CA 4010 Data Warehousing and Data Mining

Solar Radiation Prediction

Student Name	Email	Student No.
Muhammad Umar	muhammad.umer2@mail.dcu.ie	17313893
Bartlomiej Kiraga	bartlomiej.kiraga2@mail.dcu.ie	17327333

Plagiarism Declaration

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I/We have read and understood the Assignment Regulations. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

Name(s): Muhammad Umar, Bartlomiej Kiraga
Date: 10/11/2020

Introduction

- **Dataset description:**

  Solar radiation which is also known as ‘solar resource’ or ‘sunlight’, is a useful measurement that can be of great use in areas ranging from weather forecasting to renewable energy production. Solar radiation can vary greatly in different parts of Earth based on factors such as time, location and weather. Our dataset contains solar radiation readings for Hawaii which were provided by NASA’s HI-SEAS Habitat.

  Our main goal with this project was to predict the level of solar radiation during the day based on the values given in our dataset. The dataset we were looking for should have contained values such as temperature, wind speed and humidity which correlate to the measured levels of solar radiation. With the help of kaggle we were able to obtain a very large dataset which had all the attributes we needed that would help us analyze that data and predict solar radiation during the day. Meteorological data which is obtained from HI-SEAS Habitat in Hawaii is contained in this dataset. Observations made in this dataset include the readings of:

  - ➢ Solar Irradiance (W/m2)
  - ➢ Temperature (°F)
  - ➢ Barometric Pressure (Hg)
  - ➢ Humidity (%)
  - ➢ Wind Direction (°)
  - ➢ Wind Speed (mph)
  - ➢ Sun Rise/Set Time

Data Analysis

We first decided to prepare our dataset by examining all of its attributes and values. After looking at our dataset we noticed that the highest value for the solar radiation attribute seemed to be an outlier. Despite all of its other attributes being very similar to the next largest value the increase was much higher than any other leading us to believe that this specific value was an anomaly likely caused by a solar flare. We decided to remove this value to keep our predictions more realistic as these outliers may have caused the accuracy of our algorithms to show us misleading prediction values.

```
# remove outlier
data = data.drop(6465)
```

We then decided to check if there were any null values in our dataset. This is an important check as null values can lead to problems in many classification algorithms. We ran the isnull pandas command and found that our dataset contained no null values.

```
data = pd.read_csv("data/SolarPrediction.csv")

# check for null values
print(data.isnull().sum())
```

After this check we started to analyse our dataset again to see if we needed to make any changes to it. We realised that our dataset needed more columns/attributes that would help for the data visualization of our dataset.

By default we were given ‘UNIX TIME’, ‘Sunrise’ and ‘Sunset’, which didn’t seem enough columns of data for us to create graphs of our dataset for us to analyze it. For the data visualization we used ‘UNIX TIME’ to create columns like ‘Month of the year’, ‘Time of the day’ in order to map out our graphs according to these new columns.

```
hawaii= timezone('Pacific/Honolulu')
data.index =  pd.to_datetime(data['UNIXTime'], unit='s')
data.index = data.index.tz_localize(pytz.utc).tz_convert(hawaii)
data['MonthOfYear'] = data.index.strftime('%m').astype(int)
data['DayOfYear'] = data.index.strftime('%j').astype(int)
data['WeekOfYear'] = data.index.strftime('%U').astype(int)
data['TimeOfDay(h)'] = data.index.hour
data['SunElevation'] = data.apply(lambda row: timeAwayFromNight(row.TimeSunRise ,row.TimeSunSet, row.Time), axis=1)
data.drop(columns = ['UNIXTime', 'Data', 'Time', 'TimeSunRise','TimeSunSet'], inplace = True)



print(data.head())
```

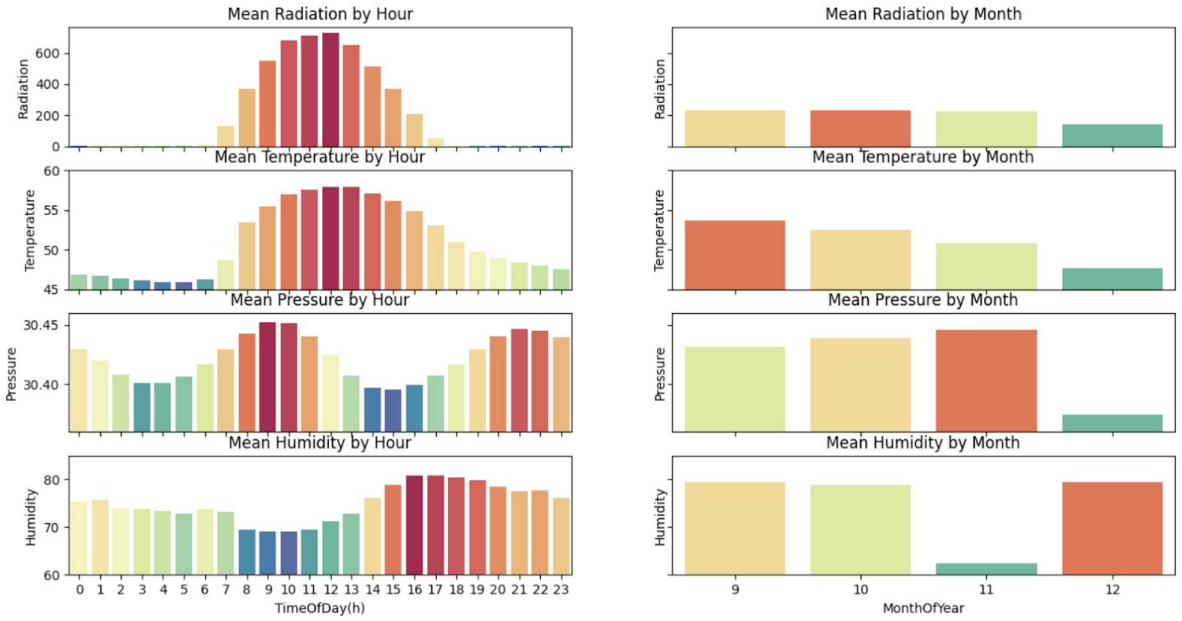
```
                           Radiation  Temperature  Pressure  Humidity  \
UNIXTime
2016-09-29 23:55:26-10:00       1.21           48     30.46        59
2016-09-29 23:50:23-10:00       1.21           48     30.46        58
2016-09-29 23:45:26-10:00       1.23           48     30.46        57
2016-09-29 23:40:21-10:00       1.21           48     30.46        60
2016-09-29 23:35:24-10:00       1.17           48     30.46        62

                           WindDirection(Degrees)  Speed  MonthOfYear  \
UNIXTime
2016-09-29 23:55:26-10:00                  177.39   5.62            9
2016-09-29 23:50:23-10:00                  176.78   3.37            9
2016-09-29 23:45:26-10:00                  158.75   3.37            9
2016-09-29 23:40:21-10:00                  137.71   3.37            9
2016-09-29 23:35:24-10:00                  104.95   5.62            9

                           DayOfYear  WeekOfYear  TimeOfDay(h)  SunElevation
UNIXTime
2016-09-29 23:55:26-10:00        273          39            23             0
2016-09-29 23:50:23-10:00        273          39            23             0
2016-09-29 23:45:26-10:00        273          39            23             0
2016-09-29 23:40:21-10:00        273          39            23             0
2016-09-29 23:35:24-10:00        273          39            23             0
```

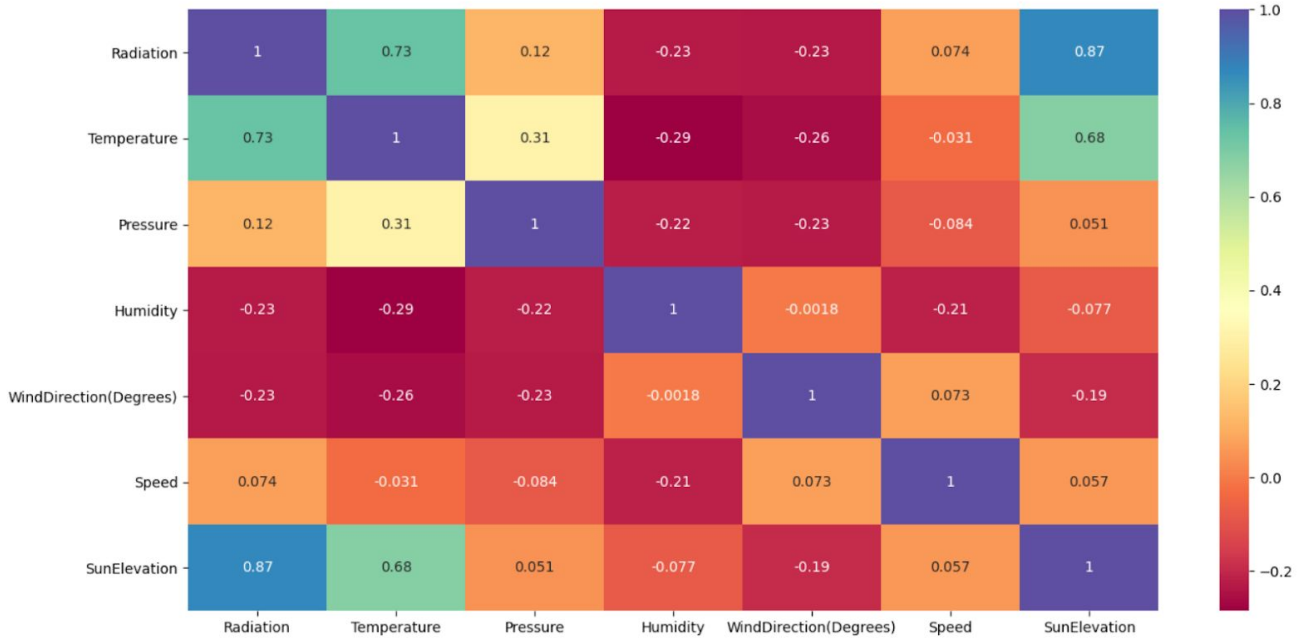
After putting together all the needed data we decided to use Graphs to plot them out to have a better view of our data set and see the relationship between different attributes with the ‘Radiation’. This way we can find out which attributes are most relevant.

We know that radiation in general is at its max during the day when the sun is out fully on a clear day with very minimum clouds and radiation is at its lowest during the night when the sun is shining on the other side of the planet.



According to the graphs we get to see that ‘Temperature’ attribute has a very good relationship with the ‘Radiation’ during the day when compared to the ‘Pressure’ and ‘WindSpeed’. This was also expected as from many resources where we researched about solar radiation said that temperature and atmospheric conditions especially shadows(Clouds) are the things that have the most effect on the radiation that controls the amount of solar radiation that reaches us.

This relationship is further concretised with the help of a pearson correlation heat map of our dataset. Correlation is used as a technique which helps investigate the relationship between two continuous, quantitative variables. We can clearly see that the ‘temperature’ and ‘SunElevation’ attributes have a very strong relationship with radiation. Both humidity and wind direction are inversely correlated with radiation meaning that lower values correlate to higher solar radiation. We contemplated removing the wind speed attribute as it has the least significant correlation with radiation but decided against it as we wanted to see if its small correlation would have any significant impact on our accuracy.



After further examination of our dataset we decided to remove the attributes “UNIXTime”, “Data”, ”Time”, ”TimeSunrise” and ”TimeSunset” as many of them are redundant and any useful data given to us by them could be simplified to a single attribute. The main way these attributes contribute to the prediction of solar radiation is by determining whether or not a measurement occurs during daytime and the stage of the day at which it was taken. We decided to create a new attribute called “SunElevation” which measures time since sunrise or time to sunset. To calculate this attribute we measure the time passed since sunrise unless current time occurs after the midpoint between sunrise and sunset in which case we measure the time between the measurement time and the sunset. Times before sunrise or after sunset get a zero value.

```
def timeAwayFromNight(sunrise, sunset, time):
    sunrise = convertHourToSec(sunrise)
    sunset = convertHourToSec(sunset)
    time = convertHourToSec(time)
    result = 0
    midpoint = (sunrise + sunset)/2
    if time <= sunrise or time >= sunset:
        result = 0
    elif time > sunrise and time <= midpoint:
        result = time - sunrise
    elif time > midpoint and time < sunset:
        result = sunset - time
    return result

# create a new column that combines time with time of sunset/sunrise
data['SunElevation'] = data.apply(lambda row: timeAwayFromNight(row.TimeSunRise ,row.TimeSunSet, row.Time), axis=1)
data.drop(columns = ['UNIXTime', 'Data', 'Time', 'TimeSunRise','TimeSunSet'], inplace = True)      You, a few seconds
```

After performing attribute selection our dataset contained the following attributes:

- Radiation
- Temperature
- Pressure
- Humidity
- WindDirection
- Speed
- SunElevation

With our attribute selection finalized we normalized our data using the min max formula and we have split our dataset into a training set for our chosen algorithms and a testing set to test their accuracy and prevent overfitting.

```
# Normalize data
data['Temperature'] = (data['Temperature'] - data['Temperature'].min()) / (data['Temperature'].max() - data['Temperature'].min())
data['Pressure'] = (data['Pressure'] - data['Pressure'].min()) / (data['Pressure'].max() - data['Pressure'].min())
data['Humidity'] = (data['Humidity'] - data['Humidity'].min()) / (data['Humidity'].max() - data['Humidity'].min())
data['Speed'] = (data['Speed'] - data['Speed'].min()) / (data['Speed'].max() - data['Speed'].min())
data['WindDirection(Degrees)'] = (data['WindDirection(Degrees)'] - data['WindDirection(Degrees)'].min()) / (data['WindDirection(Degrees)'].max() - data['WindDirection(Degrees)'].min())
data['SunElevation'] = (data['SunElevation'] - data['SunElevation'].min()) / (data['SunElevation'].max() - data['SunElevation'].min())

max = data['Radiation'].max()
min = data['Radiation'].min()
data['Radiation'] = data.apply(lambda row: rankSolarRadiationtoCategories(row.Radiation, min, max), axis=1)
```

Algorithms

- **K-Nearest Neighbour Classification**
  Due to the fact that our dataset was supervised and that all of its attributes were continuous numerical values, we have decided that the k-nearest neighbour algorithm was a good fit for our prediction.
  K-nearest neighbour algorithm works by calculating distance between a new data point with all other data points in the training set.

  First we decided to implement a k-nearest neighbour classifier using Scikit python library.
  Classification cannot be used to predict a continuous attribute therefore we had to split the Radiation attribute into different classes of discrete data. First we found the midpoint of all the radiation values. We used mean to achieve this as we had previously removed the outlier that might skew its result. We then calculated the quartiles and used them to split the values into categories that ranged from 1-4.

```
x = np.array(data.drop(['Radiation'],1))
y = np.array(data['Radiation'])

x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.2)

classifier = neighbors.KNeighborsClassifier(n_neighbors=6)
classifier.fit(x_train, y_train)
accuracy = classifier.score(x_test, y_test)
print(accuracy)
```

- **K-Nearest Neighbour Regression**
  The k-nearest neighbour algorithm can be used for both classification and regression. We decided to modify our knn classifier to use regression instead due to the fact that regression is more suitable for predicting continuous data and it would give us a more meaningful prediction without the need to split the radiation attribute into classes of discrete data.

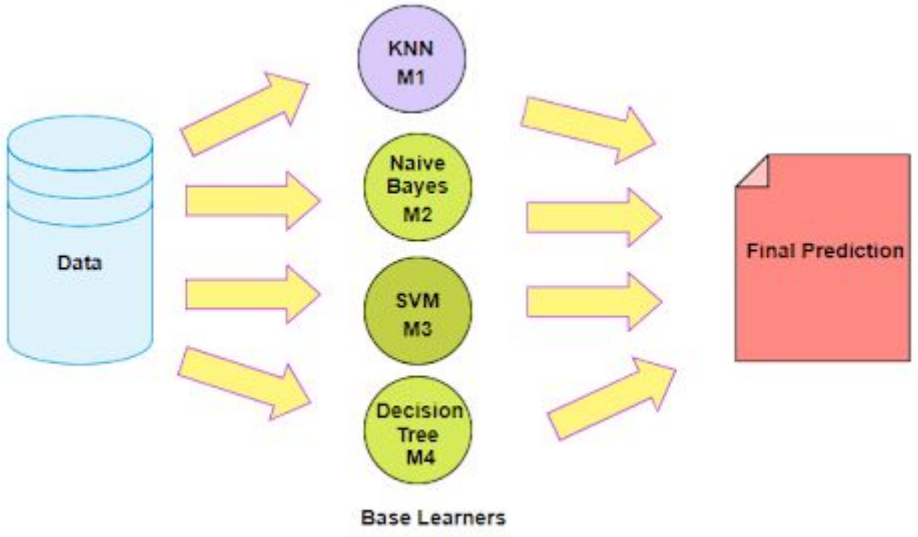
```
x = np.array(data.drop(['Radiation'],1))
y = np.array(data['Radiation'])

x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.2)

regressor = neighbors.KNeighborsRegressor(n_neighbors=6)
regressor.fit(x_train, y_train)
pred = regressor.predict(x_test)
print(r2_score(y_test, pred))
```

- **Random Forest Regression**
  With more research we found out that our data, especially radiation, is partly nonlinear because the radiation in each row is dependent on the value for that row and the value is not a straight line, it goes up and down as the day progresses.

  For this reason we decided to use another ML algorithm which is called Random Forest Regression and it uses an ensemble method of learning for its regression and classification which means that it uses multiple ML algos by creating decision trees with no interaction between them and then aggregate the results of those decision trees to make more accurate predictions. This is a supervised learning algorithm.



We use the random forest generator for its ability to handle large data files efficiently for many data sets and it is one of the most accurate ML algo available. It also tells about the importance of each variable in the classification. We thought this algo would be a useful comparison for us to make with other algos we used before.

```
for x in variations:
    x_train, x_test, y_train, y_test = model_selection.train_test_split(x, y, test_size=0.2)
    RanForReg = RandomForestRegressor()
    RanForReg.fit(x_train, y_train)
    RanForReg_pred= RanForReg.predict(x_test)
    print(x.head(0))
    print(r2_score(y_test, RanForReg_pred))
```

Results

For the results we ran the different machine learning algorithms we chose with the different columns of the data set. Picture below shows the columns which were used for the training of each algorithm. We decided to do this to see how the R^2 score affected different columns as some columns are less relevant which we know from ‘Pearson Correlation Heatmap’. Doing this will further concretise our analysis that we did in the graphs and heatmaps.

```
x1 = data[['Temperature', 'Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)', 'SunElevation']]      # Removed all of them
x2 = data[['Temperature', 'Pressure', 'Humidity', 'Speed', 'SunElevation']]                               # Removed Wind direction
x3 = data[['Temperature', 'Pressure', 'Humidity', 'WindDirection(Degrees)', 'SunElevation']]              # Removed Speed
x4 = data[['Temperature', 'Pressure', 'WindDirection(Degrees)', 'SunElevation']]                          # Removed Humidity
x5 = data[['Temperature', 'Humidity', 'Speed', 'WindDirection(Degrees)', 'SunElevation']]                 # Removed Pressure
x6 = data[['Temperature', 'Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)']]                     # Removed Sun Elevation
x7 = data[['Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)', 'SunElevation']]                    # Removed Temperature
x8 = data[['Temperature', 'SunElevation']]                                                               # Removed Everything except Temperature, SunElevation
x9 = data[['Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)']]                                    # Removed Temperature, SunElevation

variations = [x1, x2, x3, x4, x5, x6, x7, x8, x9]

y = np.array(data['Radiation'])
```

- **K-Nearest Neighbour Regression**
  Our main algorithm of interest was the k-nearest neighbour regression. We measured the accuracy of regression using the R^2 value.
  The R^2 values range from 0-1 and measure how well our regression model fits the data. Higher r^2 value means better correlation between the predictions of our algorithm and the data.
  Testing our knn regressor on our standard attribute selection with normalized data gave us an R^2 value of around 0.89-0.9. When we removed our data normalization our result dropped down to 0.82. This

shows that normalization of values in our data has a notable increase to our prediction accuracy showing that our dataset uses values of different ranges.

We then measured the r^2 values using different combinations of our attributes showing a drastic reduction in prediction accuracy when Temperature and SunElevation were removed. The removal of wind speed or the wind direction attribute had no significant impact on prediction accuracy.

```
Index(['Temperature', 'Pressure', 'Humidity', 'Speed',
       'WindDirection(Degrees)', 'SunElevation'],
      dtype='object')
R^2 score 0.8972495857060894

Index(['Temperature', 'Pressure', 'Humidity', 'Speed', 'SunElevation'], dtype='object')
R^2 score 0.8876977130072784

Index(['Temperature', 'Pressure', 'Humidity', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
R^2 score 0.9054516850826001

Index(['Temperature', 'Pressure', 'WindDirection(Degrees)', 'SunElevation'], dtype='object')
R^2 score 0.8588579837607339

Index(['Temperature', 'Humidity', 'Speed', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
R^2 score 0.8757012752888991

Index(['Temperature', 'Pressure', 'Humidity', 'Speed',
       'WindDirection(Degrees)'],
      dtype='object')
R^2 score 0.7363489439896638

Index(['Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
R^2 score 0.8790471696442217

Index(['Temperature', 'SunElevation'], dtype='object')
R^2 score 0.8081977915363862

Index(['Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)'], dtype='object')
R^2 score 0.5651121312897622
```

- **K-Nearest Neighbour Classification**
  We also measured the results of our knn classifier. The results showed high accuracy of 90% for predicting the right range of values of solar radiation based on the other attributes, however the result may be inflated by the fact that there are only 4 different possible outcomes to account for giving a less useful result compared to the regressor.

```
Index(['Temperature', 'Pressure', 'Humidity', 'Speed',
       'WindDirection(Degrees)', 'SunElevation'],
      dtype='object')
0.909744531130488

Index(['Temperature', 'Pressure', 'Humidity', 'Speed', 'SunElevation'], dtype='object')
0.9196879302432308

Index(['Temperature', 'Pressure', 'Humidity', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
0.9106623833562797

Index(['Temperature', 'Pressure', 'WindDirection(Degrees)', 'SunElevation'], dtype='object')
0.8901636836469329

Index(['Temperature', 'Humidity', 'Speed', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
0.9072969251950436

Index(['Temperature', 'Pressure', 'Humidity', 'Speed',
       'WindDirection(Degrees)'],
      dtype='object')
0.8555912498087808

Index(['Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
0.90806180204987

Index(['Temperature', 'SunElevation'], dtype='object')
0.8577329050022946

Index(['Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)'], dtype='object')
0.8110754168578859
```

- **Random Forest Regression**
  The results we got from this algorithm were as expected. As you can see any dataset which contains columns(Temperature, Sun Elevation), in general has a higher r^2 score. We further confirmed this by taking out these two columns from the dataset and we got a really bad r^2 score which further proves our hypothesis. Overall our prediction accuracy did not change in a significant way by using random forest regression. We saw at most a 0.01-0.02 difference which is not enough to conclude an improvement indicating that k-nearest neighbour is suitable for our dataset and that the Naive Bayes, Decision Trees and SVM algorithms used in random forest did not add any accuracy while slowing down the computation time.

```
Index(['Temperature', 'Pressure', 'Humidity', 'Speed',
       'WindDirection(Degrees)', 'SunElevation'],
      dtype='object')
R^2 score 0.9132804122195184

Index(['Temperature', 'Pressure', 'Humidity', 'Speed', 'SunElevation'], dtype='object')
R^2 score 0.9134132707516568

Index(['Temperature', 'Pressure', 'Humidity', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
R^2 score 0.9134893010707943

Index(['Temperature', 'Pressure', 'WindDirection(Degrees)', 'SunElevation'], dtype='object')
R^2 score 0.864063247184293

Index(['Temperature', 'Humidity', 'Speed', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
R^2 score 0.8892571275995445

Index(['Temperature', 'Pressure', 'Humidity', 'Speed',
       'WindDirection(Degrees)'],
      dtype='object')
R^2 score 0.7402147127112602

Index(['Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)',
       'SunElevation'],
      dtype='object')
R^2 score 0.8925559495903932

Index(['Temperature', 'SunElevation'], dtype='object')
R^2 score 0.7921876999776658

Index(['Pressure', 'Humidity', 'Speed', 'WindDirection(Degrees)'], dtype='object')
R^2 score 0.5654558640796068
```

Conclusion

As for the conclusion, we were satisfied with our results. The success rate we got from our chosen algorithm got the r^2 score of between 0.8 and 0.9. We have learned that higher humidity value means the radiation levels will go down as radiation from the sun is absorbed into the atmosphere due to humidity. We also learned that temperature and solar radiation goes hand in hand as higher temperature means more radiation and vice versa. The most heat caused on the planet earth is at the equator which is Sun's focus the most so that part of the planet is hit with the most radiation. Our readings were taken in HI-SEAS Habitat in Hawaii and Hawaii is 2,213 km away from the equator.

What we learned from our dataset and algos is that since our data set is non-linear, our approach of using KNN classifiers was not the optimum choice. After a bit more research we found out that regressors work best with the kind of dataset we have. So we tried two ML algos (KNN_Regressors, Random Forest Regressors) that include regressors as a method of training and making predictions. The reason we chose to do two algorithms is that we wanted to compare our results for both regression algorithms. We found them both to give pretty similar results when tested with different columns from our dataset. When running these two algos we saw that Random Forest Regression algo took longer time to predict the results as it uses decision trees and each tree represents a different ML algo and at the end the whole tree's predictions results are combined and a prediction is made which is even more accurate. But when running the same dataset with KNN_Regressors we saw it took little to no time to give us the similar results to Random Forest Regression. So we think that the amount of work which is being done in Random Forest Regression is not necessary for the prediction we

References

- https://www.tutorialspoint.com/machine_learning_with_python/index.htm
- https://in.springboard.com/blog/regression-vs-classification-in-machine-learning/#:~:text=Regression%20vs%20Classification%20in%20Machine%20Learning%3A%20Understanding%20the%20Difference,classification%20predicts%20discrete%20class%20labels.
- https://www.altexsoft.com/blog/datascience/preparing-your-dataset-for-machine-learning-8-basic-techniques-that-make-your-data-better/