Amaterasu Solar Radiation Model

Prepared by:

Amaterasu

Solar Data Specialists

1200 Broadway St • Ann Arbor, MI 48109

248-675-9827

Anisha Aggarwal, Michelle Jecmen, Andrew Li, Anish Sundaram,

Engineers in Training

Prepared for:

Professor Brian Goldsmith, Lead Technical Instructor

Professor Walburga Zahn, Technical Communications Lead

University of Michigan Engineering

April 28, 2021

Table of Contents

Abbreviations and Acronyms

CV	Cross Validation
RF	Random Forest
SD CV	Standard Deviation of Cross Validation
MAE	Mean Absolute Error
RMSE	Root Mean Absolute Error
KNN	K-Nearest Neighbors
PCA	Principal Component Analysis
MLP	Multi-Layer Perceptron

Tables

Figures

Executive Summary

As society continues to rely more on energy, it is crucial to utilize renewable energy sources. Solar power is a popular sustainable source with potential for immense energy yields. However, solar radiation is intermittent and dependent on a variety of environmental factors. The key to large-scale solar energy is accurately predicting radiation in order to optimize energy gain and economic feasibility.

Thus, *Amaterasu Solar Data Specialists* was tasked with building a solar forecasting model to predict radiation over time. Our client, an international solar company based in Honolulu, Hawaii, partnered with a weather agency to provide us detailed solar data. Our job was to analyze this dataset and answer client questions as well as provide our own analysis.

The original dataset consisted of 11 features and 32,684 samples. From this we engineered a variety of features such as time of day and day length. We analyzed feature correlations to find that temperature was the most important feature for predicting radiation. Using Python and various libraries including sklearn, pandas, and numpy, we programmed three different regression models: random forest, k-nearest neighbors, and multi-layer perceptron. We compared their accuracies on three tests (cross validation, mean absolute error, and root mean squared error) and discovered that the random forest performed the best compared to the other two models. We used gridsearch and cross validation in order to optimize our hyperparameters and ensure the highest predictive accuracy. We compared the models and concluded that the Random Forest is the best model for our client due to its performance on the accuracy tests, runtime, and the additional features it provides such as feature importances and residual plots.

This report contains a detailed analysis of the solar dataset as well as a design overview for each of the three models and their reported accuracies. The client questions are answered throughout the report.

Introduction

In February 2021, an international solar company based in Honolulu, Hawaii asked our team to develop a machine learning model to predict solar radiation versus over time. We were given a dataset from a weather company with detailed solar radiation information and tasked with analyzing the dataset, answering a set of client questions, building predictive machine learning models, and giving our final recommendations.

Producing solar energy is risky. Every five minutes, the solar company must create a "bid stack" listing the price and amount of solar energy it can produce within those five minutes. If the solar company produces less energy than estimated, they pay for the lost electricity. If the solar company produces more energy than estimated, they lose out on potential profits. Since solar radiation is intermittent and dependent on various environmental factors, accurately predicting energy output on short and long timescales is difficult. However, accurate prediction is necessary in order to maximize profits and meet energy demands.

Data Analysis

We were provided a detailed 32,684 by 11 csv file of solar radiation information. There were three main steps we used to analyze the data: Feature Engineering, Feature Correlations, and Visualization. Feature Engineering describes the process of extracting features from the raw data and creating new features, which may help our machine learning algorithm. Feature Correlations allow us to get a sense of which features may affect radiation the most. Finally, for visualization we create charts and graphs in order to better grasp on our data. After these steps, we conclude that temperature is the most important feature for predicting radiation.

Intro to Dataset & Feature Engineering

The first task *Amateratsu* accomplished was analyzing the dataset we were given. We read in the csv file as a Pandas dataframe using Python, then printed the head of the data frame (see Figure 1 on page 2). The dataset contains 11 features and 32,684 samples. Next, we dropped the samples with any null values: two samples were missing radiation values. Our task was to engineer the

'DayOfYear', 'MonthOfYear', 'TimeOfDay', and 'DayLength' from ‘Date', 'Time', 'TimeSunRise' and 'TimeSunSet' features. We converted the ‘Date’ feature into 'DayOfYear' and 'MonthOfYear’ using pd.DatetimeIndex’s “.day” and “.month” attributes. Then, we engineered ‘TimeOfDay’ by converting the time into total seconds. Lastly, we engineered 'DayLength' by subtracting ‘TimeSunSet” from ‘TimeSunRise’ and converting into seconds.

	UNIXTime	Date	Time	Radiation	Temperature	Pressure	Humidity	WindDirection	Speed	TimeSunRise	TimeSunSet
0	1475229326	9/29/2016 0:00	23:55:26	1.21	48	30.46	59	177.39	5.62	6:13:00	18:13:00
1	1475229023	9/29/2016 0:00	23:50:23	1.21	48	30.46	58	176.78	3.37	6:13:00	18:13:00
2	1475228726	9/29/2016 0:00	23:45:26	1.23	48	30.46	57	158.75	3.37	6:13:00	18:13:00
3	1475228421	9/29/2016 0:00	23:40:21	1.21	48	30.46	60	137.71	3.37	6:13:00	18:13:00
4	1475228124	9/29/2016 0:00	23:35:24	1.17	48	30.46	62	104.95	5.62	6:13:00	18:13:00

Figure 1: Head of Original Dataframe

Feature Correlation

Next, we wanted to find how the features correlated with radiation, as well as with each other. To accomplish this, we created a heatmap to plot each feature combination Pearson correlation (see Figure 2 on page 3). It is important to note that Pearson correlation only measures linear correlations. The strongest linear correlation is 0.73, between temperature and radiation. This makes sense since a higher temperature usually means the sun is shining more intensely.

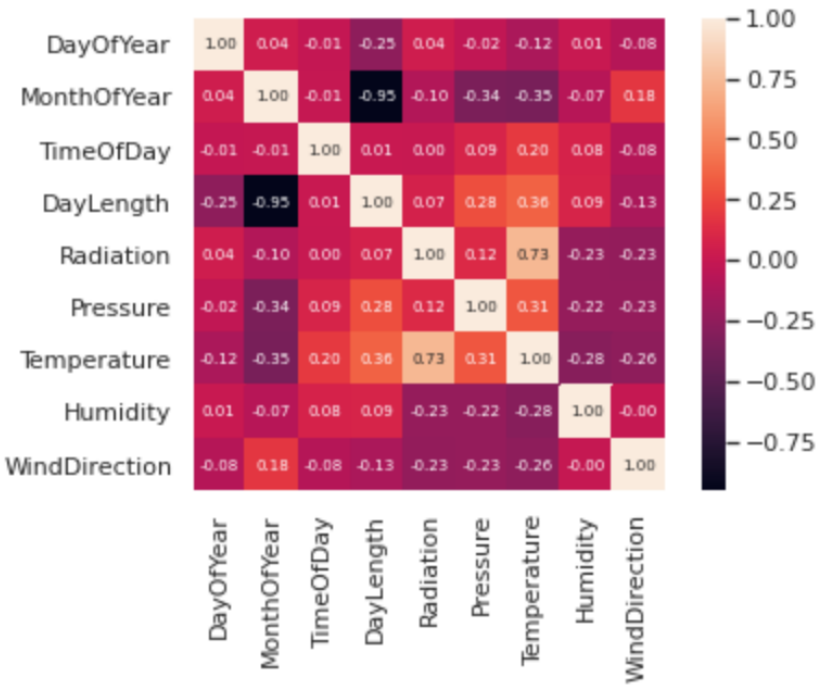

Figure 2: Heatmap of Pearson Correlations between Features

Visualization

After finding correlations between features, we wanted to plot each feature against radiation and analyze their distributions, since they could be more complex than linearly correlated. Figure 3 (page 4) shows scatter plots between our most important features, Temperature and TimeOfDay, versus Radiation. Radiation has a clear positive correlation with temperature, while radiation is the greatest in the middle of the day, and quickly decreases in intensity towards the start and end of the day.

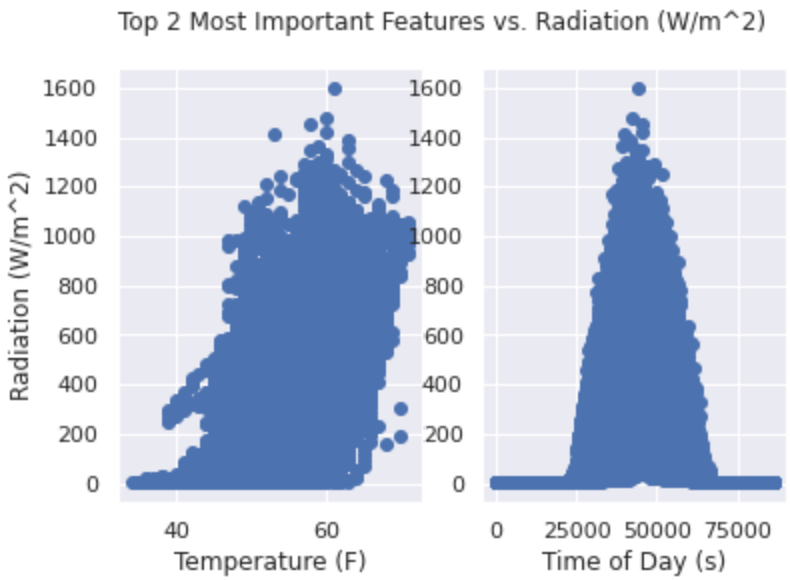

Figure 3: Temperature and Time of Day vs Radiation Plot

Regression Model Overview

Amateratsu was tasked to build two supervised regression machine learning models to predict radiation over time. We additionally built a third regression model to gain a broader understanding of the predictive abilities of different models on the solar dataset.

Train/Test Split

We used a consistent 70/30 train/test split for each model. We used a slightly larger test set because the solar dataset has repetitive, low variance samples. The dataset has about 32,000 samples over four consecutive months. The most important feature, temperature, has a standard deviation of 6.2. Additionally, radiation vs time of day shows a consistent daily trend. As a result of increasing the test set we were able to more thoroughly test each model without sacrificing training information. This allowed us to fully understand each model's accuracy before making our final recommendation and conclusions.

Model Process

A similar training and analysis process was used with each model. All models were built using Python and a corresponding sklearn library. All models were trained with the same nine features: temperature, pressure, humidity, wind direction, speed, day length, time of day, month, and year. We trained and optimized each model in the following steps:

1. Build an initial model with default hyperparameters.
2. Use k-fold cross validation (CV).
3. Report the cross validation scores.
4. Tune the hyperparameters with sklearn GridSearchCV.
5. Repeat steps 2-3 with optimized hyperparameters.
6. Visualize radiation over time.

Random Forest Regressor

The first model we built was a supervised random forest regression model. We built this model first because it provides additional dataset analysis and generally has high predictive power. A random forest model combines the predictions of many decision trees giving a more accurate prediction. The random forest model uses feature bagging to force individual decision trees to use randomized feature order rather than feature importance. This ensemble technique is notably important with our solar dataset because temperature and time of day have significantly higher feature importances than all other features.

Build Initial Model

The client requested we train and analyze an initial model before optimizing the model's hyperparameters. The random forest model used sklearn RandomForestRegressor with default hyperparameters. The notable default hyperparameters are 100 estimators and no maximum decision tree depth. These two hyperparameters are most important for determining the complexity and fit of the model.

Initial Model K-Fold CV

After fitting the model with default hyperparameters, we used 5-fold cross validation. This

allowed us to repeatedly evaluate the model during training on a withheld subsection of training data. The mean CV score reported an accuracy of 0.932 with a standard deviation of 0.006; this tells us the random forest model is accurate regardless of the withheld validation set. The mean absolute error (MAE) on the test set is 28.58 and the root mean square error (RMSE) on the test set is 77.31. These numbers initially indicate that a random forest model is well suited to this particular dataset.

Tune Hyperparameters

We used GridSearchCV to iterate over many hyperparameter combinations and determine which produce the most accurate model according to its cross validation score. We optimized the number of estimators and the maximum depth as these hyperparameters determine model complexity and thus whether the model is overfit or underfit.

The parameters we inputted into the grid search:

- 'n_estimators': [50, 100, 200, 300, 400, 500]
- 'max depth': [5, 10, 25]

The model was optimized with 500 estimators and a maximum depth of 25. Using the tuned hyperparameters we repeated 5-fold cross validation.

Cross Validation Scores

The accuracies for both the testing and training set are shown in Table 1 on page 7. Between the default and optimized hyperparameter models the MAE improved by 0.5% and the RMSE improved by 0.8%. This is not a significant improvement since the default model also had high accuracy scores. The mean CV of 0.932 and standard deviation of 0.005 tell us that the model is accurate regardless which hold-out validation set is used. Further evidence of the random forest's accuracy is shown with the relatively low MAE. The root mean square error of 77.0 shows variance in prediction error, likely due to high radiation outliers. A small increase in error between the training and test set is expected. However, a portion of the 17.2 MAE and 47.7 RMSE increase could be due to the random forest slightly overfitting the training set or an unfortunate outlier-rich test set. Overall, the random forest model is accurate.

Table 1: Optimized Random Forest Scores

	Test Set	Training Set
Mean CV	0.932	0.933
SD CV	0.005	0.005
MAE	28.4	11.2
RMSE	77.0	29.3

Visualize Radiation Over Time

We used two scatter plots to visualize radiation over time as predicted by the random forest model. To view the relative accuracy, the true data values are plotted against the corresponding time metric. The random forest model correctly models radiation trends throughout the day and over months. The model is less accurate at high radiation outliers in the middle of the day (see Figure 4) and scattered high radiation days throughout the year (see Figure 5 on page 8).

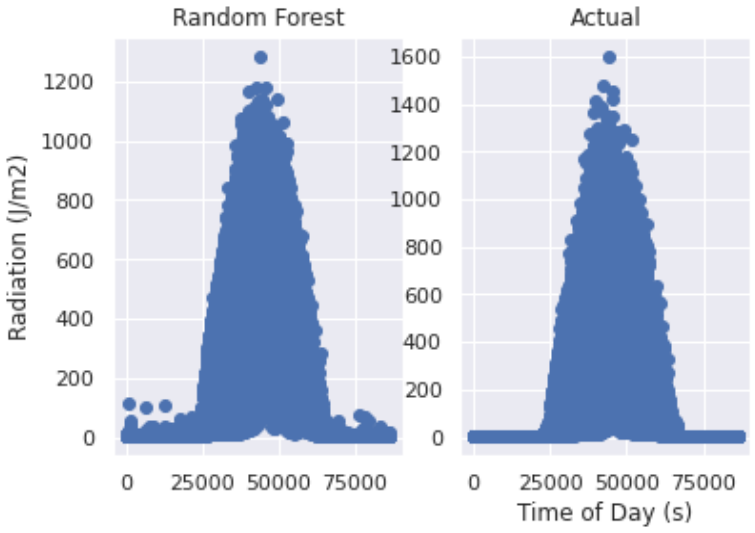

Figure 4: RF Radiation During the Day

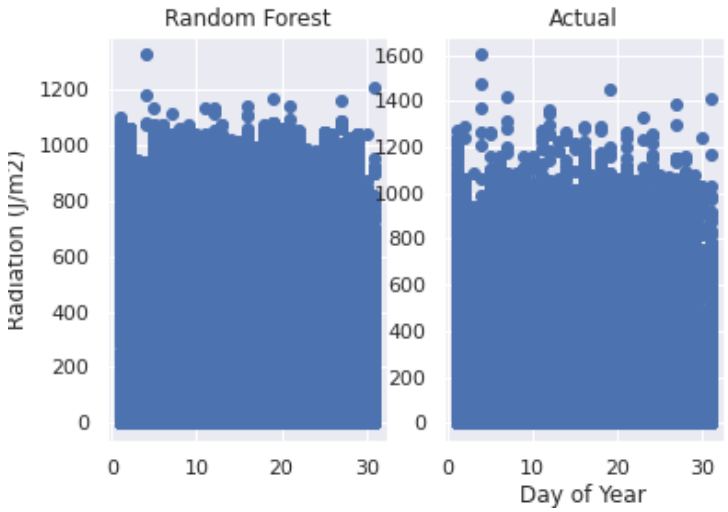

Figure 5: RF Radiation Over Days

Feature Importances

Random forests in sklearn have a feature importances attribute which calculates and displays how much each feature contributes to decreasing the weighted impurity of the model. Essentially, this attribute tells us which features are most important in our dataset. From Figure 6, we see that temperature is the most important feature followed by time of day. This confirms what we saw in our Pearson Correlation analysis as well as what we hypothesized in our dataset analysis.

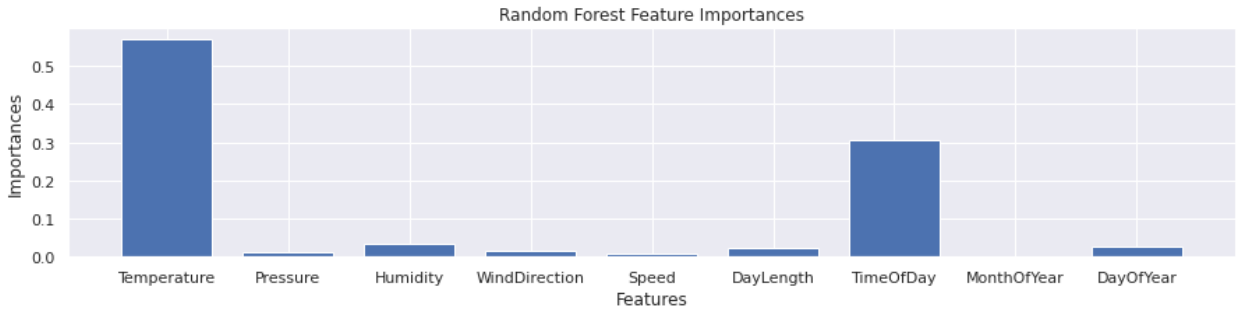

Figure 6: Random Forest Feature Importances Plot

K-Nearest Neighbor Regressor

The second model we built was a k-nearest neighbor (KNN) regression model. We choose this model because it is robust to outliers and works well on simple problems; during our data

analysis we concluded that radiation has a strong dependence on a few features with scattered high radiation outliers. The KNN model plots the data in 10-space and predicts radiation due to the average output of the nearest k neighbors. Since the model depends on distance, we standardized the ten training features.

Build Initial Model

The model used sklearn's KNeighborsRegressor initially trained with default hyperparameters. The initial model used five neighbors and a 'minkowski' distance metric. The number of neighbors is most important for determining the accuracy and fit of the model while the distance metric determines which neighbors are chosen.

Initial Model K-Fold CV

After fitting the model with default hyperparameters, we used 5-fold cross validation to improve the model during the training process. The cross validation scores reported an average accuracy of 0.876 with a standard deviation of 0.005. The mean absolute error on the test set is 48.8 and the root mean square error on the test set is 105. These numbers initially indicate that the k-nearest neighbor model is accurate at predicting the expected, average radiation but inaccurate at predicting high radiation outliers.

Tune Hyperparameters

GridSearchCV determined the number of neighbors and the distance metric that produced the most accurate model according to the cross validation score. These hyperparameters determine model accuracy and fit.

The parameters we inputted into the grid search:

- 'n_neighbors': [1, 2, 3, 5, 7, 10]
- 'metric': ['euclidean', 'minkowski', 'manhattan']

The model was optimized with 5 estimators and a 'manhattan' distance metric. Therefore, the only hyperparameter change is the distance metric.

Cross Validation Scores

We repeated 5-fold cross validation and reported the accuracy scores for the optimized model. The testing set accuracies are compared to the training set accuracies in Table 2. Between the default and optimized hyperparameter models the MAE improved by 11.8% and the RMSE improved by 8.4%. This improvement is the result of changing the distance metric from 'minkowski' to 'manhattan'. Therefore, a 'taxi-cab' method of determining the five closest points resulted in the most accurate k-neighbors model. The low increase in MAE and RMSE between the test and training set indicate that the model is not overfit. We estimate that most of the error results from high radiation outliers that appear in radiation over time graphs.

Table 2: Optimized K-Neighbors Scores

	Test Set	**Training Set**
Mean CV	0.896	0.896
SD CV	0.005	0.005
MAE	43.0	36.4
RMSE	96.1	80.7

Visualize Radiation Over Time

To visualize radiation over time as predicted by the k-nearest neighbors model, we plotted both the time of day and day of year (refer to Figure 7 and Figure 8 on page 11) against the predicted radiation. To observe the relative accuracy at different times, the plots of the corresponding true datum is shown. The model accurately represents radiation over time trends, but predicts values closer to the average. High radiation outliers are not predicted by the model.

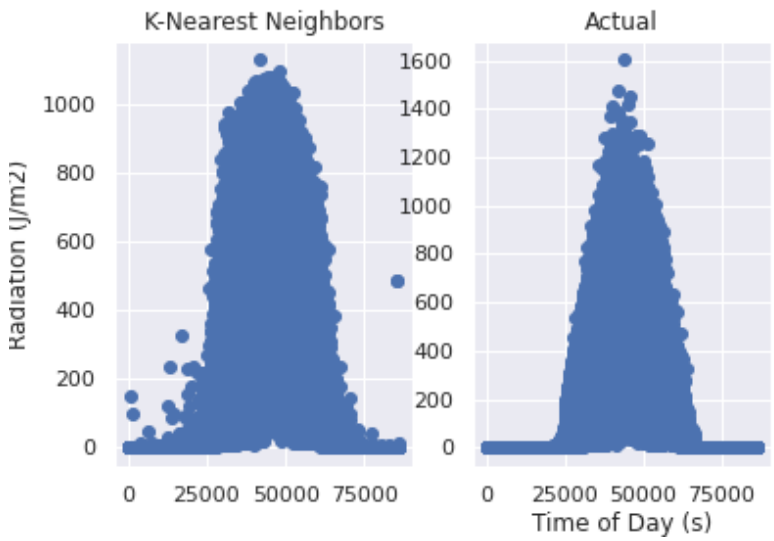

Figure 7: KNN Radiation During the Day

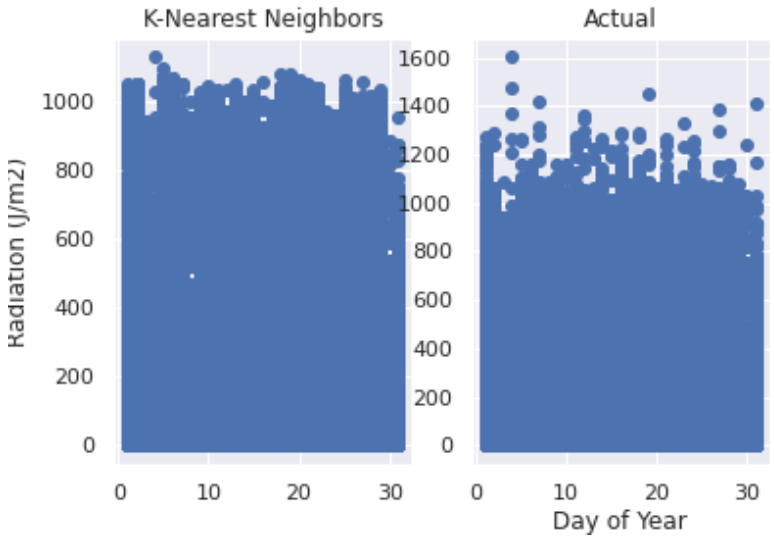

Figure 8: KNN Radiation Over Days

Principal Component Analysis

To further optimize the KNN model, our team decided to employ Principal Component Analysis (PCA). PCA allows us to test if a reduction in feature dimensionality improves model performance. This is important because KNN predicts values based on distance, which can be influenced by additional unnecessary features. We originally have nine principle components; the best one explaining 28% of the variance while the lowest two explain a combined 5.1%. We

tested the KNN model with a reduced seven components to determine if removing the two least important features improved performance. The mean CV score decreased by 5.9% and the MAE increased by 22.2%. Therefore, we determined all nine principal components were needed to achieve the best results from the KNN model.

Multi-Layer Perceptron Regressor

After building two requested supervised learning models, we built an additional multi-layer perceptron model. We selected this model because it is a typical feedforward neural network model where additional hidden layers can increase model performance. The main goal of this model was to determine if a neural network model is more accurate than the random forest and k-nearest neighbor models with respect to the solar dataset. To achieve high accuracy without spending unnecessary time training a default model, we started building the model by tuning the hyperparameters.

Tune Hyperparameters

We performed GridSearchCV to determine which hyperparameters gave the highest predictive accuracy. To tune the model without overfitting, we inputted a wide range of hidden layer sizes and maximum iterations. Increasing these two hyperparameters improves the accuracy but increases the risk of overfitting the model on the training set.
The parameters inputted into the grid search:

- ‘hidden_layer_sizes’: [10, 50, 100, 200, 500]
- ‘max_iter’: [100, 200, 500, 750]

The model was optimized with 100 hidden layers and 200 maximum iterations.

Cross Validation Scores

The cross validation scores for both the testing and training set are given in Table 3 (page 13). The average distance between our predicted datum and their true values is 80 with a variance of 120.8. These are fairly large error scores, indicating that the MLP predictions are not consistently accurate. Since the testing and training set have similar accuracy scores, it is unlikely that the

model is overfit. These results tell us that the solar dataset is not well suited for a neutral network model.

Table 3: Optimized Multi-Layer Perceptron Scores

	Test Set	**Training Set**
Mean CV	0.847	0.848
SD CV	0.010	0.010
MAE	80.0	80.9
RMSE	120.8	121.7

Visualize Radiation Over Time

We plotted radiation levels during the day as well as over many days to visualize the model's predicted radiation over time (refer to Figure 9 and Figure 10 on page 14). To understand the accuracy of these predictions, we plotted the real-value counterparts of each time metric. These plots explain the relatively high error and low accuracy scores from the multi-layer perceptron. In both the plots the datum is concentrated along the average radiation for that time. The predicted radiation variance and range appear to be significantly lower. Therefore, these plots give an idea of the average radiation over time curves.

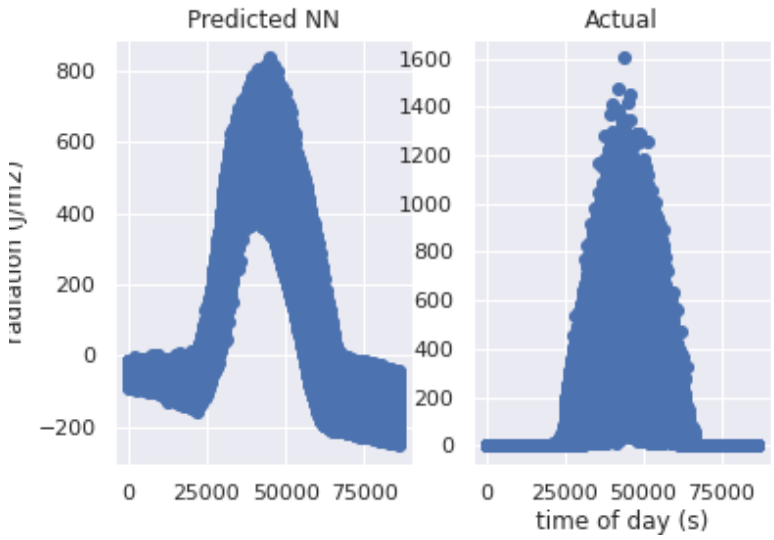

Figure 9: MLP Radiation During the Day

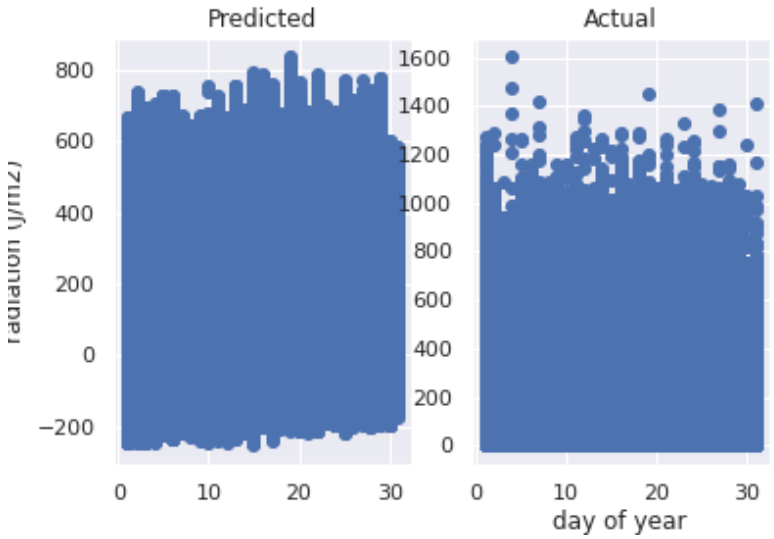

Figure 10: MLP Radiation Over Days

Comparing Models

We compared the results of each of the three machine learning models on various accuracy tests. Table 4 (page 15) shows these results. Optimal values for the mean CV and R^2 tests are values closest to 1. Optimal values for MAE and RMSE tests are low values. The random forest model has the best performance on all four tests. Its mean CV and R^2 are closest to 1 with values of 0.93 and 0.94 respectively. Its MAE and RMSE scores are the lowest out of the three models

with values of 28.39 and 76.97 respectively. The k-nearest neighbors model is a close second while the multi-layer perceptron model places last in all tests. From this comparison, it seems the random forest is the best model. Furthermore, taking a look at the residual plot for the random forest model (refer toFigure 11) we can see that most residuals cluster around zero, meaning that the predicted radiation values varied very little from the actual values.

Table 4: Accuracy Scores for All Three Models

	RF	**KNN**	**MLP**
Mean CV	0.930	0.895	0.897
MAE	28.39	48.79	101.12
RMSE	76.97	104.90	146.26
R^2	0.94	0.89	0.78

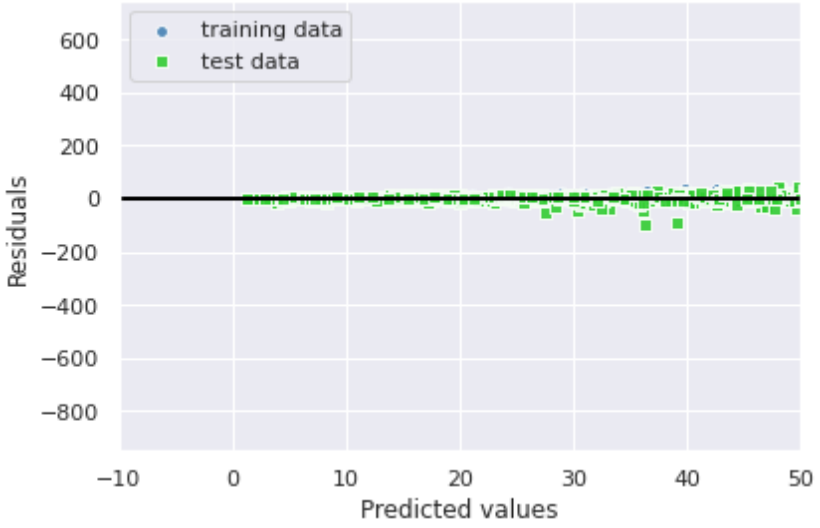

Figure 11: Residual Plot for RF Model

Random Forest

The random forest model is more accurate in predicting radiation over time than the k-nearest neighbor and multi-layer perceptron models. The random forest has the highest accuracy scores

and the lowest errors. The use of feature bagging and ensemble techniques give the random forest higher predictive accuracy than the other two models for the solar dataset.

Conclusion

This section provides our results, recommendations, and some final remarks on the project as a whole.

Team Contributions

Throughout the duration of this project our team worked collaboratively to distribute work and accomplish the tasks given to us. We met regularly via Zoom to code and discuss further steps. Michelle typically shared her screen and typed code in a shared Google Colab Sheet as we discussed it. When it came time to create the final presentation we split it up by topic and each designed the slides we would present. For the final report we took the same approach and worked on sections that we presented on and therefore were more familiar with. Anisha worked on introductory and background project information. Andrew covered the data set analysis. Michelle reported each of the three supervised models. Anish spoke about our results and conclusions. Overall, our team worked well together and was able to complete the project in a timely manner.

Working Via Zoom

Zoom was a great resource for us during the pandemic. It allowed us to quickly and easily meet. In addition, sharing screens during meetings made working together much smoother as we could all look at the same thing as we discussed. Furthermore, we relied heavily on Google Drive to share our work and edit documents together in real time. Working in person would have been ideal, but given the circumstances we would say that Zoom was a useful tool that worked well for us throughout this project.

Results and Recommendations

To summarize, our team was tasked with analyzing a dataset in order to build a machine learning model to predict radiation over time. We examined the dataset, engineered new features, and

looked at feature correlations. We determined that temperature was the feature most strongly correlated with radiation. We then proceeded to build three different machine learning models: random forest, k-nearest neighbors, and multi-layer perceptron. Comparing the performances and accuracies for each model proved that random forest is best at predicting radiation over time. After our research and testing, we can confidently recommend that our client use our random forest model. It has the best accuracy results out of all three models with a mean CV score of 0.930, a MAE score of 28.39, a RMSE score of 76.69, and a R^2 score of 0.940. In addition, the random forest provides additional features such as feature importances and residual plots which may be beneficial to our client.

References

[1] Interesting Engineering, “The UAE Has Unveiled the World's Largest Solar Power Project”, Interesting Engineering. [Online]. Available: https://interestingengineering.com/the-uae-has-unveiled-the-worlds-largest-solar-power-project. [Accessed: Apr. 12, 2021]

[2] B. Goldsmith and W. Zahn. ENGR W100. Topic: “Engineering 100 Design Project Prompt.” College of Engineering, University of Michigan, Ann Arbor, MI, Feb 6, 2021.

[3] SciKit Learn, “1. Supervised Learning”, SciKit Learn. [Online]. Available: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning. [Accessed: Mar 10, 2021] .

Appendix A: Python Code for RF Model

Here we present the code used for the Random Forest model.

```
# Libraries used throughout the project:

import io
import pandas as pd
import numpy as np

  # Used for graphics
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt

  # Used to split data and get accuracy measures
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

  # Used for cross validation and hyperparameter tuning
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score

  # The three models we used
from sklearn.ensemble import RandomForestRegressor
from sklearn import neighbors
from sklearn.neural_network import MLPRegressor

  # Additional analysis
from sklearn.decomposition import PCA
```

```
[ ]  forest = RandomForestRegressor()
     forest.fit(X_train, y_train)
```

```
[ ] # Grid search cross validation to optimize hyperparameters

    # Ran GridSearch many times, with changing parameters -
      # run with params = {'n_estimators': [50,100], 'max_depth': [2, 5, 10]}, final model: n_est = 100, max_depth = 10
      # run with params = {'n_estimators': [100,200,300], 'max_depth': [5,10,15]}, final model: n_est = 300, max_depth = 15
      # run with params = {'n_estimators': [300,400,500], 'max_depth': [20,25]}, final model: n_est = 500, max_depth = 25

    params = {'n_estimators': [100,200,400,500], 'max_depth': [10,20,25]}

    forest = GridSearchCV(RandomForestRegressor(criterion='mse', random_state=42), params, cv=5)

    forest.fit(X_train, y_train)
    y_train_pred_rf = forest.predict(X_train)
    y_test_pred_rf = forest.predict(X_test)
```

```
    # Best model used with n_estimators = 500 and max_depth = 25
    forest.best_estimator_
```

```
[ ] # Prints the CV accuracies for the test set
    accuracies = cross_val_score(estimator=forest, X=X_train_std, y=y_train, cv=5, n_jobs=-1)

    print('cv scores: ', accuracies)
    print('mean cv score: ', accuracies.mean())
    print('standard deviation of cv score: ', accuracies.std())
    print('MAE test: ', mean_absolute_error(y_test, y_test_pred_rf))
    print('RMSE test: ', np.sqrt(mean_squared_error(y_test, y_test_pred_rf)))
    print('R^2 test: ', r2_score(y_test, y_test_pred_rf))
```

```
[ ] # Prints the CV accuracies for training set
      # - Similar accuracies on test and training set show that model is not overfit
    print('cv scores: ', accuracies)
    print('mean cv score: ', accuracies.mean())
    print('standard deviation of cv score: ', accuracies.std())
    print('MAE test: ', mean_absolute_error(y_train, y_train_pred_rf))
    print('RMSE test: ', np.sqrt(mean_squared_error(y_train, y_train_pred_rf)))
    print('R^2 test: ', r2_score(y_train, y_train_pred_rf))
```

```
[ ] forest = RandomForestRegressor(criterion='mae', random_state=42, n_estimators=500, max_depth=25)
    forest.fit(X_train,y_train)
```

```
    RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mae',
                          max_depth=25, max_features='auto', max_leaf_nodes=None,
                          max_samples=None, min_impurity_decrease=0.0,
                          min_impurity_split=None, min_samples_leaf=1,
                          min_samples_split=2, min_weight_fraction_leaf=0.0,
                          n_estimators=500, n_jobs=None, oob_score=False,
                          random_state=42, verbose=0, warm_start=False)
```

```
[ ] y_train_pred_rf = forest.predict(X_train)
    y_test_pred_rf = forest.predict(X_test)
```