Edge AI Based Smart Irrigation System

Team Members : Maharudra, Nikhil, Surya

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
url=f"https://raw.githubusercontent.com/Rudra-IISC/Edge-AI-Based-Smart-Irrigation/main/Edge_AI/Output_Data__ET0.csv"
df=pd.read_csv(url)
df.tail()
df.shape
df.head()
```

	YEAR	MO	DY	T2M_MIN	T2M_MAX	RH2M	WS2M	ALLSKY_SFC_SW_DWN	ET0
0	2010	1	1	17.52	28.38	72.15	2.89	17.820	4.778
1	2010	1	2	15.41	28.93	66.76	2.69	19.080	5.200
2	2010	1	3	15.10	27.57	70.37	1.95	19.332	4.814
3	2010	1	4	14.46	29.65	64.02	1.83	20.304	5.326
4	2010	1	5	13.64	30.33	63.20	1.95	19.908	5.338

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
def remove_outliers_iqr(df, features):
    cleaned_df = df.copy()
    for feature in features:
        Q1 = cleaned_df[feature].quantile(0.25)
        Q3 = cleaned_df[feature].quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR
        cleaned_df = cleaned_df[(cleaned_df[feature] >= lower) & (cleaned_df[feature] <= upper)]
        print(f"Removed {len(df) - len(cleaned_df)} outliers from {feature}")
    return cleaned_df
features = ['T2M_MIN', 'T2M_MAX', 'RH2M', 'WS2M', 'ALLSKY_SFC_SW_DWN', 'ET0']
df_cleaned= remove_outliers_iqr(df, features)
df_cleaned.shape
```

```
df_cleaned.shape
```

```
Removed 122 outliers from T2M_MIN
Removed 124 outliers from T2M_MAX
Removed 125 outliers from RH2M
Removed 243 outliers from WS2M
Removed 298 outliers from ALLSKY_SFC_SW_DWN
Removed 311 outliers from ET0
(5258, 9)
```

```
!pip install -U ydata_profiling

import ydata_profiling
import pandas as pd # Import pandas for reading the CSV

# Load the data into a DataFrame named 'output_data_with_ET0'
output_data_with_ET0 = pd.read_csv(f"https://raw.githubusercontent.com/Rudra-IISC/Edge-AI-Based-Smart-Irrigation/main/Edge_AI/Output_Data__ET0.csv")

from ydata_profiling.utils.cache import cache_file

report = df_cleaned.profile_report(sort=None, html={"style": {"full_width": True}}, progress_bar=False)
report

profile_report = df_cleaned.profile_report(html={"style": {"full_width": True}})
profile_report.to_file("example.html")

profile_report = df_cleaned.profile_report(
    explorative=True, html={"style": {"full_width": True}})
profile_report
```

```
Collecting ydata_profiling
  Downloading ydata_profiling-4.16.1-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: scipy<1.16,>=1.4.1 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (1.15.2)
Requirement already satisfied: pandas!=1.4.0,<3.0,>1.1 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (2.2.2)
Requirement already satisfied: matplotlib<=3.10,>=3.5 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (3.10.0)
Requirement already satisfied: pydantic>=2 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (2.11.4)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (6.0.2)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (3.1.6)
Collecting visions<0.8.2,>=0.7.5 (from visions[type_image_path]<0.8.2,>=0.7.5->ydata_profiling)
  Downloading visions-0.8.1-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: numpy<2.2,>=1.16.0 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (2.0.2)
Collecting htmlmin==0.1.12 (from ydata_profiling)
  Downloading htmlmin-0.1.12.tar.gz (19 kB)
  Preparing metadata (setup.py) ... done
Collecting phik<0.13,>=0.11.1 (from ydata_profiling)
  Downloading phik-0.12.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.6 kB)
Requirement already satisfied: requests<3,>=2.24.0 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (2.32.3)
Requirement already satisfied: tqdm<5,>=4.48.2 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (4.67.1)
Requirement already satisfied: seaborn<0.14,>=0.10.1 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (0.13.2)
Collecting multimethod<2,>=1.4 (from ydata_profiling)
  Downloading multimethod-1.12-py3-none-any.whl.metadata (9.6 kB)
Requirement already satisfied: statsmodels<1,>=0.13.2 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (0.14.4)
Requirement already satisfied: typeguard<5,>=3 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (4.4.2)
Collecting imagehash==4.3.1 (from ydata_profiling)
  Downloading ImageHash-4.3.1-py2.py3-none-any.whl.metadata (8.0 kB)
Requirement already satisfied: wordcloud>=1.9.3 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (1.9.4)
Collecting dacite>=1.8 (from ydata_profiling)
  Downloading dacite-1.9.2-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: numba<=0.61,>=0.56.0 in /usr/local/lib/python3.11/dist-packages (from ydata_profiling) (0.60.0)
Collecting PyWavelets (from imagehash==4.3.1->ydata_profiling)
  Downloading pywavelets-1.8.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.0 kB)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from imagehash==4.3.1->ydata_profiling) (11.2.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2<3.2,>=2.11.1->ydata_profiling) (3.0.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<=3.10,>=3.5->ydata_profiling) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib<=3.10,>=3.5->ydata_profiling) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib<=3.10,>=3.5->ydata_profiling) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<=3.10,>=3.5->ydata_profiling) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib<=3.10,>=3.5->ydata_profiling) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<=3.10,>=3.5->ydata_profiling) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib<=3.10,>=3.5->ydata_profiling) (2.9.0.post0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba<=0.61,>=0.56.0->ydata_profiling) (0.43.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas!=1.4.0,<3.0,>1.1->ydata_profiling) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas!=1.4.0,<3.0,>1.1->ydata_profiling) (2025.2)
Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.11/dist-packages (from phik<0.13,>=0.11.1->ydata_profiling) (1.4.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic>=2->ydata_profiling) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic>=2->ydata_profiling) (2.33.2)
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydantic>=2->ydata_profiling) (4.13.2)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic>=2->ydata_profiling) (0.4.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (3.10)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (2025.4.26)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.11/dist-packages (from statsmodels<1,>=0.13.2->ydata_profiling) (1.0.1)
Requirement already satisfied: attrs>=19.3.0 in /usr/local/lib/python3.11/dist-packages (from visions<0.8.2,>=0.7.5->visions[type_image_path]<0.8.2,>=0.7.5
Requirement already satisfied: networkx>=2.4 in /usr/local/lib/python3.11/dist-packages (from visions<0.8.2,>=0.7.5->visions[type_image_path]<0.8.2,>=0.7.5
Collecting puremagic (from visions<0.8.2,>=0.7.5->visions[type_image_path]<0.8.2,>=0.7.5->ydata_profiling)
  Downloading puremagic-1.29-py3-none-any.whl.metadata (5.8 kB)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib<=3.10,>=3.5->ydata_profiling) (1.
Downloading ydata_profiling-4.16.1-py2.py3-none-any.whl (400 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 400.1/400.1 kB 7.3 MB/s eta 0:00:00
Downloading ImageHash-4.3.1-py2.py3-none-any.whl (296 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 296.5/296.5 kB 22.0 MB/s eta 0:00:00
Downloading dacite-1.9.2-py3-none-any.whl (16 kB)
Downloading multimethod-1.12-py3-none-any.whl (10 kB)
Downloading phik-0.12.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (687 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 687.8/687.8 kB 27.0 MB/s eta 0:00:00
Downloading visions-0.8.1-py3-none-any.whl (105 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 105.4/105.4 kB 10.3 MB/s eta 0:00:00
Downloading puremagic-1.29-py3-none-any.whl (43 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.3/43.3 kB 3.8 MB/s eta 0:00:00
Downloading pywavelets-1.8.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 57.4 MB/s eta 0:00:00
Building wheels for collected packages: htmlmin
  Building wheel for htmlmin (setup.py) ... done
  Created wheel for htmlmin: filename=htmlmin-0.1.12-py3-none-any.whl size=27081 sha256=f448a8c64b4fef3727c8f9c9c7dad9a8a97ded786ce2253727962990d15580f0
  Stored in directory: /root/.cache/pip/wheels/8d/55/1a/19cd535375ed1ede0c996405ebffe34b196d78e2d9545723a2
Successfully built htmlmin
Installing collected packages: puremagic, htmlmin, PyWavelets, multimethod, dacite, imagehash, visions, phik, ydata_profiling
Successfully installed PyWavelets-1.8.0 dacite-1.9.2 htmlmin-0.1.12 imagehash-4.3.1 multimethod-1.12 phik-0.12.4 puremagic-1.29 visions-0.8.1 ydata_profili
```

[Upgrade to ydata-sdk]

Improve your data and profiling with ydata-sdk, featuring data quality scoring, redundancy detection, outlier identification, text validation, and synthetic data generation.

Summarize dataset: 100% 99/99 [00:11<00:00, 8.08it/s, Completed]

```
  0%|          | 0/9 [00:00<?, ?it/s]
100%|██████████| 9/9 [00:00<00:00, 53.86it/s]
```

Generate report structure: 100% 1/1 [00:02<00:00, 2.97s/it]

Render HTML: 100% 1/1 [00:02<00:00, 2.37s/it]

Export report to file: 100% 1/1 [00:00<00:00, 25.69it/s]

Summarize dataset: 100% 99/99 [00:11<00:00, 13.28it/s, Completed]

```
  0%|          | 0/9 [00:00<?, ?it/s]
100%|██████████| 9/9 [00:00<00:00, 68.75it/s]
```

Generate report structure: 100% 1/1 [00:03<00:00, 3.21s/it]

Render HTML: 100% 1/1 [00:03<00:00, 3.20s/it]

YData Profiling Report

Overview Variables Interactions Correlations Missing values Sample

Overview

Brought to you by YData

Overview Alerts 5 Reproduction

Dataset statistics

Number of variables	9
Number of observations	5258
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	410.8 KiB
Average record size in memory	80.0 B

Variable types

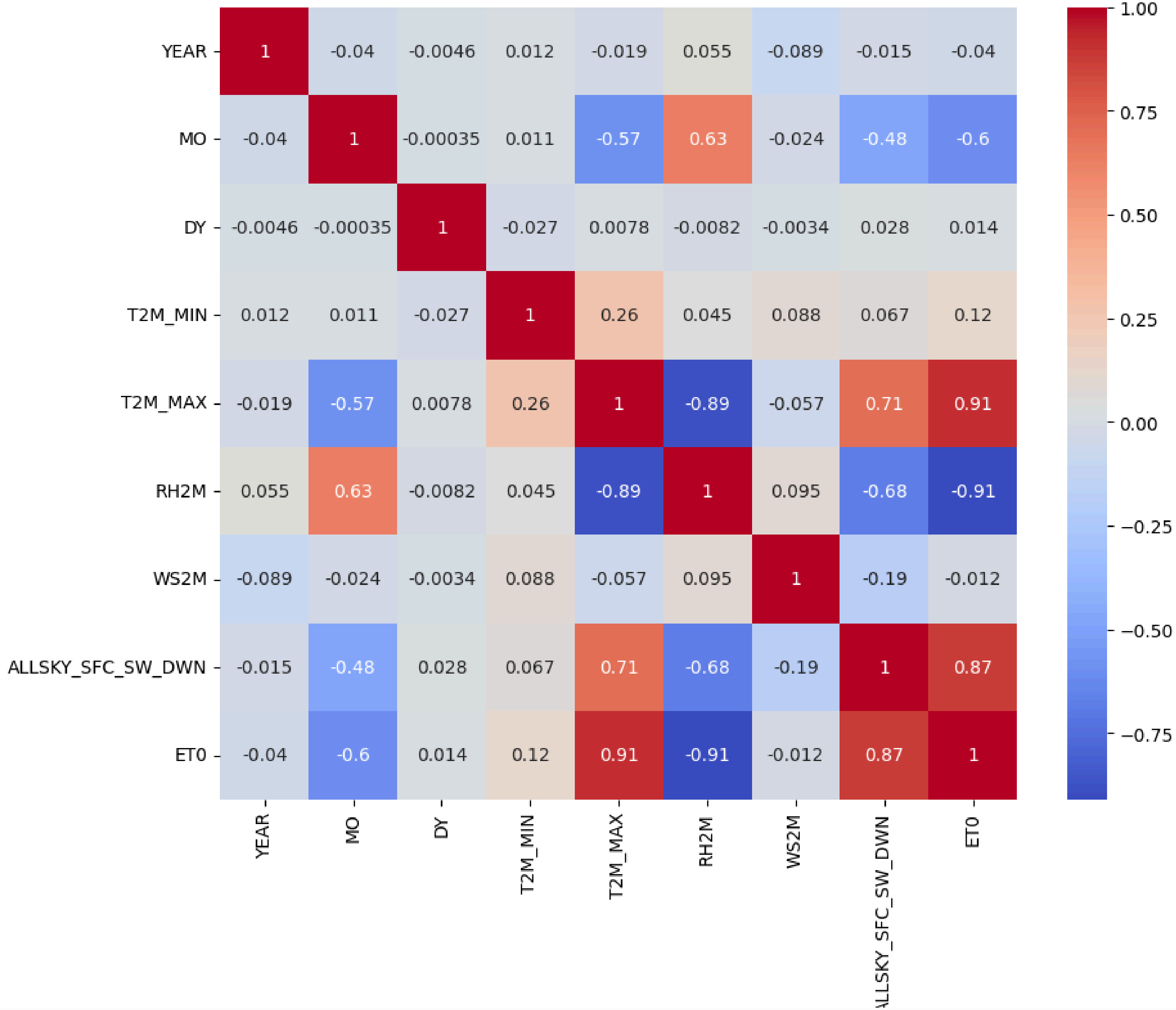
Numeric	9

Variables

Select Columns

```
df_cleaned.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(df_cleaned.corr(), annot=True, cmap='coolwarm')
```

<Axes: >


YEAR
MO
DY
T2M_MIN
T2M_MAX
RH2M
WS2M
ALLSKY_SFC_SW_DWN
ET0
1 -0.04 -0.0046 0.012 -0.019 0.055 -0.089 -0.015 -0.04
-0.04 1 -0.00035 0.011 -0.57 0.63 -0.024 -0.48 -0.6
-0.0046 -0.00035 1 -0.027 0.0078 -0.0082 -0.0034 0.028 0.014
0.012 0.011 -0.027 1 0.26 0.045 0.088 0.067 0.12
-0.019 -0.57 0.0078 0.26 1 -0.89 -0.057 0.71 0.91
0.055 0.63 -0.0082 0.045 -0.89 1 0.095 -0.68 -0.91
-0.089 -0.024 -0.0034 0.088 -0.057 0.095 1 -0.19 -0.012
-0.015 -0.48 0.028 0.067 0.71 -0.68 -0.19 1 0.87
-0.04 -0.6 0.014 0.12 0.91 -0.91 -0.012 0.87 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

```
df_cleaned.drop(columns=['YEAR', 'DY', 'MO'], inplace=True)


# from sklearn.preprocessing import MinMaxScaler
# scaler = MinMaxScaler()
# df_scaled = scaler.fit_transform(df_cleaned)
# df_scaled = pd.DataFrame(df_scaled, columns=df_cleaned.columns)
# df_scaled.head()
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_cleaned)
df_scaled = pd.DataFrame(df_scaled, columns=df_cleaned.columns)
df_scaled.head()
```

	T2M_MIN	T2M_MAX	RH2M	WS2M	ALLSKY_SFC_SW_DWN	ET0
0	-0.455899	-0.629694	0.287135	0.099587	-0.480752	-0.486189
1	-1.267821	-0.474509	-0.056696	-0.084756	-0.165699	-0.241519
2	-1.387109	-0.858239	0.173588	-0.766825	-0.102689	-0.465317
3	-1.633379	-0.271357	-0.231482	-0.877431	0.140352	-0.168466
4	-1.948913	-0.079492	-0.283790	-0.766825	0.041335	-0.161508

Next steps: Generate code with df_scaled View recommended plots New interactive sheet

```
from sklearn.model_selection import train_test_split
X = df_scaled.drop(columns=['ET0'])
y = df_scaled['ET0']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_score


model_LR = LinearRegression()
model_LR.fit(X_train, y_train)
y_pred_LR = model_LR.predict(X_test)
mse_LR = mean_squared_error(y_test, y_pred_LR)
r2_LR = r2_score(y_test, y_pred_LR)
```

```
print("Linear Regression MSE:", mse_LR)
print("Linear Regression R2:", r2_LR)
```

```
Linear Regression MSE: 0.019221886354103167
Linear Regression R2: 0.9801178170726929
```

```
best_features = ['T2M_MAX', 'RH2M', 'ALLSKY_SFC_SW_DWN']
```

```
df_new = df_scaled[best_features]
df_new['ET0'] = df_scaled['ET0']
df_new.head()
```

```
<ipython-input-17-629fe5febc3a>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new['ET0'] = df_scaled['ET0']
```

	T2M_MAX	RH2M	ALLSKY_SFC_SW_DWN	ET0
0	-0.629694	0.287135	-0.480752	-0.486189
1	-0.474509	-0.056696	-0.165699	-0.241519
2	-0.858239	0.173588	-0.102689	-0.465317
3	-0.271357	-0.231482	0.140352	-0.168466
4	-0.079492	-0.283790	0.041335	-0.161508

Next steps: Generate code with df_new | View recommended plots | New interactive sheet

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(df_new.drop(columns=['ET0']), df_new['ET0'], test_size=0.2, random_state=42)
```

```
model_LR1 = LinearRegression()
model_LR.fit(X_train1, y_train1)
y_pred_LR = model_LR.predict(X_test1)
mse_LR = mean_squared_error(y_test1, y_pred_LR)
r2_LR = r2_score(y_test1, y_pred_LR)
print("Linear Regression MSE:", mse_LR)
print("Linear Regression R2:", r2_LR)
```

```
Linear Regression MSE: 0.0369463661022275
Linear Regression R2: 0.9618963672174179
```

```
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score

Model1_SVR_df_new = SVR()
Model1_SVR_df_new.fit(X_train1, y_train1)
y_pred_SVR = Model1_SVR_df_new.predict(X_test1)
mse_SVR = mean_squared_error(y_test1, y_pred_SVR)
r2_SVR = r2_score(y_test1, y_pred_SVR)
print("Support Vector Regressor MSE:", mse_SVR)
print("Support Vector Regressor R2:", r2_SVR)
```

```
Support Vector Regressor MSE: 0.031656185785660554
Support Vector Regressor R2: 0.9673522512298908
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# --- Function to estimate MLP model size ---
def estimate_mlp_size(model, dtype_size=8):
    """
    Estimate memory size of a trained MLPRegressor in bytes.
    Parameters:
        model: Trained sklearn.neural_network.MLPRegressor model
        dtype_size: Size of each parameter in bytes (default 8 for float64)
    Returns:
        Total size in bytes
    """
    total_params = 0
    for coef in model.coefs_:
        total_params += coef.size
    for intercept in model.intercepts_:
        total_params += intercept.size

    total_size_bytes = total_params * dtype_size
    total_size_kb = total_size_bytes / 1024
```

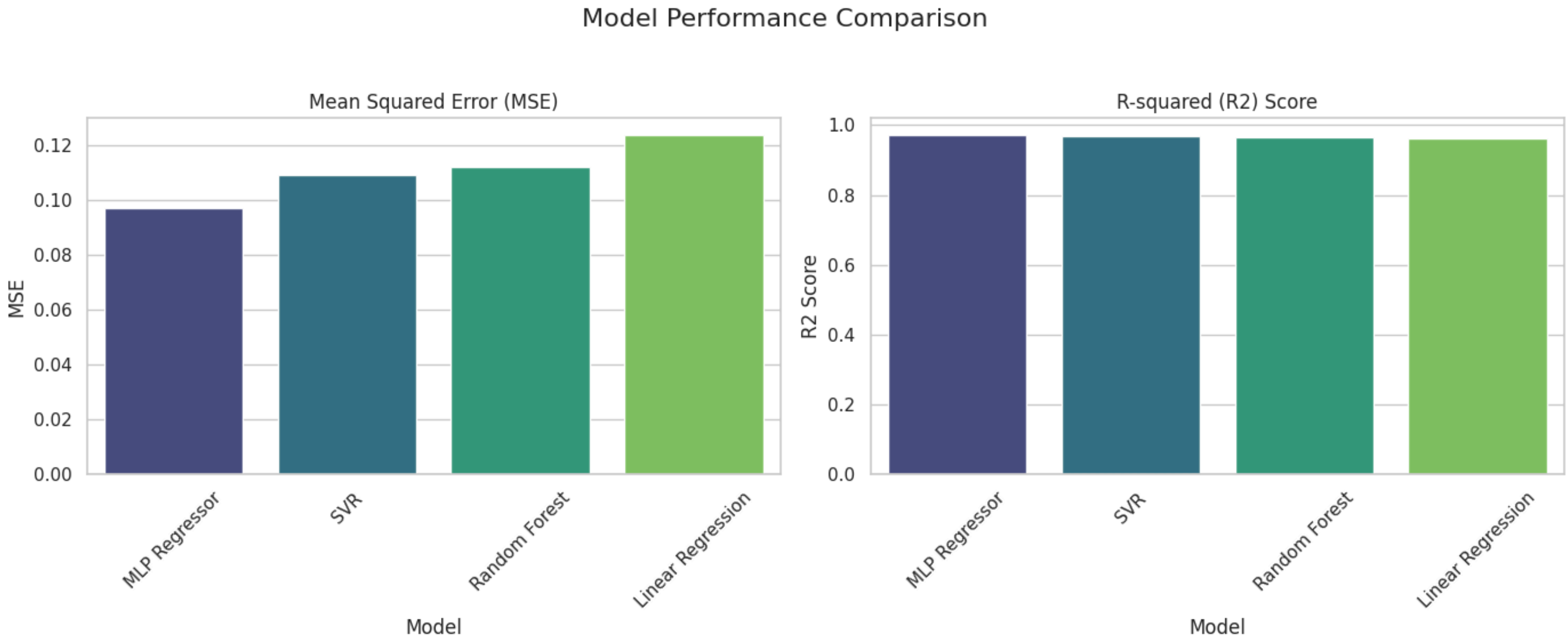
```
        total_size_mb = total_size_kb / 1024

        print(f"MLP Model Parameter Count: {total_params}")
        print(f"Estimated Model Size: {total_size_bytes:.2f} bytes ({total_size_kb:.2f} KB / {total_size_mb:.2f} MB)")
        return total_size_bytes

    # --- Data Preparation ---
    try:
        df = pd.read_csv('https://raw.githubusercontent.com/Rudra-IISC/Edge-AI-Based-Smart-Irrigation/main/Edge_AI/Output_Data__ET0.csv')

        print("--- Raw Data Head ---")
        print(df.head())
        print("\n--- Raw Data Info ---")
        df.info()
        print("\n")

        best_features = ['T2M_MAX', 'RH2M', 'ALLSKY_SFC_SW_DWN']
        target_variable = 'ET0'
        required_columns = best_features + [target_variable]
        if not all(col in df.columns for col in required_columns):
            missing_cols = [col for col in required_columns if col not in df.columns]
            raise ValueError(f"Missing required columns in CSV: {missing_cols}")

        X = df[best_features].copy()
        y = df[target_variable].copy()

        print("--- Scaling Features ---")
        scaler = StandardScaler()
        X_scaled = scaler.fit_transform(X)
        X = pd.DataFrame(X_scaled, columns=best_features, index=X.index)
        print("Features scaled using StandardScaler.\n")

        X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.2, random_state=42)
        print(f"Training set size: {X_train1.shape[0]} samples")
        print(f"Test set size: {X_test1.shape[0]} samples\n")

        results = {}

        # 1. Linear Regression
        print("--- Training Linear Regression ---")
        model_LR = LinearRegression()
        model_LR.fit(X_train1, y_train1)
        y_pred_LR = model_LR.predict(X_test1)
        mse_LR = mean_squared_error(y_test1, y_pred_LR)
        r2_LR = r2_score(y_test1, y_pred_LR)
        results['Linear Regression'] = {'MSE': mse_LR, 'R2': r2_LR}
```

```
    print(f"Linear Regression MSE: {mse_LR:.4f}")
    print(f"Linear Regression R2: {r2_LR:.4f}\n")

    # 2. Support Vector Regressor (SVR)
    print("--- Training Support Vector Regressor (SVR) ---")
    model_SVR = SVR()
    model_SVR.fit(X_train1, y_train1)
    y_pred_SVR = model_SVR.predict(X_test1)
    mse_SVR = mean_squared_error(y_test1, y_pred_SVR)
    r2_SVR = r2_score(y_test1, y_pred_SVR)
    results['SVR'] = {'MSE': mse_SVR, 'R2': r2_SVR}
    print(f"Support Vector Regressor MSE: {mse_SVR:.4f}")
    print(f"Support Vector Regressor R2: {r2_SVR:.4f}\n")

    # 3. Random Forest Regressor
    print("--- Training Random Forest Regressor ---")
    model_RF = RandomForestRegressor(n_estimators=100, random_state=42)
    model_RF.fit(X_train1, y_train1)
    y_pred_RF = model_RF.predict(X_test1)
    mse_RF = mean_squared_error(y_test1, y_pred_RF)
    r2_RF = r2_score(y_test1, y_pred_RF)
    results['Random Forest'] = {'MSE': mse_RF, 'R2': r2_RF}
    print(f"Random Forest Regressor MSE: {mse_RF:.4f}")
    print(f"Random Forest Regressor R2: {r2_RF:.4f}\n")

    # 4. MLP Regressor
    print("--- Training MLP Regressor ---")
    model_MLP = MLPRegressor(hidden_layer_sizes=(64, 32), max_iter=1000, random_state=42, early_stopping=True, n_iter_no_change=10)
    model_MLP.fit(X_train1, y_train1)
    y_pred_MLP = model_MLP.predict(X_test1)
    mse_MLP = mean_squared_error(y_test1, y_pred_MLP)
    r2_MLP = r2_score(y_test1, y_pred_MLP)
    results['MLP Regressor'] = {'MSE': mse_MLP, 'R2': r2_MLP}
    print(f"MLP Regressor MSE: {mse_MLP:.4f}")
    print(f"MLP Regressor R2: {r2_MLP:.4f}")

    print("--- Estimating MLP Regressor Model Size ---")
    original_size_bytes = estimate_mlp_size(model_MLP)

    # Simulated compression techniques:
    # 1. Quantization to 8-bit (1 byte per parameter)
    quantized_size_bytes = original_size_bytes / 8

    # 2. Example pruning - assume 30% of weights are pruned
    pruned_size_bytes = original_size_bytes * 0.7
```

```
        print(f"\nSimulated Quantized Model Size (8-bit): {quantized_size_bytes:.2f} bytes ({quantized_size_bytes/1024:.2f} KB)")
        print(f"Simulated Pruned Model Size (30% smaller): {pruned_size_bytes:.2f} bytes ({pruned_size_bytes/1024:.2f} KB)")

        # --- Model Comparison ---
        print("--- Model Comparison Results ---")
        results_df = pd.DataFrame(results).T
        print(results_df)
        print("\n")

        # --- Visualization ---
        print("--- Generating Comparison Plots ---")
        sns.set(style="whitegrid")
        fig, axes = plt.subplots(1, 2, figsize=(14, 6))
        fig.suptitle('Model Performance Comparison', fontsize=16)

        mse_sorted = results_df.sort_values('MSE')
        sns.barplot(x=mse_sorted.index, y='MSE', data=mse_sorted, ax=axes[0], palette='viridis')
        axes[0].set_title('Mean Squared Error (MSE)')
        axes[0].set_ylabel('MSE')
        axes[0].set_xlabel('Model')
        axes[0].tick_params(axis='x', rotation=45)

        r2_sorted = results_df.sort_values('R2', ascending=False)
        sns.barplot(x=r2_sorted.index, y='R2', data=r2_sorted, ax=axes[1], palette='viridis')
        axes[1].set_title('R-squared (R2) Score')
        axes[1].set_ylabel('R2 Score')
        axes[1].set_xlabel('Model')
        axes[1].tick_params(axis='x', rotation=45)
        axes[1].axhline(0, color='grey', lw=1, linestyle='--')

        plt.tight_layout(rect=[0, 0.03, 1, 0.95])
        plt.show()

    except FileNotFoundError:
        print("Error: The file 'output_data_with_ET0.csv' was not found.")
    except ValueError as ve:
        print(f"Data Error: {ve}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

    print("--- Finished ---")
```

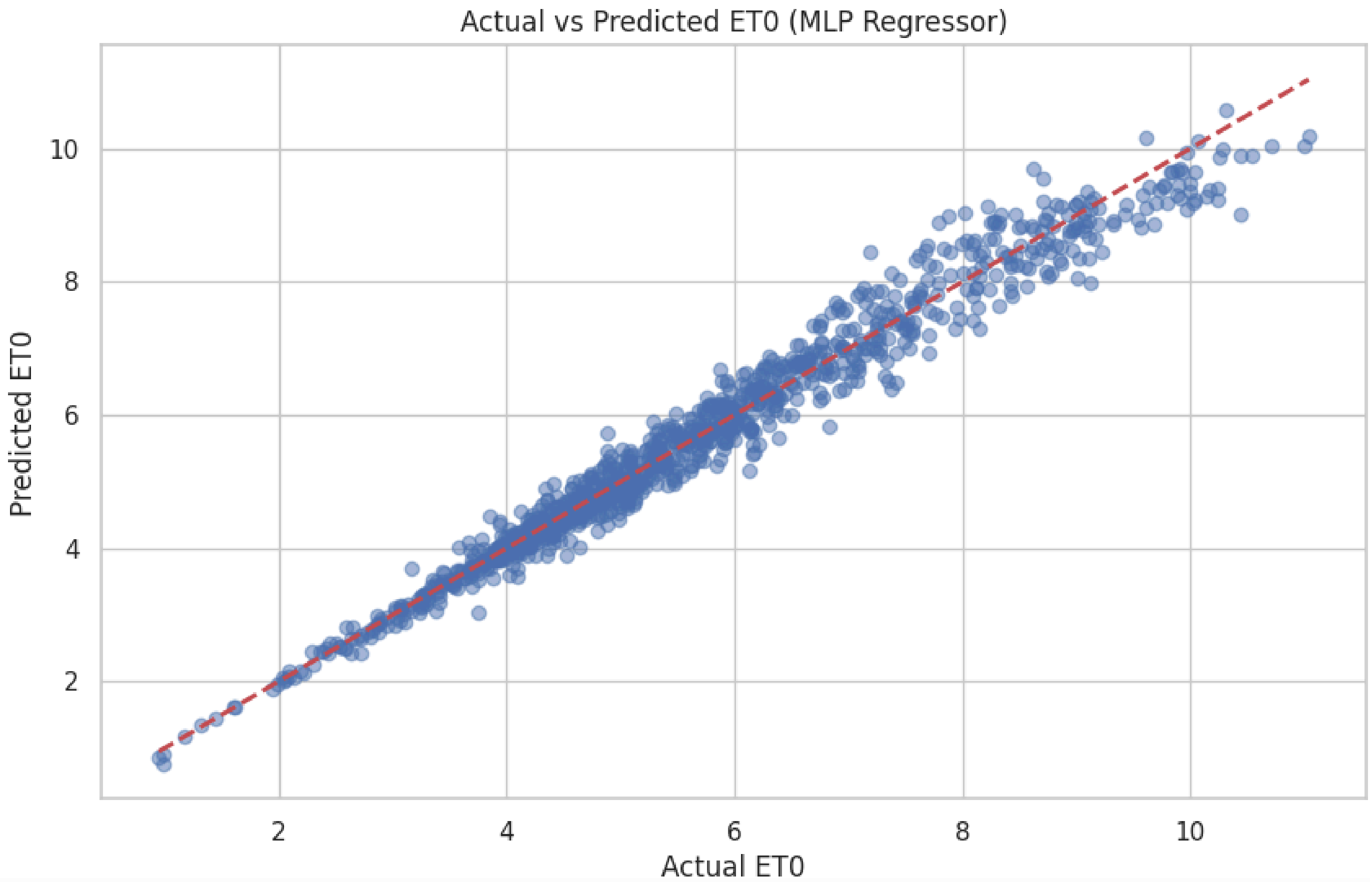
```
--- Raw Data Head ---
   YEAR  MO  DY  T2M_MIN  T2M_MAX   RH2M  WS2M  ALLSKY_SFC_SW_DWN    ET0
0  2010   1   1    17.52    28.38  72.15  2.89             17.820  4.778
1  2010   1   2    15.41    28.93  66.76  2.69             19.080  5.200
2  2010   1   3    15.10    27.57  70.37  1.95             19.332  4.814
3  2010   1   4    14.46    29.65  64.02  1.83             20.304  5.326
4  2010   1   5    13.64    30.33  63.20  1.95             19.908  5.338

--- Raw Data Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5569 entries, 0 to 5568
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   YEAR               5569 non-null   int64  
 1   MO                 5569 non-null   int64  
 2   DY                 5569 non-null   int64  
 3   T2M_MIN            5569 non-null   float64
 4   T2M_MAX            5569 non-null   float64
 5   RH2M               5569 non-null   float64
 6   WS2M               5569 non-null   float64
 7   ALLSKY_SFC_SW_DWN  5569 non-null   float64
 8   ET0                5569 non-null   float64
dtypes: float64(6), int64(3)
memory usage: 391.7 KB


--- Scaling Features ---
Features scaled using StandardScaler.

Training set size: 4455 samples
Test set size: 1114 samples

--- Training Linear Regression ---
Linear Regression MSE: 0.1237
Linear Regression R2: 0.9632

--- Training Support Vector Regressor (SVR) ---
Support Vector Regressor MSE: 0.1093
Support Vector Regressor R2: 0.9675

--- Training Random Forest Regressor ---
Random Forest Regressor MSE: 0.1120
Random Forest Regressor R2: 0.9667

--- Training MLP Regressor ---
MLP Regressor MSE: 0.0971
MLP Regressor R2: 0.9711
--- Estimating MLP Regressor Model Size ---
MLP Model Parameter Count: 2369
```

```
Estimated Model Size: 18952.00 bytes (18.51 KB / 0.02 MB)

Simulated Quantized Model Size (8-bit): 2369.00 bytes (2.31 KB)
Simulated Pruned Model Size (30% smaller): 13266.40 bytes (12.96 KB)
--- Model Comparison Results ---
                        MSE        R2
Linear Regression  0.123728  0.963197
SVR                0.109338  0.967477
Random Forest      0.111998  0.966686
MLP Regressor      0.097086  0.971122


--- Generating Comparison Plots ---
<ipython-input-22-ed4c99ffa341>:139: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the sam

  sns.barplot(x=mse_sorted.index, y='MSE', data=mse_sorted, ax=axes[0], palette='viridis')
<ipython-input-22-ed4c99ffa341>:146: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the sam

  sns.barplot(x=r2_sorted.index, y='R2', data=r2_sorted, ax=axes[1], palette='viridis')
```



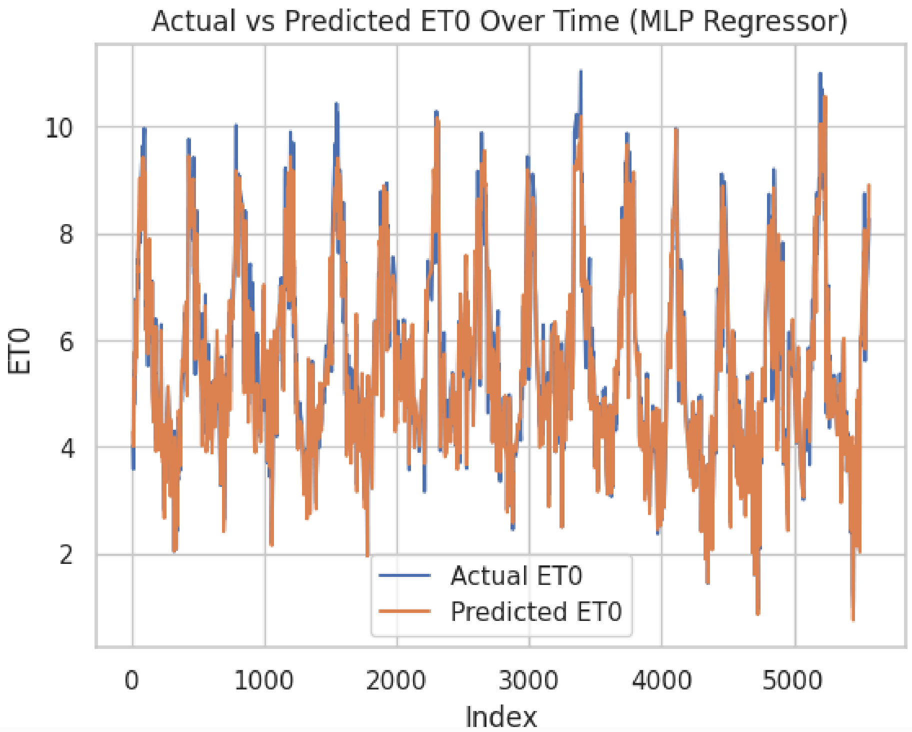
```
--- Finished ---
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test1, y_pred_MLP, alpha=0.5)
plt.plot([y_test1.min(), y_test1.max()], [y_test1.min(), y_test1.max()], 'r--', lw=2)
plt.xlabel('Actual ET0')
plt.ylabel('Predicted ET0')
plt.title('Actual vs Predicted ET0 (MLP Regressor)')
plt.show()
```




```
sns.lineplot(x=y_test1.index, y=y_test1, label='Actual ET0')
sns.lineplot(x=y_test1.index, y=y_pred_MLP, label='Predicted ET0')
plt.xlabel('Index')
plt.ylabel('ET0')
plt.title('Actual vs Predicted ET0 Over Time (MLP Regressor)')
```

```
plt.legend()
plt.show()
```




```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler
import warnings

# Suppress ConvergenceWarning for cleaner output during extraction focus
from sklearn.exceptions import ConvergenceWarning
warnings.filterwarnings("ignore", category=ConvergenceWarning)

# --- Configuration ---
CSV_FILE = 'https://raw.githubusercontent.com/Rudra-IISC/Edge-AI-Based-Smart-Irrigation/main/Edge_AI/Output_Data__ET0.csv'
BEST_FEATURES = ['T2M_MAX', 'RH2M', 'ALLSKY_SFC_SW_DWN']
TARGET_VARIABLE = 'ET0'
TEST_SIZE = 0.2
RANDOM_STATE = 42
```

```
# MLP Parameters (Keep it relatively simple for Pico)
# You might need to experiment with smaller layers if memory is still an issue
HIDDEN_LAYER_SIZES = (16, 8) # Reduced layers for Pico suitability
MAX_ITER = 1500 # Increased iterations might be needed for smaller networks
EARLY_STOPPING = True
N_ITER_NO_CHANGE = 15

# --- Data Preparation ---
print(f"--- Loading Data from {CSV_FILE} ---")
try:
    df = pd.read_csv(CSV_FILE)
    print("Data loaded successfully.")

    # Check for required columns
    required_columns = BEST_FEATURES + [TARGET_VARIABLE]
    if not all(col in df.columns for col in required_columns):
        missing_cols = [col for col in required_columns if col not in df.columns]
        raise ValueError(f"Missing required columns in CSV: {missing_cols}")

    X = df[BEST_FEATURES].copy()
    y = df[TARGET_VARIABLE].copy()

    # Handle potential NaN values (replace with mean, or choose another strategy)
    if X.isnull().values.any():
        print("Warning: NaN values found in features. Filling with mean.")
        X = X.fillna(X.mean())
    if y.isnull().values.any():
        print("Warning: NaN values found in target. Filling with mean.")
        y = y.fillna(y.mean())


    # --- Feature Scaling ---
    print("--- Scaling Features ---")
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    print("Features scaled using StandardScaler.")
    # Keep X_scaled as numpy array for training

    # --- Train/Test Split ---
    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled, y, test_size=TEST_SIZE, random_state=RANDOM_STATE
    )
    print(f"Training set size: {X_train.shape[0]}, Test set size: {X_test.shape[0]}\n")

    # --- Train MLP Regressor ---
    print(f"--- Training MLP Regressor {HIDDEN_LAYER_SIZES} ---")
```

```
    model_MLP = MLPRegressor(
        hidden_layer_sizes=HIDDEN_LAYER_SIZES,
        activation='relu', # Standard activation, easy to implement
        solver='adam',
        max_iter=MAX_ITER,
        random_state=RANDOM_STATE,
        early_stopping=EARLY_STOPPING,
        n_iter_no_change=N_ITER_NO_CHANGE,
        learning_rate_init=0.001 # Default, adjust if needed
    )
    model_MLP.fit(X_train, y_train)
    print("MLP Model training complete.")

    # --- Evaluate (Optional but recommended) ---
    from sklearn.metrics import mean_squared_error, r2_score
    y_pred_MLP = model_MLP.predict(X_test)
    mse_MLP = mean_squared_error(y_test, y_pred_MLP)
    r2_MLP = r2_score(y_test, y_pred_MLP)
    print(f"\n--- Model Evaluation ---")
    print(f"MLP Regressor MSE: {mse_MLP:.4f}")
    print(f"MLP Regressor R2: {r2_MLP:.4f}\n")


    # --- Extract Parameters ---
    print("--- Extracting Model Parameters for MicroPython ---")

    # 1. Scaler Parameters
    scaler_mean = scaler.mean_.tolist()
    scaler_scale = scaler.scale_.tolist() # Use scale_ (standard deviation)

    # 2. MLP Weights and Biases
    # Coefs are weights, Intercepts are biases
    weights = [coef.tolist() for coef in model_MLP.coefs_]
    biases = [intercept.tolist() for intercept in model_MLP.intercepts_]

    # --- Print Parameters in MicroPython Format ---
    # Clear instructions for the user
    print("\n" + "="*50)
    print("COPY THE FOLLOWING PARAMETERS INTO YOUR MicroPython SCRIPT")
    print("="*50 + "\n")

    print("# --- Scaler Parameters ---")
    print(f"SCALER_MEAN = {scaler_mean}")
    print(f"SCALER_SCALE = {scaler_scale}\n")

    print("# --- MLP Parameters ---")
```

```
        # Print weights layer by layer
        for i, w in enumerate(weights):
            print(f"# Weights: Layer {i} (Input/Previous Layer -> Layer {i+1})")
            print(f"WEIGHTS_{i} = [")
            for row in w:
                print(f"    {row},")
            print("]\n")

        # Print biases layer by layer
        for i, b in enumerate(biases):
            print(f"# Biases: Layer {i+1}")
            print(f"BIASES_{i} = {b}\n")

        print("="*50)
        print("PARAMETER EXTRACTION COMPLETE")
        print("="*50)


    except FileNotFoundError:
        print(f"Error: The file '{CSV_FILE}' was not found.")
    except ValueError as ve:
        print(f"Data Error: {ve}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

```
# --- Scaler Parameters ---
SCALER_MEAN = [30.523708026575687, 67.71459507990662, 18.924402944873407]
SCALER_SCALE = [3.5751311909811117, 15.723214825593704, 48.63370888697871]
```