```
import pandas as pd
import numpy as np

generation2021 = pd.read_csv("ASI - 2021.csv")
generation2021 = generation2021.drop(columns = [
    " Energy BOARD 1 3MW",
    " Energy BOARD 1 5MW",
    " Energy BOARD 10 3MW",
    " Energy BOARD 10 5MW",
    " Energy BOARD 11 3MW",
    " Energy BOARD 11 5MW",
    " Energy BOARD 12 5MW",
    " Energy BOARD 13 5MW",
    " Energy BOARD 14 5MW",
    " Energy BOARD 15 5MW",
    " Energy BOARD 16 5MW",
    " Energy BOARD 17 5MW",
    " Energy BOARD 18 5MW",
    " Energy BOARD 19 5MW",
    " Energy BOARD 2 3MW",
    " Energy BOARD 2 5MW",
    " Energy BOARD 3 3MW",
    " Energy BOARD 3 5MW",
    " Energy BOARD 4 3MW",
    " Energy BOARD 4 5MW",
    " Energy BOARD 5 3MW",
    " Energy BOARD 5 5MW",
    " Energy BOARD 6 3MW",
    " Energy BOARD 6 5MW",
    " Energy BOARD 7 3MW",
    " Energy BOARD 7 5MW",
    " Energy BOARD 8 3MW",
    " Energy BOARD 8 5MW",
    " Energy BOARD 9 3MW",
    " Energy BOARD 9 5MW",
    " Energy MSB  5MW 3200A",
    " Energy MSB 3MW",
    " Energy MSB 5MW 6300A"
]
)

generation2021["Time"] = pd.to_datetime(generation2021["Time"],
format='mixed')
generation2021.set_index("Time", inplace = True)

daily_sum = generation2021.resample('10T').sum()

daily_sum['10-min mean Solar Power (MW)'] = daily_sum.sum(axis=1)

data2021 = daily_sum[['10-min mean Solar Power (MW)']].reset_index()
```

```
print(data2021)
```

```
                          Time  10-min mean Solar Power (MW)
0      2021-01-01 00:00:00                             0
1      2021-01-01 00:10:00                             0
2      2021-01-01 00:20:00                             0
3      2021-01-01 00:30:00                             0
4      2021-01-01 00:40:00                             0
...                    ...                           ...
52555 2021-12-31 23:10:00                             0
52556 2021-12-31 23:20:00                             0
52557 2021-12-31 23:30:00                             0
52558 2021-12-31 23:40:00                             0
52559 2021-12-31 23:50:00                             0

[52560 rows x 2 columns]

C:\Users\AbdullahHarithJamadi\AppData\Local\Temp\
ipykernel_12920\4184560830.py:45: FutureWarning: 'T' is deprecated and
will be removed in a future version, please use 'min' instead.
  daily_sum = generation2021.resample('10T').sum()
```

```
import pandas as pd
import numpy as np

generation2022 = pd.read_csv("ASI - 2022.csv")
generation2022 = generation2022.drop(columns = [
    " Energy BOARD 1 3MW",
    " Energy BOARD 1 5MW",
    " Energy BOARD 10 3MW",
    " Energy BOARD 10 5MW",
    " Energy BOARD 11 3MW",
    " Energy BOARD 11 5MW",
    " Energy BOARD 12 5MW",
    " Energy BOARD 13 5MW",
    " Energy BOARD 14 5MW",
    " Energy BOARD 15 5MW",
    " Energy BOARD 16 5MW",
    " Energy BOARD 17 5MW",
    " Energy BOARD 18 5MW",
    " Energy BOARD 19 5MW",
    " Energy BOARD 2 3MW",
    " Energy BOARD 2 5MW",
    " Energy BOARD 3 3MW",
    " Energy BOARD 3 5MW",
    " Energy BOARD 4 3MW",
    " Energy BOARD 4 5MW",
    " Energy BOARD 5 3MW",
    " Energy BOARD 5 5MW",
```

```
    " Energy BOARD 6 3MW",
    " Energy BOARD 6 5MW",
    " Energy BOARD 7 3MW",
    " Energy BOARD 7 5MW",
    " Energy BOARD 8 3MW",
    " Energy BOARD 8 5MW",
    " Energy BOARD 9 3MW",
    " Energy BOARD 9 5MW",
    " Energy MSB  5MW 3200A",
    " Energy MSB 3MW",
    " Energy MSB 5MW 6300A"
]
)

generation2022["Time"] = pd.to_datetime(generation2022["Time"],
format='mixed')
generation2022.set_index("Time", inplace = True)

daily_sum = generation2022.resample('10T').sum()

daily_sum['10-min mean Solar Power (MW)'] = daily_sum.sum(axis=1)

data2022 = daily_sum[['10-min mean Solar Power (MW)']].reset_index()

print(data2022)
# print(data2022.to_string())
```

```
                     Time  10-min mean Solar Power (MW)
0     2022-01-01 00:00:00                           0.0
1     2022-01-01 00:10:00                           0.0
2     2022-01-01 00:20:00                           0.0
3     2022-01-01 00:30:00                           0.0
4     2022-01-01 00:40:00                           0.0
...                   ...                           ...
52555 2022-12-31 23:10:00                           0.0
52556 2022-12-31 23:20:00                           0.0
52557 2022-12-31 23:30:00                           0.0
52558 2022-12-31 23:40:00                           0.0
52559 2022-12-31 23:50:00                           0.0

[52560 rows x 2 columns]

C:\Users\AbdullahHarithJamadi\AppData\Local\Temp\
ipykernel_12920\2344569143.py:45: FutureWarning: 'T' is deprecated and
will be removed in a future version, please use 'min' instead.
  daily_sum = generation2022.resample('10T').sum()
```

```
import pandas as pd
import numpy as np

generation2023 = pd.read_csv("Generation - 2023 (January -
```

```
August).csv")
generation2023 = generation2023.drop(columns = [
    " Energy BOARD 1 3MW",
    " Energy BOARD 1 5MW",
    " Energy BOARD 10 3MW",
    " Energy BOARD 10 5MW",
    " Energy BOARD 11 3MW",
    " Energy BOARD 11 5MW",
    " Energy BOARD 12 5MW",
    " Energy BOARD 13 5MW",
    " Energy BOARD 14 5MW",
    " Energy BOARD 15 5MW",
    " Energy BOARD 16 5MW",
    " Energy BOARD 17 5MW",
    " Energy BOARD 18 5MW",
    " Energy BOARD 19 5MW",
    " Energy BOARD 2 3MW",
    " Energy BOARD 2 5MW",
    " Energy BOARD 3 3MW",
    " Energy BOARD 3 5MW",
    " Energy BOARD 4 3MW",
    " Energy BOARD 4 5MW",
    " Energy BOARD 5 3MW",
    " Energy BOARD 5 5MW",
    " Energy BOARD 6 3MW",
    " Energy BOARD 6 5MW",
    " Energy BOARD 7 3MW",
    " Energy BOARD 7 5MW",
    " Energy BOARD 8 3MW",
    " Energy BOARD 8 5MW",
    " Energy BOARD 9 3MW",
    " Energy BOARD 9 5MW",
    " Energy MSB  5MW 3200A",
    " Energy MSB 3MW",
    " Energy MSB 5MW 6300A"
]
)

generation2023["Time"] = pd.to_datetime(generation2023["Time"],
format='mixed')
generation2023.set_index("Time", inplace = True)

daily_sum = generation2023.resample('10T').sum()

daily_sum['10-min mean Solar Power (MW)'] = daily_sum.sum(axis=1)

data2023 = daily_sum[['10-min mean Solar Power (MW)']].reset_index()

print(data2023)
```

```
                     Time  10-min mean Solar Power (MW)
0     2023-01-01 00:00:00                             0
1     2023-01-01 00:10:00                             0
2     2023-01-01 00:20:00                             0
3     2023-01-01 00:30:00                             0
4     2023-01-01 00:40:00                             0
...                   ...                           ...
34987 2023-08-31 23:10:00                             0
34988 2023-08-31 23:20:00                             0
34989 2023-08-31 23:30:00                             0
34990 2023-08-31 23:40:00                             0
34991 2023-08-31 23:50:00                             0

[34992 rows x 2 columns]

C:\Users\AbdullahHarithJamadi\AppData\Local\Temp\
ipykernel_12920\318052064.py:45: FutureWarning: 'T' is deprecated and
will be removed in a future version, please use 'min' instead.
  daily_sum = generation2023.resample('10T').sum()
```

```
data = pd.concat([data2021, data2022, data2023], ignore_index=True)
data.describe()
```

```
                                Time  10-min mean Solar Power (MW)
count                         140112                 140112.000000
mean   2022-05-02 11:54:59.999999488                   5093.091541
min              2021-01-01 00:00:00                      0.000000
25%              2021-09-01 05:57:30                      0.000000
50%              2022-05-02 11:55:00                     29.000000
75%              2022-12-31 17:52:30                   9599.250000
max              2023-08-31 23:50:00                  28509.000000
std                              NaN                   7231.336095
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam

# Select relevant columns
selected_columns = [
    '10-min mean Solar Power (MW)'
    # '24 Hour mean solar power from solar panel (MW)'

]
data_selected = data[selected_columns]
split_ratio = 0.8

train_size = int(len(data_selected) * split_ratio)
```

```python
train_data = data_selected[:train_size]
test_data = data_selected[train_size:]

# Normalize the data
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train_data)
test_scaled = scaler.transform(test_data)

# Prepare the data for LSTM
def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps)])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

time_steps = 144 # 6: Short Term, 12: Medium Term, 144: Daily cycle
X_train, y_train = create_dataset(train_scaled, train_scaled[:, 0],
time_steps)
X_test, y_test = create_dataset(test_scaled, test_scaled[:, 0],
time_steps)

# Define the LSTM model architecture
model = Sequential([
    LSTM(units=64, input_shape=(X_train.shape[1], X_train.shape[2])),
    # LSTM(units=64),
    Dense(units=1)
])

from keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.0005),
loss='mean_squared_error')

# Define the EarlyStopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the LSTM model
history = model.fit(X_train, y_train, epochs=200, batch_size=32,
validation_split=0.1, verbose=1, callbacks=[early_stopping])

# Evaluate the model performance
model.evaluate(X_test, y_test)

model.save("lstm_model.h5")

# Make predictions
predictions = model.predict(X_test)
```

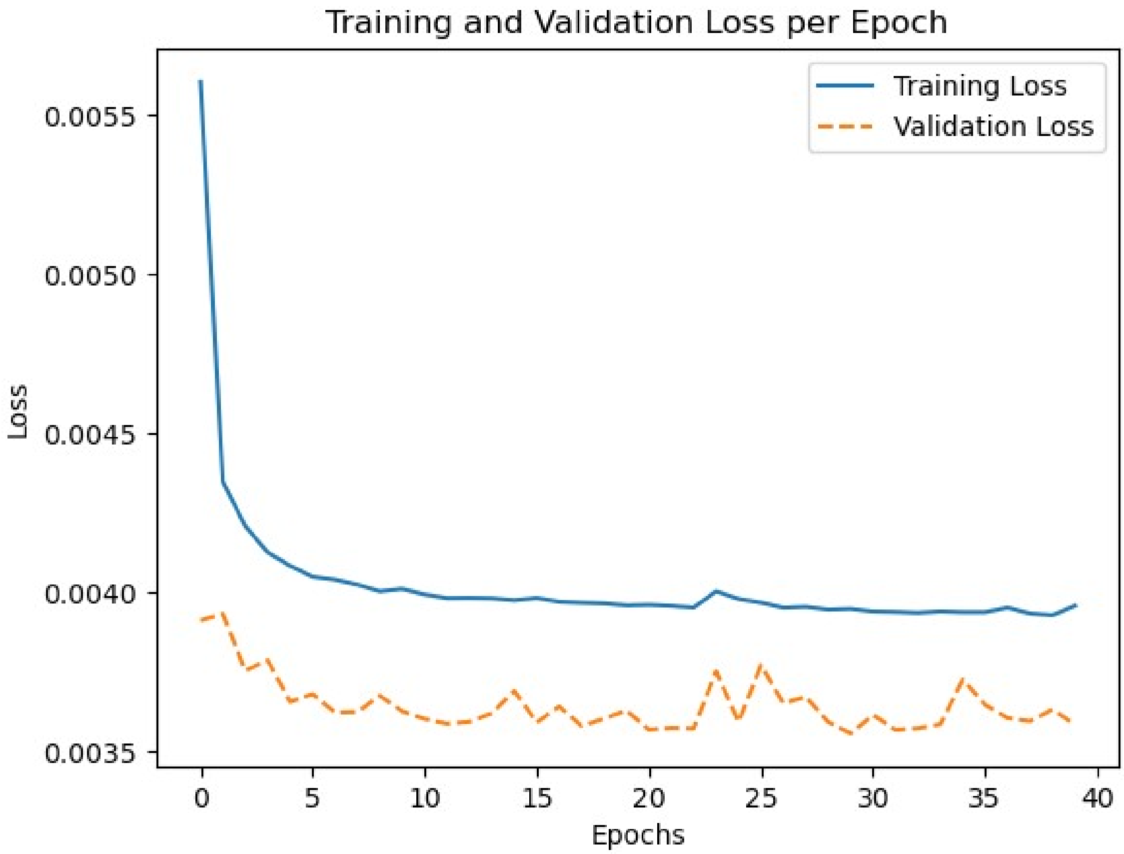
```
Epoch 1/200
```

```
c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

3149/3149 ━━━━━━━━━━━━━━━━━━━━ 64s 20ms/step - loss: 0.0082 -
val_loss: 0.0039
Epoch 2/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0044 -
val_loss: 0.0039
Epoch 3/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 68s 22ms/step - loss: 0.0043 -
val_loss: 0.0038
Epoch 4/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 218s 69ms/step - loss: 0.0041 -
val_loss: 0.0038
Epoch 5/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 218s 69ms/step - loss: 0.0040 -
val_loss: 0.0037
Epoch 6/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 187s 60ms/step - loss: 0.0040 -
val_loss: 0.0037
Epoch 7/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 173s 55ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 8/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 64s 20ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 9/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 67s 21ms/step - loss: 0.0040 -
val_loss: 0.0037
Epoch 10/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 11/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 12/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0039 -
val_loss: 0.0036
Epoch 13/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 14/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 15/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0039 -
```

```
val_loss: 0.0037
Epoch 16/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 17/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 18/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 19/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 20/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0039 -
val_loss: 0.0036
Epoch 21/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 22/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 23/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0039 -
val_loss: 0.0036
Epoch 24/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0041 -
val_loss: 0.0038
Epoch 25/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 26/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0038
Epoch 27/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 28/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0039 -
val_loss: 0.0037
Epoch 29/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 30/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0039 -
val_loss: 0.0036
Epoch 31/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 61s 19ms/step - loss: 0.0039 -
val_loss: 0.0036
```

```
Epoch 32/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 33/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0039 -
val_loss: 0.0036
Epoch 34/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 66s 21ms/step - loss: 0.0039 -
val_loss: 0.0036
Epoch 35/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 64s 20ms/step - loss: 0.0041 -
val_loss: 0.0037
Epoch 36/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 37/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0040 -
val_loss: 0.0036
Epoch 38/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 64s 20ms/step - loss: 0.0039 -
val_loss: 0.0036
Epoch 39/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0038 -
val_loss: 0.0036
Epoch 40/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 65s 21ms/step - loss: 0.0040 -
val_loss: 0.0036
872/872 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - loss: 0.0050

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

872/872 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step
```

```
import matplotlib.pyplot as plt

# Plotting training and validation loss per epoch
plt.plot(history.history['loss'], label='Training Loss',
linestyle='-')
plt.plot(history.history['val_loss'], label='Validation Loss',
linestyle='--')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss per Epoch')
plt.show()
```




```
y_pred = model.predict(X_test)
y_true = y_test

# Now can print y_true and y_pred
print("Actual values (y_true):", y_true)
print("Predicted values (y_pred):", y_pred)
```

```
872/872 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step
Actual values (y_true): [0.41200516 0.45729346 0.49099971 ... 0.
0.         0.        ]
Predicted values (y_pred): [[3.9197886e-01]
 [4.2735302e-01]
 [4.6667087e-01]
 ...
 [3.4280866e-04]
 [4.0030479e-04]
 [4.8271567e-04]]
```

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_true, y_pred)
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 0.004798272029538745
```

```python
import numpy as np
import pandas as pd

# Flatten the X_test array
X_test_flat = X_test.reshape(X_test.shape[0], -1)

# Convert y_test and y_pred to 1D arrays
y_test_flat = y_test.flatten()
y_pred_flat = y_pred.flatten()

# Convert X_test_flat to DataFrame
X_test_df = pd.DataFrame(X_test_flat, columns=[f'Feature_{i}' for i in
range(X_test_flat.shape[1])])

# Create DataFrame for y_test and y_pred
y_test_df = pd.DataFrame({'Actual values (y_true)': y_test_flat})
y_pred_df = pd.DataFrame({'Predicted values (y_pred)': y_pred_flat})

# Concatenate X_test_df, y_test_df, and y_pred_df along columns
result_df = pd.concat([X_test_df, y_test_df, y_pred_df], axis=1)

# Print the result DataFrame
result_df.head(10)
```

```
   Feature_0  Feature_1  Feature_2  Feature_3  Feature_4  Feature_5  \
0   0.415304   0.454640   0.501004   0.539013   0.578600   0.609366
1   0.454640   0.501004   0.539013   0.578600   0.609366   0.644937
2   0.501004   0.539013   0.578600   0.609366   0.644937   0.673049
3   0.539013   0.578600   0.609366   0.644937   0.673049   0.688791
4   0.578600   0.609366   0.644937   0.673049   0.688791   0.744012
5   0.609366   0.644937   0.673049   0.688791   0.744012   0.736804
6   0.644937   0.673049   0.688791   0.744012   0.736804   0.670970
7   0.673049   0.688791   0.744012   0.736804   0.670970   0.612665
8   0.688791   0.744012   0.736804   0.670970   0.612665   0.705680
9   0.744012   0.736804   0.670970   0.612665   0.705680   0.838282

   Feature_6  Feature_7  Feature_8  Feature_9  ...  Feature_136
Feature_137  \
0   0.644937   0.673049   0.688791   0.744012  ...     0.048444
0.063217
1   0.673049   0.688791   0.744012   0.736804  ...     0.063217
0.134000
2   0.688791   0.744012   0.736804   0.670970  ...     0.134000
0.180436
3   0.744012   0.736804   0.670970   0.612665  ...     0.180436
0.223788
4   0.736804   0.670970   0.612665   0.705680  ...     0.223788
0.276320
5   0.670970   0.612665   0.705680   0.838282  ...     0.276320
0.327775
```

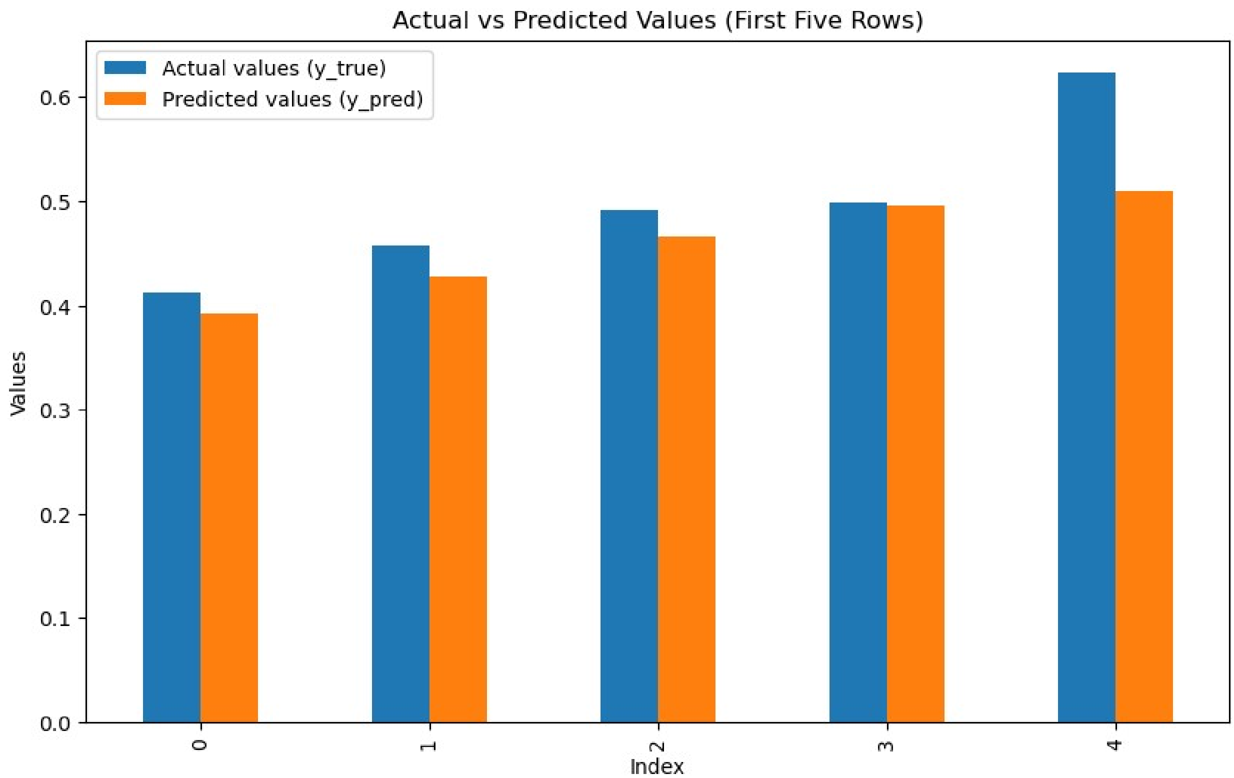
```
6   0.612665   0.705680   0.838282   0.834696  ...     0.327775
0.373207
7   0.705680   0.838282   0.834696   0.842477  ...     0.373207
0.412005
8   0.838282   0.834696   0.842477   0.884323  ...     0.412005
0.457293
9   0.834696   0.842477   0.884323   0.890419  ...     0.457293
0.491000

   Feature_138  Feature_139  Feature_140  Feature_141  Feature_142  \
0     0.134000     0.180436     0.223788     0.276320     0.327775
1     0.180436     0.223788     0.276320     0.327775     0.373207
2     0.223788     0.276320     0.327775     0.373207     0.412005
3     0.276320     0.327775     0.373207     0.412005     0.457293
4     0.327775     0.373207     0.412005     0.457293     0.491000
5     0.373207     0.412005     0.457293     0.491000     0.498924
6     0.412005     0.457293     0.491000     0.498924     0.623530
7     0.457293     0.491000     0.498924     0.623530     0.558018
8     0.491000     0.498924     0.623530     0.558018     0.474182
9     0.498924     0.623530     0.558018     0.474182     0.385686

   Feature_143  Actual values (y_true)  Predicted values (y_pred)
0     0.373207                0.412005                   0.391979
1     0.412005                0.457293                   0.427353
2     0.457293                0.491000                   0.466671
3     0.491000                0.498924                   0.495900
4     0.498924                0.623530                   0.509462
5     0.623530                0.558018                   0.590933
6     0.558018                0.474182                   0.560533
7     0.474182                0.385686                   0.523240
8     0.385686                0.356713                   0.466437
9     0.356713                0.380809                   0.444794

[10 rows x 146 columns]
```

```python
result_df = result_df.iloc[:, -2:]
result_df.head()
```

```
   Actual values (y_true)  Predicted values (y_pred)
0                0.412005                   0.391979
1                0.457293                   0.427353
2                0.491000                   0.466671
3                0.498924                   0.495900
4                0.623530                   0.509462
```

```python
import matplotlib.pyplot as plt

# Selecting only the first five rows
result_df_first_five = result_df.iloc[:5]
```

```python
# Plotting the actual vs predicted values for the first five rows
result_df_first_five.plot(kind='bar', figsize=(10, 6))
plt.title('Actual vs Predicted Values (First Five Rows)')
plt.xlabel('Index')
plt.ylabel('Values')
plt.show()
```

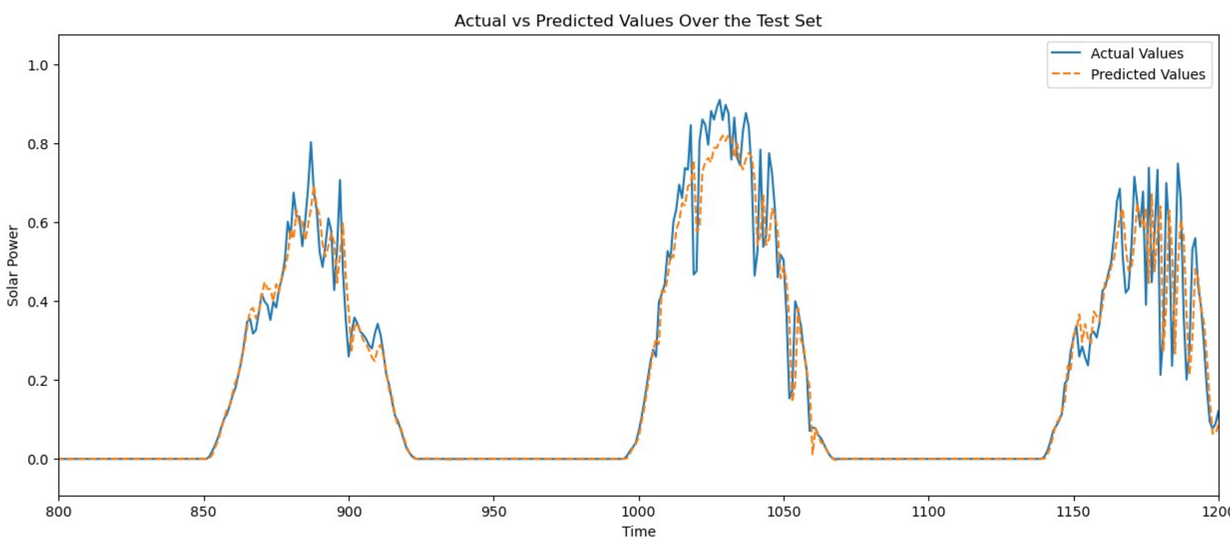



```python
from sklearn.metrics import mean_absolute_error, mean_squared_error

y_true = result_df['Actual values (y_true)']
y_pred = result_df['Predicted values (y_pred)']

mae = mean_absolute_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
rmse = mean_squared_error(y_true, y_pred, squared=False)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
```

```
Mean Absolute Error: 0.0316790132187954
Mean Squared Error: 0.004798272029538745
Root Mean Squared Error: 0.0692695606275855
```

```
c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\sklearn\
metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in
version 1.4 and will be removed in 1.6. To calculate the root mean
squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

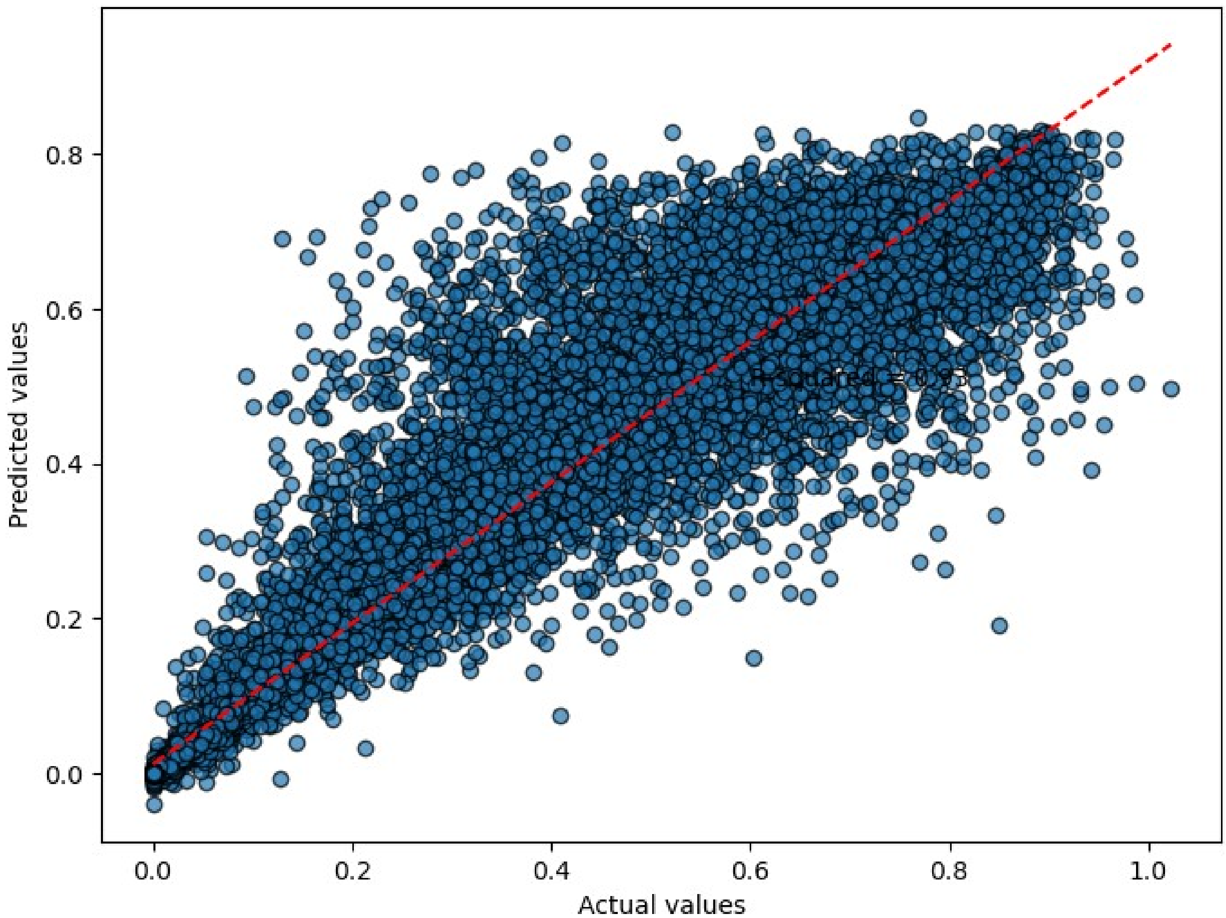
```
import matplotlib.pyplot as plt

# Plotting the actual vs predicted values over the entire test set
plt.figure(figsize=(15, 6))
plt.plot(result_df['Actual values (y_true)'], label='Actual Values')
plt.plot(result_df['Predicted values (y_pred)'], label='Predicted
Values', linestyle='dashed')
plt.xlim(800,1200)
plt.title('Actual vs Predicted Values Over the Test Set')
plt.xlabel('Time')
plt.ylabel('Solar Power')
plt.legend()
plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score

r_squared = r2_score(y_true, y_pred)
plt.figure(figsize=(8, 6))
plt.scatter(y_true,y_pred, cmap='viridis', edgecolor='k', alpha=0.7)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')

plt.plot(np.unique(y_true), np.poly1d(np.polyfit(y_true, y_pred, 1))
(np.unique(y_true)), 'r--')
```

```
plt.text(0.6, 0.5, 'R-squared = %0.2f' % r_squared)
plt.show()
```

```
C:\Users\AbdullahHarithJamadi\AppData\Local\Temp\
ipykernel_12920\702279148.py:7: UserWarning: No data for colormapping
provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(y_true,y_pred, cmap='viridis', edgecolor='k', alpha=0.7)
```



UNIVARIATE USING ASI GENERATION 2021 TO COMPARE WITH MULTIVARIATE ASI GENERATION 2021

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
from keras.callbacks import EarlyStopping

# Select relevant columns
selected_columns = [
    '10-min mean Solar Power (MW)'
    # '24 Hour mean solar power from solar panel (MW)'
```

```
]
data_selected = data2021[selected_columns]
split_ratio = 0.8

train_size = int(len(data_selected) * split_ratio)

train_data = data_selected[:train_size]
test_data = data_selected[train_size:]

# Normalize the data
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train_data)
test_scaled = scaler.transform(test_data)

# Prepare the data for LSTM
def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps)])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

time_steps = 144 # 6: Short Term, 12: Medium Term, 144: Daily cycle
X_train, y_train = create_dataset(train_scaled, train_scaled[:, 0],
time_steps)
X_test, y_test = create_dataset(test_scaled, test_scaled[:, 0],
time_steps)

# Define the LSTM model architecture
model = Sequential([
    LSTM(units=64, input_shape=(X_train.shape[1], X_train.shape[2])),
    Dense(units=1)
])
from keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.0005), loss='mse',
metrics=['mae'])

# Define the EarlyStopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the LSTM model
history = model.fit(X_train, y_train, epochs=200, batch_size=32,
validation_split=0.1, verbose=1, callbacks=[early_stopping])

# Evaluate the model performance
model.evaluate(X_test, y_test)

model.save("lstm_univariate_model.h5")
```

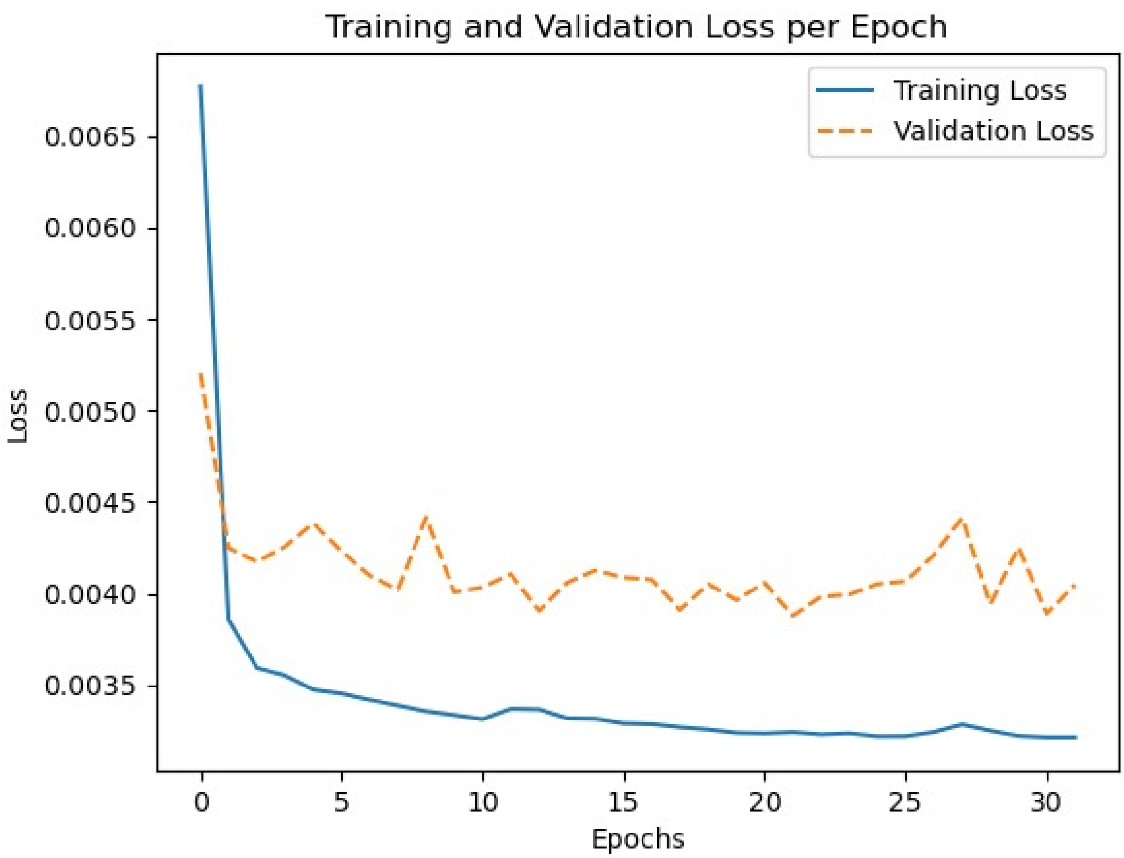
```python
# Make predictions
predictions = model.predict(X_test)
```

```
Epoch 1/200

c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

1179/1179 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - loss: 0.0124 - mae:
0.0674 - val_loss: 0.0052 - val_mae: 0.0434
Epoch 2/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0040 - mae:
0.0362 - val_loss: 0.0042 - val_mae: 0.0359
Epoch 3/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0035 - mae:
0.0320 - val_loss: 0.0042 - val_mae: 0.0359
Epoch 4/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0035 - mae:
0.0311 - val_loss: 0.0043 - val_mae: 0.0326
Epoch 5/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0034 - mae:
0.0305 - val_loss: 0.0044 - val_mae: 0.0367
Epoch 6/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0034 - mae:
0.0301 - val_loss: 0.0042 - val_mae: 0.0353
Epoch 7/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0033 - mae:
0.0291 - val_loss: 0.0041 - val_mae: 0.0321
Epoch 8/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 26s 22ms/step - loss: 0.0035 - mae:
0.0292 - val_loss: 0.0040 - val_mae: 0.0312
Epoch 9/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0033 - mae:
0.0282 - val_loss: 0.0044 - val_mae: 0.0327
Epoch 10/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0033 - mae:
0.0283 - val_loss: 0.0040 - val_mae: 0.0320
Epoch 11/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0033 - mae:
0.0272 - val_loss: 0.0040 - val_mae: 0.0311
Epoch 12/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0033 - mae:
0.0278 - val_loss: 0.0041 - val_mae: 0.0371
Epoch 13/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0034 - mae:
```

```
0.0295 - val_loss: 0.0039 - val_mae: 0.0307
Epoch 14/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0034 - mae:
0.0285 - val_loss: 0.0041 - val_mae: 0.0308
Epoch 15/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0032 - mae:
0.0275 - val_loss: 0.0041 - val_mae: 0.0351
Epoch 16/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0033 - mae:
0.0280 - val_loss: 0.0041 - val_mae: 0.0304
Epoch 17/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 28s 23ms/step - loss: 0.0032 - mae:
0.0271 - val_loss: 0.0041 - val_mae: 0.0301
Epoch 18/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 25s 21ms/step - loss: 0.0033 - mae:
0.0274 - val_loss: 0.0039 - val_mae: 0.0296
Epoch 19/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0033 - mae:
0.0276 - val_loss: 0.0041 - val_mae: 0.0303
Epoch 20/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0032 - mae:
0.0268 - val_loss: 0.0040 - val_mae: 0.0299
Epoch 21/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 19ms/step - loss: 0.0033 - mae:
0.0271 - val_loss: 0.0041 - val_mae: 0.0305
Epoch 22/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0032 - mae:
0.0266 - val_loss: 0.0039 - val_mae: 0.0290
Epoch 23/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 27s 23ms/step - loss: 0.0032 - mae:
0.0263 - val_loss: 0.0040 - val_mae: 0.0295
Epoch 24/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0031 - mae:
0.0262 - val_loss: 0.0040 - val_mae: 0.0301
Epoch 25/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 21ms/step - loss: 0.0031 - mae:
0.0264 - val_loss: 0.0041 - val_mae: 0.0304
Epoch 26/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0033 - mae:
0.0269 - val_loss: 0.0041 - val_mae: 0.0315
Epoch 27/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0032 - mae:
0.0266 - val_loss: 0.0042 - val_mae: 0.0312
Epoch 28/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 21ms/step - loss: 0.0033 - mae:
0.0276 - val_loss: 0.0044 - val_mae: 0.0332
Epoch 29/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 26s 22ms/step - loss: 0.0033 - mae:
0.0275 - val_loss: 0.0039 - val_mae: 0.0300
```

```
Epoch 30/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 26s 22ms/step - loss: 0.0032 - mae:
0.0266 - val_loss: 0.0043 - val_mae: 0.0326
Epoch 31/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 26s 22ms/step - loss: 0.0033 - mae:
0.0271 - val_loss: 0.0039 - val_mae: 0.0292
Epoch 32/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 19ms/step - loss: 0.0031 - mae:
0.0262 - val_loss: 0.0040 - val_mae: 0.0301
324/324 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0054 - mae: 0.0336

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

324/324 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
```

```python
import matplotlib.pyplot as plt

# Plotting training and validation loss per epoch
plt.plot(history.history['loss'], label='Training Loss',
linestyle='-')
plt.plot(history.history['val_loss'], label='Validation Loss',
linestyle='--')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss per Epoch')
plt.show()
```




```
y_pred = model.predict(X_test)
y_true = y_test

# Now can print y_true and y_pred
print("Actual values (y_true):", y_true)
print("Predicted values (y_pred):", y_pred)

324/324 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
Actual values (y_true): [0. 0. 0. ... 0. 0. 0.]
Predicted values (y_pred): [[0.00032096]
 [0.00047453]
 [0.00060094]
 ...
 [0.0008688 ]
 [0.00050082]
 [0.00017584]]

from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_true, y_pred)
print("Mean Squared Error (MSE):", mse)

Mean Squared Error (MSE): 0.004551565612015819
```

```python
import numpy as np
import pandas as pd

# Flatten the X_test array
X_test_flat = X_test.reshape(X_test.shape[0], -1)

# Convert y_test and y_pred to 1D arrays
y_test_flat = y_test.flatten()
y_pred_flat = y_pred.flatten()

# Convert X_test_flat to DataFrame
X_test_df = pd.DataFrame(X_test_flat, columns=[f'Feature_{i}' for i in range(X_test_flat.shape[1])])

# Create DataFrame for y_test and y_pred
y_test_df = pd.DataFrame({'Actual values (y_true)': y_test_flat})
y_pred_df = pd.DataFrame({'Predicted values (y_pred)': y_pred_flat})

# Concatenate X_test_df, y_test_df, and y_pred_df along columns
result_df = pd.concat([X_test_df, y_test_df, y_pred_df], axis=1)

# Print the result DataFrame
result_df.head(10)
```

```
   Feature_0  Feature_1  Feature_2  Feature_3  Feature_4  Feature_5  \
0        0.0        0.0        0.0        0.0        0.0        0.0
1        0.0        0.0        0.0        0.0        0.0        0.0
2        0.0        0.0        0.0        0.0        0.0        0.0
3        0.0        0.0        0.0        0.0        0.0        0.0
4        0.0        0.0        0.0        0.0        0.0        0.0
5        0.0        0.0        0.0        0.0        0.0        0.0
6        0.0        0.0        0.0        0.0        0.0        0.0
7        0.0        0.0        0.0        0.0        0.0        0.0
8        0.0        0.0        0.0        0.0        0.0        0.0
9        0.0        0.0        0.0        0.0        0.0        0.0

   Feature_6  Feature_7  Feature_8  Feature_9  ...  Feature_136
Feature_137  \
0        0.0        0.0        0.0        0.0  ...          0.0
0.0
1        0.0        0.0        0.0        0.0  ...          0.0
0.0
2        0.0        0.0        0.0        0.0  ...          0.0
0.0
3        0.0        0.0        0.0        0.0  ...          0.0
0.0
4        0.0        0.0        0.0        0.0  ...          0.0
0.0
5        0.0        0.0        0.0        0.0  ...          0.0
0.0
```

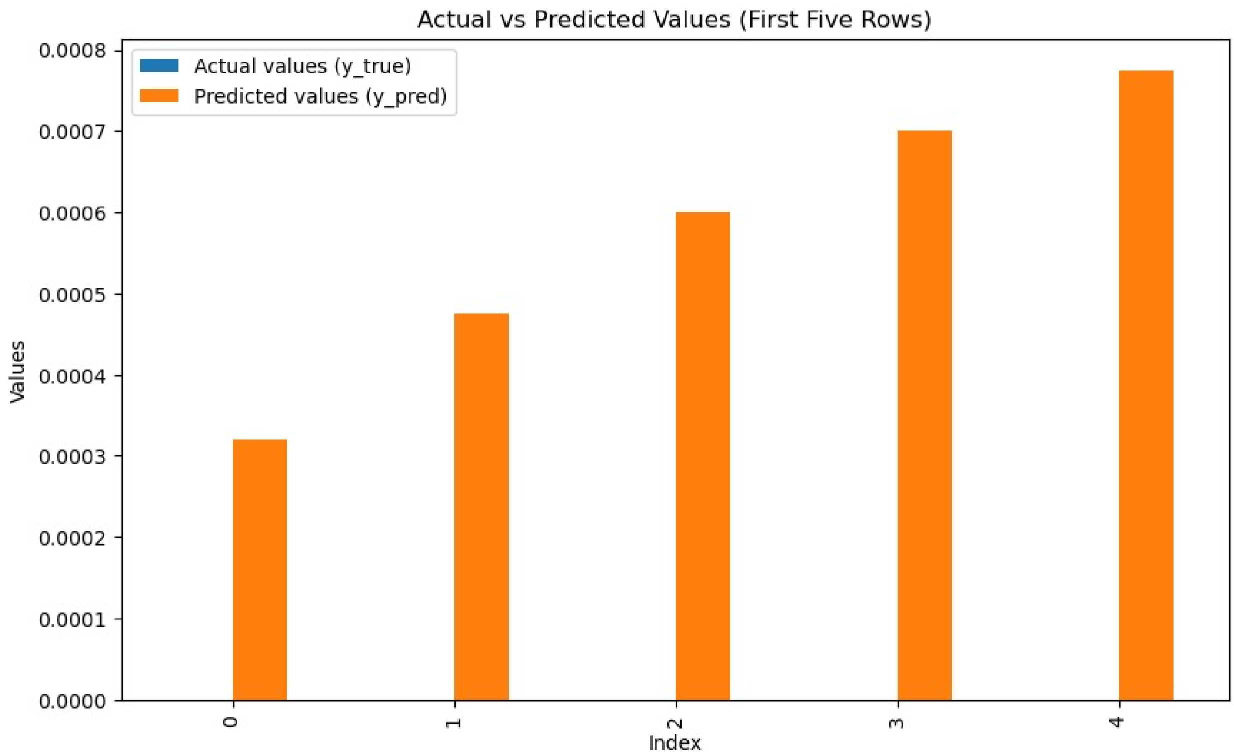
```
6        0.0        0.0        0.0        0.0  ...          0.0
0.0
7        0.0        0.0        0.0        0.0  ...          0.0
0.0
8        0.0        0.0        0.0        0.0  ...          0.0
0.0
9        0.0        0.0        0.0        0.0  ...          0.0
0.0

   Feature_138  Feature_139  Feature_140  Feature_141  Feature_142  \
0          0.0          0.0          0.0          0.0          0.0
1          0.0          0.0          0.0          0.0          0.0
2          0.0          0.0          0.0          0.0          0.0
3          0.0          0.0          0.0          0.0          0.0
4          0.0          0.0          0.0          0.0          0.0
5          0.0          0.0          0.0          0.0          0.0
6          0.0          0.0          0.0          0.0          0.0
7          0.0          0.0          0.0          0.0          0.0
8          0.0          0.0          0.0          0.0          0.0
9          0.0          0.0          0.0          0.0          0.0

   Feature_143  Actual values (y_true)  Predicted values (y_pred)
0          0.0                     0.0                   0.000321
1          0.0                     0.0                   0.000475
2          0.0                     0.0                   0.000601
3          0.0                     0.0                   0.000700
4          0.0                     0.0                   0.000775
5          0.0                     0.0                   0.000826
6          0.0                     0.0                   0.000859
7          0.0                     0.0                   0.000875
8          0.0                     0.0                   0.000880
9          0.0                     0.0                   0.000876

[10 rows x 146 columns]
```

```
result_df = result_df.iloc[:, -2:]
result_df.head()
```

```
   Actual values (y_true)  Predicted values (y_pred)
0                     0.0                   0.000321
1                     0.0                   0.000475
2                     0.0                   0.000601
3                     0.0                   0.000700
4                     0.0                   0.000775
```

```
import matplotlib.pyplot as plt

# Selecting only the first five rows
result_df_first_five = result_df.iloc[:5]
```

```python
# Plotting the actual vs predicted values for the first five rows
result_df_first_five.plot(kind='bar', figsize=(10, 6))
plt.title('Actual vs Predicted Values (First Five Rows)')
plt.xlabel('Index')
plt.ylabel('Values')
plt.show()
```

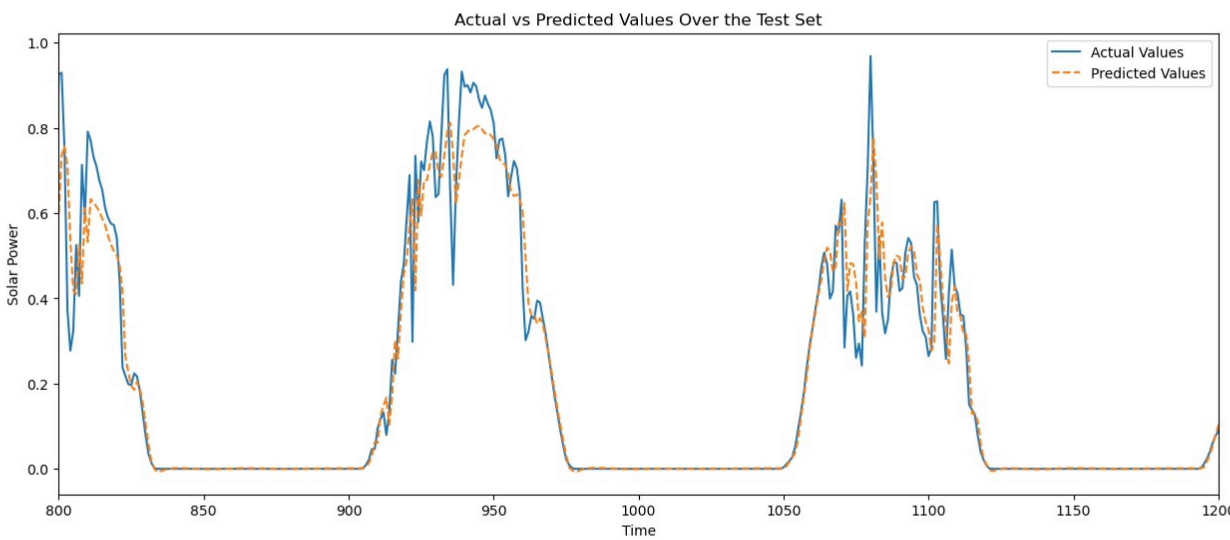



```python
from sklearn.metrics import mean_absolute_error, mean_squared_error

y_true = result_df['Actual values (y_true)']
y_pred = result_df['Predicted values (y_pred)']

mae = mean_absolute_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
rmse = mean_squared_error(y_true, y_pred, squared=False)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
```

```
Mean Absolute Error: 0.031088219474760287
Mean Squared Error: 0.004551565612015819
Root Mean Squared Error: 0.06746529190640042

c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\sklearn\
metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in
```

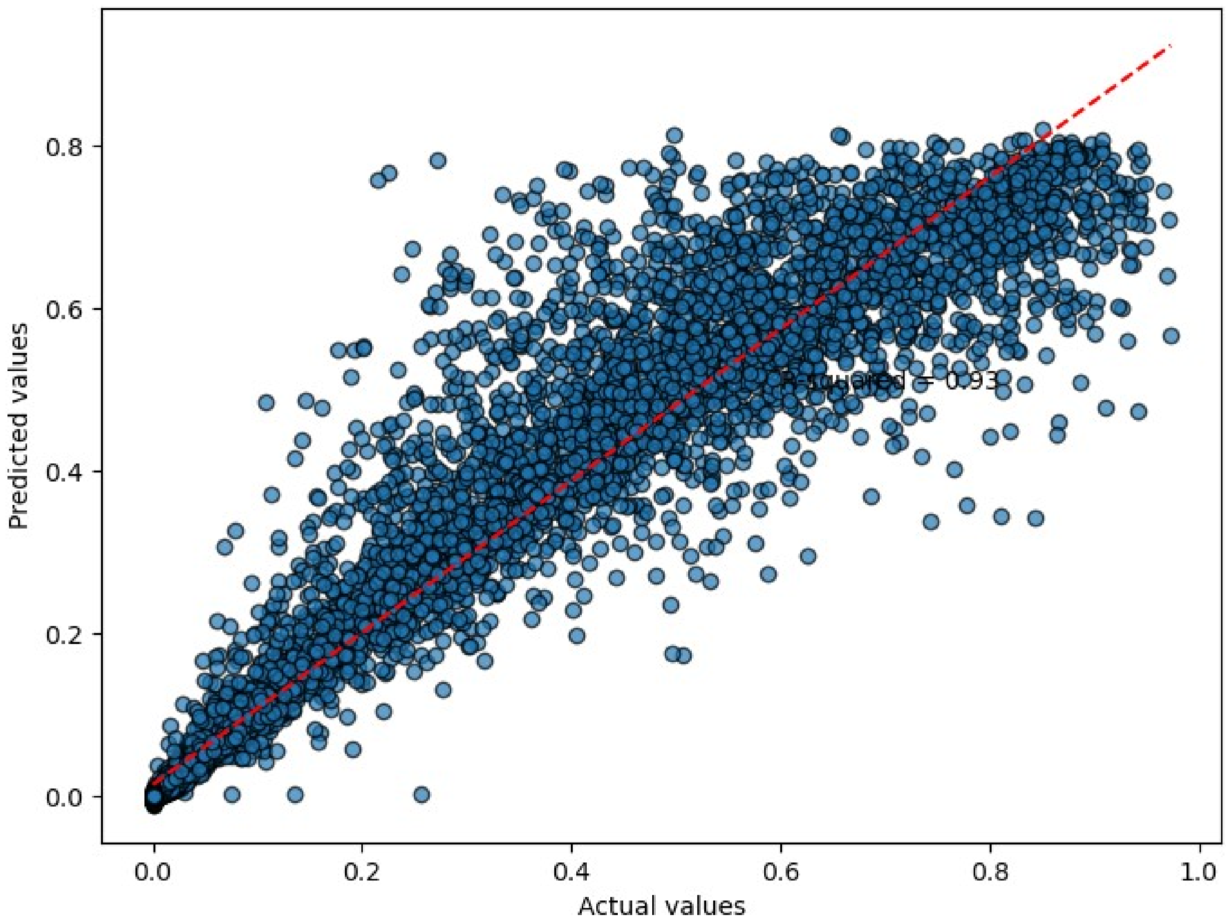
```
version 1.4 and will be removed in 1.6. To calculate the root mean
squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

```
import matplotlib.pyplot as plt

# Plotting the actual vs predicted values over the entire test set
plt.figure(figsize=(15, 6))
plt.plot(result_df['Actual values (y_true)'], label='Actual Values')
plt.plot(result_df['Predicted values (y_pred)'], label='Predicted
Values', linestyle='dashed')
plt.xlim(800,1200)
plt.title('Actual vs Predicted Values Over the Test Set')
plt.xlabel('Time')
plt.ylabel('Solar Power')
plt.legend()
plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score

r_squared = r2_score(y_true, y_pred)
plt.figure(figsize=(8, 6))
plt.scatter(y_true,y_pred, cmap='viridis', edgecolor='k', alpha=0.7)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')

plt.plot(np.unique(y_true), np.poly1d(np.polyfit(y_true, y_pred, 1))
(np.unique(y_true)), 'r--')

plt.text(0.6, 0.5, 'R-squared = %0.2f' % r_squared)
plt.show()
```

```
C:\Users\AbdullahHarithJamadi\AppData\Local\Temp\
ipykernel_12920\702279148.py:7: UserWarning: No data for colormapping
provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(y_true,y_pred, cmap='viridis', edgecolor='k', alpha=0.7)
```



MULTIVARIATE USING ASI GENERATION 2021 + WEATHER 2021 TO COMPARE WITH UNIVARIATE ASI GENERATION 2021

```
import pandas as pd
import numpy as np

environment2021 = pd.read_csv("Weather sensor - 2021.csv",
encoding='unicode_escape')
environment2021 = environment2021[['Time', 'W/m^2', '°C']]

environment2021["Time"] = pd.to_datetime(environment2021["Time"],
format='mixed')
environment2021.set_index("Time", inplace=True)

environment2021_resampled = environment2021.resample('10T').mean()
```

```
weatherdata2021 = environment2021_resampled[['W/m^2',
'°C']].reset_index()

weatherdata2021.rename(columns={'W/m^2': '10-min mean W/m^2', '°C':
'10-min mean °C'}, inplace=True)

print(weatherdata2021)
```

```
C:\Users\AbdullahHarithJamadi\AppData\Local\Temp\
ipykernel_12920\3430815374.py:4: DtypeWarning: Columns (3,5) have
mixed types. Specify dtype option on import or set low_memory=False.
  environment2021 = pd.read_csv("Weather sensor - 2021.csv",
encoding='unicode_escape')

                     Time  10-min mean W/m^2  10-min mean °C
0     2021-01-01 00:00:00                0.0          24.665
1     2021-01-01 00:10:00                0.0          24.520
2     2021-01-01 00:20:00                0.0          24.395
3     2021-01-01 00:30:00                0.0          24.350
4     2021-01-01 00:40:00                0.0          24.235
...                   ...                ...             ...
52555 2021-12-31 23:10:00                0.0          22.750
52556 2021-12-31 23:20:00                0.0          22.600
52557 2021-12-31 23:30:00                0.0          22.600
52558 2021-12-31 23:40:00                0.0          22.650
52559 2021-12-31 23:50:00                0.0          22.650

[52560 rows x 3 columns]

C:\Users\AbdullahHarithJamadi\AppData\Local\Temp\
ipykernel_12920\3430815374.py:10: FutureWarning: 'T' is deprecated and
will be removed in a future version, please use 'min' instead.
  environment2021_resampled = environment2021.resample('10T').mean()
```

```
weatherdata2021 = weatherdata2021.drop(columns = ['Time'])
combined_data2021 = pd.concat([data2021, weatherdata2021], axis=1,
join='inner')

print(combined_data2021)
```

```
                     Time  10-min mean Solar Power (MW)  10-min mean
W/m^2  \
0     2021-01-01 00:00:00                             0
0.0
1     2021-01-01 00:10:00                             0
0.0
2     2021-01-01 00:20:00                             0
0.0
3     2021-01-01 00:30:00                             0
0.0
4     2021-01-01 00:40:00                             0
```

```
0.0
...                    ...                           ...
...
52555 2021-12-31 23:10:00                              0
0.0
52556 2021-12-31 23:20:00                              0
0.0
52557 2021-12-31 23:30:00                              0
0.0
52558 2021-12-31 23:40:00                              0
0.0
52559 2021-12-31 23:50:00                              0
0.0

       10-min mean °C
0              24.665
1              24.520
2              24.395
3              24.350
4              24.235
...               ...
52555          22.750
52556          22.600
52557          22.600
52558          22.650
52559          22.650

[52560 rows x 4 columns]
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense
from keras.callbacks import EarlyStopping

# Select relevant columns
selected_columns = [
    '10-min mean Solar Power (MW)',
    '10-min mean W/m^2',
    '10-min mean °C'

]
data_selected = combined_data2021[selected_columns]
split_ratio = 0.8

train_size = int(len(data_selected) * split_ratio)

train_data = data_selected[:train_size]
test_data = data_selected[train_size:]
```

```python
# Normalize the data
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train_data)
test_scaled = scaler.transform(test_data)

# Prepare the data for LSTM
def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps)])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

time_steps = 144 # 6: Short Term, 12: Medium Term, 144: Daily cycle
X_train, y_train = create_dataset(train_scaled, train_scaled[:, 0],
time_steps)
X_test, y_test = create_dataset(test_scaled, test_scaled[:, 0],
time_steps)

# Define the LSTM model architecture
model = Sequential([
    LSTM(units=64, input_shape=(X_train.shape[1], X_train.shape[2])),
    Dense(units=1)
])
from keras.optimizers import Adam
model.compile(optimizer=Adam(learning_rate=0.0005), loss='mse',
metrics=['mae'])

# Define the EarlyStopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the LSTM model
history = model.fit(X_train, y_train, epochs=200, batch_size=32,
validation_split=0.1, verbose=1, callbacks=[early_stopping])

# Evaluate the model performance
model.evaluate(X_test, y_test)

model.save("lstm_multivariate_model.h5")

# Make predictions
predictions = model.predict(X_test)
```

```
Epoch 1/200
```

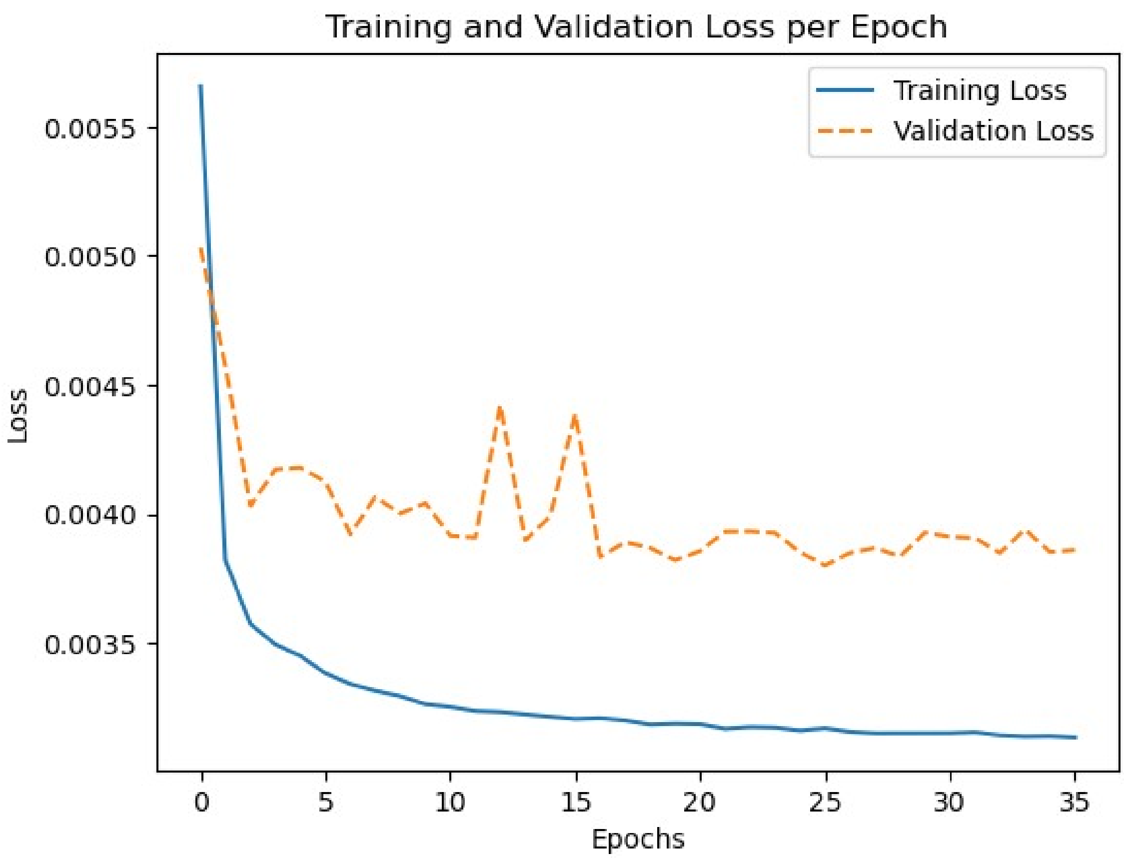
```
c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
```

```
the model instead.
  super().__init__(**kwargs)

1179/1179 ━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - loss: 0.0095 - mae: 
0.0571 - val_loss: 0.0050 - val_mae: 0.0403
Epoch 2/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 27s 23ms/step - loss: 0.0039 - mae: 
0.0339 - val_loss: 0.0046 - val_mae: 0.0361
Epoch 3/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 26s 22ms/step - loss: 0.0036 - mae: 
0.0307 - val_loss: 0.0040 - val_mae: 0.0321
Epoch 4/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 27s 23ms/step - loss: 0.0035 - mae: 
0.0298 - val_loss: 0.0042 - val_mae: 0.0346
Epoch 5/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 26s 22ms/step - loss: 0.0034 - mae: 
0.0294 - val_loss: 0.0042 - val_mae: 0.0318
Epoch 6/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 26s 22ms/step - loss: 0.0034 - mae: 
0.0284 - val_loss: 0.0041 - val_mae: 0.0343
Epoch 7/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 26s 22ms/step - loss: 0.0033 - mae: 
0.0286 - val_loss: 0.0039 - val_mae: 0.0316
Epoch 8/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 27s 23ms/step - loss: 0.0034 - mae: 
0.0283 - val_loss: 0.0041 - val_mae: 0.0317
Epoch 9/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 32s 27ms/step - loss: 0.0033 - mae: 
0.0278 - val_loss: 0.0040 - val_mae: 0.0325
Epoch 10/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0033 - mae: 
0.0277 - val_loss: 0.0040 - val_mae: 0.0309
Epoch 11/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0032 - mae: 
0.0272 - val_loss: 0.0039 - val_mae: 0.0296
Epoch 12/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0031 - mae: 
0.0265 - val_loss: 0.0039 - val_mae: 0.0306
Epoch 13/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0032 - mae: 
0.0269 - val_loss: 0.0044 - val_mae: 0.0331
Epoch 14/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0033 - mae: 
0.0271 - val_loss: 0.0039 - val_mae: 0.0289
Epoch 15/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0033 - mae: 
0.0273 - val_loss: 0.0040 - val_mae: 0.0314
Epoch 16/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0032 - mae: 
0.0265 - val_loss: 0.0044 - val_mae: 0.0337
```

```
Epoch 17/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0031 - mae:
0.0267 - val_loss: 0.0038 - val_mae: 0.0291
Epoch 18/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0032 - mae:
0.0267 - val_loss: 0.0039 - val_mae: 0.0307
Epoch 19/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0032 - mae:
0.0268 - val_loss: 0.0039 - val_mae: 0.0290
Epoch 20/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0032 - mae:
0.0266 - val_loss: 0.0038 - val_mae: 0.0290
Epoch 21/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 25s 21ms/step - loss: 0.0032 - mae:
0.0265 - val_loss: 0.0039 - val_mae: 0.0298
Epoch 22/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0032 - mae:
0.0267 - val_loss: 0.0039 - val_mae: 0.0311
Epoch 23/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0031 - mae:
0.0264 - val_loss: 0.0039 - val_mae: 0.0305
Epoch 24/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0031 - mae:
0.0266 - val_loss: 0.0039 - val_mae: 0.0297
Epoch 25/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 20ms/step - loss: 0.0032 - mae:
0.0265 - val_loss: 0.0039 - val_mae: 0.0288
Epoch 26/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 25s 21ms/step - loss: 0.0031 - mae:
0.0261 - val_loss: 0.0038 - val_mae: 0.0296
Epoch 27/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 21ms/step - loss: 0.0032 - mae:
0.0269 - val_loss: 0.0039 - val_mae: 0.0303
Epoch 28/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 25s 21ms/step - loss: 0.0031 - mae:
0.0263 - val_loss: 0.0039 - val_mae: 0.0292
Epoch 29/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 27s 23ms/step - loss: 0.0031 - mae:
0.0259 - val_loss: 0.0038 - val_mae: 0.0297
Epoch 30/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 28s 24ms/step - loss: 0.0032 - mae:
0.0263 - val_loss: 0.0039 - val_mae: 0.0302
Epoch 31/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 27s 23ms/step - loss: 0.0032 - mae:
0.0263 - val_loss: 0.0039 - val_mae: 0.0291
Epoch 32/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 27s 23ms/step - loss: 0.0031 - mae:
0.0262 - val_loss: 0.0039 - val_mae: 0.0303
Epoch 33/200
```

```
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 25s 21ms/step - loss: 0.0032 - mae:
0.0265 - val_loss: 0.0039 - val_mae: 0.0289
Epoch 34/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 24s 21ms/step - loss: 0.0031 - mae:
0.0263 - val_loss: 0.0039 - val_mae: 0.0296
Epoch 35/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0031 - mae:
0.0263 - val_loss: 0.0039 - val_mae: 0.0289
Epoch 36/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 23s 20ms/step - loss: 0.0031 - mae:
0.0261 - val_loss: 0.0039 - val_mae: 0.0299
324/324 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0052 - mae: 0.0343

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

324/324 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
```

```
import matplotlib.pyplot as plt

# Plotting training and validation loss per epoch
plt.plot(history.history['loss'], label='Training Loss',
linestyle='-')
plt.plot(history.history['val_loss'], label='Validation Loss',
linestyle='--')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss per Epoch')
plt.show()
```




```
y_pred = model.predict(X_test)
y_true = y_test

# Now can print y_true and y_pred
print("Actual values (y_true):", y_true)
print("Predicted values (y_pred):", y_pred)
```

```
324/324 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
Actual values (y_true): [0. 0. 0. ... 0. 0. 0.]
Predicted values (y_pred): [[-0.00425411]
 [-0.00413324]
 [-0.00382844]
 ...
 [-0.00433243]
 [-0.00482577]
 [-0.00502641]]
```

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_true, y_pred)
print("Mean Squared Error (MSE):", mse)
```

```
Mean Squared Error (MSE): 0.0045079275010198585
```

```python
import numpy as np
import pandas as pd

# Flatten the X_test array
X_test_flat = X_test.reshape(X_test.shape[0], -1)

# Convert y_test and y_pred to 1D arrays
y_test_flat = y_test.flatten()
y_pred_flat = y_pred.flatten()

# Convert X_test_flat to DataFrame
X_test_df = pd.DataFrame(X_test_flat, columns=[f'Feature_{i}' for i in
range(X_test_flat.shape[1])])

# Create DataFrame for y_test and y_pred
y_test_df = pd.DataFrame({'Actual values (y_true)': y_test_flat})
y_pred_df = pd.DataFrame({'Predicted values (y_pred)': y_pred_flat})

# Concatenate X_test_df, y_test_df, and y_pred_df along columns
result_df = pd.concat([X_test_df, y_test_df, y_pred_df], axis=1)

# Print the result DataFrame
result_df.head(10)
```

```
   Feature_0  Feature_1  Feature_2  Feature_3  Feature_4  Feature_5  \
0        0.0        0.0   0.153933        0.0        0.0   0.153933
1        0.0        0.0   0.153933        0.0        0.0   0.154878
2        0.0        0.0   0.154878        0.0        0.0   0.156766
3        0.0        0.0   0.156766        0.0        0.0   0.156766
4        0.0        0.0   0.156766        0.0        0.0   0.154878
5        0.0        0.0   0.154878        0.0        0.0   0.154878
6        0.0        0.0   0.154878        0.0        0.0   0.154878
7        0.0        0.0   0.154878        0.0        0.0   0.156766
8        0.0        0.0   0.156766        0.0        0.0   0.159600
9        0.0        0.0   0.159600        0.0        0.0   0.160544

   Feature_6  Feature_7  Feature_8  Feature_9  ...  Feature_424
Feature_425  \
0        0.0        0.0   0.154878        0.0  ...          0.0
0.138823
1        0.0        0.0   0.156766        0.0  ...          0.0
0.137879
2        0.0        0.0   0.156766        0.0  ...          0.0
0.137879
3        0.0        0.0   0.154878        0.0  ...          0.0
0.137879
4        0.0        0.0   0.154878        0.0  ...          0.0
0.135990
5        0.0        0.0   0.154878        0.0  ...          0.0
0.134101
```

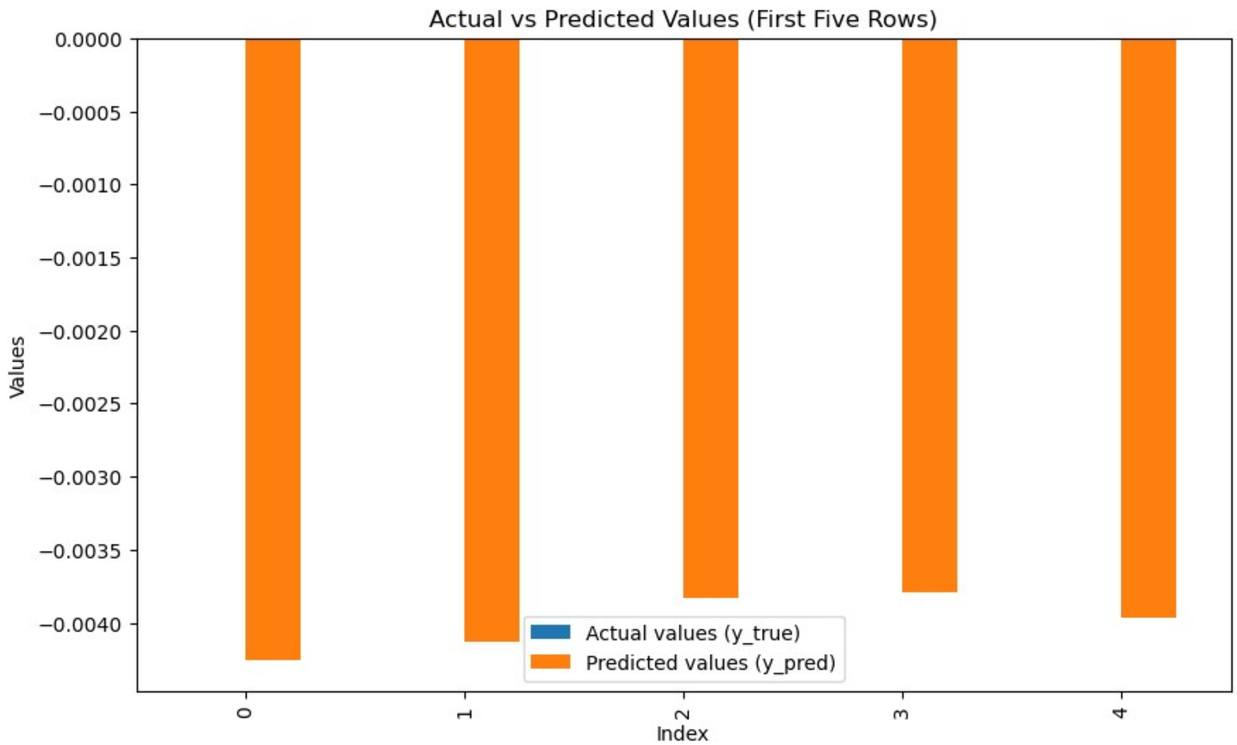
```
6        0.0         0.0   0.156766        0.0  ...          0.0
0.134101
7        0.0         0.0   0.159600        0.0  ...          0.0
0.134101
8        0.0         0.0   0.160544        0.0  ...          0.0
0.133157
9        0.0         0.0   0.159600        0.0  ...          0.0
0.132213

   Feature_426  Feature_427  Feature_428  Feature_429  Feature_430  \
0          0.0          0.0     0.137879          0.0          0.0
1          0.0          0.0     0.137879          0.0          0.0
2          0.0          0.0     0.137879          0.0          0.0
3          0.0          0.0     0.135990          0.0          0.0
4          0.0          0.0     0.134101          0.0          0.0
5          0.0          0.0     0.134101          0.0          0.0
6          0.0          0.0     0.134101          0.0          0.0
7          0.0          0.0     0.133157          0.0          0.0
8          0.0          0.0     0.132213          0.0          0.0
9          0.0          0.0     0.132213          0.0          0.0

   Feature_431  Actual values (y_true)  Predicted values (y_pred)
0     0.137879                     0.0                  -0.004254
1     0.137879                     0.0                  -0.004133
2     0.135990                     0.0                  -0.003828
3     0.134101                     0.0                  -0.003793
4     0.134101                     0.0                  -0.003970
5     0.134101                     0.0                  -0.003901
6     0.133157                     0.0                  -0.003706
7     0.132213                     0.0                  -0.003624
8     0.132213                     0.0                  -0.003653
9     0.132213                     0.0                  -0.003572

[10 rows x 434 columns]
```

```
result_df = result_df.iloc[:, -2:]
result_df.head()
```

```
   Actual values (y_true)  Predicted values (y_pred)
0                     0.0                  -0.004254
1                     0.0                  -0.004133
2                     0.0                  -0.003828
3                     0.0                  -0.003793
4                     0.0                  -0.003970
```

```
import matplotlib.pyplot as plt

# Selecting only the first five rows
result_df_first_five = result_df.iloc[:5]
```

```python
# Plotting the actual vs predicted values for the first five rows
result_df_first_five.plot(kind='bar', figsize=(10, 6))
plt.title('Actual vs Predicted Values (First Five Rows)')
plt.xlabel('Index')
plt.ylabel('Values')
plt.show()
```



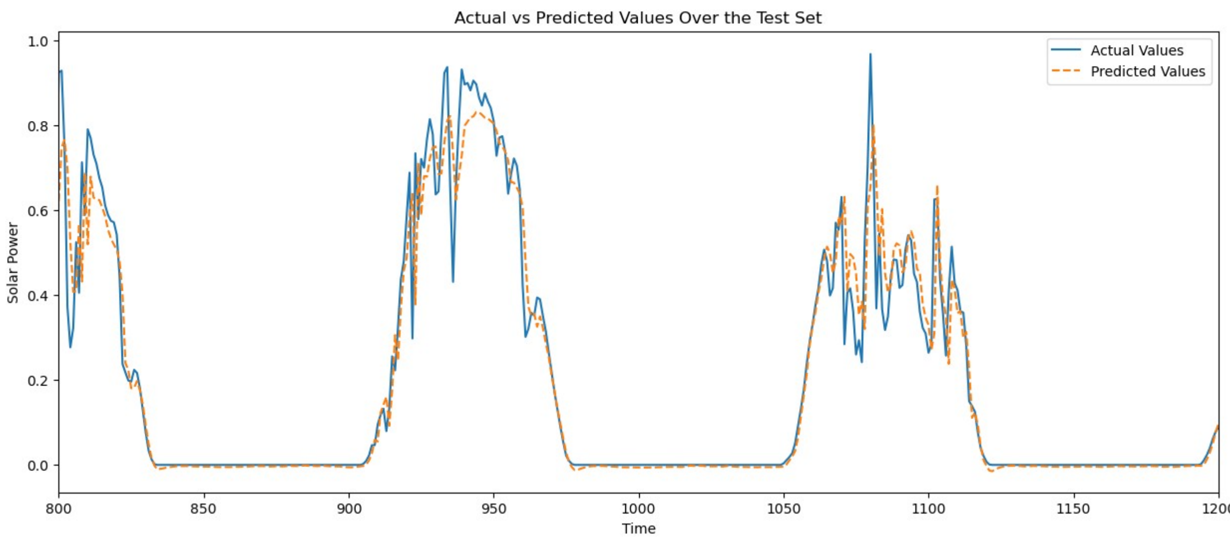

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error

y_true = result_df['Actual values (y_true)']
y_pred = result_df['Predicted values (y_pred)']

mae = mean_absolute_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
rmse = mean_squared_error(y_true, y_pred, squared=False)

print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
```

```
Mean Absolute Error: 0.03206214443827581
Mean Squared Error: 0.0045079275010198585
Root Mean Squared Error: 0.06714110142840865
```

```
c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\sklearn\
metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in
```

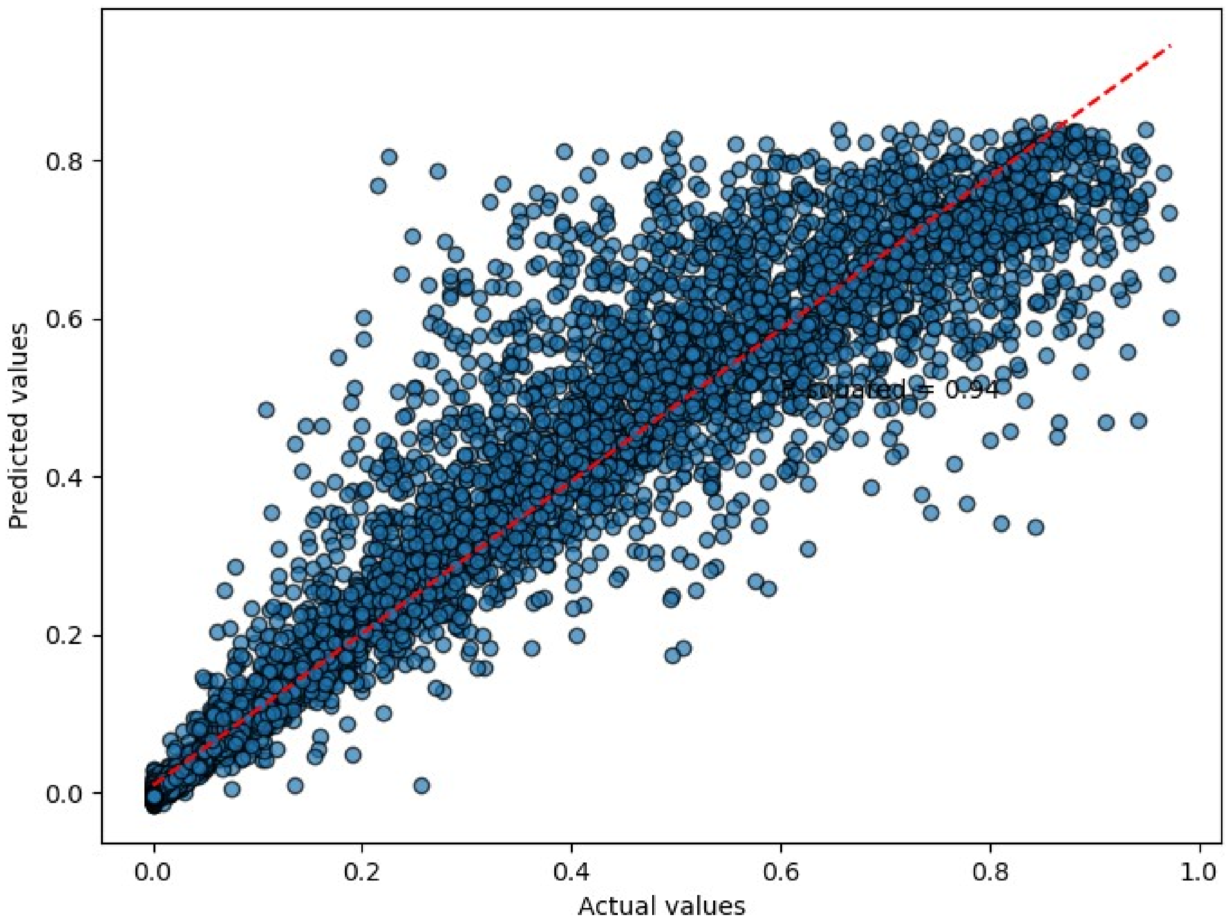
```
version 1.4 and will be removed in 1.6. To calculate the root mean
squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

```
import matplotlib.pyplot as plt

# Plotting the actual vs predicted values over the entire test set
plt.figure(figsize=(15, 6))
plt.plot(result_df['Actual values (y_true)'], label='Actual Values')
plt.plot(result_df['Predicted values (y_pred)'], label='Predicted
Values', linestyle='dashed')
plt.xlim(800,1200)
plt.title('Actual vs Predicted Values Over the Test Set')
plt.xlabel('Time')
plt.ylabel('Solar Power')
plt.legend()
plt.show()
```




```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score

r_squared = r2_score(y_true, y_pred)
plt.figure(figsize=(8, 6))
plt.scatter(y_true,y_pred, cmap='viridis', edgecolor='k', alpha=0.7)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')

plt.plot(np.unique(y_true), np.poly1d(np.polyfit(y_true, y_pred, 1))
(np.unique(y_true)), 'r--')

plt.text(0.6, 0.5, 'R-squared = %0.2f' % r_squared)
plt.show()
```

```
C:\Users\AbdullahHarithJamadi\AppData\Local\Temp\
ipykernel_12920\702279148.py:7: UserWarning: No data for colormapping
provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(y_true,y_pred, cmap='viridis', edgecolor='k', alpha=0.7)
```




```python
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam

selected_columns = [
    '10-min mean Solar Power (MW)'
]
data_selected = data[selected_columns]
split_ratio = 0.8
```

```
train_size = int(len(data_selected) * split_ratio)

train_data = data_selected[:train_size]
test_data = data_selected[train_size:]

scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train_data)
test_scaled = scaler.transform(test_data)

def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps)])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

time_horizons = {
    '30_min': 3,
    '1_hour': 6,
    '1_day': 144
}

results = {}

for horizon, time_steps in time_horizons.items():
    print(f"\nTesting for {horizon} horizon...")

    X_train, y_train = create_dataset(train_scaled, train_scaled[:,
0], time_steps)
    X_test, y_test = create_dataset(test_scaled, test_scaled[:, 0],
time_steps)

    model = Sequential([
        LSTM(units=64, input_shape=(X_train.shape[1],
X_train.shape[2])),
        # Dropout(0.2),
        # LSTM(units=64),
        # Dropout(0.2),
        Dense(units=1)
    ])

    model.compile(optimizer=Adam(learning_rate=0.0005), loss='mse',
metrics=['mae'])

    early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

    history = model.fit(X_train, y_train, epochs=200, batch_size=32,
validation_split=0.1, verbose=1, callbacks=[early_stopping])
```

```
    test_loss, test_mae = model.evaluate(X_test, y_test)
    print(f"Test Loss: {test_loss}, Test MAE: {test_mae}")

    predictions = model.predict(X_test)

    predictions_rescaled = scaler.inverse_transform(predictions)
    y_test_rescaled = scaler.inverse_transform(y_test.reshape(-1, 1))

    rmse = np.sqrt(mean_squared_error(y_test_rescaled,
predictions_rescaled))
    r2 = r2_score(y_test_rescaled, predictions_rescaled)

    print(f"RMSE for {horizon}: {rmse}")
    print(f"R² for {horizon}: {r2}")

    results[horizon] = {
        'MAE': test_mae,
        'MSE': test_loss,
        'RMSE': rmse,
        'R²': r2
    }

    results_df = pd.DataFrame({
        'Predictions': predictions_rescaled.flatten(),
        'Actual': y_test_rescaled.flatten()
    })

results_df.to_csv(f"univariate_predictions_vs_actual_{horizon}.csv",
index=False)

print("\nSummary of Results:")
for horizon, metrics in results.items():
    print(f"\n{horizon}:")
    for metric, value in metrics.items():
        print(f"{metric}: {value}")
```

```
c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)


Testing for 30_min horizon...
Epoch 1/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0114 - mae:
0.0588 - val_loss: 0.0042 - val_mae: 0.0318
Epoch 2/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0047 - mae:
0.0349 - val_loss: 0.0041 - val_mae: 0.0295
```

```
Epoch 3/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0045 - mae:
0.0330 - val_loss: 0.0040 - val_mae: 0.0285
Epoch 4/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0044 - mae:
0.0317 - val_loss: 0.0040 - val_mae: 0.0306
Epoch 5/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0044 - mae:
0.0313 - val_loss: 0.0040 - val_mae: 0.0308
Epoch 6/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0044 - mae:
0.0312 - val_loss: 0.0040 - val_mae: 0.0291
Epoch 7/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0044 - mae:
0.0313 - val_loss: 0.0040 - val_mae: 0.0302
Epoch 8/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0309 - val_loss: 0.0040 - val_mae: 0.0301
Epoch 9/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0310 - val_loss: 0.0040 - val_mae: 0.0288
Epoch 10/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0270
Epoch 11/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0044 - mae:
0.0310 - val_loss: 0.0039 - val_mae: 0.0286
Epoch 12/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0309 - val_loss: 0.0039 - val_mae: 0.0280
Epoch 13/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0309 - val_loss: 0.0040 - val_mae: 0.0273
Epoch 14/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0310 - val_loss: 0.0039 - val_mae: 0.0270
Epoch 15/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0306 - val_loss: 0.0039 - val_mae: 0.0274
Epoch 16/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0042 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0283
Epoch 17/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0306 - val_loss: 0.0041 - val_mae: 0.0293
Epoch 18/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - loss: 0.0043 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0273
Epoch 19/200
```

```
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - loss: 0.0043 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0271
Epoch 20/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - loss: 0.0044 - mae:
0.0309 - val_loss: 0.0039 - val_mae: 0.0287
Epoch 21/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0042 - mae:
0.0303 - val_loss: 0.0039 - val_mae: 0.0291
Epoch 22/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0044 - mae:
0.0309 - val_loss: 0.0039 - val_mae: 0.0279
Epoch 23/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0307 - val_loss: 0.0039 - val_mae: 0.0271
Epoch 24/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0307 - val_loss: 0.0039 - val_mae: 0.0273
Epoch 25/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0307 - val_loss: 0.0040 - val_mae: 0.0282
Epoch 26/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0306 - val_loss: 0.0039 - val_mae: 0.0287
Epoch 27/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0303 - val_loss: 0.0039 - val_mae: 0.0280
Epoch 28/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0044 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0272
Epoch 29/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0276
Epoch 30/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0275
Epoch 31/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0042 - mae:
0.0300 - val_loss: 0.0039 - val_mae: 0.0280
Epoch 32/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0291
Epoch 33/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0305 - val_loss: 0.0040 - val_mae: 0.0283
Epoch 34/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0304 - val_loss: 0.0040 - val_mae: 0.0289
Epoch 35/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0042 - mae:
```

```
0.0302 - val_loss: 0.0040 - val_mae: 0.0288
Epoch 36/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0044 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0279
Epoch 37/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0305 - val_loss: 0.0039 - val_mae: 0.0282
Epoch 38/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0304 - val_loss: 0.0040 - val_mae: 0.0277
Epoch 39/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0302 - val_loss: 0.0039 - val_mae: 0.0276
Epoch 40/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0042 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0270
Epoch 41/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0284
Epoch 42/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0305 - val_loss: 0.0039 - val_mae: 0.0274
Epoch 43/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0306 - val_loss: 0.0039 - val_mae: 0.0284
Epoch 44/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0306 - val_loss: 0.0039 - val_mae: 0.0272
Epoch 45/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0302 - val_loss: 0.0040 - val_mae: 0.0278
Epoch 46/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - loss: 0.0043 - mae:
0.0305 - val_loss: 0.0039 - val_mae: 0.0271
Epoch 47/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - loss: 0.0042 - mae:
0.0300 - val_loss: 0.0039 - val_mae: 0.0277
Epoch 48/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0280
Epoch 49/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - loss: 0.0042 - mae:
0.0302 - val_loss: 0.0039 - val_mae: 0.0277
Epoch 50/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step - loss: 0.0043 - mae:
0.0303 - val_loss: 0.0039 - val_mae: 0.0286
876/876 ━━━━━━━━━━━━━━━━━━━━ 1s 661us/step - loss: 0.0054 - mae:
0.0344
Test Loss: 0.005200345069169998, Test MAE: 0.03332371637225151
```

```
876/876 ━━━━━━━━━━━━━━━━━━━━ 1s 758us/step
RMSE for 30_min: 2011.1009025317298
R² for 30_min: 0.9290627894070121

Testing for 1_hour horizon...
Epoch 1/200

c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

3153/3153 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - loss: 0.0123 - mae:
0.0614 - val_loss: 0.0043 - val_mae: 0.0350
Epoch 2/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0048 - mae:
0.0349 - val_loss: 0.0042 - val_mae: 0.0300
Epoch 3/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0046 - mae:
0.0334 - val_loss: 0.0040 - val_mae: 0.0285
Epoch 4/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0044 - mae:
0.0316 - val_loss: 0.0039 - val_mae: 0.0273
Epoch 5/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0310 - val_loss: 0.0039 - val_mae: 0.0288
Epoch 6/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0312 - val_loss: 0.0039 - val_mae: 0.0284
Epoch 7/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0314 - val_loss: 0.0039 - val_mae: 0.0283
Epoch 8/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0312 - val_loss: 0.0040 - val_mae: 0.0309
Epoch 9/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0311 - val_loss: 0.0039 - val_mae: 0.0277
Epoch 10/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0044 - mae:
0.0310 - val_loss: 0.0039 - val_mae: 0.0272
Epoch 11/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0310 - val_loss: 0.0039 - val_mae: 0.0278
Epoch 12/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0309 - val_loss: 0.0039 - val_mae: 0.0292
Epoch 13/200
```

```
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0287
Epoch 14/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0308 - val_loss: 0.0040 - val_mae: 0.0294
Epoch 15/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0276
Epoch 16/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0272
Epoch 17/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0306 - val_loss: 0.0039 - val_mae: 0.0276
Epoch 18/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0306 - val_loss: 0.0039 - val_mae: 0.0285
Epoch 19/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0306 - val_loss: 0.0039 - val_mae: 0.0287
Epoch 20/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0308 - val_loss: 0.0039 - val_mae: 0.0298
Epoch 21/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0042 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0282
Epoch 22/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0041 - mae:
0.0300 - val_loss: 0.0039 - val_mae: 0.0294
Epoch 23/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0305 - val_loss: 0.0039 - val_mae: 0.0271
Epoch 24/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0304 - val_loss: 0.0038 - val_mae: 0.0279
Epoch 25/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0305 - val_loss: 0.0039 - val_mae: 0.0270
Epoch 26/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0042 - mae:
0.0304 - val_loss: 0.0039 - val_mae: 0.0289
Epoch 27/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0305 - val_loss: 0.0039 - val_mae: 0.0277
Epoch 28/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0042 - mae:
0.0301 - val_loss: 0.0038 - val_mae: 0.0266
Epoch 29/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
```

```
0.0306 - val_loss: 0.0038 - val_mae: 0.0271
Epoch 30/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0044 - mae:
0.0306 - val_loss: 0.0039 - val_mae: 0.0275
Epoch 31/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0305 - val_loss: 0.0039 - val_mae: 0.0280
Epoch 32/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0043 - mae:
0.0302 - val_loss: 0.0038 - val_mae: 0.0284
Epoch 33/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 0.0042 - mae:
0.0300 - val_loss: 0.0038 - val_mae: 0.0279
Epoch 34/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0300 - val_loss: 0.0039 - val_mae: 0.0281
Epoch 35/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0298 - val_loss: 0.0038 - val_mae: 0.0269
Epoch 36/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0303 - val_loss: 0.0038 - val_mae: 0.0284
Epoch 37/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0299 - val_loss: 0.0038 - val_mae: 0.0270
Epoch 38/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0299 - val_loss: 0.0039 - val_mae: 0.0288
Epoch 39/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0304 - val_loss: 0.0038 - val_mae: 0.0269
Epoch 40/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0301 - val_loss: 0.0038 - val_mae: 0.0273
Epoch 41/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0298 - val_loss: 0.0039 - val_mae: 0.0278
Epoch 42/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0298 - val_loss: 0.0038 - val_mae: 0.0271
Epoch 43/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0297 - val_loss: 0.0038 - val_mae: 0.0279
Epoch 44/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0043 - mae:
0.0301 - val_loss: 0.0039 - val_mae: 0.0283
Epoch 45/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0300 - val_loss: 0.0039 - val_mae: 0.0274
```

```
Epoch 46/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0042 - mae:
0.0300 - val_loss: 0.0038 - val_mae: 0.0280
Epoch 47/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0296 - val_loss: 0.0038 - val_mae: 0.0267
Epoch 48/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 4s 1ms/step - loss: 0.0042 - mae:
0.0299 - val_loss: 0.0038 - val_mae: 0.0270
Epoch 49/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0041 - mae:
0.0296 - val_loss: 0.0039 - val_mae: 0.0294
Epoch 50/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0299 - val_loss: 0.0038 - val_mae: 0.0267
Epoch 51/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0041 - mae:
0.0295 - val_loss: 0.0038 - val_mae: 0.0272
Epoch 52/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0299 - val_loss: 0.0038 - val_mae: 0.0290
Epoch 53/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0300 - val_loss: 0.0038 - val_mae: 0.0269
Epoch 54/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0297 - val_loss: 0.0038 - val_mae: 0.0274
Epoch 55/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0296 - val_loss: 0.0038 - val_mae: 0.0277
Epoch 56/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0296 - val_loss: 0.0039 - val_mae: 0.0275
Epoch 57/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0295 - val_loss: 0.0038 - val_mae: 0.0284
Epoch 58/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0043 - mae:
0.0299 - val_loss: 0.0038 - val_mae: 0.0265
Epoch 59/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0296 - val_loss: 0.0038 - val_mae: 0.0283
Epoch 60/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0297 - val_loss: 0.0038 - val_mae: 0.0268
Epoch 61/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0041 - mae:
0.0293 - val_loss: 0.0038 - val_mae: 0.0266
Epoch 62/200
```

```
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0041 - mae:
0.0294 - val_loss: 0.0039 - val_mae: 0.0269
Epoch 63/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0297 - val_loss: 0.0039 - val_mae: 0.0276
Epoch 64/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0293 - val_loss: 0.0038 - val_mae: 0.0276
Epoch 65/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0296 - val_loss: 0.0038 - val_mae: 0.0272
Epoch 66/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0295 - val_loss: 0.0039 - val_mae: 0.0272
Epoch 67/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0041 - mae:
0.0293 - val_loss: 0.0038 - val_mae: 0.0266
Epoch 68/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0298 - val_loss: 0.0039 - val_mae: 0.0268
Epoch 69/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0041 - mae:
0.0293 - val_loss: 0.0038 - val_mae: 0.0276
Epoch 70/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0041 - mae:
0.0293 - val_loss: 0.0038 - val_mae: 0.0271
Epoch 71/200
3153/3153 ━━━━━━━━━━━━━━━━━━━━ 5s 1ms/step - loss: 0.0042 - mae:
0.0295 - val_loss: 0.0038 - val_mae: 0.0269
876/876 ━━━━━━━━━━━━━━━━━━━━ 1s 894us/step - loss: 0.0053 - mae:
0.0335
Test Loss: 0.0050838724710047245, Test MAE: 0.03253071755170822
876/876 ━━━━━━━━━━━━━━━━━━━━ 1s 900us/step
RMSE for 1_hour: 1988.4510379490528
R² for 1_hour: 0.9306442033115193

Testing for 1_day horizon...
Epoch 1/200

c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

3149/3149 ━━━━━━━━━━━━━━━━━━━━ 74s 23ms/step - loss: 0.0108 - mae:
0.0590 - val_loss: 0.0041 - val_mae: 0.0318
Epoch 2/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0045 - mae:
```

```
0.0354 - val_loss: 0.0039 - val_mae: 0.0294
Epoch 3/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0042 - mae:
0.0322 - val_loss: 0.0040 - val_mae: 0.0314
Epoch 4/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0041 - mae:
0.0312 - val_loss: 0.0039 - val_mae: 0.0308
Epoch 5/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0040 - mae:
0.0300 - val_loss: 0.0038 - val_mae: 0.0282
Epoch 6/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0041 - mae:
0.0299 - val_loss: 0.0037 - val_mae: 0.0290
Epoch 7/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 65s 21ms/step - loss: 0.0041 - mae:
0.0302 - val_loss: 0.0036 - val_mae: 0.0262
Epoch 8/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 66s 21ms/step - loss: 0.0040 - mae:
0.0296 - val_loss: 0.0039 - val_mae: 0.0300
Epoch 9/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 65s 21ms/step - loss: 0.0042 - mae:
0.0304 - val_loss: 0.0037 - val_mae: 0.0277
Epoch 10/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0040 - mae:
0.0296 - val_loss: 0.0036 - val_mae: 0.0266
Epoch 11/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0039 - mae:
0.0289 - val_loss: 0.0036 - val_mae: 0.0269
Epoch 12/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0040 - mae:
0.0291 - val_loss: 0.0036 - val_mae: 0.0267
Epoch 13/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0040 - mae:
0.0295 - val_loss: 0.0037 - val_mae: 0.0265
Epoch 14/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0040 - mae:
0.0293 - val_loss: 0.0038 - val_mae: 0.0281
Epoch 15/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 67s 21ms/step - loss: 0.0040 - mae:
0.0293 - val_loss: 0.0036 - val_mae: 0.0263
Epoch 16/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 66s 21ms/step - loss: 0.0040 - mae:
0.0291 - val_loss: 0.0037 - val_mae: 0.0267
Epoch 17/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 66s 21ms/step - loss: 0.0040 - mae:
0.0290 - val_loss: 0.0036 - val_mae: 0.0273
Epoch 18/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 66s 21ms/step - loss: 0.0039 - mae:
0.0289 - val_loss: 0.0036 - val_mae: 0.0267
```

```
Epoch 19/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 66s 21ms/step - loss: 0.0039 - mae:
0.0289 - val_loss: 0.0038 - val_mae: 0.0273
Epoch 20/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 67s 21ms/step - loss: 0.0040 - mae:
0.0295 - val_loss: 0.0036 - val_mae: 0.0260
Epoch 21/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 66s 21ms/step - loss: 0.0040 - mae:
0.0293 - val_loss: 0.0036 - val_mae: 0.0270
Epoch 22/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 62s 20ms/step - loss: 0.0040 - mae:
0.0288 - val_loss: 0.0036 - val_mae: 0.0269
Epoch 23/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0039 - mae:
0.0285 - val_loss: 0.0036 - val_mae: 0.0266
Epoch 24/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 63s 20ms/step - loss: 0.0038 - mae:
0.0285 - val_loss: 0.0036 - val_mae: 0.0261
Epoch 25/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 65s 21ms/step - loss: 0.0040 - mae:
0.0290 - val_loss: 0.0036 - val_mae: 0.0263
Epoch 26/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 76s 24ms/step - loss: 0.0039 - mae:
0.0286 - val_loss: 0.0036 - val_mae: 0.0267
Epoch 27/200
3149/3149 ━━━━━━━━━━━━━━━━━━━━ 70s 22ms/step - loss: 0.0040 - mae:
0.0288 - val_loss: 0.0036 - val_mae: 0.0259
872/872 ━━━━━━━━━━━━━━━━━━━━ 7s 7ms/step - loss: 0.0049 - mae: 0.0340
Test Loss: 0.00479432987049222, Test MAE: 0.033309586346149445
872/872 ━━━━━━━━━━━━━━━━━━━━ 7s 7ms/step
RMSE for 1_day: 1930.9955007127676
R² for 1_day: 0.9344705360646872

Summary of Results:

30_min:
MAE: 0.03332371637225151
MSE: 0.005200345069169998
RMSE: 2011.1009025317298
R²: 0.9290627894070121

1_hour:
MAE: 0.03253071755170822
MSE: 0.005083872471004724 5
RMSE: 1988.4510379490528
R²: 0.9306442033115193

1_day:
MAE: 0.033309586346149445
MSE: 0.00479432987049222
```

```
RMSE: 1930.9955007127676
R²: 0.9344705360646872
```

```python
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam

selected_columns = [
    '10-min mean Solar Power (MW)',
    '10-min mean W/m^2',
    '10-min mean °C'
]
data_selected = combined_data2021[selected_columns]
split_ratio = 0.8

train_size = int(len(data_selected) * split_ratio)

train_data = data_selected[:train_size]
test_data = data_selected[train_size:]

scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train_data)
test_scaled = scaler.transform(test_data)

def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps)])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)

time_horizons = {
    '30_min': 3,
    '1_hour': 6,
    '1_day': 144
}

results = {}

for horizon, time_steps in time_horizons.items():
    print(f"\nTesting for {horizon} horizon...")

    X_train, y_train = create_dataset(train_scaled, train_scaled[:,
0], time_steps)
    X_test, y_test = create_dataset(test_scaled, test_scaled[:, 0],
```

```
time_steps)

    model = Sequential([
        LSTM(units=64, input_shape=(X_train.shape[1],
X_train.shape[2])),
        # Dropout(0.2),
        # LSTM(units=64),
        # Dropout(0.2),
        Dense(units=1)
    ])

    model.compile(optimizer=Adam(learning_rate=0.0005), loss='mse',
metrics=['mae'])

    early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

    history = model.fit(X_train, y_train, epochs=200, batch_size=32,
validation_split=0.1, verbose=1, callbacks=[early_stopping])

    test_loss, test_mae = model.evaluate(X_test, y_test)
    print(f"Test Loss: {test_loss}, Test MAE: {test_mae}")

    predictions = model.predict(X_test)

    predictions_placeholder = np.zeros((predictions.shape[0],
train_scaled.shape[1]))
    predictions_placeholder[:, 0] = predictions.flatten()

    predictions_rescaled =
scaler.inverse_transform(predictions_placeholder)[:, 0]

    y_test_placeholder = np.zeros((y_test.shape[0],
train_scaled.shape[1]))
    y_test_placeholder[:, 0] = y_test.flatten()

    y_test_rescaled = scaler.inverse_transform(y_test_placeholder)[:,
0]

    rmse = np.sqrt(mean_squared_error(y_test_rescaled,
predictions_rescaled))
    r2 = r2_score(y_test_rescaled, predictions_rescaled)

    print(f"RMSE for {horizon}: {rmse}")
    print(f"R² for {horizon}: {r2}")

    results[horizon] = {
        'MAE': test_mae,
        'MSE': test_loss,
        'RMSE': rmse,
```

```
        'R²': r2
    }

    results_df = pd.DataFrame({
        'Predictions': predictions_rescaled.flatten(),
        'Actual': y_test_rescaled.flatten()
    })

results_df.to_csv(f"multivariate_predictions_vs_actual_{horizon}.csv",
index=False)

print("\nSummary of Results:")
for horizon, metrics in results.items():
    print(f"\n{horizon}:")
    for metric, value in metrics.items():
        print(f"{metric}: {value}")
```

```
Testing for 30_min horizon...
Epoch 1/200

c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - loss: 0.0108 - mae:
0.0612 - val_loss: 0.0053 - val_mae: 0.0379
Epoch 2/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.0042 - mae:
0.0341 - val_loss: 0.0048 - val_mae: 0.0361
Epoch 3/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.0038 - mae:
0.0319 - val_loss: 0.0047 - val_mae: 0.0375
Epoch 4/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - loss: 0.0037 - mae:
0.0317 - val_loss: 0.0048 - val_mae: 0.0387
Epoch 5/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 0.0037 - mae:
0.0319 - val_loss: 0.0044 - val_mae: 0.0367
Epoch 6/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - loss: 0.0037 - mae:
0.0309 - val_loss: 0.0044 - val_mae: 0.0350
Epoch 7/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0036 - mae:
0.0301 - val_loss: 0.0042 - val_mae: 0.0327
Epoch 8/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0035 - mae:
```

```
0.0297 - val_loss: 0.0042 - val_mae: 0.0337
Epoch 9/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0036 - mae:
0.0295 - val_loss: 0.0042 - val_mae: 0.0324
Epoch 10/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0034 - mae:
0.0283 - val_loss: 0.0041 - val_mae: 0.0319
Epoch 11/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0035 - mae:
0.0292 - val_loss: 0.0042 - val_mae: 0.0317
Epoch 12/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0035 - mae:
0.0286 - val_loss: 0.0042 - val_mae: 0.0312
Epoch 13/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0036 - mae:
0.0290 - val_loss: 0.0043 - val_mae: 0.0320
Epoch 14/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0277 - val_loss: 0.0043 - val_mae: 0.0316
Epoch 15/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0283 - val_loss: 0.0042 - val_mae: 0.0357
Epoch 16/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.0034 - mae:
0.0285 - val_loss: 0.0041 - val_mae: 0.0318
Epoch 17/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0036 - mae:
0.0290 - val_loss: 0.0041 - val_mae: 0.0313
Epoch 18/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0281 - val_loss: 0.0041 - val_mae: 0.0333
Epoch 19/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0033 - mae:
0.0279 - val_loss: 0.0041 - val_mae: 0.0335
Epoch 20/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0035 - mae:
0.0285 - val_loss: 0.0042 - val_mae: 0.0305
Epoch 21/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0035 - mae:
0.0285 - val_loss: 0.0041 - val_mae: 0.0314
Epoch 22/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0034 - mae:
0.0279 - val_loss: 0.0041 - val_mae: 0.0334
Epoch 23/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0034 - mae:
0.0283 - val_loss: 0.0041 - val_mae: 0.0299
Epoch 24/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0035 - mae:
0.0282 - val_loss: 0.0042 - val_mae: 0.0308
```

```
Epoch 25/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0034 - mae:
0.0277 - val_loss: 0.0041 - val_mae: 0.0328
Epoch 26/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0033 - mae:
0.0278 - val_loss: 0.0041 - val_mae: 0.0301
Epoch 27/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0034 - mae:
0.0279 - val_loss: 0.0041 - val_mae: 0.0334
Epoch 28/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0033 - mae:
0.0275 - val_loss: 0.0041 - val_mae: 0.0304
Epoch 29/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0033 - mae:
0.0272 - val_loss: 0.0042 - val_mae: 0.0315
Epoch 30/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0035 - mae:
0.0280 - val_loss: 0.0042 - val_mae: 0.0309
Epoch 31/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0033 - mae:
0.0276 - val_loss: 0.0041 - val_mae: 0.0304
Epoch 32/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0035 - mae:
0.0282 - val_loss: 0.0042 - val_mae: 0.0320
Epoch 33/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0035 - mae:
0.0278 - val_loss: 0.0041 - val_mae: 0.0318
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0054 - mae: 0.0333
Test Loss: 0.004702538717538118, Test MAE: 0.03146672621369362
329/329 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
RMSE for 30_min: 1912.010049011503
R² for 30_min: 0.9321078969840211

Testing for 1_hour horizon...
Epoch 1/200

c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(**kwargs)

1183/1183 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - loss: 0.0091 - mae:
0.0581 - val_loss: 0.0054 - val_mae: 0.0405
Epoch 2/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 7s 5ms/step - loss: 0.0043 - mae:
0.0350 - val_loss: 0.0048 - val_mae: 0.0376
Epoch 3/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - loss: 0.0038 - mae:
```

```
0.0324 - val_loss: 0.0045 - val_mae: 0.0356
Epoch 4/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0036 - mae:
0.0316 - val_loss: 0.0043 - val_mae: 0.0332
Epoch 5/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0035 - mae:
0.0297 - val_loss: 0.0043 - val_mae: 0.0329
Epoch 6/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0036 - mae:
0.0298 - val_loss: 0.0042 - val_mae: 0.0317
Epoch 7/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0285 - val_loss: 0.0043 - val_mae: 0.0327
Epoch 8/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0036 - mae:
0.0289 - val_loss: 0.0042 - val_mae: 0.0308
Epoch 9/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0278 - val_loss: 0.0042 - val_mae: 0.0326
Epoch 10/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0036 - mae:
0.0289 - val_loss: 0.0041 - val_mae: 0.0307
Epoch 11/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0281 - val_loss: 0.0042 - val_mae: 0.0331
Epoch 12/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0035 - mae:
0.0284 - val_loss: 0.0042 - val_mae: 0.0330
Epoch 13/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0281 - val_loss: 0.0042 - val_mae: 0.0327
Epoch 14/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0280 - val_loss: 0.0041 - val_mae: 0.0309
Epoch 15/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0276 - val_loss: 0.0041 - val_mae: 0.0308
Epoch 16/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0032 - mae:
0.0268 - val_loss: 0.0041 - val_mae: 0.0352
Epoch 17/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0035 - mae:
0.0282 - val_loss: 0.0041 - val_mae: 0.0303
Epoch 18/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0279 - val_loss: 0.0041 - val_mae: 0.0312
Epoch 19/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0276 - val_loss: 0.0041 - val_mae: 0.0319
```

```
Epoch 20/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0275 - val_loss: 0.0041 - val_mae: 0.0311
Epoch 21/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0276 - val_loss: 0.0041 - val_mae: 0.0304
Epoch 22/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0276 - val_loss: 0.0041 - val_mae: 0.0303
Epoch 23/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.0033 - mae:
0.0272 - val_loss: 0.0041 - val_mae: 0.0314
Epoch 24/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0276 - val_loss: 0.0044 - val_mae: 0.0342
Epoch 25/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0274 - val_loss: 0.0041 - val_mae: 0.0326
Epoch 26/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.0034 - mae:
0.0281 - val_loss: 0.0041 - val_mae: 0.0313
Epoch 27/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0276 - val_loss: 0.0041 - val_mae: 0.0319
Epoch 28/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0272 - val_loss: 0.0040 - val_mae: 0.0299
Epoch 29/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0034 - mae:
0.0275 - val_loss: 0.0041 - val_mae: 0.0312
Epoch 30/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0268 - val_loss: 0.0042 - val_mae: 0.0315
Epoch 31/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae:
0.0274 - val_loss: 0.0040 - val_mae: 0.0299
Epoch 32/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0272 - val_loss: 0.0042 - val_mae: 0.0322
Epoch 33/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0273 - val_loss: 0.0041 - val_mae: 0.0303
Epoch 34/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.0035 - mae:
0.0280 - val_loss: 0.0040 - val_mae: 0.0298
Epoch 35/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae:
0.0270 - val_loss: 0.0041 - val_mae: 0.0317
Epoch 36/200
```

```
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae: 
0.0276 - val_loss: 0.0042 - val_mae: 0.0317
Epoch 37/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae: 
0.0274 - val_loss: 0.0041 - val_mae: 0.0320
Epoch 38/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.0033 - mae: 
0.0271 - val_loss: 0.0040 - val_mae: 0.0309
Epoch 39/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae: 
0.0276 - val_loss: 0.0042 - val_mae: 0.0329
Epoch 40/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae: 
0.0272 - val_loss: 0.0040 - val_mae: 0.0304
Epoch 41/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0033 - mae: 
0.0271 - val_loss: 0.0041 - val_mae: 0.0305
Epoch 42/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - loss: 0.0034 - mae: 
0.0274 - val_loss: 0.0041 - val_mae: 0.0299
Epoch 43/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - loss: 0.0035 - mae: 
0.0277 - val_loss: 0.0041 - val_mae: 0.0297
Epoch 44/200
1183/1183 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - loss: 0.0033 - mae: 
0.0273 - val_loss: 0.0041 - val_mae: 0.0318
329/329 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0054 - mae: 0.0332
Test Loss: 0.004701569210737944, Test MAE: 0.031379152089357376
329/329 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
RMSE for 1_hour: 1911.8127856039905
R² for 1_hour: 0.9321320086401613

Testing for 1_day horizon...

c:\Users\AbdullahHarithJamadi\anaconda3\Lib\site-packages\keras\src\
layers\rnn\rnn.py:204: UserWarning: Do not pass an 
`input_shape`/`input_dim` argument to a layer. When using Sequential 
models, prefer using an `Input(shape)` object as the first layer in 
the model instead.
  super().__init__(**kwargs)

Epoch 1/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 73s 58ms/step - loss: 0.0094 - mae: 
0.0592 - val_loss: 0.0055 - val_mae: 0.0442
Epoch 2/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 66s 56ms/step - loss: 0.0041 - mae: 
0.0349 - val_loss: 0.0045 - val_mae: 0.0372
Epoch 3/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 67s 57ms/step - loss: 0.0036 - mae: 
0.0315 - val_loss: 0.0043 - val_mae: 0.0344
```

```
Epoch 4/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 68s 58ms/step - loss: 0.0035 - mae:
0.0303 - val_loss: 0.0040 - val_mae: 0.0322
Epoch 5/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 66s 56ms/step - loss: 0.0035 - mae:
0.0298 - val_loss: 0.0042 - val_mae: 0.0375
Epoch 6/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 67s 57ms/step - loss: 0.0034 - mae:
0.0291 - val_loss: 0.0040 - val_mae: 0.0317
Epoch 7/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 68s 57ms/step - loss: 0.0033 - mae:
0.0284 - val_loss: 0.0041 - val_mae: 0.0313
Epoch 8/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 68s 57ms/step - loss: 0.0034 - mae:
0.0283 - val_loss: 0.0039 - val_mae: 0.0309
Epoch 9/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 64s 54ms/step - loss: 0.0033 - mae:
0.0276 - val_loss: 0.0040 - val_mae: 0.0309
Epoch 10/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 66s 56ms/step - loss: 0.0033 - mae:
0.0278 - val_loss: 0.0039 - val_mae: 0.0301
Epoch 11/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 68s 58ms/step - loss: 0.0033 - mae:
0.0275 - val_loss: 0.0044 - val_mae: 0.0346
Epoch 12/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 69s 58ms/step - loss: 0.0033 - mae:
0.0278 - val_loss: 0.0041 - val_mae: 0.0313
Epoch 13/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 64s 55ms/step - loss: 0.0032 - mae:
0.0271 - val_loss: 0.0040 - val_mae: 0.0303
Epoch 14/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 68s 57ms/step - loss: 0.0032 - mae:
0.0271 - val_loss: 0.0040 - val_mae: 0.0306
Epoch 15/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 66s 56ms/step - loss: 0.0031 - mae:
0.0267 - val_loss: 0.0039 - val_mae: 0.0301
Epoch 16/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 68s 58ms/step - loss: 0.0033 - mae:
0.0272 - val_loss: 0.0039 - val_mae: 0.0306
Epoch 17/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 67s 57ms/step - loss: 0.0033 - mae:
0.0270 - val_loss: 0.0039 - val_mae: 0.0296
Epoch 18/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 69s 58ms/step - loss: 0.0031 - mae:
0.0266 - val_loss: 0.0040 - val_mae: 0.0302
Epoch 19/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 67s 57ms/step - loss: 0.0032 - mae:
0.0271 - val_loss: 0.0038 - val_mae: 0.0301
Epoch 20/200
```

```
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 87s 74ms/step - loss: 0.0032 - mae:
0.0266 - val_loss: 0.0040 - val_mae: 0.0310
Epoch 21/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 68s 58ms/step - loss: 0.0032 - mae:
0.0268 - val_loss: 0.0039 - val_mae: 0.0302
Epoch 22/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 67s 57ms/step - loss: 0.0032 - mae:
0.0272 - val_loss: 0.0038 - val_mae: 0.0291
Epoch 23/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 87s 61ms/step - loss: 0.0031 - mae:
0.0267 - val_loss: 0.0039 - val_mae: 0.0299
Epoch 24/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 72s 61ms/step - loss: 0.0032 - mae:
0.0265 - val_loss: 0.0039 - val_mae: 0.0293
Epoch 25/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 70s 59ms/step - loss: 0.0032 - mae:
0.0264 - val_loss: 0.0040 - val_mae: 0.0320
Epoch 26/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 68s 58ms/step - loss: 0.0032 - mae:
0.0269 - val_loss: 0.0038 - val_mae: 0.0298
Epoch 27/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 69s 58ms/step - loss: 0.0031 - mae:
0.0265 - val_loss: 0.0042 - val_mae: 0.0319
Epoch 28/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 72s 61ms/step - loss: 0.0033 - mae:
0.0270 - val_loss: 0.0041 - val_mae: 0.0312
Epoch 29/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 92s 78ms/step - loss: 0.0031 - mae:
0.0262 - val_loss: 0.0039 - val_mae: 0.0298
Epoch 30/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 109s 93ms/step - loss: 0.0031 - mae:
0.0259 - val_loss: 0.0038 - val_mae: 0.0298
Epoch 31/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 143s 94ms/step - loss: 0.0031 - mae:
0.0262 - val_loss: 0.0038 - val_mae: 0.0286
Epoch 32/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 143s 95ms/step - loss: 0.0031 - mae:
0.0260 - val_loss: 0.0039 - val_mae: 0.0301
Epoch 33/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 143s 96ms/step - loss: 0.0032 - mae:
0.0269 - val_loss: 0.0039 - val_mae: 0.0299
Epoch 34/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 113s 96ms/step - loss: 0.0031 - mae:
0.0261 - val_loss: 0.0039 - val_mae: 0.0291
Epoch 35/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 104s 88ms/step - loss: 0.0032 - mae:
0.0263 - val_loss: 0.0038 - val_mae: 0.0291
Epoch 36/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 145s 90ms/step - loss: 0.0030 - mae:
```

```
0.0261 - val_loss: 0.0039 - val_mae: 0.0294
Epoch 37/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 100s 85ms/step - loss: 0.0031 - mae:
0.0263 - val_loss: 0.0039 - val_mae: 0.0295
Epoch 38/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 141s 84ms/step - loss: 0.0031 - mae:
0.0259 - val_loss: 0.0038 - val_mae: 0.0281
Epoch 39/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 99s 84ms/step - loss: 0.0031 - mae:
0.0260 - val_loss: 0.0039 - val_mae: 0.0299
Epoch 40/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 96s 81ms/step - loss: 0.0031 - mae:
0.0262 - val_loss: 0.0040 - val_mae: 0.0297
Epoch 41/200
1179/1179 ━━━━━━━━━━━━━━━━━━━━ 96s 81ms/step - loss: 0.0033 - mae:
0.0266 - val_loss: 0.0039 - val_mae: 0.0297
324/324 ━━━━━━━━━━━━━━━━━━━━ 11s 33ms/step - loss: 0.0052 - mae:
0.0330
Test Loss: 0.004439511336386204, Test MAE: 0.030453678220510483
324/324 ━━━━━━━━━━━━━━━━━━━━ 12s 36ms/step
RMSE for 1_day: 1857.7680843463302
R² for 1_day: 0.9362637303647137

Summary of Results:

30_min:
MAE: 0.03146672621369362
MSE: 0.004702538717538118
RMSE: 1912.010049011503
R²: 0.9321078969840211

1_hour:
MAE: 0.031379152089357376
MSE: 0.004701569210737944
RMSE: 1911.8127856039905
R²: 0.9321320086401613

1_day:
MAE: 0.030453678220510483
MSE: 0.004439511336386204
RMSE: 1857.7680843463302
R²: 0.9362637303647137
```