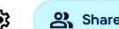


FEB
24

Week 2 Task (as per your instructions)

-  1. Fit dataset into an ML model
 - means: load your clean dataset and prepare it for machine learning.
-  2. Split the data (train/test)
 - you already did this in Step 4 – train_test_split ✓
-  3. Train the model
 - you'll now build and train a Neural Network (Step 5).
-  4. Predict the output
 - after training, the model will predict future or unseen values.
-  5. Compare predicted vs actual values
 - you'll print both to see how close your model is.
-  6. Find the accuracy / error
 - you'll calculate metrics like MAE, MSE, or R² Score.



Q

Commands + Code + Text ▶ Run all

✓ RAM Disk

[10]

```
from google.colab import files
uploaded = files.upload()

Choose file: No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving dataset1.xlsx to dataset1 (3).xlsx
Saving dataset2.xlsx to dataset2 (3).xlsx
```

[11]

```
import pandas as pd

# Read the Excel files
data1 = pd.read_excel("dataset1.xlsx")
data2 = pd.read_excel("dataset2.xlsx")

# Show first few rows from each
print("Dataset 1:")
display(data1.head())

print("\nDataset 2:")
display(data2.head())

# Check the column names
print("\nColumns in Dataset 1:", data1.columns.tolist())
print("Columns in Dataset 2:", data2.columns.tolist())
```

Dataset 1:

	datetime	National Hourly Demand	Northern Region Hourly Demand	Western Region Hourly Demand	Eastern Region Hourly Demand	Southern Region Hourly Demand	North-Eastern Region Hourly Demand
0	2019-01-01 00:00:00	118690.67	33692.02	38522.22	13128.89	31681.83	1665.72
1	2019-01-01 01:00:00	116029.23	32534.39	38071.09	12737.53	31129.97	1556.24
2	2019-01-01 02:00:00	114044.14	31730.37	37680.10	12387.36	30760.87	1485.44
3	2019-01-01 03:00:00	113648.97	31529.25	37747.37	12301.12	30616.27	1454.96
4	2019-01-01 04:00:00	116290.05	32406.61	38101.80	12479.13	31839.38	1463.14

Dataset 2:

	Year	Month	Monthly_load	max_temp
0	2019	Jan	137954.776707	25.86
1	2019	Feb	140545.370461	27.72
2	2019	Mar	147127.426398	31.08
3	2019	Apr	154253.928819	34.74
4	2019	May	162908.707527	35.81

Columns in Dataset 1: ['datetime', 'National Hourly Demand', 'Northern Region Hourly Demand', 'Western Region Hourly Demand', 'Eastern Region Hourly Demand', 'Southern Region Hourly Demand', 'North-Eastern Region Hourly Demand']
 Columns in Dataset 2: ['Year', 'Month', 'Monthly_load', 'max_temp']

[12]

✓ 0s

```
# Check data health for Dataset 1
print("Dataset 1 Info:")
data1.info()
print("\nMissing values in Dataset 1:")
print(data1.isnull().sum())

print("\n-----\n")

# Check data health for Dataset 2
print("Dataset 2 Info:")
data2.info()
print("\nMissing values in Dataset 2:")
print(data2.isnull().sum())
```

Variables

Terminal

✓ 1:34 AM Python 3





Vo

LTE1

4G

LTE2

.11

.11

13%

.11

.11

.11

.11

File Edit View Insert Runtime Tools Help



Share



Q Commands + Code + Text ▶ Run all ▾

✓ RAM Disk

	1	2019	Feb	140545.370461	27.72
1	2	2019	Mar	147127.426398	31.08
2	3	2019	Apr	154253.928819	34.74
3	4	2019	May	162908.707527	35.81



Columns in Dataset 1: ['datetime', 'National Hourly Demand', 'Northern Region Hourly Demand', 'Western Region Hourly Demand', 'Eastern Region Hourly Demand', 'Southern Region Hourly Demand', 'North-Eastern Region Hourly Demand']
Columns in Dataset 2: ['Year', 'Month', 'M']

[12]

✓ 0s

```
# Check data health for Dataset 1
print("Dataset 1 Info:")
data1.info()
print("\nMissing values in Dataset 1:")
print(data1.isnull().sum())

print("\n-----\n")

# Check data health for Dataset 2
print("Dataset 2 Info:")
data2.info()
print("\nMissing values in Dataset 2:")
print(data2.isnull().sum())
```

Dataset 1 Info:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46728 entries, 0 to 46727
Data columns (total 7 columns):
 #   Column
 --- 
 0   datetime
 1   National Hourly Demand
 2   Northern Region Hourly Demand
 3   Western Region Hourly Demand
 4   Eastern Region Hourly Demand
 5   Southern Region Hourly Demand
 6   North-Eastern Region Hourly Demand
dtypes: datetime64[ns](1), float64(6)
memory usage: 2.5 MB
```

Missing values in Dataset 1:

```
datetime          0
National Hourly Demand      0
Northern Region Hourly Demand 0
Western Region Hourly Demand 0
Eastern Region Hourly Demand 0
Southern Region Hourly Demand 0
North-Eastern Region Hourly Demand 0
dtype: int64
```

1191

Variables Terminal



✓ 1:34 AM Python 3





Q Commands

+ Code

+ Text

> Run all

✓ RAM

Disk

North-Eastern Region Hourly Demand

```
dtype: int64
```

```
[19] ✓ 5s
```

```
import pandas as pd
```

```
# Load both datasets (make sure they're uploaded on the left side under "Files")
```

```
dataset1 = pd.read_excel('/content/dataset1.xlsx')
```

```
dataset2 = pd.read_excel('/content/dataset2.xlsx')
```

```
print("✅ Datasets Loaded Successfully")
```

```
✅ Datasets Loaded Successfully
```

[16]

✓ 0s

dataset1.head()

	datetime	National Hourly Demand	Northen Region Hourly Demand	Western Region Hourly Demand	Eastern Region Hourly Demand	Southern Region Hourly Demand	North-Eastern Region Hourly Demand
0	2019-01-01 00:00:00	118690.67	33692.02	38522.22	13128.89	31681.83	1665.72
1	2019-01-01 01:00:00	116029.23	32534.39	38071.09	12737.53	31129.97	1556.24
2	2019-01-01 02:00:00	114044.14	31730.37	37680.10	12387.36	30760.87	1485.44
3	2019-01-01 03:00:00	113648.97	31529.25	37747.37	12301.12	30616.27	1454.96
4	2019-01-01 04:00:00	116290.05	32406.61	38101.80	12479.13	31839.38	1463.14

[22]

✓ 0s

print("Dataset 1 Info:")

dataset1.info()

Dataset 1 Info:

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 46728 entries, 0 to 46727
```

```
Data columns (total 7 columns):
```

```
 #   Column
```

```
---  -----
```

```
 0   datetime
```

```
 1   National Hourly Demand
```

```
 2   Northen Region Hourly Demand
```

```
 3   Western Region Hourly Demand
```

```
 4   Eastern Region Hourly Demand
```

```
 5   Southern Region Hourly Demand
```

```
 6   North-Eastern Region Hourly Demand
```

```
dtypes: datetime64[ns](1), float64(6)
```

```
memory usage: 2.5 MB
```

[24]

```
from sklearn.model_selection import train_test_split
```

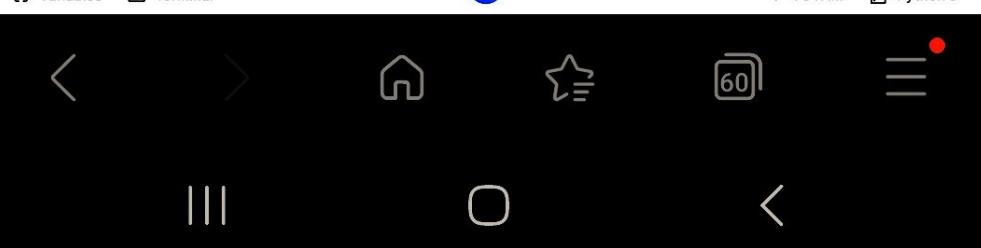
```
X = dataset1[['Northen Region Hourly Demand',  
              'Western Region Hourly Demand',  
              'Eastern Region Hourly Demand',  
              'Southern Region Hourly Demand',  
              'North-Eastern Region Hourly Demand']]
```

```
y = dataset1['National Hourly Demand']  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Variables

Terminal

✓ 1:34 AM Python 3





Q Commands

+ Code

+ Text

> Run all

✓ RAM

Disk

```
Data columns (total 7 columns):
 #   Column
 --- 
 0   datetime
 1   National Hourly Demand
 2   Northen Region Hourly Demand
 3   Western Region Hourly Demand
 4   Eastern Region Hourly Demand
 5   Southern Region Hourly Demand
 6   North-Eastern Region Hourly Demand
dtypes: datetime64[ns](1), float64(6)
memory usage: 2.5 MB
```

```
[24] from sklearn.model_selection import train_test_split
X = dataset1[['Northen Region Hourly Demand',
               'Western Region Hourly Demand',
               'Eastern Region Hourly Demand',
               'Southern Region Hourly Demand',
               'North-Eastern Region Hourly Demand']]

y = dataset1['National Hourly Demand']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Data Split Successful!")
print("Training samples:", X_train.shape[0])
print("Testing samples:", X_test.shape[0])
```

Data Split Successful!
 Training samples: 37382
 Testing samples: 9346

```
[31] # ----- Random Forest Regressor -----
# 1 Import necessary libraries
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error

# 2 Initialize the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# 3 Train the model
rf_model.fit(X_train, y_train)
print("Random Forest Training Complete!")

# 4 Make predictions
y_pred_rf = rf_model.predict(X_test)

# 5 Evaluate the model
r2_rf = r2_score(y_test, y_pred_rf)
mse_rf = mean_squared_error(y_test, y_pred_rf)

# 6 Print results
print("Random Forest Predictions Complete!")
print(f"Accuracy (R²) : {r2_rf*100:.2f}%")
print(f"Mean Squared Error : {mse_rf:.2f}")

 Random Forest Training Complete!
 Random Forest Predictions Complete!
Accuracy (R²) : 99.90%
Mean Squared Error : 562603.41
```

```
[29] # Step 5.1: Import the Linear Regression model
from sklearn.linear_model import LinearRegression

# Step 5.2: Create the model
model = LinearRegression()

# Step 5.3: Train (fit) the model with training data
model.fit(X_train, y_train)

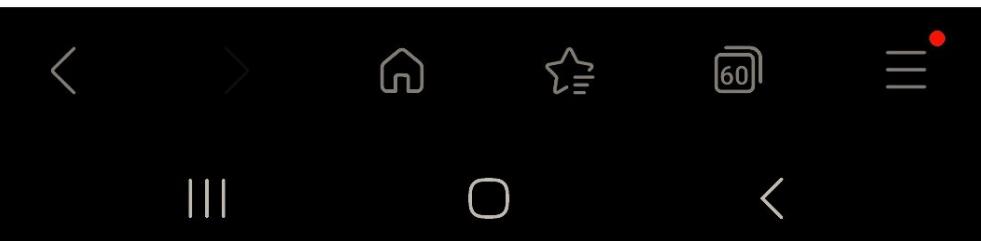
# Step 5.4: Print confirmation
print("Model training completed successfully!")
```

Variables

Terminal



✓ 1:34 AM Python 3





Q Commands + Code + Text ▶ Run all ✓ RAM Disk

```
[24] y = dataset1['National Hourly Demand']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Data Split Successful!")
print("Training samples:", X_train.shape[0])
print("Testing samples:", X_test.shape[0])
```

```
✓ Data Split Successful!
Training samples: 37382
Testing samples: 9346
```

```
[31] # ----- Random Forest Regressor -----
# 1 Import necessary libraries
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

```
# 2 Initialize the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
# 3 Train the model
rf_model.fit(X_train, y_train)
print("Random Forest Training Complete!")
```

```
# 4 Make predictions
y_pred_rf = rf_model.predict(X_test)
```

```
# 5 Evaluate the model
r2_rf = r2_score(y_test, y_pred_rf)
mse_rf = mean_squared_error(y_test, y_pred_rf)
```

```
# 6 Print results
print("Random Forest Predictions Complete!")
print(f"Accuracy (R²) : {r2_rf*100:.2f}%")
print(f"Mean Squared Error : {mse_rf:.2f}")
```

```
✓ Random Forest Training Complete!
✓ Random Forest Predictions Complete!
Accuracy (R²) : 99.90%
Mean Squared Error : 562603.41
```

```
[29] # Step 5.1: Import the Linear Regression model
from sklearn.linear_model import LinearRegression
```

```
# Step 5.2: Create the model
model = LinearRegression()
```

```
# Step 5.3: Train (fit) the model with training data
model.fit(X_train, y_train)
```

```
# Step 5.4: Print confirmation
print("Model training completed successfully!")
```

```
✓ Model training completed successfully!
```

```
[32] from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
```

```
# Step 6: Evaluate Model
y_pred = model.predict(X_test)
```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
# Convert R² to percentage
accuracy_percent = r2 * 100
```

```
print("Model Evaluation Results")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R² Score: {r2:.2f}")
print(f"Accuracy: {accuracy_percent:.2f}%")
```

```
... Model Evaluation Results
Mean Absolute Error (MAE): 0.00
Mean Squared Error (MSE): 0.00
R² Score: 1.00
✓ Accuracy: 100.00%
```

