Weather-Based Prediction of Wind Turbine Energy Output: A Next-Generation Approach to Renewable Energy Management

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90651
PROJECT NAME	Weather-Based Prediction of Wind Turbine Energy Output: A Next-Generation Approach to Renewable Energy Management
MAXIMUM MARKS	3 MARKS

6.3 - Data Pre-Processing:

In this milestone, we will be preprocessing the dataset that is collected. Preprocessing includes:

1. Processing the dataset.
2. Handling the null values.
3. Handling the categorical values if any.
4. Normalize the data if required.
5. Identify the dependent and independent variables.
6. Split the dataset into train and test sets.

Import required libraries:

```
# Importing Necessary Libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
```

Analyse the datasets:

Step 1: The datasets are imported as data frames using the panda's library. Rename the columns with suitable column names for better understanding.

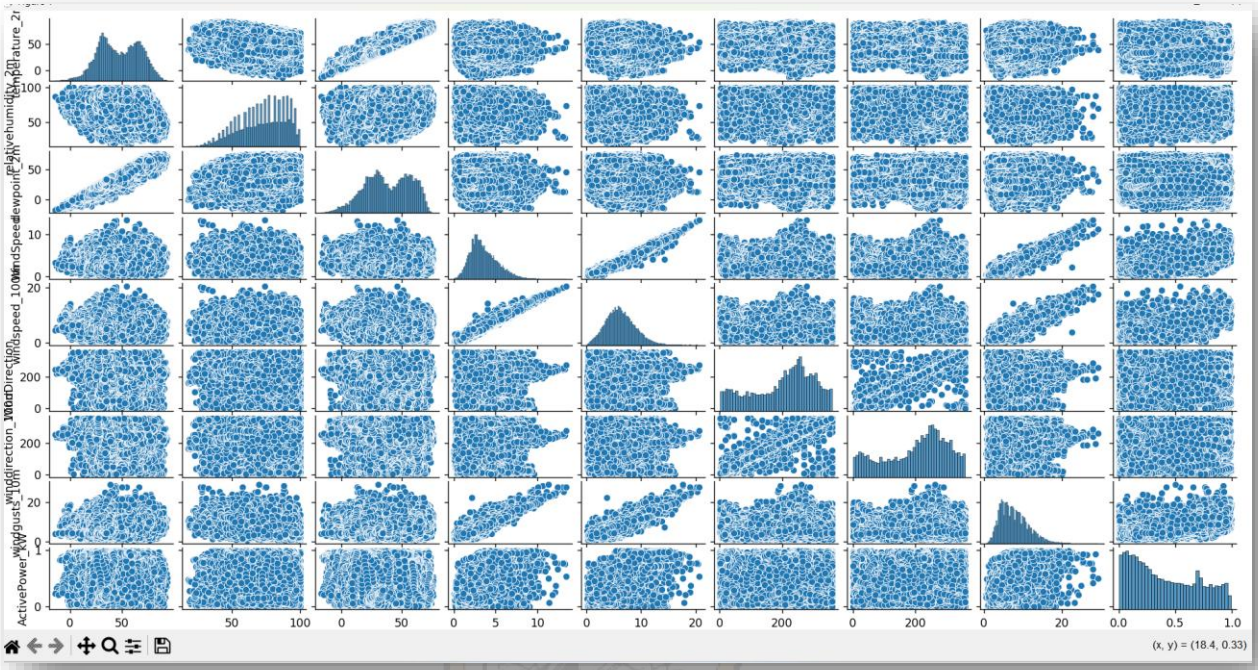
*Dataset contains the wind speed and wind direction along with the power generated**.**

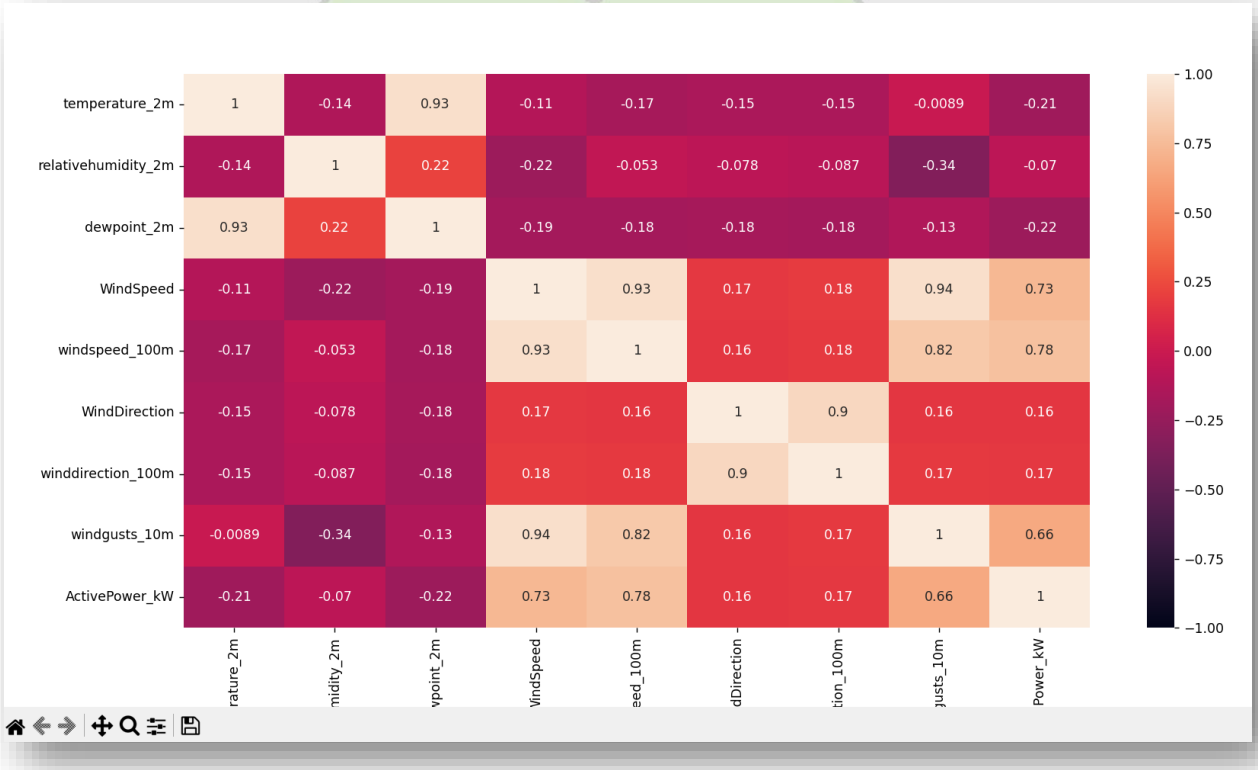
```
path = "data/Location1.csv"

df = pd.read_csv(path, encoding="latin1")

df.columns = df.columns.str.strip()

df.rename(columns={

    'Date/Time': 'Time',
```

```
        'LV ActivePower (kW)': 'ActivePower_kW',
        'Wind Speed (m/s)': 'WindSpeed',
        'Wind Direction (°)': 'WindDirection'
    }, inplace=True)
```

Step 2: Check the correlation between the columns for dimensionality reduction (knowing which columns are necessary and which are not)

```
sns.pairplot(df)
plt.figure(figsize=(10, 8))
corr = df.corr(numeric_only=True)
ax = sns.heatmap(corr, vmin=-1, vmax=1, annot=True)
bottom, top = ax.get_ylim()
plt.show();
ax.set_ylim(bottom + 0.5, top - 0.5)
print(corr)
df["Time"] = pd.to_datetime(df["Time"], errors="coerce")
```


temperature_2m
relativehumidity_2m
dewpoint_2m
WindSpeed
windspeed_100m
WindDirection
winddirection_100m
windgusts_10m
ActivePower_kW
(x, y) = (18.4, 0.33)


temperature_2m
relativehumidity_2m
dewpoint_2m
WindSpeed
windspeed_100m
WindDirection
winddirection_100m
windgusts_10m
ActivePower_kW
1 -0.14 0.93 -0.11 -0.17 -0.15 -0.15 -0.0089 -0.21
-0.14 1 0.22 -0.22 -0.053 -0.078 -0.087 -0.34 -0.07
0.93 0.22 1 -0.19 -0.18 -0.18 -0.18 -0.13 -0.22
-0.11 -0.22 -0.19 1 0.93 0.17 0.18 0.94 0.73
-0.17 -0.053 -0.18 0.93 1 0.16 0.18 0.82 0.78
-0.15 -0.078 -0.18 0.17 0.16 1 0.9 0.16 0.16
-0.15 -0.087 -0.18 0.18 0.18 0.9 1 0.17 0.17
-0.0089 -0.34 -0.13 0.94 0.82 0.16 0.17 1 0.66
-0.21 -0.07 -0.22 0.73 0.78 0.16 0.17 0.66 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Splitting data into independent and dependent variables

In this activity, the dependent and independent variables are to be identified. The independent columns are considered as x and the dependent column as y.

After identifying the dependent and independent variables, the dataset now has to be split into two sets, one set is used for training the model and the second set is used for testing how good the model is built. The split ratio we consider is 80% for training and 20% for testing.

```
df["Hour"] = df["Time"].dt.hour
df["Month"] = df["Time"].dt.month
df["DayOfYear"] = df["Time"].dt.dayofyear
required_cols = ["ActivePower_kW", "WindSpeed"]
if "Theoretical_Power_Curve (KWh)" in df.columns:
    required_cols.append("Theoretical_Power_Curve (KWh)")
    print("✓ Using Theoretical Power Curve feature")
else:
    print("⚠ Theoretical Power Curve column NOT found")
df = df.dropna(subset=required_cols)
features = ["WindSpeed", "Hour", "Month", "DayOfYear"]
if "Theoretical_Power_Curve (KWh)" in df.columns:
    features.insert(1, "Theoretical_Power_Curve (KWh)")
X = df[features]
y = df["ActivePower_kW"]
train_X, val_X, train_y, val_y = train_test_split(
    X, y, test_size=0.25, shuffle=False
)
```