage Error

    def MAPE (y_test, y_pred):
        y_test, y_pred = np.array(y_test), np.array(y_pred)
        return np.mean(np.abs((y_test - y_pred) / y_test)) * 100

    # Evaluation of MAPE

    result = MAPE(y_test, y_pred)
    print('Mean Absolute Percentage Error (MAPE):', round(result, 2), '%')
```

```
# Calculate Adjusted R squared values

r_squared = round(metrics.r2_score(y_test, y_pred),6)
adjusted_r_squared = round(1 - (1-r_squared)*(len(y)-1)/(len(y)-x.shape[1]-1),6)
print('Adj R Square: ', adjusted_r_squared)
print('-----------------------------------------------------------------------------------------------------------------------')
#-------------------------------------------------------------------------------------
new_row = {'Model Name' : models,
           'Mean_Absolute_Error_MAE' : metrics.mean_absolute_error(y_test, y_pred),
           'Adj_R_Square' : adjusted_r_squared,
           'Root_Mean_Squared_Error_RMSE' : np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
           'Mean_Absolute_Percentage_Error_MAPE' : result,
           'Mean_Squared_Error_MSE' : metrics.mean_squared_error(y_test, y_pred),
           'Root_Mean_Squared_Log_Error_RMSLE': np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),
           'R2_score' : metrics.r2_score(y_test, y_pred)}
RGRResult = RGRResult.append(new_row, ignore_index=True)
#-------------------------------------------------------------------------------------
```

Statified Sampling: Extra Tree model performance is good,by considering the Adj_R_square and R2_score.This is because ExtraTree uses bootstrap aggregation which can reduce bias and variance in the data and can lead to good predictions with dataset**.**

Simple Random Sampling: ExtraTree / Random Forest are out performed Linear Regression model, by considering the Adj_R_Square,R2_Score. This is because ExtraTree

have hidden and complex patterns between different variables and can leads to good predictions with claims dataset.

4.0 Conclusions and Future Work

The model results in the following order by considering the Adj_ R_square, R2_score.

1) Extra Tree Regressor with stratified sampling and Random Sampling
2) Random Forest Regressor with stratified sampling and Random Sampling
3) XGB Regressor with stratified sampling and Random Sampling

We recommend model – Extra Tree Regressor with stratified sampling and Random Sampling technique as a best fit for the given solar radiation dataset.Because it uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good prediction with the given dataset.

	Model Name	Mean_Absolute_Error_MAE	Adj_R_Square	Root_Mean_Squared_Error_RMSE	Mean_Absolute_Percentage_Error_MAPE	Mean_Squared_Error_MSE	Root_Mean_Squared_Log_Error_RMSLE	R2_score
0	LinearRegression()	144.48729	0.620566	192.864336	4112.866125	37196.651964	5.261987	0.620694
1	DecisionTreeRegressor()	42.980469	0.859398	117.403472	21.762219	13783.575307	4.765616	0.859445
2	(DecisionTreeRegressor(max_features=1.0, rando...	32.170248	0.9326	81.285727	20.204679	6607.369443	4.39797	0.932623
3	SVR()	143.153875	0.38599	245.341915	1353.728383	60192.655152	5.502653	0.386197
4	XGBRegressor(base_score=0.5, booster='gbtree',...	40.733248	0.922174	87.346722	292.715358	7629.44993	4.469886	0.9222
5	(ExtraTreeRegressor(random_state=315679246), E...	30.895153	0.936502	78.897728	29.168474	6224.851559	4.368152	0.936523
6	KNeighborsRegressor()	58.566034	0.857643	118.133535	189.096694	13955.532021	4.771816	0.857691
7	([DecisionTreeRegressor(criterion='friedman_ms...	58.384756	0.877435	109.614611	560.792106	12015.362874	4.696971	0.877476

Different model Results

```
Model Name:  ExtraTreesRegressor()
Mean Absolute Error (MAE): 30.668
Mean Squared Error (MSE): 6250.759
Root Mean Squared Error (RMSE): 79.062
R2_score: 0.936259
Root Mean Squared Log Error (RMSLE): 4.37
Mean Absolute Percentage Error (MAPE): 28.28 %
Adj R Square:  0.936238
------------------------------------------------
--------
```

Extra tree Regressor

```
Model Name:  RandomForestRegressor()
Mean Absolute Error (MAE): 32.299
Mean Squared Error (MSE): 6654.11
Root Mean Squared Error (RMSE): 81.573
R2_score: 0.932146
Root Mean Squared Log Error (RMSLE): 4.401
Mean Absolute Percentage Error (MAPE): 20.19 %
Adj R Square:  0.932123
------------------------------------------------
--------
```

Random Forest Regressor

5.0 Reference

Kaggle link : https://www.kaggle.com

Git hub link: https://github.com

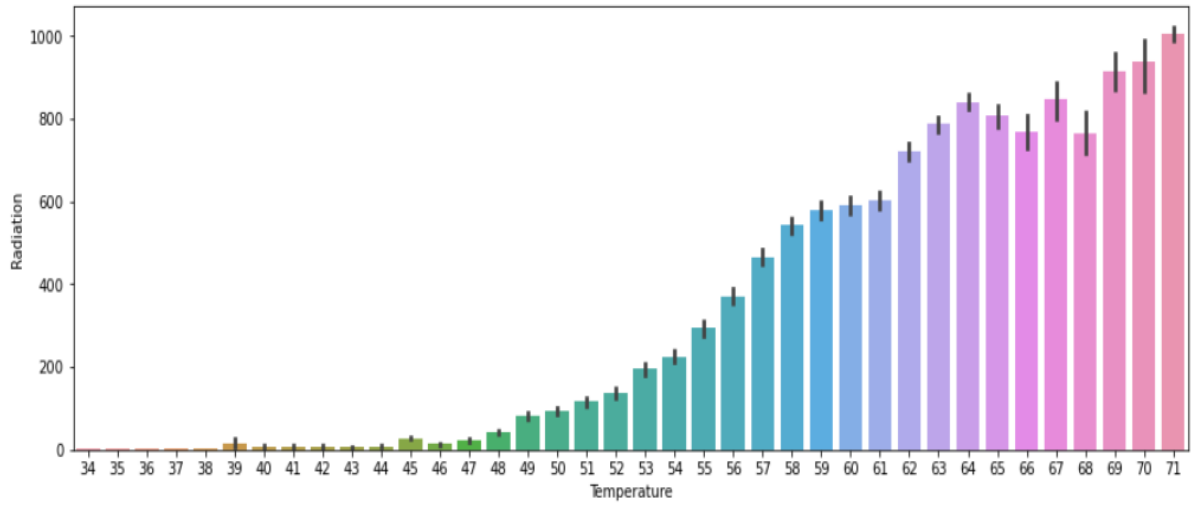
6.0 Appendices

- Kernel methods
- Graphical methods
- Reinforcement learning
- Convex analysis
- Optimization
- Bio informatics
- Minimal description length principle
- Topics in information theory
- Decision theory
- Network algorithms

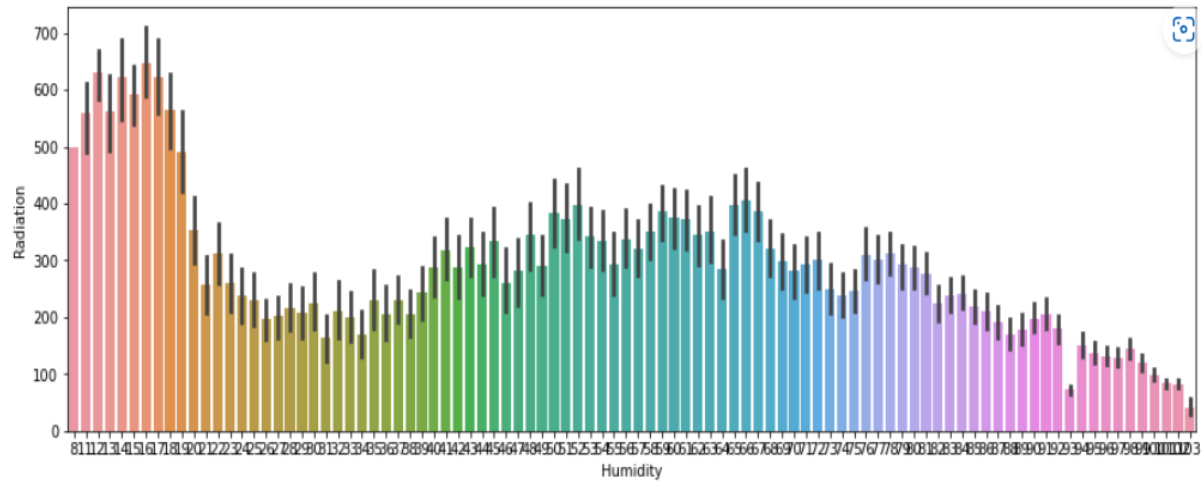
6.1 Python Code Results

	Model Name	Mean_Absolute_Error_MAE	Adj_R_Square	Root_Mean_Squared_Error_RMSE	Mean_Absolute_Percentage_Error_MAPE	Mean_Squared_Error_MSE	Root_Mean_Squared_Log_Error_RMSLE	R2_score
0	LinearRegression()	144.48729	0.620566	192.864336	4112.866125	37196.651964	5.261987	0.620694
1	DecisionTreeRegressor()	42.980469	0.859398	117.403472	21.762219	13783.575307	4.765616	0.859445
2	(DecisionTreeRegressor(max_features=1.0, rando...	32.170248	0.9326	81.285727	20.204679	6607.369443	4.39797	0.932623
3	SVR()	143.153875	0.38599	245.341915	1353.728383	60192.655152	5.502653	0.386197
4	XGBRegressor(base_score=0.5, booster='gbtree',...	40.733248	0.922174	87.346722	292.715358	7629.44993	4.469886	0.9222
5	(ExtraTreeRegressor(random_state=315679246), E...	30.895153	0.936502	78.897728	29.168474	6224.851559	4.368152	0.936523
6	KNeighborsRegressor()	58.566034	0.857643	118.133535	189.096694	13955.532021	4.771816	0.857691
7	([DecisionTreeRegressor(criterion='friedman_ms...	58.384756	0.877435	109.614611	560.792106	12015.362874	4.696971	0.877476

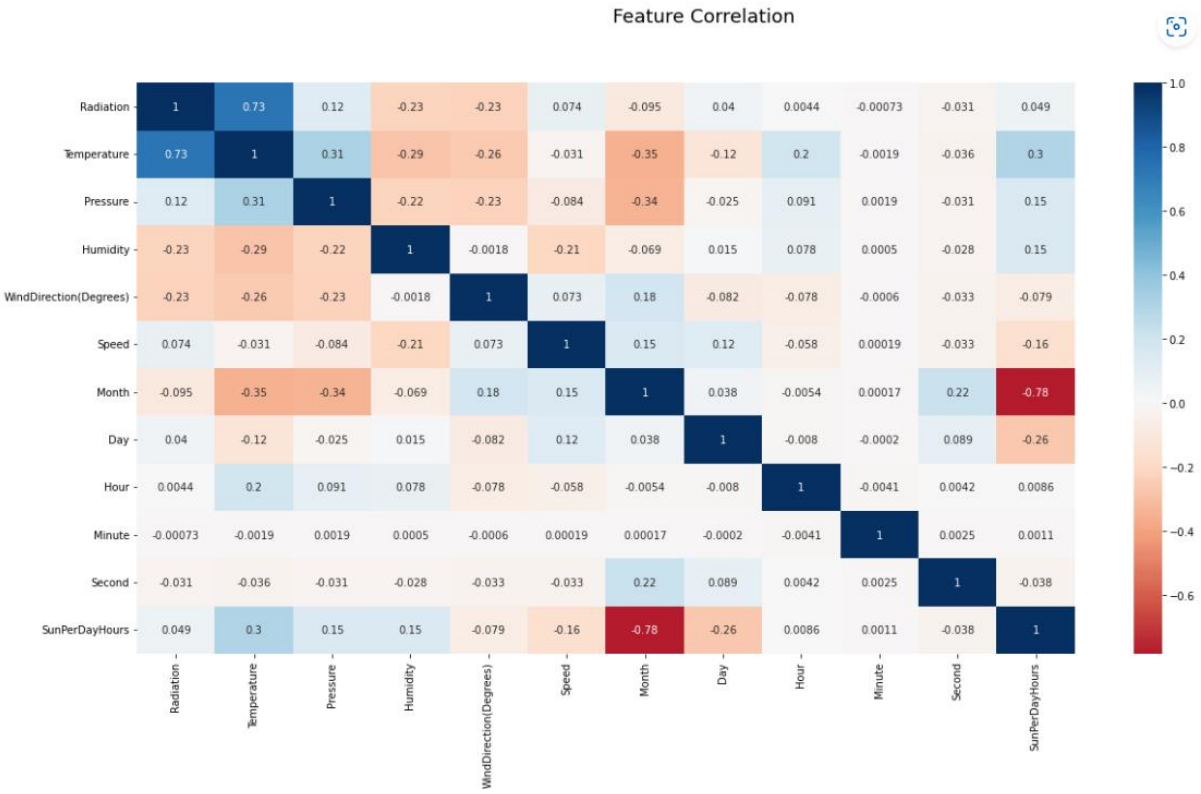
6.2 List of Charts



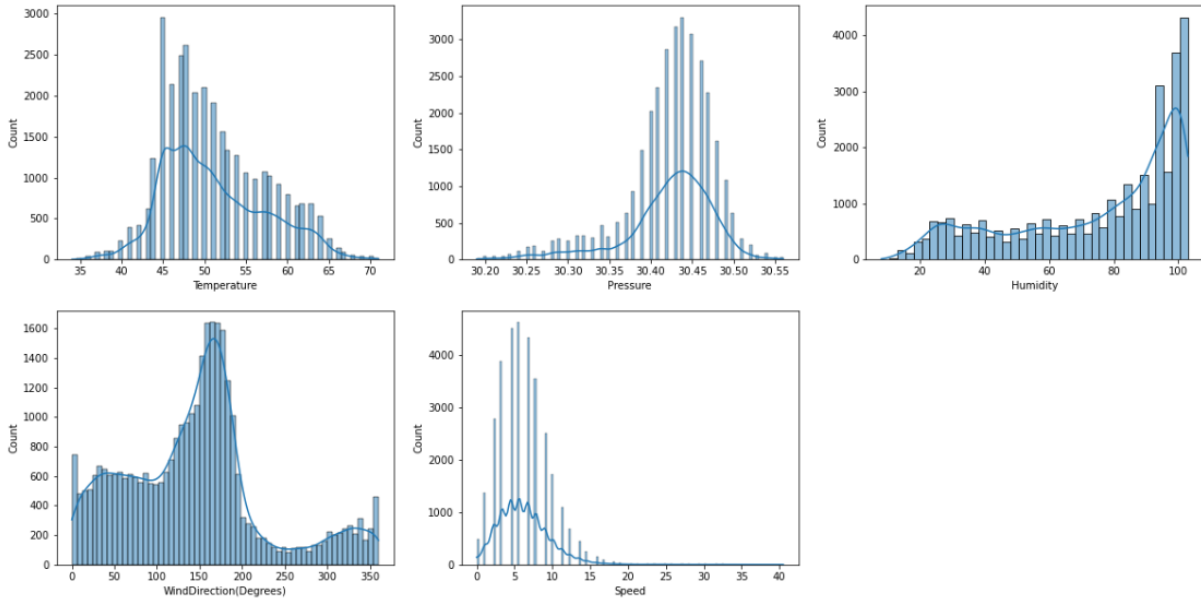
Temperature



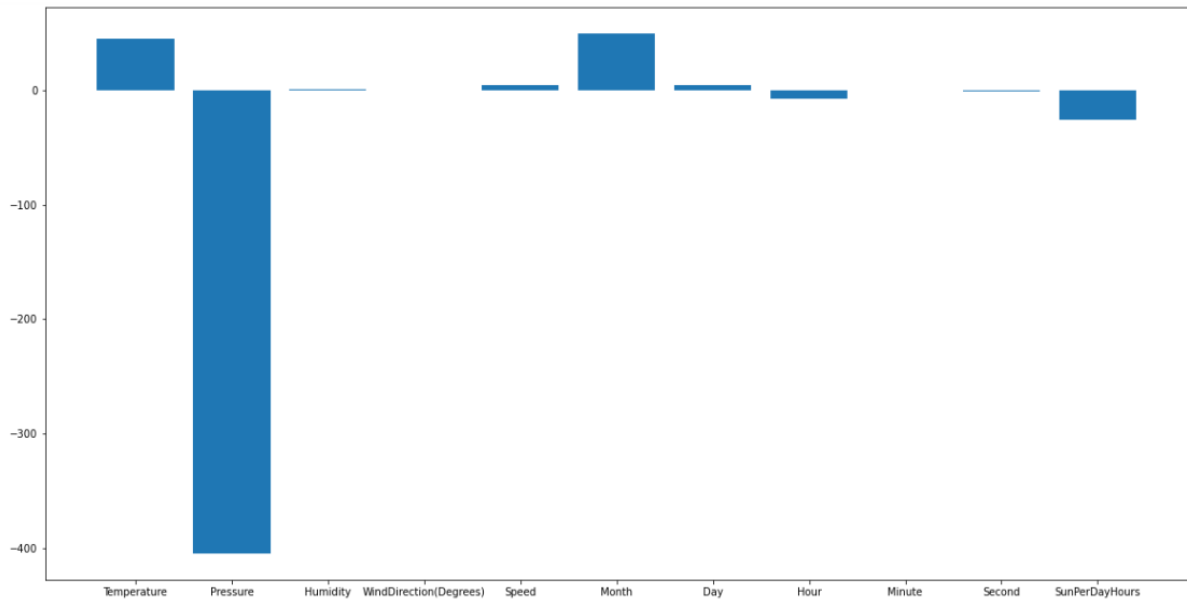
Humidity



Feature Correlation



HistPlot


0
−100
−200
−300
−400
Temperature
Pressure
Humidity
WindDirection(Degrees)
Speed
Month
Day
Hour
Minute
Second
SunPerDayHours

Bar Plot