INTRODUCTION

Solar energy is one of the most important renewable resources we have and knowing how much energy a system will produce is essential for running it efficiently. This study takes a close look at a solar energy system in Berkeley, California, to see how things like temperature and wind speed affect power generation. By using machine learning models, it's working toward creating accurate predictions for future energy output, an effort that could make sustainable energy management even more effective.

PURPOSE OF THE PROJECT

- This project dives into the energy production patterns of a solar energy system in Berkeley, CA, to figure out how environmental factors impact power generation. Using machine learning, the goal is to build predictive models that can accurately forecast future energy output.

About Dataset: The dataset used in this project contains detailed records of a solar energy system installed in Berkeley, California and it has 2929 rows and 16 columns.

LIST OF OBJECTIVES

- Analyze the relationship between environmental factors (temperature, wind speed, solar noon distance) and solar energy production.
- Perform exploratory data analysis to uncover trends and patterns in power generation.
- Handle missing data and preprocess the dataset to ensure accuracy and consistency.
- Build and optimize machine learning models for predicting solar energy output.
- Evaluate model performance using metrics such as Mean Squared Error (MSE) and R-squared (R^2).

Data Description

The dataset appears to be a Solar Panel dataset, which is typically used to predict Solar Power Generation based on various features.

a. Overview of Dataset:

Total Entries:2929
Features:16

b. Columns:

The dataset includes various columns that capture environmental and operational data related to solar energy production. Below is a detailed list of the columns:

Numerical Columns:

Day of Year: The day number within the year (1–365).

Month: The month of the year (1–12).

Day: The day of the month (1–31).

Average Temperature (Day): The average temperature of the day in degrees Celsius.

Average Wind Direction (Day): The wind direction during the day, measured in degrees.

Relative Humidity: The relative humidity percentage during the day.

Power Generated: The amount of power generated by the solar energy system, measured in kilowatts (target variable).

Distance to Solar Noon: The time difference in hours between the observation and solar noon, affecting energy production.

Average Wind Speed (Day): The average wind speed for the day in meters per second.

Visibility: The atmospheric visibility, affecting sunlight intensity.

Average Wind Speed (Period): Wind speed during the specified observation period.

Average Barometric Pressure (Period): The air pressure during the observation period.

Categorical Columns:

Is Daylight: A Boolean value indicating whether it is daylight (True/False).

Sky Cover: The amount of cloud cover during the day, typically represented as categorical values (clear, partly cloudy, overcast).

Year: The year of the observation (2020, 2021).

First Hour of Period: The first hour of the observation period, which helps in understanding the time of day (morning or evening).

Choice of Dependent and Independent Variables and Selection of Algorithms

a. Independent Variables:

This is the variable that is manipulated or controlled by the researcher to observe its effect on the dependent variable. It's the "cause" in a cause-and-effect relationship.

Average Temperature (Day): The mean daily temperature affecting solar panel efficiency.
Average Wind Speed (Day): Wind speed that may influence the cooling of solar panels.
Relative Humidity: Moisture content in the air, potentially impacting energy production.
Distance to Solar Noon: Measures how close the observation time is to solar noon, strongly correlated with solar energy output.
Visibility: Indicates atmospheric clarity, which can affect sunlight intensity.
Average Barometric Pressure (Period): Air pressure during the observation period.
Categorical Variables:

Is Daylight: Indicates whether the observation occurs during daylight hours (True/False).
Sky Cover: Reflects cloud coverage, which affects sunlight reaching the solar panels.
Year, Month, Day: Date-related features to account for seasonal and temporal patterns.
First Hour of Period: Starting hour of the observation period, influencing sunlight availability.

b. Dependent Variables:

This variable is what you measure in the experiment and what is affected during the experiment. It is the "effect" or the outcome that depends on the independent variable.

Power Generated: This is the primary variable of interest, representing the amount of energy (in kilowatts) produced by the solar energy system. It is the output the project aims to predict using machine learning models.

c. Selection of Algorithms:

For this project, the **Random Forest Regressor** algorithm was selected to model and predict solar energy production. Below are the reasons for this choice:

- Solar energy production is influenced by complex, non-linear relationships between environmental factors. Random Forest is well-suited for capturing such patterns without requiring explicit assumptions about the underlying relationships.
- The dataset includes potential outliers in power generation and other variables. Random Forest is less sensitive to outliers compared to linear models.
- Random Forest provides insights into the importance of each predictor, helping to identify the most significant factors influencing energy production.
- It works well with both numerical and categorical variables, making it suitable for this dataset with a mix of feature types.
- By aggregating predictions from multiple decision trees, Random Forest reduces the risk of overfitting, especially in datasets with noise.

Exploratory Data Analysis (EDA)

Feature Analysis and Data Conversion

Firstly, the dataset was analyzed to ensure that all features were represented by appropriate data types. “Numerical features” such as “temperature”, “wind speed” and “power generated” were converted to integer or float formats, while categorical features like “daylight” status and “sky cover” were assigned as categorical data types. This data transformation was essential for ensuring compatibility with the predictive model and enabling efficient data analysis.

Missing Value Analysis

A missing value inspection revealed one missing data in the "Average Wind Speed (Period)" column. This single missing value accounted for 0.03% of the dataset, a seemingly unnecessary proportion. However, removing the observation would disrupt the dataset's structure. Specifically, Figure 1.0 highlights that each "First Hour of Period" typically contains 365 records, matching the days of a year. Removing this data point would reduce one group to 364, potentially compromising the model’s ability to capture patterns effectively.

Handling Missing Data

To address this, the missing value was imputed using the column's mean value rather than deleting the observation. This approach preserved the dataset's completeness and consistency across features while maintaining the annual patterns critical to predictive modeling. Below, Code Block is demonstrating how to handling missing data check data percentages again specifically.

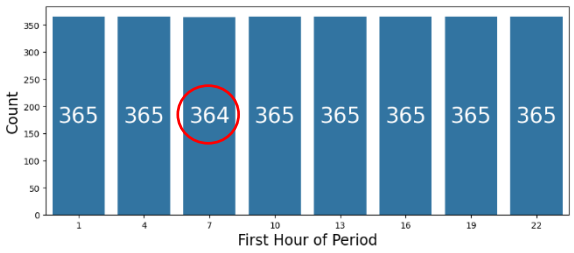

Figure 1.0: First Hour of Period Data Distribution

Code block:

```
data['Average Wind Speed (Period)'] = data['Average Wind Speed
(Period)'].fillna(data['Average Wind Speed (Period)'].mean())
null_counts = data.isnull().sum()
null_percentage = (null_counts / len(data))*100
formatted_null_percentage = null_percentage.apply(lambda x: f"%{x:.2f}")
print("----------Null Percentages After Cleaning-------------")
print(formatted_null_percentage)
```

Examination of Categorical Values

This step analyzes the dataset's categorical variables to ensure there are no unexpected or excessive values outside the expected categories. The goal is to validate the dataset's categorical columns and identify any anomalies that might affect data analysis or predictive modeling. As a result, the categorical variables are well-structured and contain no excessive or unexpected values.

```
Unique Values for Is Daylight Column:
[False, True]
Categories (2, bool): [False, True]
Count: 2

Unique Values for Sky Cover Column:
[0, 1, 3, 4, 2]
Categories (5, int64): [0, 1, 2, 3, 4]
Count: 5

Unique Values for Year Column:
[2008, 2009]
Categories (2, int64): [2008, 2009]
Count: 2

Unique Values for First Hour of Period Column:
[1, 4, 7, 10, 13, 16, 19, 22]
Categories (8, int64): [1, 4, 7, 10, 13, 16, 19, 22]
Count: 8
```

Figure 1.1: Unique Values and Counts

Numeric Statistical Summary

The purpose of this step is to examine the numeric variables in the dataset and provide statistical insights, such as their mean, standard deviation, and distribution across the dataset.

The numeric summary shows important details about the dataset. “Day of Year” covers all days of the year. “Distance to Solar Noon” averages 0.50, meaning many records are close to peak solar times. “Temperature” and “Wind Speed” are mostly normal but sometimes extreme. “Power Generated” averages 6979.85 kW, with zero values likely during the night. Percentiles help understand data trends and prepare for further analysis.

```
----------Numberic Statistical  Summary----------
                                    count         mean           std  \
Day of Year                        2920.0   183.334247    105.769919
Month                              2920.0     6.526027      3.448442
Day                                2920.0    15.720548      8.797754
Distance to Solar Noon             2920.0     0.503294      0.298024
Average Temperature (Day)          2920.0    58.468493      6.841200
Average Wind Direction (Day)       2920.0    24.953425      6.915178
Average Wind Speed (Day)           2920.0    10.096986      4.838185
Visibility                         2920.0     9.557705      1.383884
Relative Humidity                  2920.0    73.513699     15.077139
Average Wind Speed (Period)        2920.0    10.129154      7.260303
Average Barometric Pressure (Period)  2920.0    30.017760      0.142006
Power Generated                    2920.0  6979.846233  10312.336413

                                        min         25%         50%  \
Day of Year                        1.000000   92.000000   183.000000
Month                              1.000000    4.000000     7.000000
Day                                1.000000    8.000000    16.000000
Distance to Solar Noon             0.050401    0.243714     0.478957
Average Temperature (Day)         42.000000   53.000000    59.000000
Average Wind Direction (Day)       1.000000   25.000000    27.000000
Average Wind Speed (Day)           1.100000    6.600000    10.000000
Visibility                         0.000000   10.000000    10.000000
Relative Humidity                 14.000000   65.000000    77.000000
Average Wind Speed (Period)        0.000000    5.000000     9.000000
Average Barometric Pressure (Period)  29.480000   29.920000    30.000000
Power Generated                    0.000000    0.000000   404.000000

                                        75%          max
Day of Year                     275.000000   366.000000
Month                            10.000000    12.000000
Day                              23.000000    31.000000
Distance to Solar Noon            0.739528     1.141361
Average Temperature (Day)        63.000000    78.000000
Average Wind Direction (Day)     29.000000    36.000000
Average Wind Speed (Day)         13.100000    26.600000
Visibility                       10.000000    10.000000
Relative Humidity                84.000000   100.000000
Average Wind Speed (Period)      15.000000    40.000000
Average Barometric Pressure (Period)   30.110000    30.530000
Power Generated                12723.500000  36580.000000
```

Figure 1.2: Numeric data description by pandas describe () option

Categorical Statistical Summary

The goal of this step is to summarize the categorical variables in the dataset, providing insights into their unique values, most frequent categories, and distribution.

The categorical summary shows that the dataset is clear and organized. There are records for two years, with more data from 2009 (1944 entries). Observations are spread across eight time periods (like 1, 4, 7), with 365 records for each time. Most data (1805 entries) come from daylight hours, which is important for studying solar energy. Sky cover has five levels, and partial cloud cover (1) is the most common with 776 entries. This makes the dataset easy to use for analysis.

```
----------Categorical Statistical Summary----------

        Year  First Hour of Period Is Daylight  Sky Cover
count   2920                  2920        2920       2920
unique     2                     8           2          5
top     2009                     1        True          1
freq    1944                   365        1805        776
```

Figure 1.3: Categorical Data Description by pandas describe () option

Histogram Visualizations

The visualizations show a clear and balanced distribution of the dataset's categorical variables. Most data was collected in 2009, with 1944 records compared to 976 in 2008. Observations are evenly distributed across eight time periods, each recorded 365 times, ensuring comprehensive time coverage. The majority of the data (1805 entries) comes from daylight hours, aligning with the focus on solar energy production. For sky cover, five levels are present, with partial cloud cover being the most frequent at 776 entries. This structure supports reliable and detailed analysis.

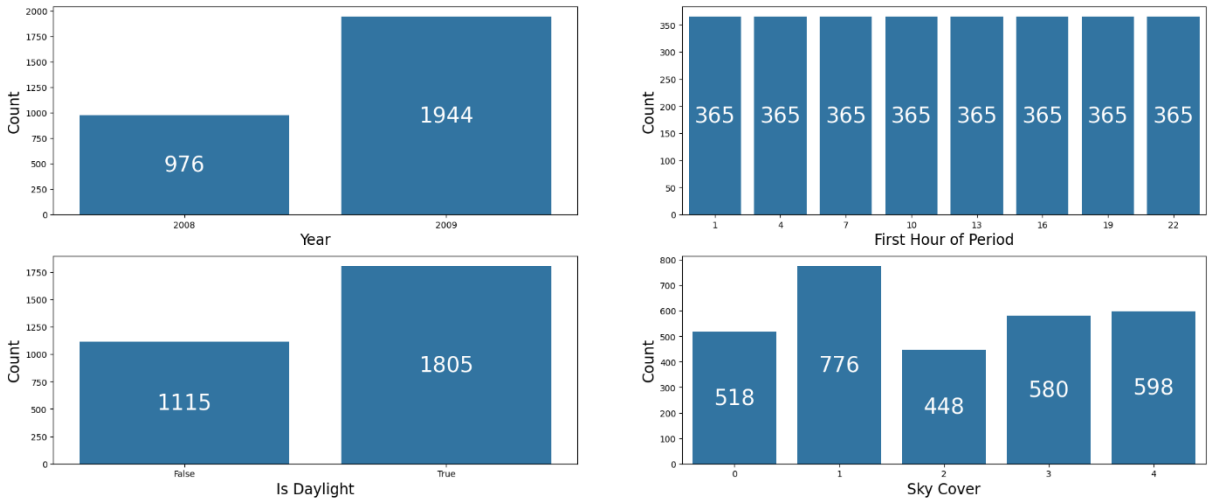

Figure 1.4: Distribution of Categorical Data

The histogram illustrates the distribution of the target variable, Power Generated. The data is highly skewed, with the most observations concentrated near zero power output. Over 1,500 records show minimal or no power generation, likely representing nighttime or periods of low solar activity.

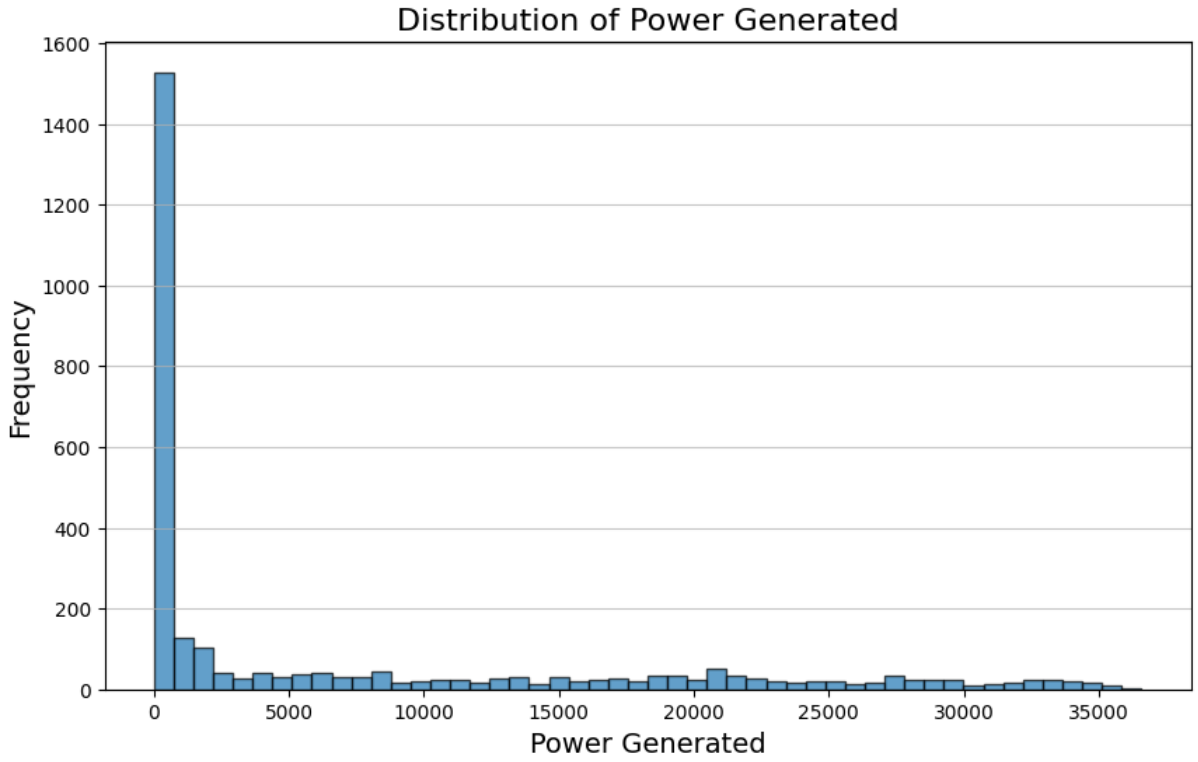

Figure 1.5: Distribution of Power Generation

Scatter Plots

The scatter plots reveal the relationships between environmental factors and power generation. Power output is highest at moderate temperatures (55–65°C), while extremely high or low temperatures are less available. Wind speed shows minimal impact, with most data clustering below 15 m/s. Power generation decreases as sky cover increases, confirming the negative effect of clouds. Moderate humidity (50–70%) aligns with higher outputs, while high humidity reduces energy production. A strong negative relationship is seen with distance to solar noon, where power peaks near noon and declines as the distance increases. These insights highlight key factors affecting solar energy efficiency.

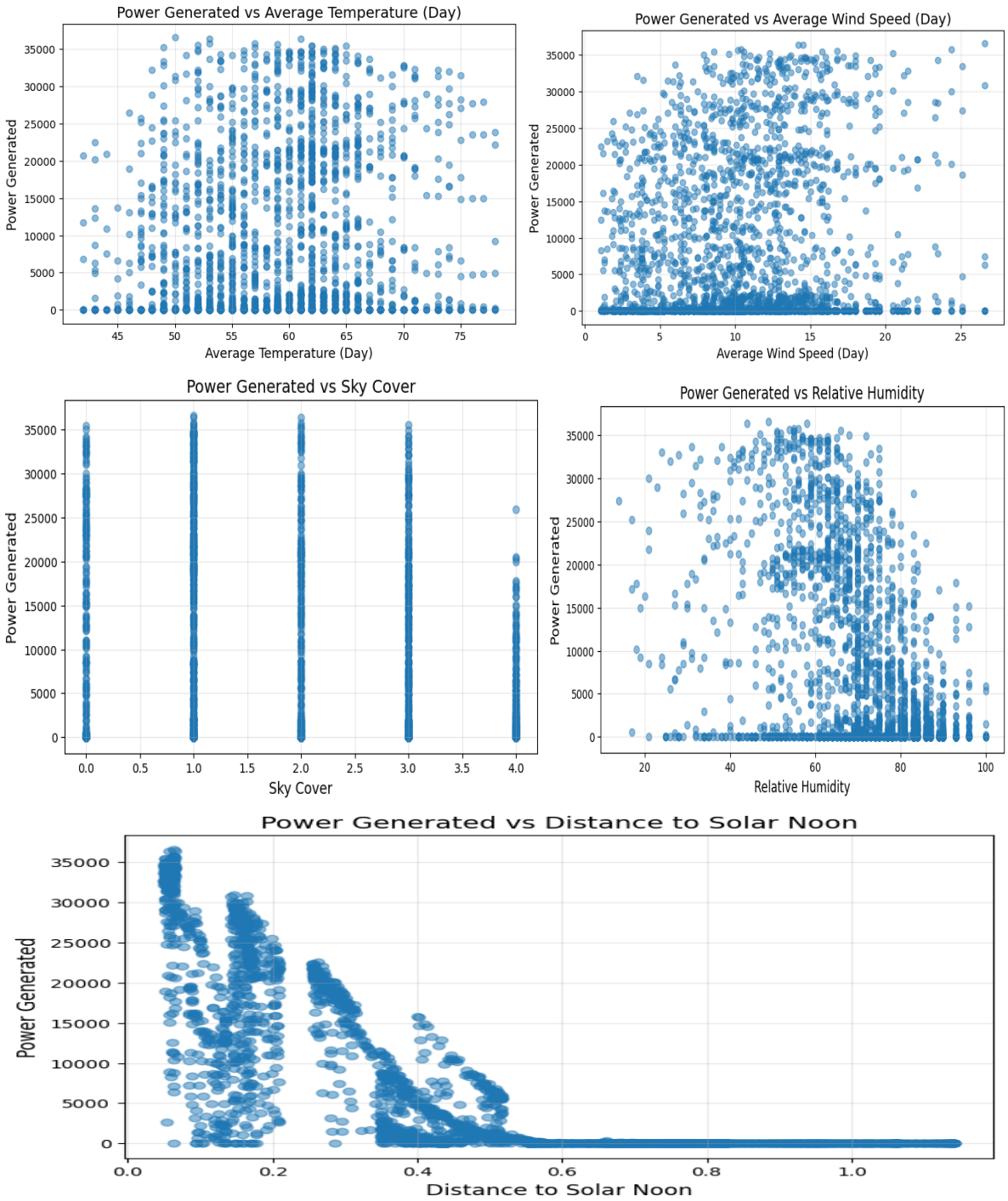

Figure 1.6: Relation of Environmental Variables

Correlation Matrix

The correlation matrix provides insights into the relationships between numerical features and power generation. A strong negative correlation (-0.75) exists between Distance to Solar Noon and Power Generated, confirming that energy production is highest near solar noon. Average Temperature (Day) has a weak positive correlation (0.13) with power generation, suggesting a minor influence. Relative Humidity shows a moderate negative correlation (-0.52), indicating its negative impact on energy output. Other variables, such as wind speed and barometric pressure, show minimal correlation with power generation. These findings highlight key predictors for solar energy production and guide feature selection for modeling.

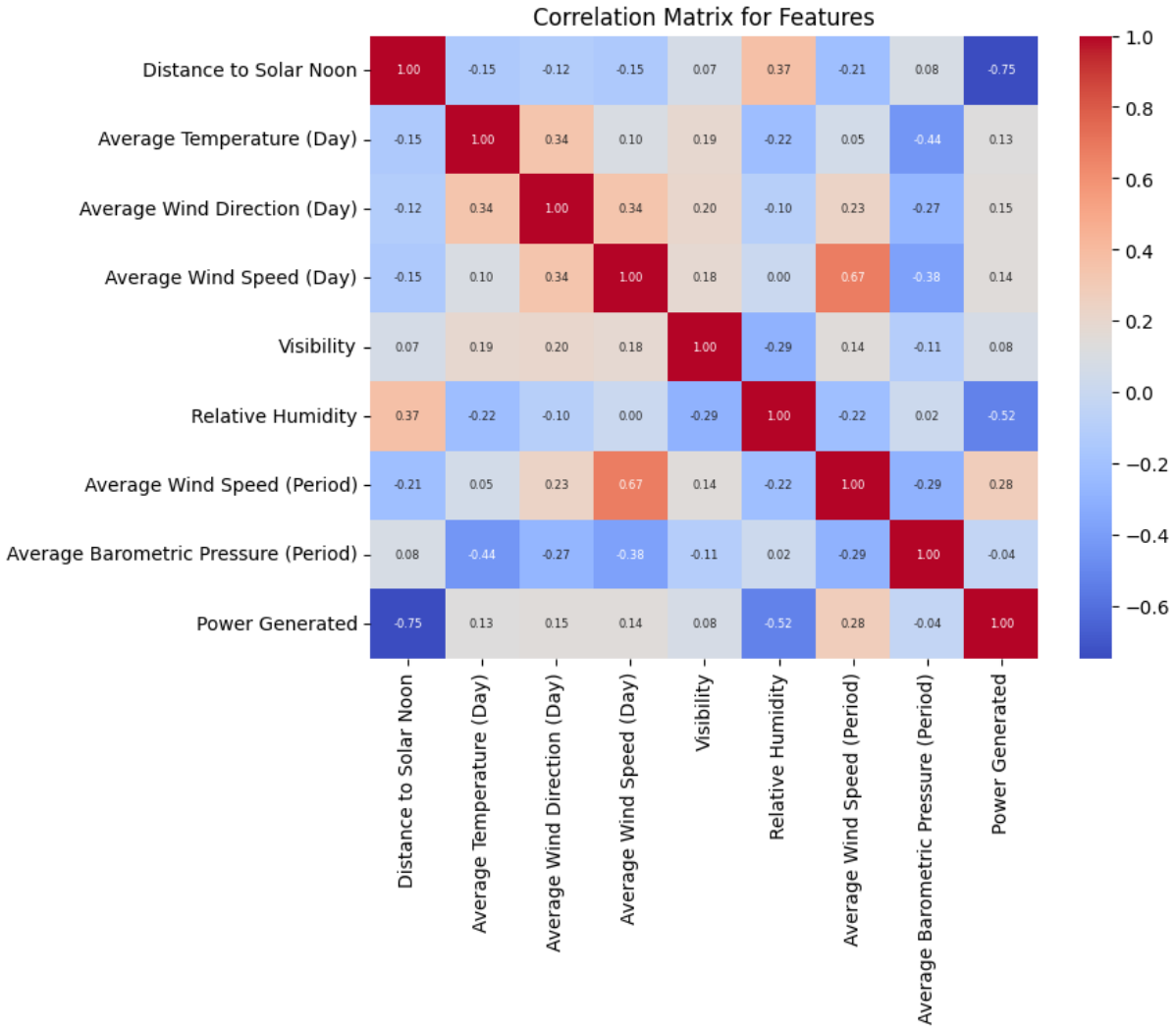

Figure 1.7: Correlation Matrix

Line Plot

The line plot shows daily total power generation over time, revealing seasonal trends and fluctuations. After Integrate separated Month,Day and Year features as a creation of “Date” feauture, observed that energy production is lower during the winter months (late 2008) and increases significantly as the year progresses into spring and summer, peaking in mid-2009. This pattern aligns with expected solar energy trends and confirms the seasonal impact on solar.

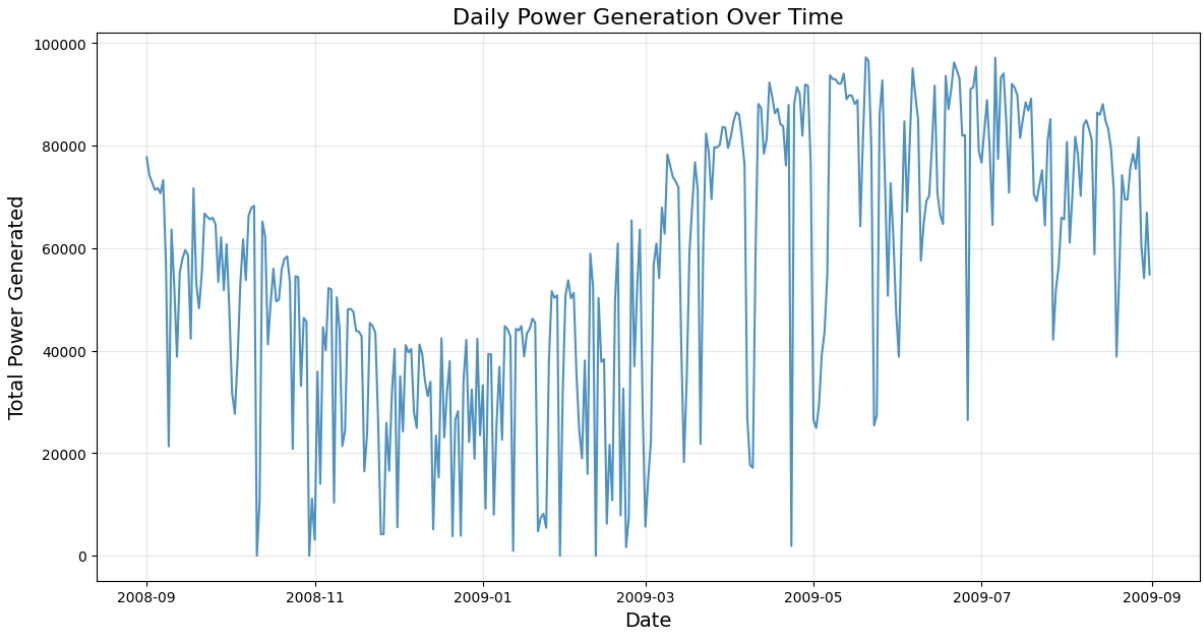

Figure 1.8: Daily Power Generation

Data Preprocessing and Feature Engineering

1. Changes the “is daylight” column from True/False values to binary (1 for True, 0 for False) and then converts the column back to the bool data type for better representation.

```
df["is daylight"]=df["is daylight"].apply(lambda x: 1 if x==True else 0)
df["is daylight"]=df["is daylight"].astype("bool")
```

2. Three different functions created to feature engineering, such as create season, time of day and solar proximity

```
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Autumn'

#-----------------------------------------------------

def time_of_day(hour):
    if hour < 6:
        return 'Night'
```

```
    elif hour < 12:
        return 'Morning'
    elif hour < 18:
        return 'Afternoon'
    else:
        return 'Evening'

#-------------------------------------------------------

def solar_proximity(distance):
    if distance < 0.2:
        return 'Near Noon'
    elif distance < 0.5:
        return 'Midday'
    else:
        return 'Far from Noon'


df['season'] = df['month'].apply(get_season) #create season feature from month value
df['time of day'] = df['first hour of period'].apply(time_of_day)
df['solar proximity'] = df['distance to solar noon'].apply(solar_proximity)
```

3. One-hot encoding applied for these new features.

```
df = pd.get_dummies(df, columns=['season', 'time of day', 'solar proximity','sky
cover'], drop_first=True) #turn these columns' clusters into the columns typed as
bool
```

4. Drops unnecessary columns such as year, month, day, day of year, and first hour of period since new features (season, time of day) already incorporate this information.

```
columns_to_drop = ['year', 'month', 'day', 'day of year','first hour of period']
df.drop(columns=[col for col in columns_to_drop if col in df.columns], inplace=True)
df.head()
```

5. dataset columns are standardized to lowercase.

```
print("\nOriginal Column Names:")
print(df.columns.tolist())

df.columns = df.columns.str.strip().str.lower()
print("\nColumn Names after standardization:")
print(df.columns.tolist())
```

6. As a result, final dataset is created as below.

	is daylight	distance to solar noon	average temperature (day)	average wind direction (day)	average wind speed (day)	visibility	relative humidity	average wind speed (period)	average barometric pressure (period)	power generated	...	season_Winter	time of day_Evening	time of day_Morning	time of day_Night	solar proximity_Midday	solar proximity_Near Noon	sky cover_1	sky cover_2	sky cover_3	sky cover_4
0	False	0.859897	69	28	7.5	10.0	75	8.0	29.82	0	...	False	False	False	True	False	False	False	False	False	False
1	False	0.628535	69	28	7.5	10.0	77	5.0	29.85	0	...	False	False	False	True	False	False	False	False	False	False
2	True	0.397172	69	28	7.5	10.0	70	0.0	29.89	5418	...	False	False	True	False	True	False	False	False	False	False
3	True	0.165810	69	28	7.5	10.0	33	0.0	29.91	25477	...	False	False	True	False	False	True	False	False	False	False
4	True	0.065553	69	28	7.5	10.0	21	3.0	29.89	30069	...	False	False	False	False	False	True	False	False	False	False

5 rows × 23 columns

Figure 2: Processed dataset

Creation of Prediction Model

Preparation

Firstly, I need to split the dataset into training and testing subsets, which is crucial for building and evaluating machine learning models:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

This split ensures a balanced evaluation of the model by using part of the data for training and reserving another part for validation.

Standardization of the numerical variables in the dataset, as following step is an essential preprocessing step for many machine learning algorithms.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train) # Scales the training data (X_train).
X_test = scaler.transform(X_test)# Scales the test data.
```

Prediction Model

The Prediction model that I used is "Random Forest Regressor", I chose that because:

1. Random Forest is an ensemble learning method that combines the predictions of multiple decision trees to improve accuracy and reduce overfitting.
2. It is particularly effective when dealing with non-linear relationships in the data, as it can capture complex patterns.

There are 2 metrics to interpret outputs:

1. **MSE:**

   -Useful to understand the magnitude of errors.

   -However, looking at MSE alone is not enough because this value varies with the magnitude of the target variable.

2. **R^2:**

   -Useful to understand how well the model explains the target variable.

   -Since it is dimensionless, it is more useful for comparing different models.

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)# A Random Forest model with 100 trees
is created.
# random_state=42 provides randomness control to ensure repeatability of results.
rf_model.fit(X_train, y_train)# The model is trained using the training data (X_train, y_train).

# Prediction on Test Data
y_pred = rf_model.predict(X_test) # After training, the model makes predictions on the test data (X_test).

mse = mean_squared_error(y_test, y_pred) # Mean Squared Error (MSE) is calculated.
r2 = r2_score(y_test, y_pred) # The R^2 score is calculated. This shows the explanatory power of the model.
print(f"Mean Squared Error: {mse}") #The R^2 score is calculated. This shows the explanatory power of the
model.
```

```
print(f"R^2 Score: {r2}") #print the metrics
```

Result

Mean Squared Error (MSE): 10776163.01

Indicates the average squared error between predicted and actual values.

R^2 Score: 0.8976

Suggests that the model explains approximately 89.76% of the variance in the target variable, indicating strong performance.

To better understand the result, may use scatter plot:

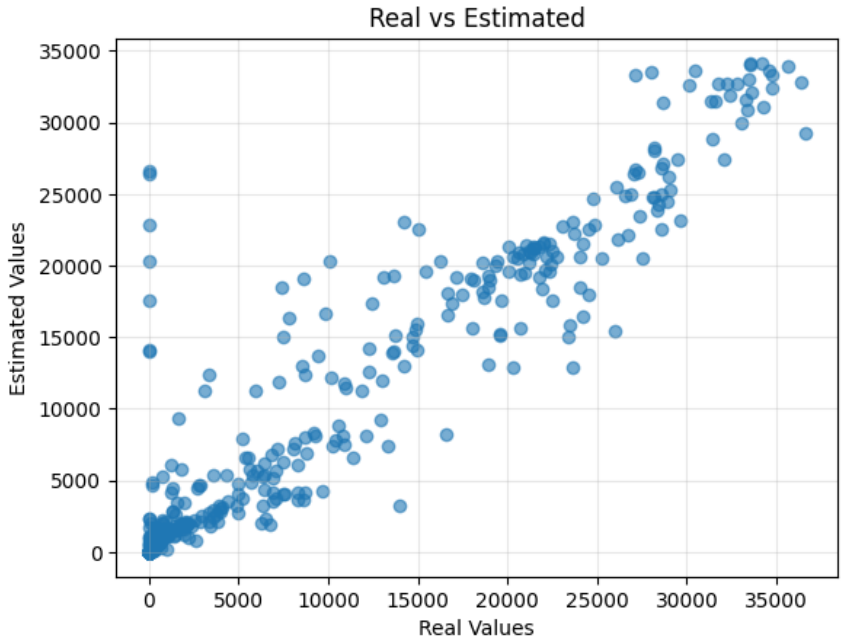

Figure 3: Scatter Plot of Real and Estimated Values

Scatter Plot (Figure 3) shows that the Random Forest model shows good performance, with an R^2 score of about 0.89. This matches the close alignment between actual and predicted values. While most predictions are accurate, there are some differences that suggest areas to improve. For example, better feature selection or trying different models might help handle higher values more effectively.

Improvement of The Random Forest Regressor by Using GridSearchCV Method

This section optimizes the Random Forest model using GridSearchCV, which systematically tests multiple combinations of hyperparameters to find the best-performing configuration.

1. Prepare Parameters of Grid:

```
# Specify hyperparameter search ranges
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
}
```

Defines the possible values for each hyperparameter:

n_estimators: Number of decision trees.
max_depth: Limits tree depth to prevent overfitting.
min_samples_split: Minimum data points required to split a node.
min_samples_leaf: Minimum data points in each leaf node.
max_features: Strategy for feature selection (square root or logarithm of the total number of features).

2. Configuring Grid SearchCV and training with it then printing best parameters and results.

```
rf_model = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, scoring='r2', cv=5,
verbose=2, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best Parameters
print("Best Parameters:", grid_search.best_params_)

# Estimate by Optimum model
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Performance evaluation
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R^2 Score: {r2}")
```

3. After this step, it tests all hyperparameter combinations and identifies the best configuration based on the R^2 score. Results are shown below.

```
Best Parameters: {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Mean Squared Error: 9884502.312705677
R^2 Score: 0.9061631535953113
```

Figure 4: Result after GridSearchCV

According to figure 4, the hyperparameter tuning process using GridSearchCV optimized the Random Forest model by testing various parameter combinations. The best configuration included 200 trees (n_estimators), a maximum tree depth of 20 (max_depth), a square root strategy for feature selection (max_features='sqrt'), and specific criteria for node splitting and leaf size. The optimized model achieved a Mean Squared Error (MSE) of 9834850.31 and an R^2 Score of 0.9061, indicating

that it explains 90.61% of the variance in the data. This represents a significant improvement in predictive performance compared to the default settings, highlighting the effectiveness of hyperparameter tuning in enhancing model accuracy.

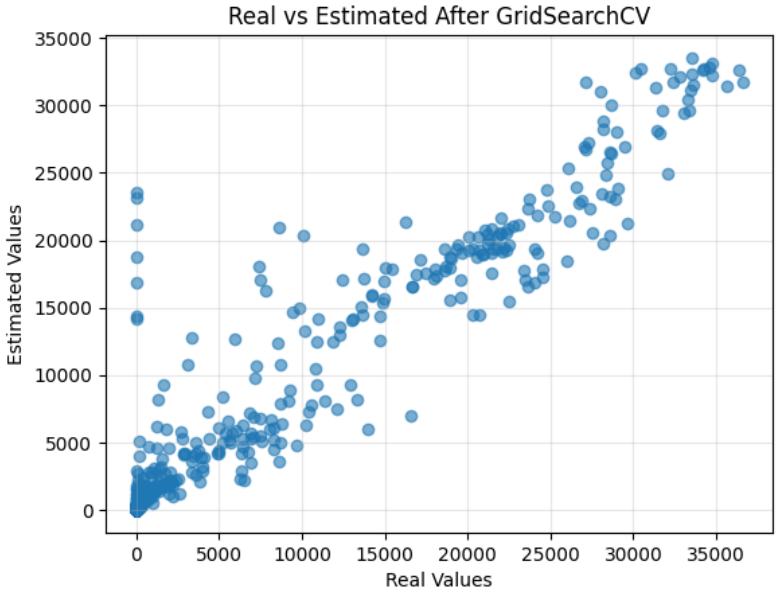

Figure 5: Scatter Plot after Improvement

After the improvement, the scatter plot shows that the optimized Random Forest model makes more accurate and consistent predictions. The higher R^2 score (0.9061) shows that tuning the model's settings worked well to improve its performance.

Conclusion

This study looked at the solar energy production system in Berkeley, California, using machine learning to predict future energy outputs. By analyzing the data, cleaning it, and creating useful features, we found important environmental factors that affect energy generation.

A Random Forest model was chosen because it handles complex relationships well and works effectively even with outliers. The model's first performance, with an R^2 score of 0.8976, showed strong predictive ability. After tuning the model's settings using GridSearchCV, the R^2 score improved to 0.9061, and prediction errors were reduced, as seen in the lower Mean Squared Error (MSE). This shows how important it is to fine-tune machine learning models for better accuracy.

The study highlights how environmental factors, like distance to solar noon and humidity, affect energy production. It also shows the potential of machine learning to improve renewable energy systems. In the future, adding more data and trying different machine learning methods could further improve predictions and system efficiency.

Bibliography

Anon., n.d. *medium.* [Online]
Available at: https://medium.com/@byanalytixlabs/random-forest-regression-how-it-helps-in-predictive-analytics-01c31897c1d4
[Accessed 23 11 2024].

Boswell, D., 2002. Introduction to Support Vector Machines. *Introduction to Support Vector Machines,* p. 1.

Camastra, F. & Vinciarelli, A., 2008. *Machine Learning for Audio, Image and Video Analysis Theory and Applications.* s.l.:Springer.

Constantin, A., 2020. *kaggle.* [Online]
Available at: https://www.kaggle.com/datasets/vipulgote4/solar-power-generation/data
[Accessed 1 11 2024].

Fieguth, P., 2022. *An Introduction to Pattern Recognition and Machine Learning.* s.l.:Springer.