Interview

AUTHOR: ERIC MWANGI

Djricmwas@gmail.com

Instructions to candidates:

Preparation Time: You have been provided with a document from the analytics and insights team. Dedicate 15 minutes exclusively for reading and understanding this document before commencing the exercise.

Objective & Scope: The purpose of this exercise is threefold:

1. Analyze the provided data and construct a model.
2. Systematically document your approach and methodology, ensuring that it's comprehensible for both a fellow candidate and a senior analyst.
3. Critically evaluate the methods employed and the resulting outputs.

Evaluation & Communication: Post-completion, be prepared to discuss with a senior analyst, detailing your chosen approach, the results derived, and any conclusions drawn. All questions within the exercise must be attempted.

First we will import the required libraries for this analysis. Panda for data wrangling Numpy for fatser calculations Seaborn and matplotlib for visualization Scikit and xgboast for liner regression and xgboast models.

In [83]:
```
import pandas as pd
import numpy as np
import seaborn  as sn

from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import xgboost as xgb
```

In [84]:
```
df= pd.read_excel("Data for candidate.xlsx", sheet_name='Raw Data', skiprows=2)
df.columns= ['Hours', 'Datetime', 'Solar_gen', 'Elec_usage']
#Create on meore column for the date without the hows.
df['Date']= pd.to_datetime(df['Datetime'].dt.date)
#df['date'] = pd.to_datetime(df['date'])
df.tail()
```

Out[84]:

	Hours	Datetime	Solar_gen	Elec_usage	Date
8755	19	2020-12-31 19:00:00	0.012	4.395600	2020-12-31
8756	20	2020-12-31 20:00:00	0.003	4.560600	2020-12-31
8757	21	2020-12-31 21:00:00	0.000	2.022000	2020-12-31
8758	22	2020-12-31 22:00:00	0.015	1.668000	2020-12-31
8759	23	2020-12-31 23:00:00	0.000	0.805919	2020-12-31

The data contains details on hourly solar generation and electricity consumption for the year 2020. We will narrow it down to daily data, hence the need to create a new variable called 'Date,' which will be used for aggregation

In [85]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Hours       8760 non-null   int64
 1   Datetime    8760 non-null   datetime64[ns]
 2   Solar_gen   8760 non-null   float64
 3   Elec_usage  8760 non-null   float64
 4   Date        8760 non-null   datetime64[ns]
dtypes: datetime64[ns](2), float64(2), int64(1)
memory usage: 342.3 KB
```

The data contains 8760 rows and 5 colums. Each data type for each varible is shown above.

In [86]:
```
df.describe()
```

Out[86]:

	Hours	Datetime	Solar_gen	Elec_usage	Date
count	8760.000000	8760	8760.000000	8760.000000	8760
mean	11.500000	2020-07-02 07:37:13.972602624	1.116750	7.312704	2020-07-01 20:07:13.972602880
min	0.000000	2020-01-01 00:00:00	0.000000	-12.624000	2020-01-01 00:00:00
25%	5.750000	2020-04-02 05:45:00	0.000000	0.300000	2020-04-02 00:00:00
50%	11.500000	2020-07-02 11:30:00	0.024000	0.621000	2020-07-02 00:00:00
75%	17.250000	2020-10-01 17:15:00	1.272750	1.686000	2020-10-01 00:00:00
max	23.000000	2020-12-31 23:00:00	13.050000	46000.000000	2020-12-31 00:00:00
std	6.922582	NaN	2.026098	491.479806	NaN

The describe fuctions shows descriptive static of the data. The average hourly solar generation is approximately 1.12 kWh, with a maximum of 13.05 kWh and a minimum of 0 kWh, indicating periods of no solar production. Electricity usage varies significantly, with an average of 7.31 kWh, a minimum of -12.62 kWh (likely due to data errors or net energy export), and a maximum of 46,000 kWh, suggesting extreme consumption outliers. The median electricity usage is 0.62 kWh, indicating that most values are relatively low. The distribution of hours is uniform, ranging from 0 to 23, covering a full day. The 46,000 kWh looks suspicious and a suggestion that the data contains outliers. We will use the z-score to identify outliers and replace them with median.

In [87]:
```
from scipy import stats
df['z_score'] = np.abs(stats.zscore(df['Elec_usage']))
outliers = df[df['z_score'] > 3]
print("Outliers:\n", outliers)
```

```
Outliers:
      Hours            Datetime  Solar_gen  Elec_usage        Date    z_score
276     12 2020-01-12 12:00:00      5.214     46000.0 2020-01-12  93.585356
```

In [88]:
```
# Replace the outlier with median.
median_value = df['Elec_usage'].median()
df['Elec_usage'] = np.where((df['z_score'] > 3) , median_value, df['Elec_usage'])
df.head()
```

Out[88]:

	Hours	Datetime	Solar_gen	Elec_usage	Date	z_score
0	0	2020-01-01 00:00:00	0.0	1.509849	2020-01-01	0.011808
1	1	2020-01-01 01:00:00	0.0	1.411859	2020-01-01	0.012007
2	2	2020-01-01 02:00:00	0.0	1.023898	2020-01-01	0.012796
3	3	2020-01-01 03:00:00	0.0	0.642000	2020-01-01	0.013573
4	4	2020-01-01 04:00:00	0.0	0.960000	2020-01-01	0.012926

Next, we will conduct further analysis on daily electricity usage, which will serve as our dependent variable. The independent variables will be derived from daily solar generation data, specifically the previous day's solar generation and the solar generation from the past seven days. This approach means that we will use both the previous day's solar generation and the solar generation from the preceding seven days to predict daily electricity usage.

In [89]:
```
#Check the average power usage vs power generation per day.
daily_data= df.groupby('Date').agg({
    'Solar_gen': 'mean',    # Average solar_generation per day
    'Elec_usage': 'mean',  # Average electricity usge per day
}).reset_index()
#Adda alag variablg for the solar gen
daily_data['Solar_gen_lag_1'] = daily_data['Solar_gen'].shift(1)
```

```python
daily_data['Solar_gen_lag_7'] = daily_data['Solar_gen'].shift(7)
daily_data.head(10)
```

Out[89]:

	Date	Solar_gen	Elec_usage	Solar_gen_lag_1	Solar_gen_lag_7
0	2020-01-01	0.103375	1.217771	NaN	NaN
1	2020-01-02	0.094125	2.944890	0.103375	NaN
2	2020-01-03	0.388375	2.795390	0.094125	NaN
3	2020-01-04	0.132875	1.571921	0.388375	NaN
4	2020-01-05	0.234250	1.779347	0.132875	NaN
5	2020-01-06	0.321750	2.012027	0.234250	NaN
6	2020-01-07	0.677000	2.684235	0.321750	NaN
7	2020-01-08	0.054750	1.550418	0.677000	0.103375
8	2020-01-09	0.060625	1.093298	0.054750	0.094125
9	2020-01-10	0.097625	1.978569	0.060625	0.388375

In [90]:

```python
import plotly
import plotly.express as px

# Reshaping the Data for Plotly
df_melted = daily_data.melt(id_vars=["Date"], value_vars=["Solar_gen", "Elec_usage", 'Solar_gen_lag_1','Solar_gen_lag
                     var_name="Type", value_name="Power")
# Create Line Plot
fig = px.line(df_melted, x="Date", y="Power", color="Type",
              markers=True, title="Solar Genaration & Electricity Usage Over Time")
# Customize Layout
fig.update_layout(
    xaxis_title="Week",
    yaxis_title="Score",
    template="plotly_dark",  # Try "plotly", "ggplot2", "seaborn" for different styles
    hovermode="x unified"
)
# Show Plot
```

```
fig.show()
plotly.offline.init_notebook_mode()
```

The plot reveals an interesting relationship between electricity usage and solar generation. Specifically, when solar generation is low, electricity usage tends to be high, and vice versa. Additionally, we observe a steady upward trend in solar generation from January to June 2020, followed by a consistent downward trend. Electricity usage mirrors this pattern, but in reverse: a downward trend until June, followed by an upward trend starting in June. The lag variables are also depicted in the graph. To explore this relationship further, we will use a correlation matrix for deeper analysis.

In [91]:
```
daily_data[['Solar_gen', 'Elec_usage', 'Solar_gen_lag_1','Solar_gen_lag_7']].corr()
```

Out[91]:

	Solar_gen	Elec_usage	Solar_gen_lag_1	Solar_gen_lag_7
Solar_gen	1.000000	-0.319258	0.705050	0.529160
Elec_usage	-0.319258	1.000000	-0.295627	-0.230072
Solar_gen_lag_1	0.705050	-0.295627	1.000000	0.551516
Solar_gen_lag_7	0.529160	-0.230072	0.551516	1.000000

The correlation matrix reveals a moderate negative correlation between solar generation and electricity usage. This indicates that as solar generation increases, electricity usage tends to decrease.

We will build both a linear regression model and an XGBoost model, then compare their performance using the Root Mean Square Error (RMSE). RMSE measures the average magnitude of error between predicted and actual values, with lower RMSE indicating better model performance.

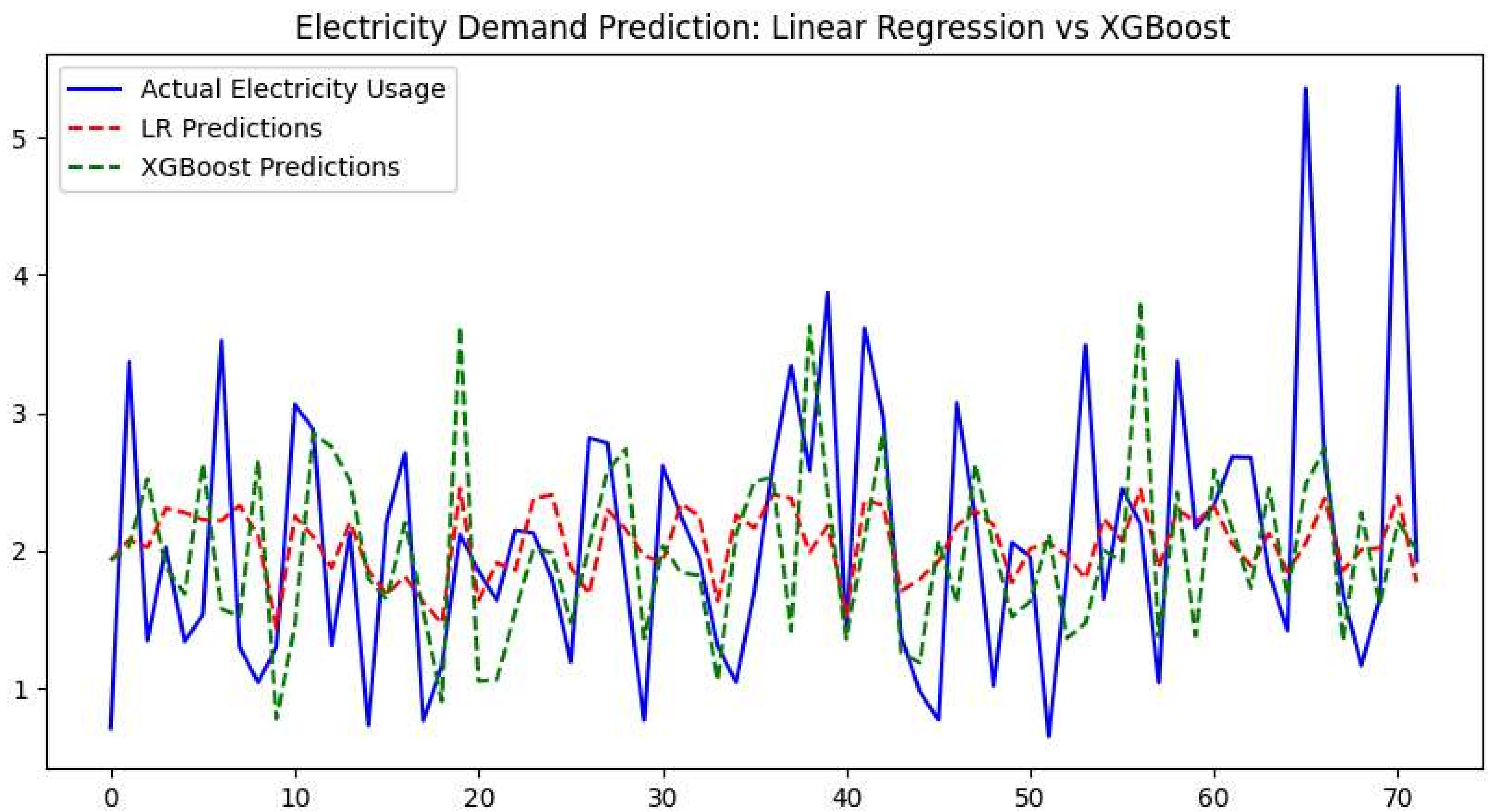
In [92]:
```
# Drop first row with NaN due to lagging
daily_data = daily_data.dropna()
# Features and target
X = daily_data[[ 'Solar_gen_lag_1',     'Solar_gen_lag_7']]
y = daily_data['Elec_usage']
# Split into training (80%) and testing (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# === 1. Train Linear Regression Model ===
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))

# === 2. Train XGBoost Model ===
xgb_model = xgb.XGBRegressor(objective="reg:squarederror", n_estimators=100, random_state=42)
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
rmse_xgb = np.sqrt(mean_squared_error(y_test, y_pred_xgb))
# Print results
print(f"Linear Regression RMSE: {rmse_lr:.2f}")
print(f"XGBoost RMSE: {rmse_xgb:.2f}")
```

```
Linear Regression RMSE: 0.91
XGBoost RMSE: 1.02
```

The Linear Regression model has an RMSE of 0.91, while the XGBoost model has an RMSE of 1.02. Since RMSE measures the average error between predicted and actual values, a lower RMSE indicates better model performance. Therefore, the Linear Regression model performs slightly better than the XGBoost model in terms of prediction accuracy, as it has a lower RMSE.

In [93]:
```
# === Plot Predictions ===
plt.figure(figsize=(10, 5))
plt.plot(y_test.values, label="Actual Electricity Usage", color="blue")
plt.plot(y_pred_lr, label="LR Predictions", linestyle="dashed", color="red")
plt.plot(y_pred_xgb, label="XGBoost Predictions", linestyle="dashed", color="green")
plt.legend()
```

```
plt.title("Electricity Demand Prediction: Linear Regression vs XGBoost")
plt.show()
```



The graph shows that both model perfoemed well and appears to follow the genral trend of the actual electicity usage but with some deviation. This show there some room for improvement and we can try and do fine-tuning to improve the prediction

THANK YOU