```
# importing dependencies
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

```
#Load dataset
file_path = '/content/cropwatMeerut.xlsx - Sheet1.csv'
df = pd.read_csv(file_path)
```

```
print(df.head())
```

```
   Year     Month  Min Temp (°C)  Max Temp (°C)  Humidity (%)  Wind (km/day)  \
0  2005   January            7.7           20.1            50              6   
1  2005  February           10.8           23.2            49              7   
2  2005     March           17.1           30.4            45              6   
3  2005     April           20.4           36.3            22              8   
4  2005       May           24.8           39.5            20              9   

   Sun (hours)  Rad (MJ/m²/day)  ETo (mm/day)  
0          4.2              9.9          1.12  
1          6.1             13.8          1.65  
2          7.2             17.7          2.64  
3          8.0             20.8          3.32  
4          7.2             20.7          3.72  
```

```
#checking if missing values present in dataset
print(df.isnull().sum())
```

```
Year               0
Month              0
Min Temp (°C)      0
Max Temp (°C)      0
Humidity (%)       0
Wind (km/day)      0
Sun (hours)        0
Rad (MJ/m²/day)    0
ETo (mm/day)       0
dtype: int64
```

```
#dividing x(input values) parameters and Y(target) Parameters
X= df.drop(columns=['ETo (mm/day)','Year','Month'])
Y=df['ETo (mm/day)']
```

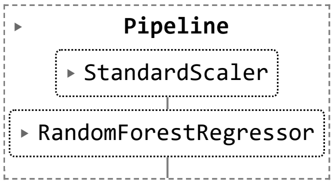
```
#Spliting data
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X,Y, test_size=0.2,random_state=42)
```

```
#Creating a random forest regression model model
rf_model= RandomForestRegressor(n_estimators=100,random_state=42)
```

```
#creating pipeline for StandardScalar and RandomForestRegression
pipeline=Pipeline([
    ('scaler', StandardScaler()),
    ('rf',rf_model)
]

)


#training
pipeline.fit(X_Train,Y_Train)
```



▸ **Pipeline**

▸ StandardScaler

▸ RandomForestRegressor

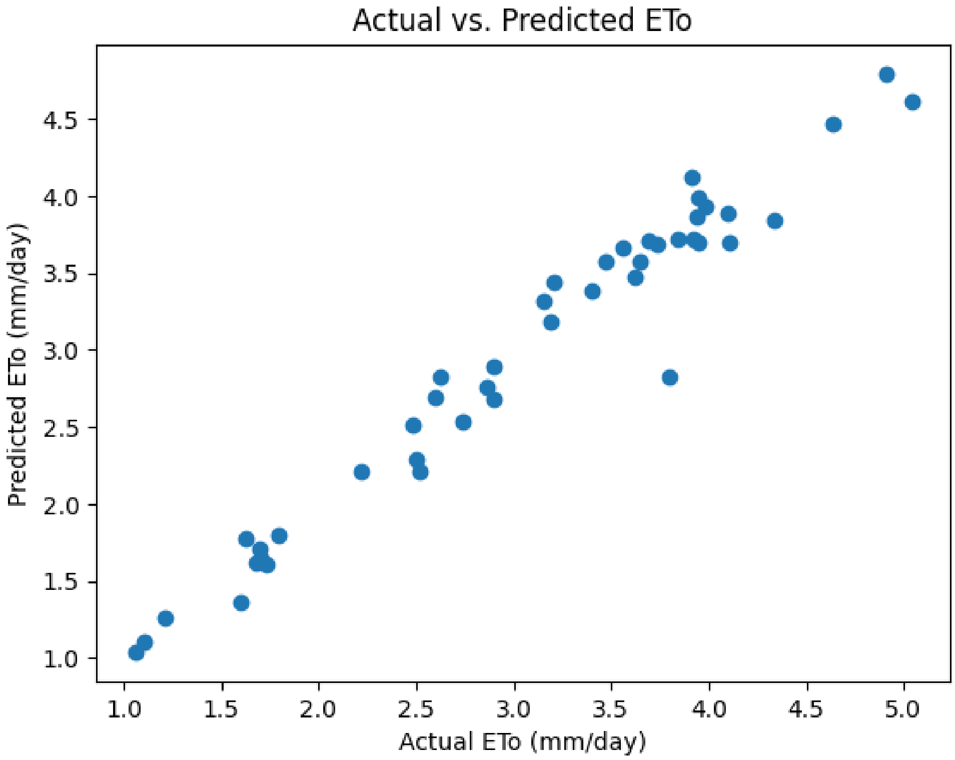
```
#making predictions
predictions=pipeline.predict(X_Test)


# Evaluate the model
mse = mean_squared_error(Y_Test, predictions)
print(f'Random Forest Regression MSE on the test set: {mse}')
```

```
     Random Forest Regression MSE on the test set: 0.05321705522727278
```

```
import matplotlib.pyplot as plt

plt.scatter(Y_Test, predictions)
plt.xlabel('Actual ETo (mm/day)')
plt.ylabel('Predicted ETo (mm/day)')
plt.title('Actual vs. Predicted ETo')
plt.show()
```



Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV

# Define the hyperparameter grid
param_grid = {
    'rf__n_estimators': [50, 100, 150],
    'rf__max_depth': [None, 10, 20],
    'rf__min_samples_split': [2, 5, 10]
}

# Create the grid search object
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)

# Fit the grid search to the data
grid_search.fit(X_Train, Y_Train)

# Print the best hyperparameters
print("Best Hyperparameters:", grid_search.best_params_)

# Get the best model from grid search
best_model = grid_search.best_estimator_

# Make predictions using the best model
best_predictions = best_model.predict(X_Test)

# Evaluate the best model
best_mse = mean_squared_error(Y_Test, best_predictions)
print(f'Best Random Forest Regression MSE on the test set: {best_mse}')
```

```
Best Hyperparameters: {'rf__max_depth': 10, 'rf__min_samples_split': 2, 'rf__n_estimators': 150}
Best Random Forest Regression MSE on the test set: 0.051938496921612495
```

Let's make some predictions

```
# Assuming 'new_data' is a DataFrame containing new data with the same features as our training data
new_data = pd.DataFrame({
    'Min Temp (°C)': [7],
    'Max Temp (°C)': [20],
    'Humidity (%)': [55],
    'Wind (km/day)': [8],
    'Sun (hours)': [4],
    'Rad (MJ/m²/day)': [9.2]
})

# Using best_model to make predictions
new_predictions = best_model.predict(new_data)

# Display the predictions
print('Predicted ETo (mm/day):', new_predictions[0])
```

```
Predicted ETo (mm/day): 1.1363897924297928
```