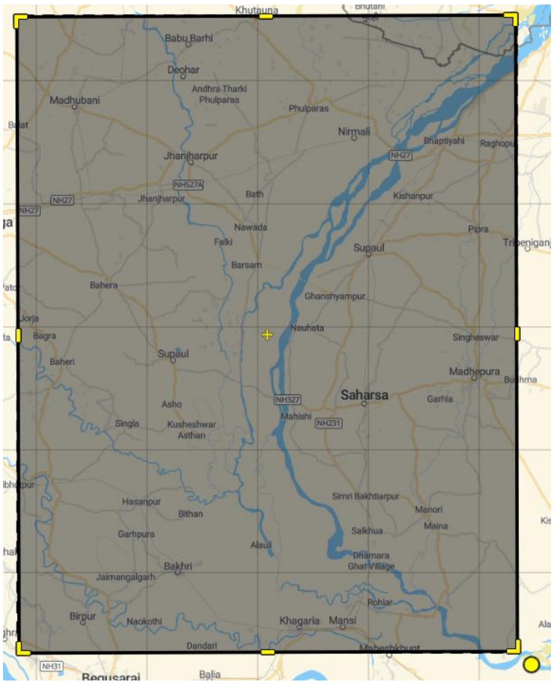
CSV Parameterd for Kosi Region:

Area of Interest (aoi):

Got the coordinates of the area of interest from BoundingBox

westlimit=86.2461; southlimit=25.4653; eastlimit=86.7172; northlimit=26.2192

https://boundingbox.klokantech.com/



CSV DATASET:

Got the csv parameters from the CDS (Climate Data Store) API

https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels-monthly-means?tab=form

dataset used - **ERA5 monthly averaged data on single levels from 1940 to present**

Daily average dataset used for the aoi coordinates westlimit=86.2461; southlimit=25.4653; eastlimit=86.7172; northlimit=26.2192 with duration= 2014 – 2023 (10 YEARS)

Made API request

parameters used – Convective precipitation, Convective rain rate, Instantaneous large-scale surface precipitation fraction, Large scale rain rate, Large-scale precipitation, Large-scale precipitation fraction, Maximum total precipitation rate since previous post-processing, Minimum total precipitation rate since previous post-processing, Precipitation type, Total column rain water, Total precipitation, Vertical integral of eastward water vapour flux, Vertical integral of northward water vapour flux, Vertically integrated moisture divergence

'cp', 'crr', 'ilspf', 'lsp', 'lspf', 'lsrr', 'mxtpr', 'mntpr', 'ptype', 'tcrw', 'tp', 'p71.162', 'p72.162', 'vimd

Sample dataset preview:

	A	B	C	D	E	F	G	H	I	J	K
1	LAT	LON	YEAR	MO	DY	PRECTOTCORR	WS10M	RH2M	QV2M	T2M_RANGE	ALLSKY_SFC_UV_INDEX
2	25.75	86.25	2014	1	1	0	3.7	60.5	7.2	15.83	0.81
3	25.75	86.25	2014	1	2	0	5.36	67.69	7.84	12.05	0.76
4	25.75	86.25	2014	1	3	0	3.61	59.84	5.28	14.14	0.58
5	25.75	86.25	2014	1	4	0	3.07	56.06	4.79	15.85	0.6
6	25.75	86.25	2014	1	5	0.01	2.98	56.59	5.58	16.07	0.49
7	25.75	86.25	2014	1	6	0.01	3.68	71.5	7.93	13.23	0.59
8	25.75	86.25	2014	1	7	0	4.03	54.31	4.82	14.87	0.58
9	25.75	86.25	2014	1	8	0	3.21	46.38	3.81	16.48	0.57
10	25.75	86.25	2014	1	9	0.08	2.06	42.75	3.85	15.03	0.62

PRE-PROCESS THE CSV PARAMETER DATASET:

PSEUDOCODE:

 # Preprocess and normalize the dataset

 # Load the dataset

file_path = "/content/Kosi Rainfall + metrics daily (2014-2023).csv"

data = pd.read_csv(file_path)

 # Handle any missing values (if necessary)

data = data.dropna()

 # Fill missing values with a method suitable for your data

data.fillna(method='ffill', inplace=True)

 # Selecting the columns of interest

columns_of_interest = ['PRECTOTCORR', 'WS10M', 'RH2M', 'QV2M', 'T2M_RANGE', 'ALLSKY_SFC_UV_INDEX']

data = data[columns_of_interest]

 # Normalizing the data

scaler = MinMaxScaler()

scaled_data = scaler.fit_transform(data)

 # Splitting the data into sequences

def create_sequences(data, seq_length):

  xs, ys = [], []

  for i in range(len(data)-seq_length):

```
        x = data[i:i+seq_length]
        y = data[i+seq_length]
        xs.append(x)
        ys.append(y)
    return np.array(xs), np.array(ys)

# for prediction of number of days
SEQ_LENGTH = 10
X, y = create_sequences(scaled_data, SEQ_LENGTH)

# Split the data into training and testing sets 80% and 20%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

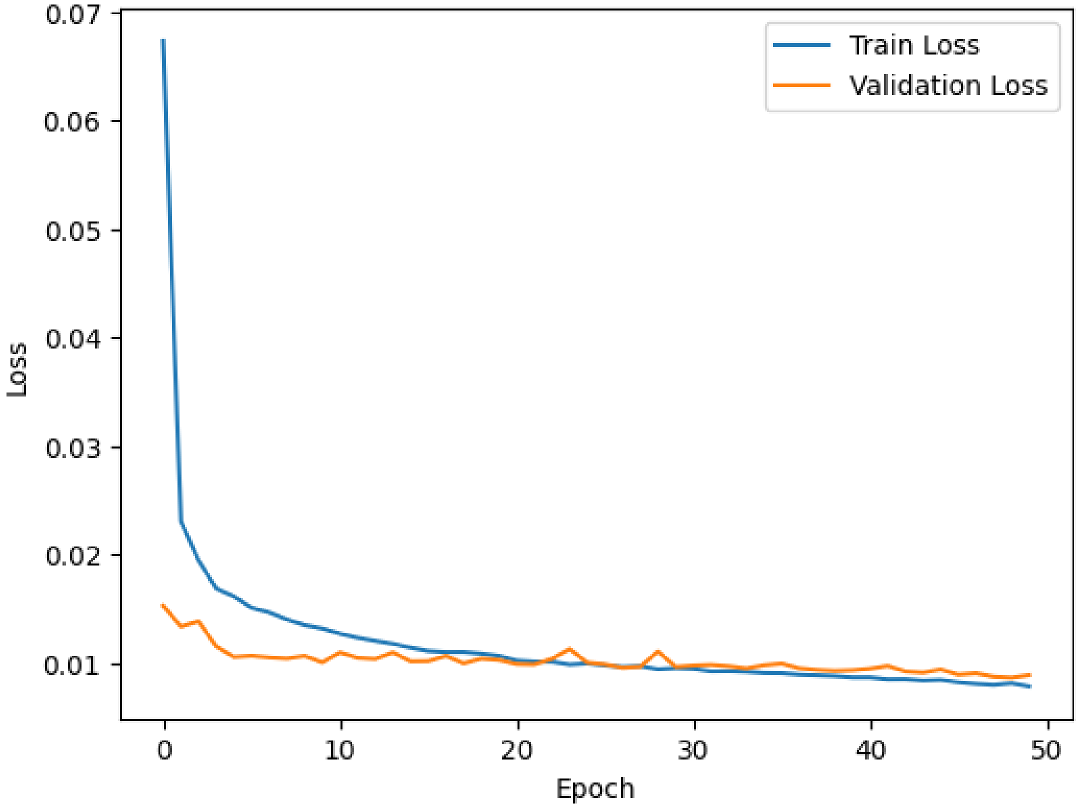
APPLIED PRE-PROCESSED CSV DATASET INTO ML/DL MODELS:

1. **LSTM**

```python
# Define the LSTM model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(SEQ_LENGTH, 6)))
model.add(Dropout(0.3))
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(50))
model.add(Dropout(0.3))
model.add(Dense(25))
model.add(Dense(6))  # Output layer

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 100)           42800

 dropout (Dropout)           (None, 30, 100)           0

 lstm_1 (LSTM)               (None, 30, 100)           80400

 dropout_1 (Dropout)         (None, 30, 100)           0

 lstm_2 (LSTM)               (None, 50)                30200

 dropout_2 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 25)                1275

 dense_1 (Dense)             (None, 6)                 156

=================================================================
Total params: 154831 (604.81 KB)
Trainable params: 154831 (604.81 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

RESULTS:



PRECTOTCORR- MSE: 0.005509878491951068, MAE: 0.033541882158388985, RMSE: 0.07422855577169118, R^2: 0.20674122660768435

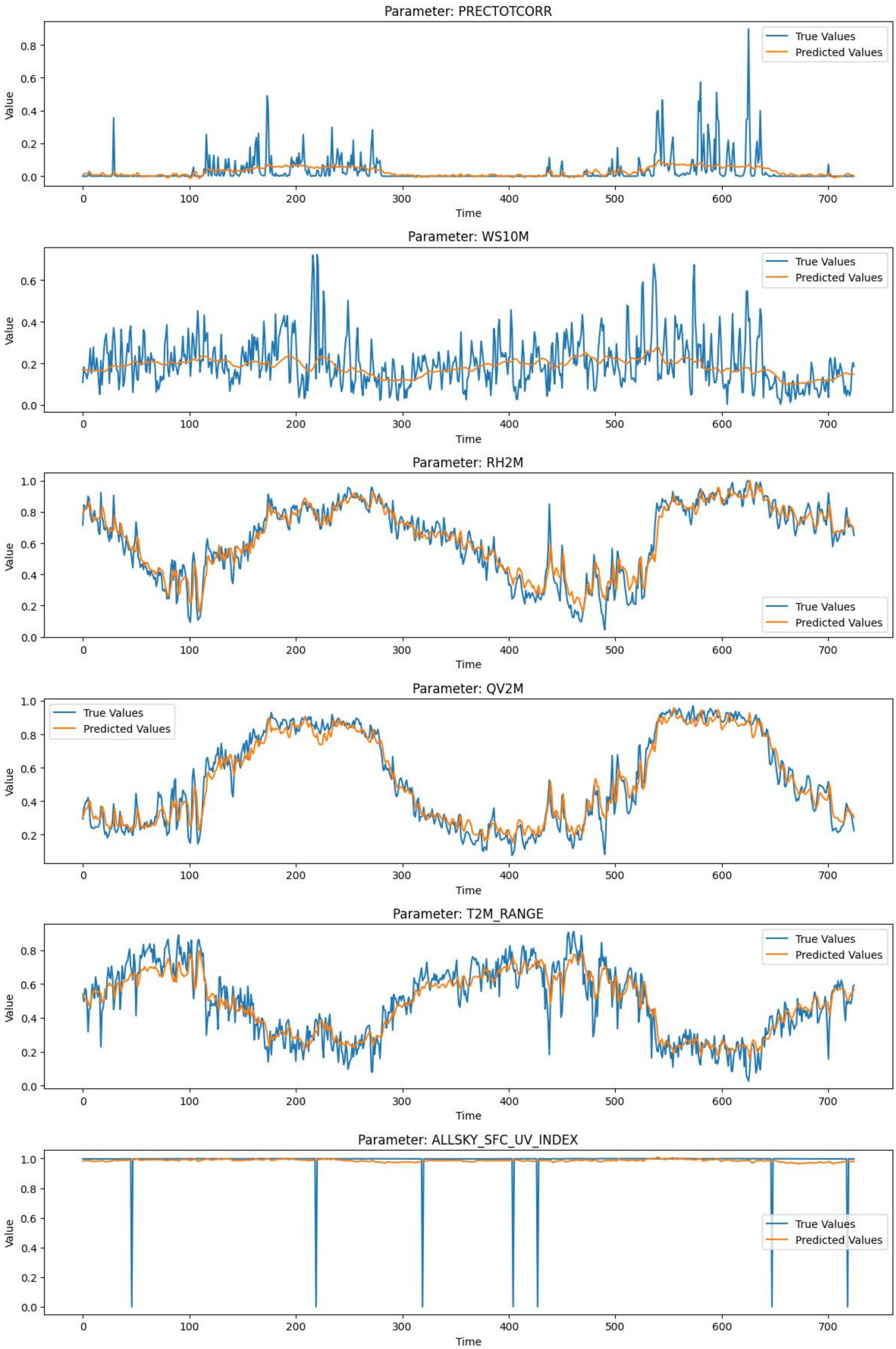
WS10M - MSE: 0.011977256002227762, MAE: 0.08003778268565791, RMSE: 0.10944065059303952, R^2: 0.12330569498672339

RH2M - MSE: 0.004811742026239228, MAE: 0.05238782780455773, RMSE: 0.06936672131677572, R^2: 0.9094222955794752

QV2M - MSE: 0.004265433783190752, MAE: 0.05170731303653358, RMSE: 0.06531028849416264, R^2: 0.9431018518443797

T2M_RANGE - MSE: 0.007029784658533242, MAE: 0.06434565153712091, RMSE: 0.08384381109260983, R^2: 0.8353440252279535

ALLSKY_SFC_UV_INDEX- MSE: 0.009567821386559138, MAE: 0.021783227585812363, RMSE: 0.09781524107499372, R^2: -0.004220316750685127



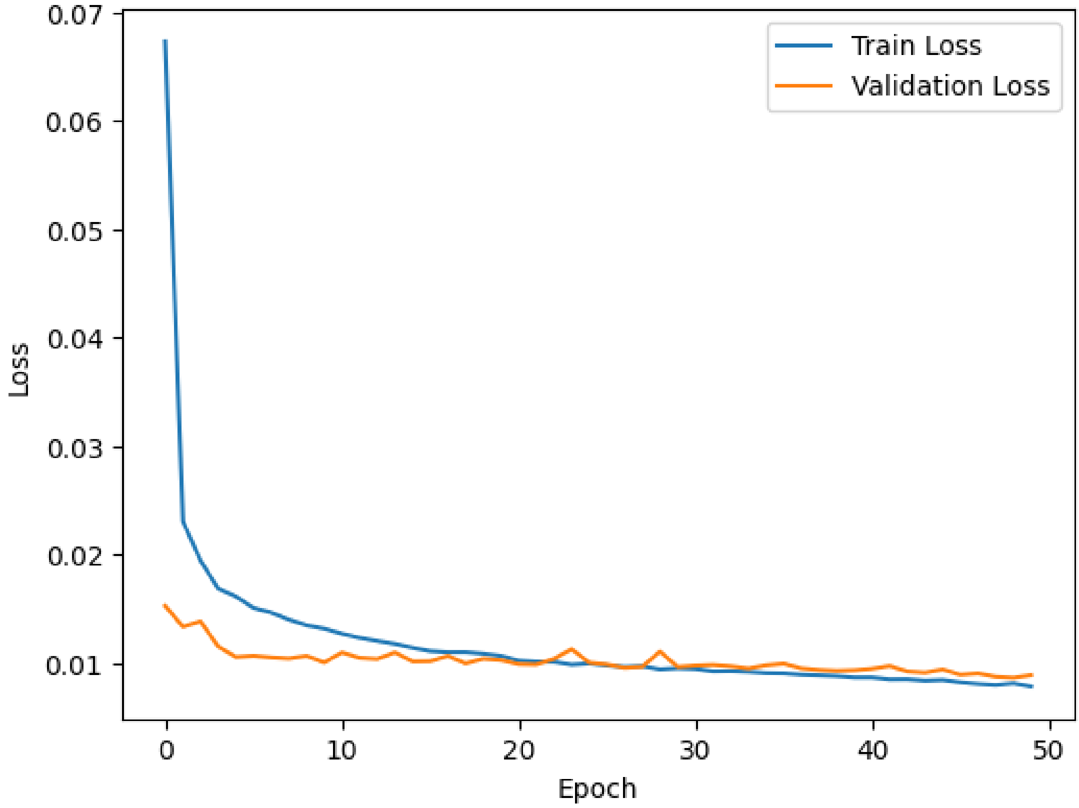
Parameter: PRECTOTCORR
Parameter: WS10M
Parameter: RH2M
Parameter: QV2M
Parameter: T2M_RANGE
Parameter: ALLSKY_SFC_UV_INDEX
Value
Time
True Values
Predicted Values
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0

2. CNN+LSTM

```
model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(SEQ_LENGTH, len(parameters))),
    MaxPooling1D(pool_size=2),
    LSTM(100, return_sequences=True),
    Dropout(0.2),
    LSTM(50),
    Dense(len(parameters))
])

model.compile(optimizer='adam', loss='mse')
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_5 (Conv1D)           (None, 28, 64)            1216

 max_pooling1d_5 (MaxPoolin  (None, 14, 64)            0
 g1D)

 lstm_10 (LSTM)              (None, 14, 100)           66000

 dropout_9 (Dropout)         (None, 14, 100)           0

 lstm_11 (LSTM)              (None, 50)                30200

 dense_5 (Dense)             (None, 6)                 306

=================================================================
Total params: 97722 (381.73 KB)
Trainable params: 97722 (381.73 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

RESULTS:



PRECTOTCORR - MSE: 48.367071489993855, MAE: 3.445699959255209, RMSE: 6.954643879451618, R2: 0.294942283017806

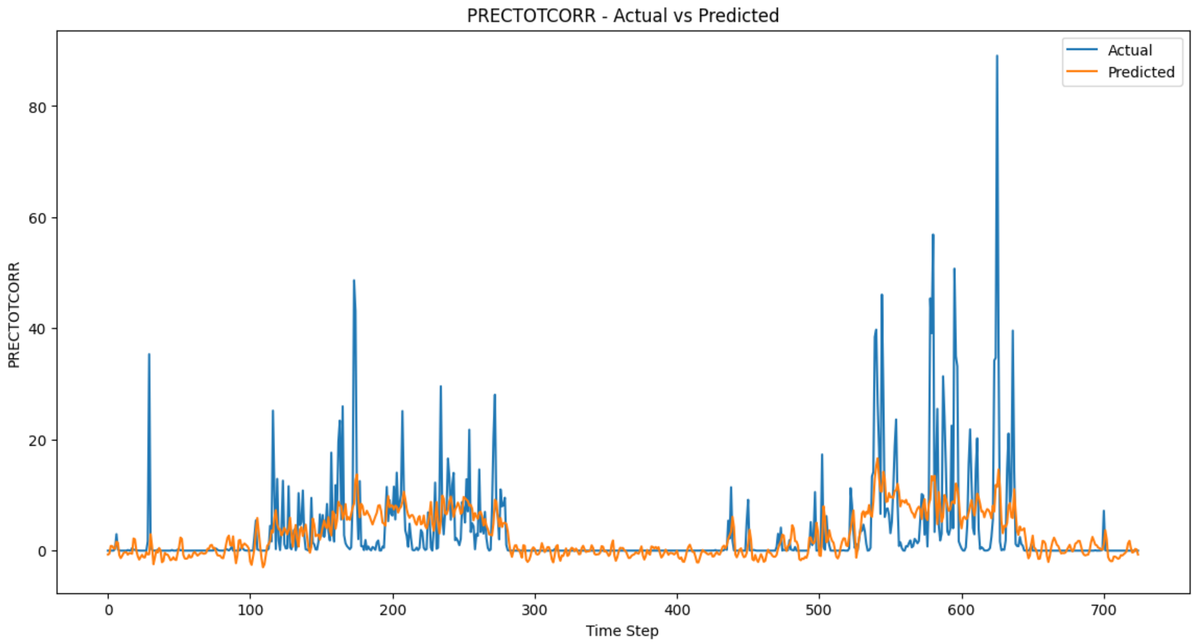
WS10M - MSE: 1.3213525861292739, MAE: 0.8622876688463934, RMSE: 1.1495010161497352, R2: 0.21783208373339302

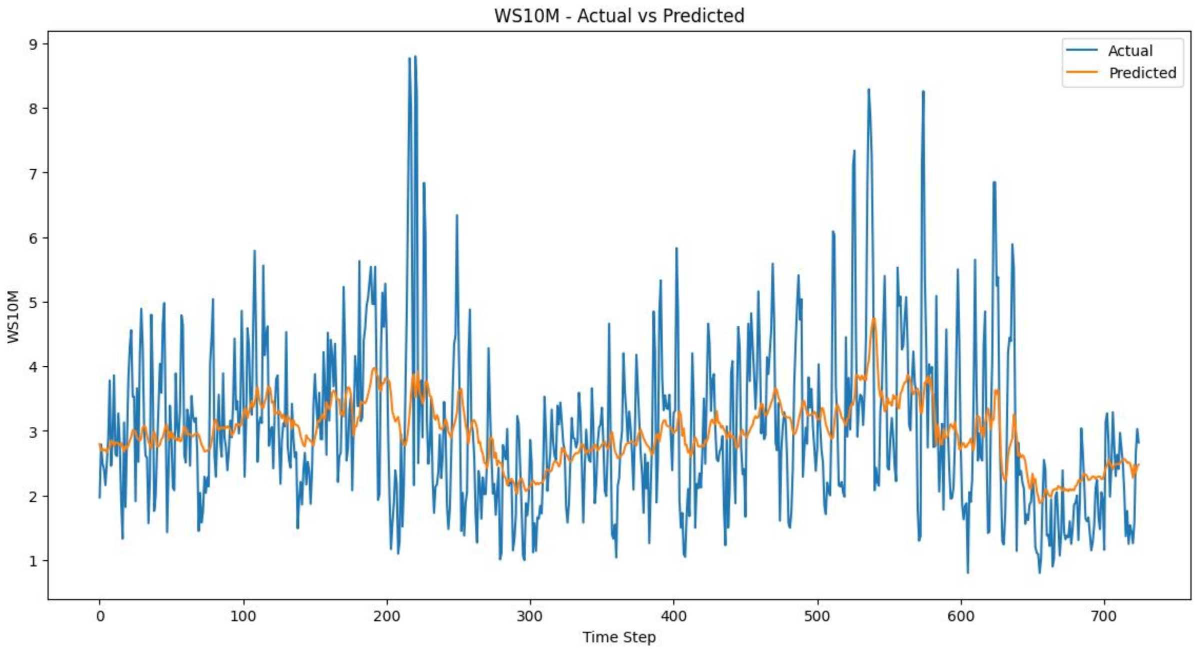
RH2M - MSE: 39.115718962057606, MAE: 4.758565859880119, RMSE: 6.254256067835535, R2: 0.9026510320820771

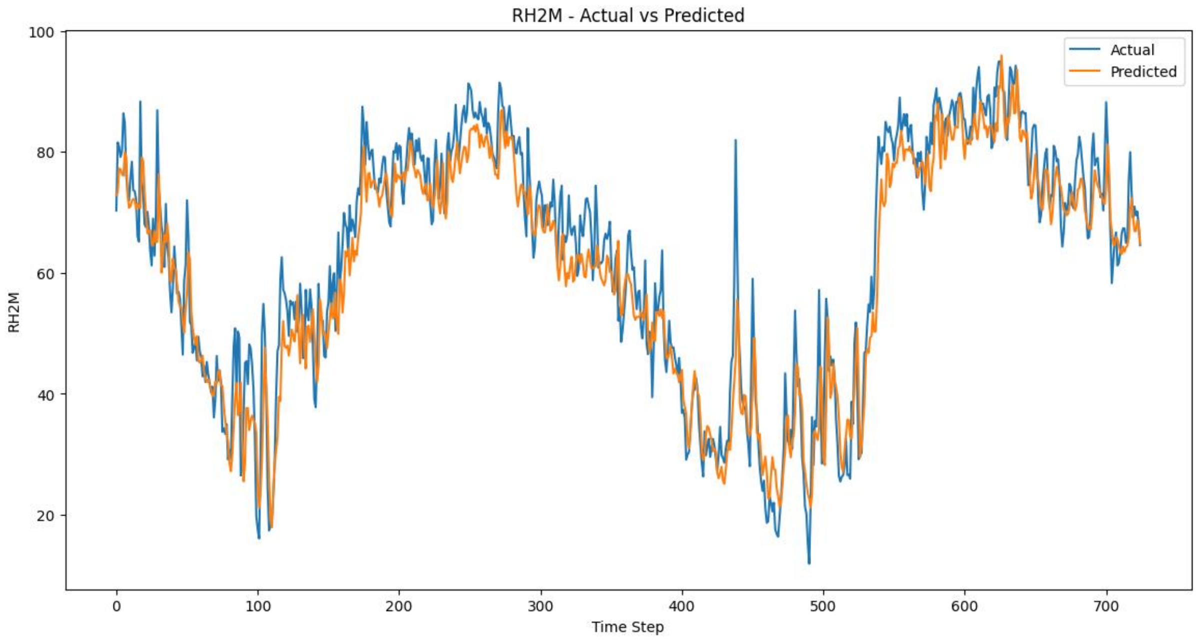
QV2M - MSE: 1.957222030185225, MAE: 1.0682949267420276, RMSE: 1.3990075161289253, R2: 0.9451640299673679

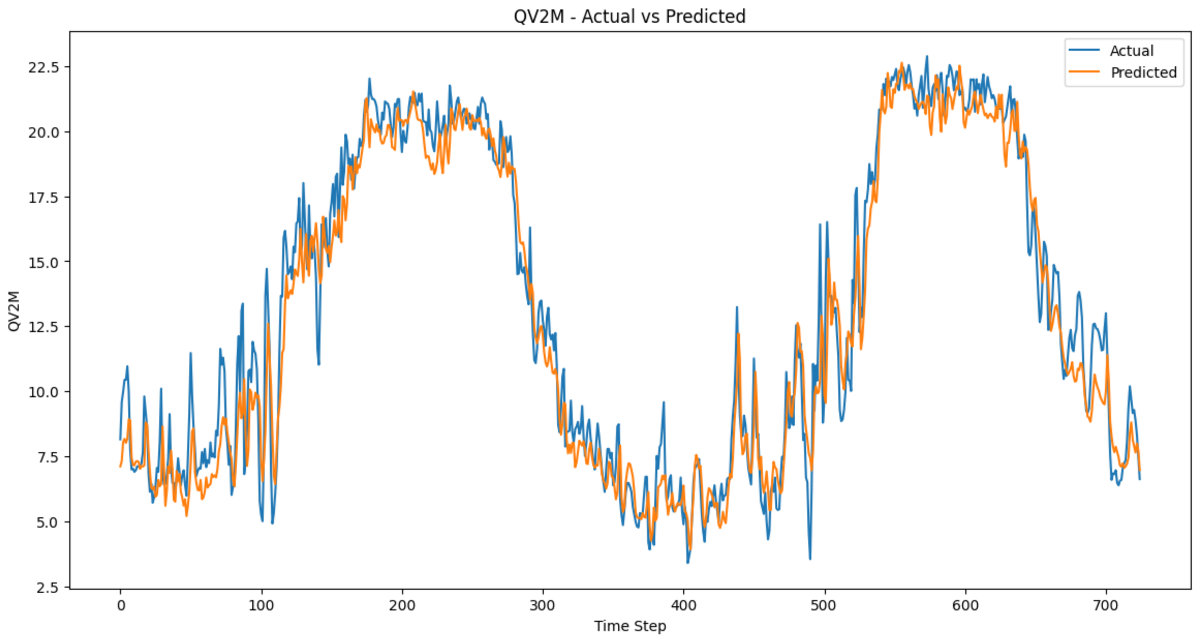
T2M_RANGE - MSE: 2.751994270052257, MAE: 1.235148481618947, RMSE: 1.6589135812489622, R2: 0.8446536906149253

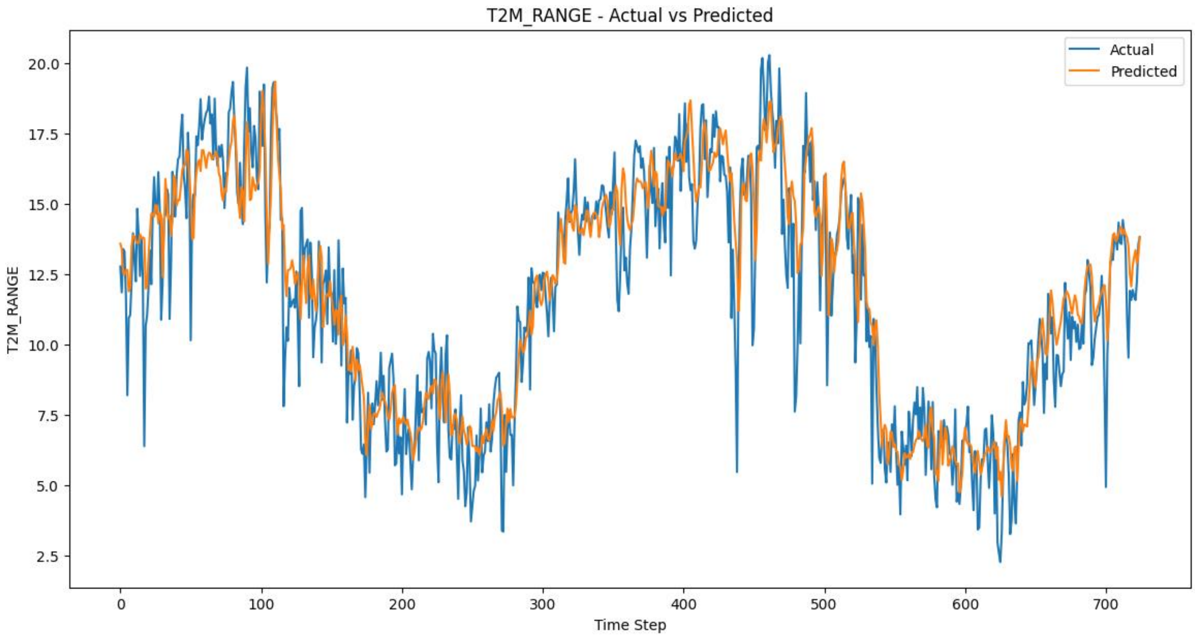
ALLSKY_SFC_UV_INDEX - MSE: 9607.86644016263, MAE: 22.400149466175048, RMSE: 98.01972475049412, R2: -0.0039007337864325198

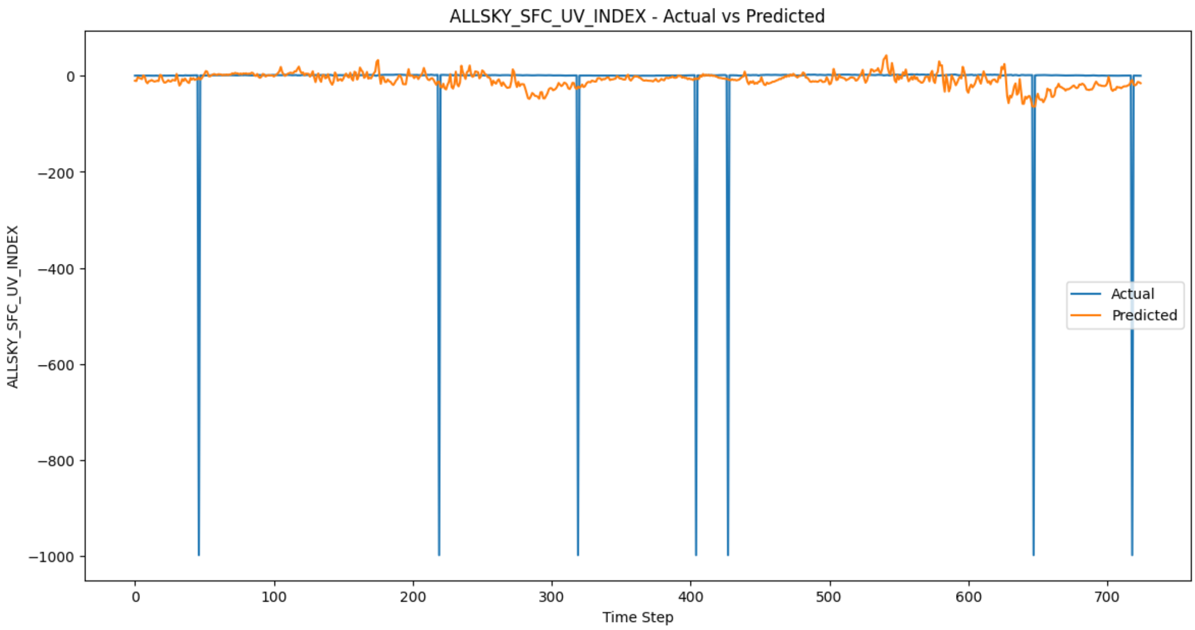

PRECTOTCORR - Actual vs Predicted
Actual
Predicted
PRECTOTCORR
0
20
40
60
80
0
100
200
300
400
500
600
700
Time Step


WS10M - Actual vs Predicted
Actual
Predicted
WS10M
1
2
3
4
5
6
7
8
9
0
100
200
300
400
500
600
700
Time Step


RH2M - Actual vs Predicted
Actual
Predicted
RH2M
Time Step
100
80
60
40
20
0
100
200
300
400
500
600
700


QV2M - Actual vs Predicted
Actual
Predicted
QV2M
Time Step
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
100
200
300
400
500
600
700


T2M_RANGE - Actual vs Predicted
Actual
Predicted
T2M_RANGE
Time Step
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
100
200
300
400
500
600
700

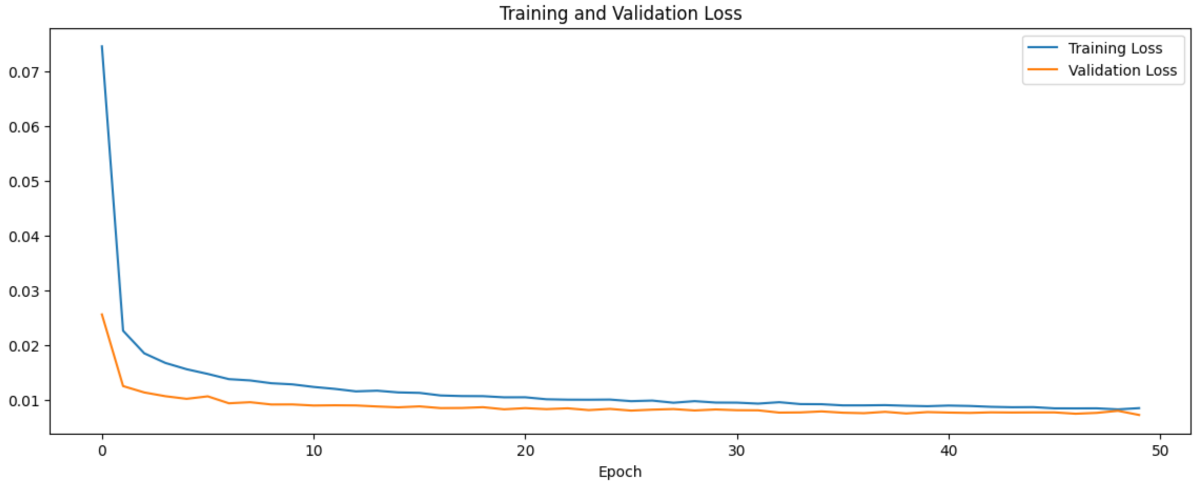

ALLSKY_SFC_UV_INDEX - Actual vs Predicted
Actual
Predicted
ALLSKY_SFC_UV_INDEX
Time Step
0
−200
−400
−600
−800
−1000
0
100
200
300
400
500
600
700

3. GRU

```
# Build the GRU model
model = Sequential()
model.add(GRU(100, return_sequences=True, input_shape=(SEQ_LENGTH, X_train.shape[2])))
model.add(Dropout(0.2))
model.add(GRU(100))
model.add(Dropout(0.2))
model.add(Dense(X_train.shape[2]))

model.compile(optimizer='adam', loss='mse')
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_3 (GRU)                 (None, 30, 100)           33300

 dropout_3 (Dropout)         (None, 30, 100)           0

 gru_4 (GRU)                 (None, 100)               60600

 dropout_4 (Dropout)         (None, 100)               0

 dense_2 (Dense)             (None, 9)                 909

=================================================================
Total params: 94809 (370.35 KB)
Trainable params: 94809 (370.35 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

RESULTS:



PRECTOTCORR - MSE: 45.412340586030965

PRECTOTCORR - MAE: 3.282139829780203

PRECTOTCORR - RMSE: 6.738867900918593

PRECTOTCORR - R-squared: 0.3380140622524821

WS10M - MSE: 0.8957806304428567

WS10M - MAE: 0.7118709068364112

WS10M - RMSE: 0.9464568825059368

WS10M - R-squared: 0.4697472298457901

RH2M - MSE: 27.784692523512867

RH2M - MAE: 4.006415741598196

RH2M - RMSE: 5.2711187165072335

RH2M - R-squared: 0.9308510437017793

QV2M - MSE: 1.4147341467679373

QV2M - MAE: 0.8983136160094163

QV2M - RMSE: 1.189425973639359

QV2M - R-squared: 0.9603630461542648

T2M_RANGE - MSE: 2.4279720289225866

T2M_RANGE - MAE: 1.1956903326231858

T2M_RANGE - RMSE: 1.5581951190151337

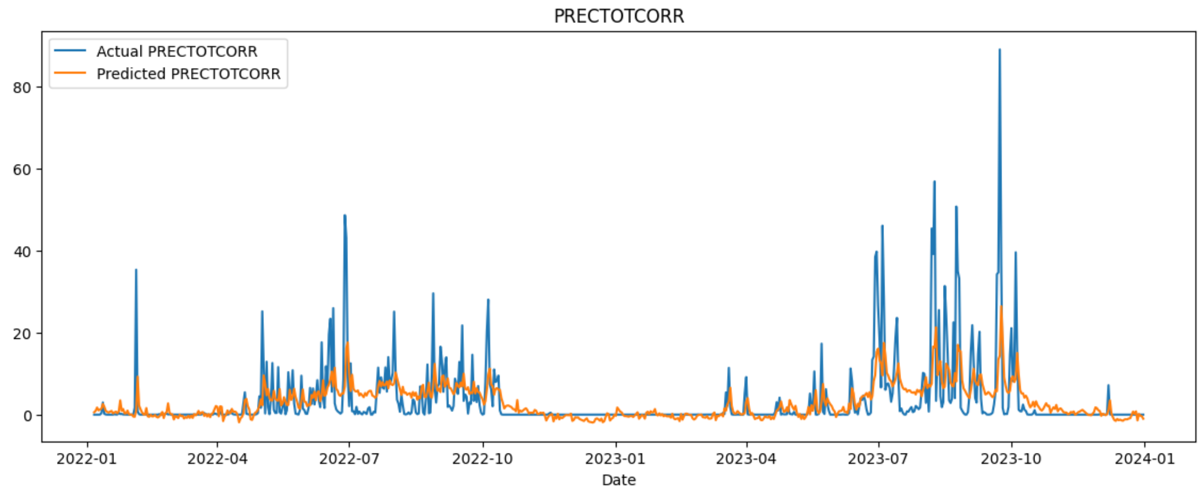
T2M_RANGE - R-squared: 0.8629443025780161

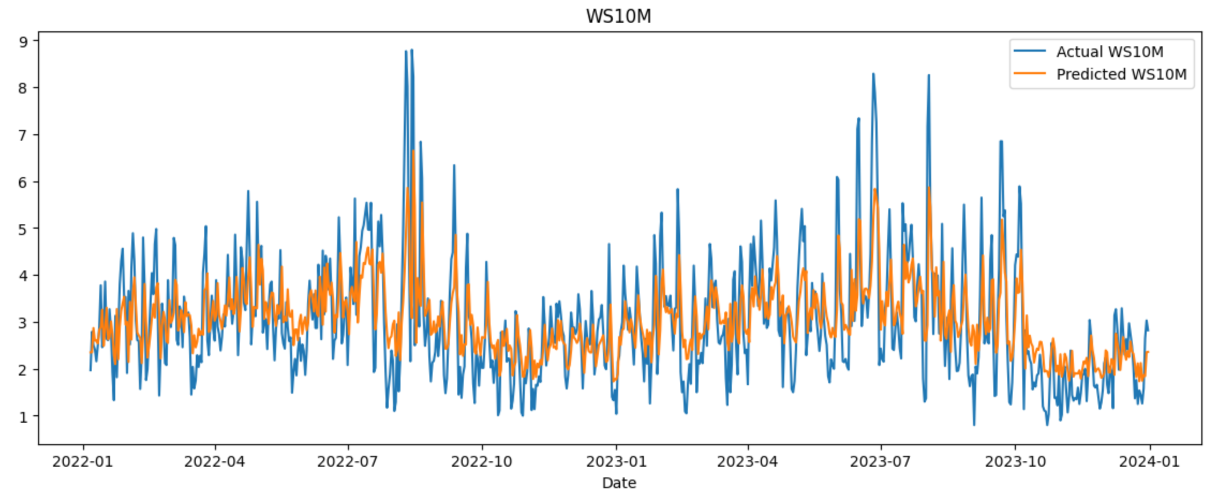
ALLSKY_SFC_UV_INDEX - MSE: 10223.685548752297

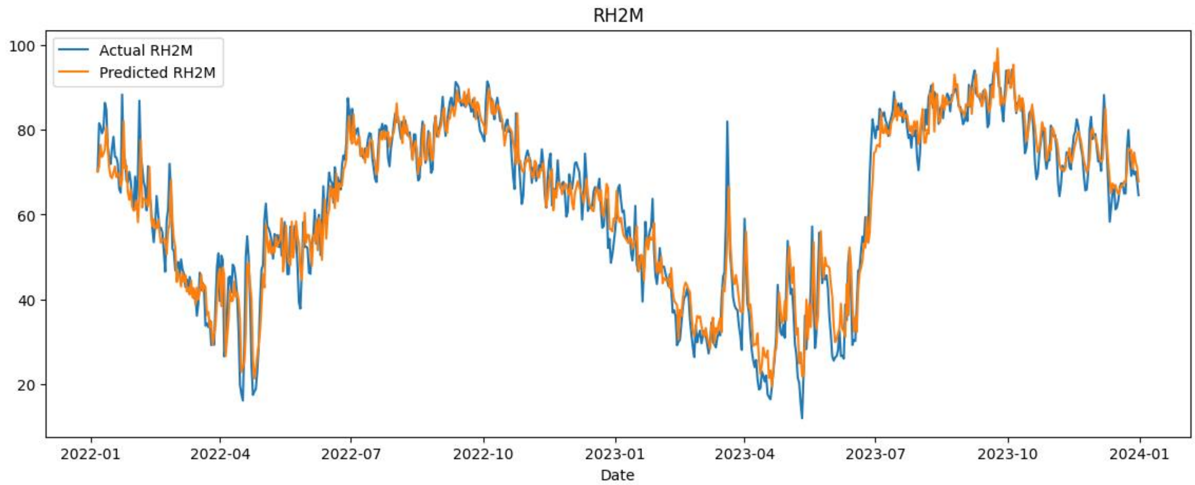
ALLSKY_SFC_UV_INDEX - MAE: 40.58522459114832

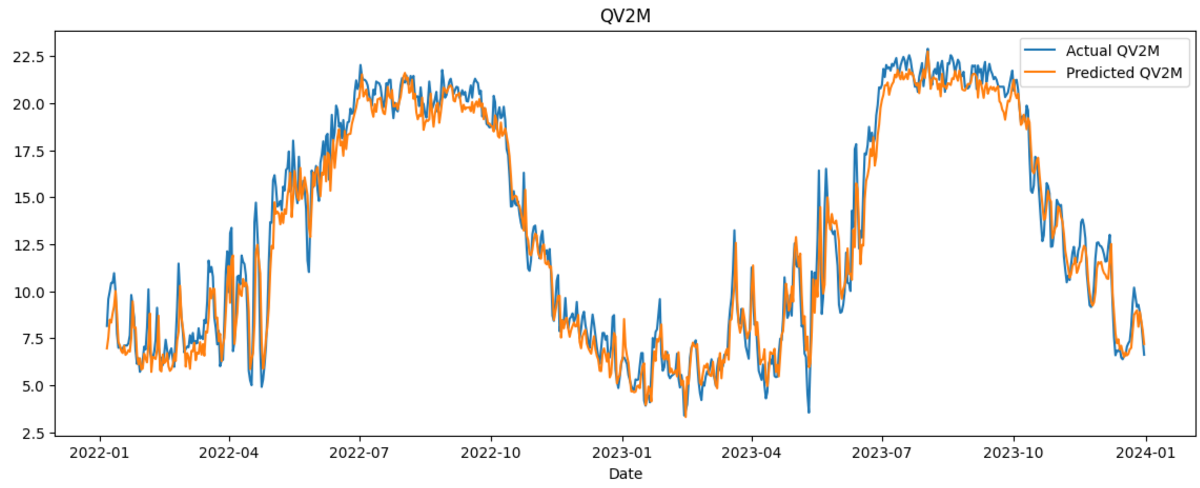
ALLSKY_SFC_UV_INDEX - RMSE: 101.11224232877193

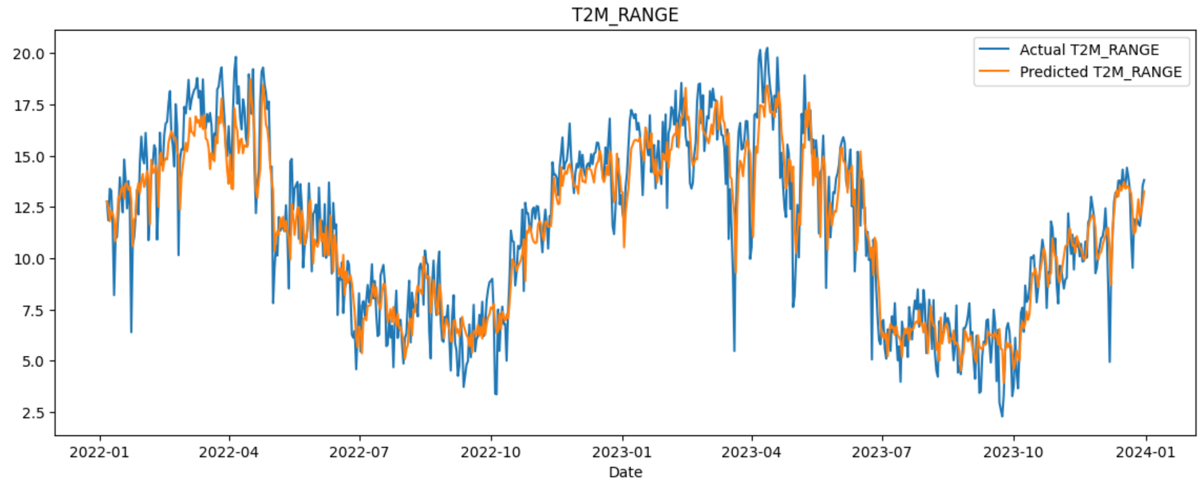
ALLSKY_SFC_UV_INDEX - R-squared: -0.06824605528347116

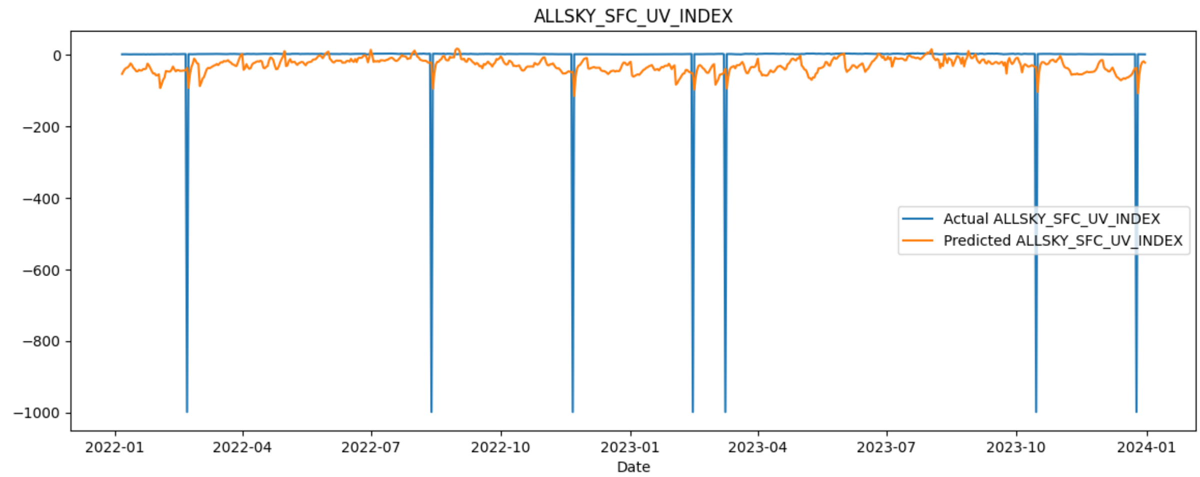



WS10M
Actual WS10M
Predicted WS10M
9
8
7
6
5
4
3
2
1
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
Date


RH2M
Actual RH2M
Predicted RH2M
100
80
60
40
20
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
Date


QV2M
Actual QV2M
Predicted QV2M
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
Date


T2M_RANGE
Actual T2M_RANGE
Predicted T2M_RANGE
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
Date


ALLSKY_SFC_UV_INDEX
Actual ALLSKY_SFC_UV_INDEX
Predicted ALLSKY_SFC_UV_INDEX
0
−200
−400
−600
−800
−1000
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
Date

4. CNN+TRANSFORMER

****INTRODUCED LAG FEATURE TO INCREASE THE EVALUATION METRICS AND ACCURACY OF MODEL****

```
# Feature Engineering: Adding lag features
for lag in range(1, 4):
    data[f'PRECTOTCORR_lag{lag}'] = data['PRECTOTCORR'].shift(lag)
    data[f'WS10M_lag{lag}'] = data['WS10M'].shift(lag)




# CNN part
def build_cnn(input_shape):
    inputs = Input(shape=input_shape)
    x = Conv1D(filters=64, kernel_size=3, activation='relu')(inputs)
    x = MaxPooling1D(pool_size=2)(x)
    x = Conv1D(filters=128, kernel_size=3, activation='relu')(x)
    x = MaxPooling1D(pool_size=2)(x)
    x = Flatten()(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.2)(x)
    model = Model(inputs, x)
    return model

# Transformer part
def build_transformer(input_shape):
    inputs = Input(shape=input_shape)
    x = LayerNormalization(epsilon=1e-6)(inputs)
```

```
    x = MultiHeadAttention(num_heads=4, key_dim=input_shape[-1])(x, x)
    x = Dropout(0.1)(x)
    x = Add()([inputs, x])
    x = LayerNormalization(epsilon=1e-6)(x)
    x = Dense(units=128, activation='relu')(x)
    model = Model(inputs, x)
    return model

input_shape = (seq_length, data_normalized.shape[1])
cnn_model = build_cnn(input_shape)

# The output of the CNN model should be reshaped to match the Transformer input requirements
cnn_output_shape = cnn_model.output_shape[1:]  # (features,)
cnn_output = Reshape((1, cnn_output_shape[0]))(cnn_model.output)

transformer_model = build_transformer(cnn_output.shape[1:])

# Combine CNN and Transformer
combined_input = Input(shape=input_shape)
cnn_output = cnn_model(combined_input)
cnn_output = Reshape((1, cnn_output_shape[0]))(cnn_output)
transformer_output = transformer_model(cnn_output)
flattened_output = Flatten()(transformer_output)
output = Dense(data_normalized.shape[1])(flattened_output)

model = Model(combined_input, output)
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 30, 12)]          0

 model (Functional)          (None, 128)               125504

 reshape_1 (Reshape)         (None, 1, 128)            0

 model_1 (Functional)        (None, 1, 128)            280832

 flatten_1 (Flatten)         (None, 128)               0

 dense_2 (Dense)             (None, 12)                1548

=================================================================
Total params: 407884 (1.56 MB)
Trainable params: 407884 (1.56 MB)
Non-trainable params: 0 (0.00 Byte)
```

This model architecture combines a Convolutional Neural Network (CNN) and a Transformer to process sequential data. Here's a breakdown of each part of the architecture and how they work together:

CNN Part

The CNN is used to extract features from the input data. The structure is as follows:

1. **Input Layer**: Takes input with shape (seq_length, num_features).
2. **First Convolutional Layer**:
   - Conv1D: 64 filters, kernel size of 3, ReLU activation.
   - MaxPooling1D: Pool size of 2.
3. **Second Convolutional Layer**:
   - Conv1D: 128 filters, kernel size of 3, ReLU activation.
   - MaxPooling1D: Pool size of 2.
4. **Flatten Layer**: Flattens the 3D output to 1D.
5. **Dense Layer**: 128 units, ReLU activation.
6. **Dropout Layer**: 20% dropout to prevent overfitting.

The output of this part is a flattened feature vector of length 128.

Transformer Part

The Transformer processes the feature vector to capture temporal dependencies. The structure is as follows:

1. **Input Layer**: Takes input with shape (1, cnn_output_features), where cnn_output_features is 128 from the CNN part.
2. **Layer Normalization**: Normalizes the input.
3. **Multi-Head Attention**: 4 heads, key dimension is the same as the last dimension of the input.
4. **Dropout Layer**: 10% dropout.
5. **Add & Layer Normalization**: Adds the original input to the output of the attention layer and normalizes it.
6. **Dense Layer**: 128 units, ReLU activation.

Combining CNN and Transformer

The steps to combine them are as follows:

1. **Input Layer**: Takes input with shape (seq_length, num_features).
2. **CNN Model**: Processes the input and produces an output of shape (cnn_output_features), where cnn_output_features is 128.
3. **Reshape Layer**: Reshapes the CNN output to match the input shape required by the Transformer, i.e., (1, 128).
4. **Transformer Model**: Processes the reshaped output from the CNN.
5. **Flatten Layer**: Flattens the output of the Transformer to a 1D vector.
6. **Dense Layer**: Produces the final output with num_features units (same as the number of features in the input).

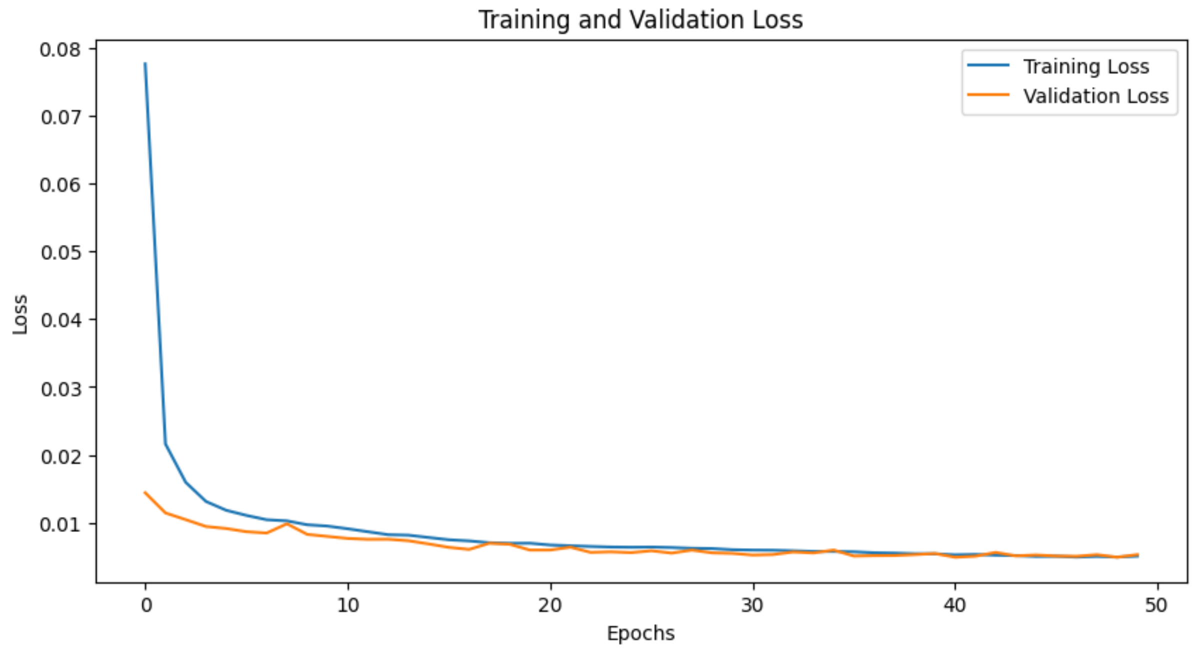
Compilation

The model is compiled using the Adam optimizer, mean squared error (MSE) loss, and mean absolute error (MAE) as a metric.

Summary

The combined model first uses a CNN to extract features from the sequential input data. These features are then reshaped and fed into a Transformer model to capture temporal dependencies. The final output is produced by a dense layer, with the model being trained to minimize MSE.

RESULTS:



BEFORE LAG:

MODEL METRIC: MSE: 1046.7978906589167

MAE: 3.980997078764154

RMSE: 32.35425614442274

R-squared: 0.6066104305024839

Metrics for PRECTOTCORR:

MSE: 37.940194875600355

MAE: 2.9508248136124897

RMSE: 6.159561256745512

R-squared: 0.3252810894674911

Metrics for WS10M:

MSE: 1.1552844384649028

MAE: 0.8335356996336035

RMSE: 1.0748415876141484

R-squared: 0.2508321204902041

Metrics for RH2M:

MSE: 41.329438172827075

MAE: 4.9201600069499145

RMSE: 6.428797568194777

R-squared: 0.9129092099399148

Metrics for QV2M:

MSE: 3.223309397708382

MAE: 1.3806825279267454

RMSE: 1.7953577353019041

R-squared: 0.9196476969878116

Metrics for T2M_RANGE:

MSE: 3.4382402967853443

MAE: 1.3864313835070279

RMSE: 1.8542492542226745

R-squared: 0.8225402923194522

Metrics for ALLSKY_SFC_UV_INDEX:

MSE: 12424.38756013743

MAE: 27.340753416608383

RMSE: 111.46473684595244

R-squared: -0.011269491452342573

AFTER LAG:

Metrics for PRECTOTCORR_lag1:

MSE: 12.770644820016896

MAE: 2.179339694664376

RMSE: 3.573603898030236

R-squared: 0.706091060972394

Metrics for WS10M_lag1:

MSE: 0.6911616561057519

MAE: 0.6837652206816067

RMSE: 0.8313613270448367

R-squared: 0.59040650632335

Metrics for PRECTOTCORR_lag2:

MSE: 16.230507664554573

MAE: 2.1697284923665077

RMSE: 4.028710422027696

R-squared: 0.6486395439814066

Metrics for WS10M_lag2:

MSE: 0.42431155210954535

MAE: 0.49670446235830606

RMSE: 0.651392011088212

R-squared: 0.7577168889805308

Metrics for PRECTOTCORR_lag3:

MSE: 19.574259304537637

MAE: 2.9440178673934994

RMSE: 4.424280653907213

R-squared: 0.6216794566552987

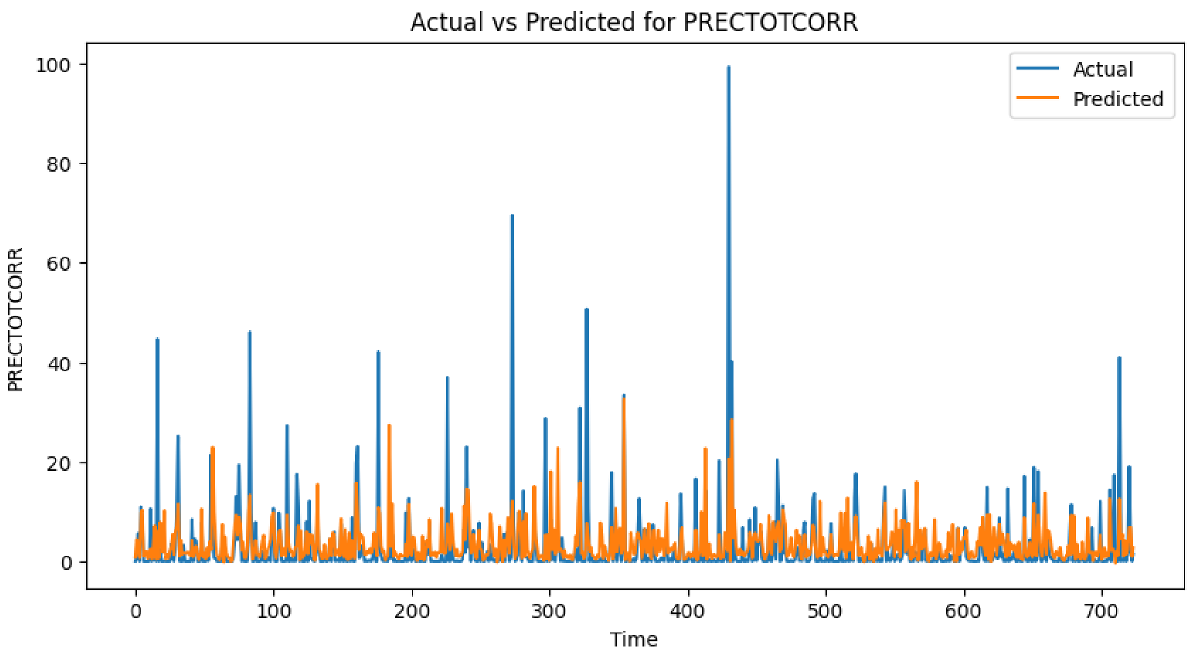
Metrics for WS10M_lag3:

MSE: 0.40977559085302434

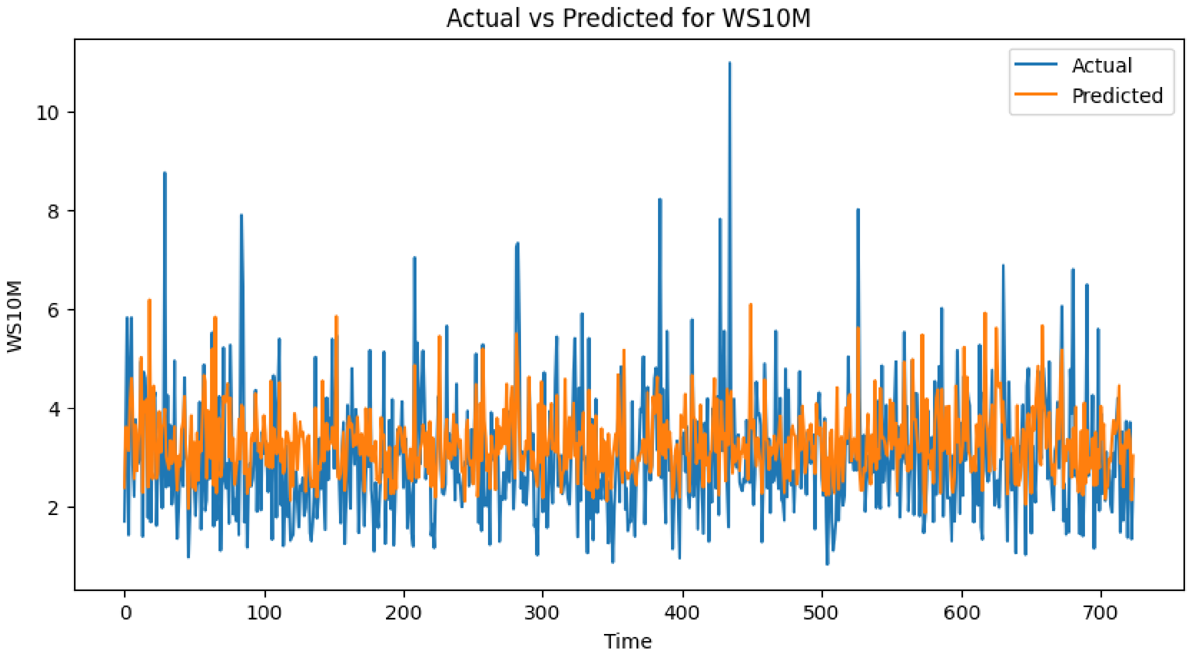
MAE: 0.4860213594673747

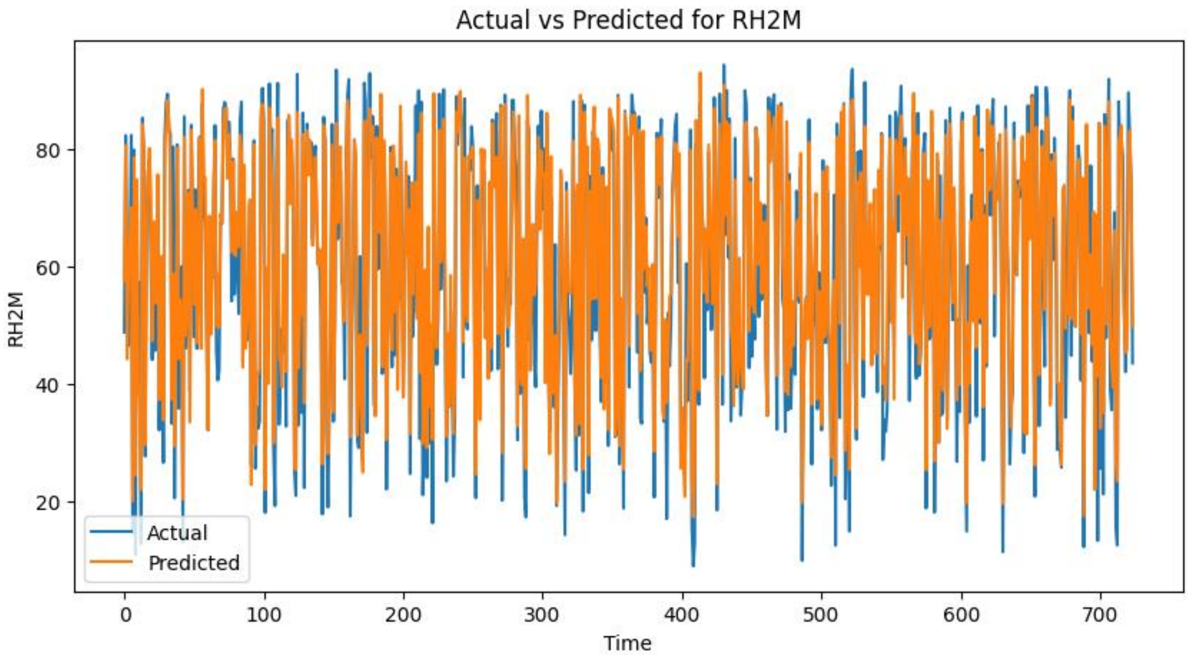
RMSE: 0.640137165655162

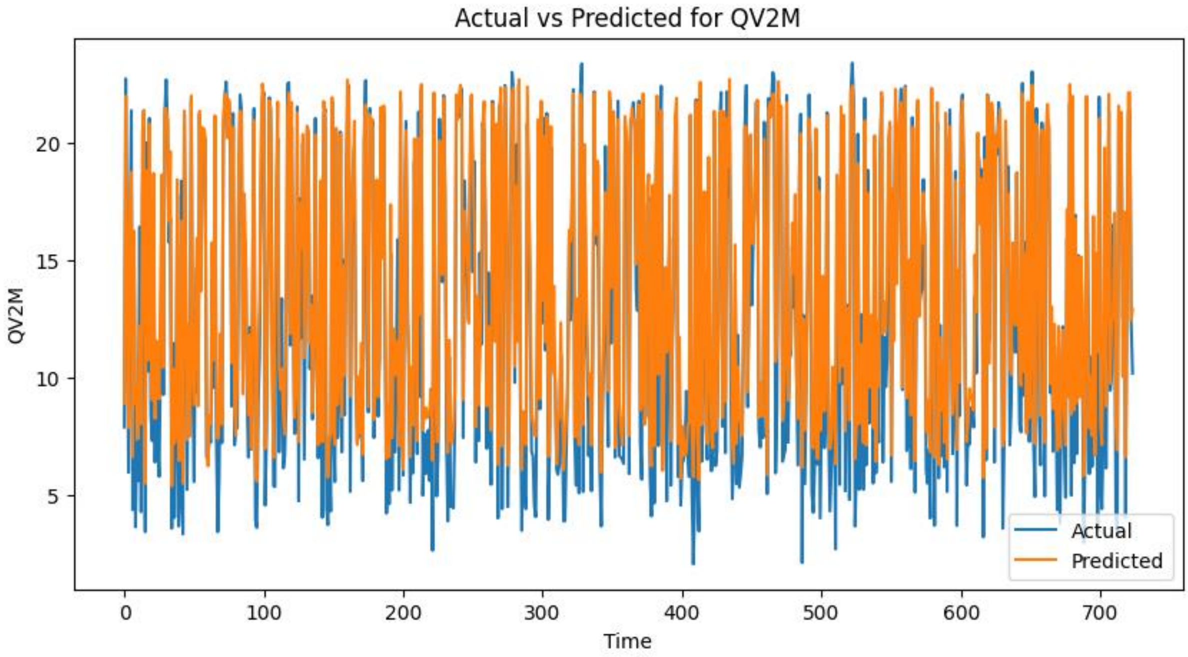
R-squared: 0.734850791364298

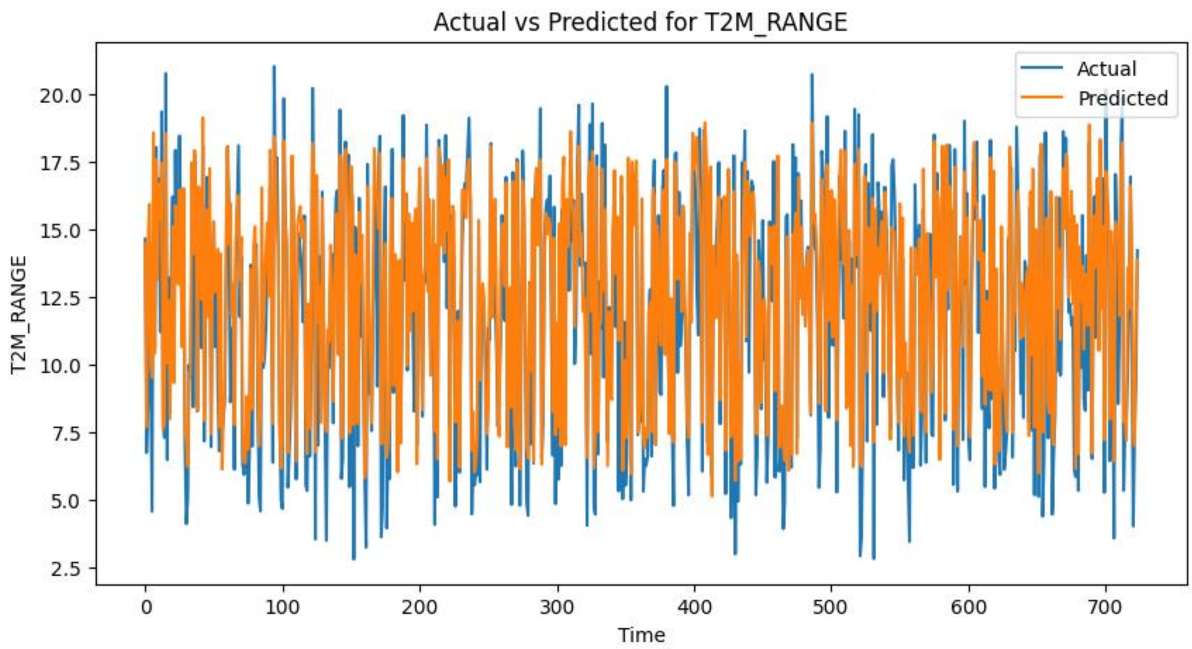


Actual vs Predicted for PRECTOTCORR
Actual
Predicted
PRECTOTCORR
Time
0
20
40
60
80
100
0
100
200
300
400
500
600
700

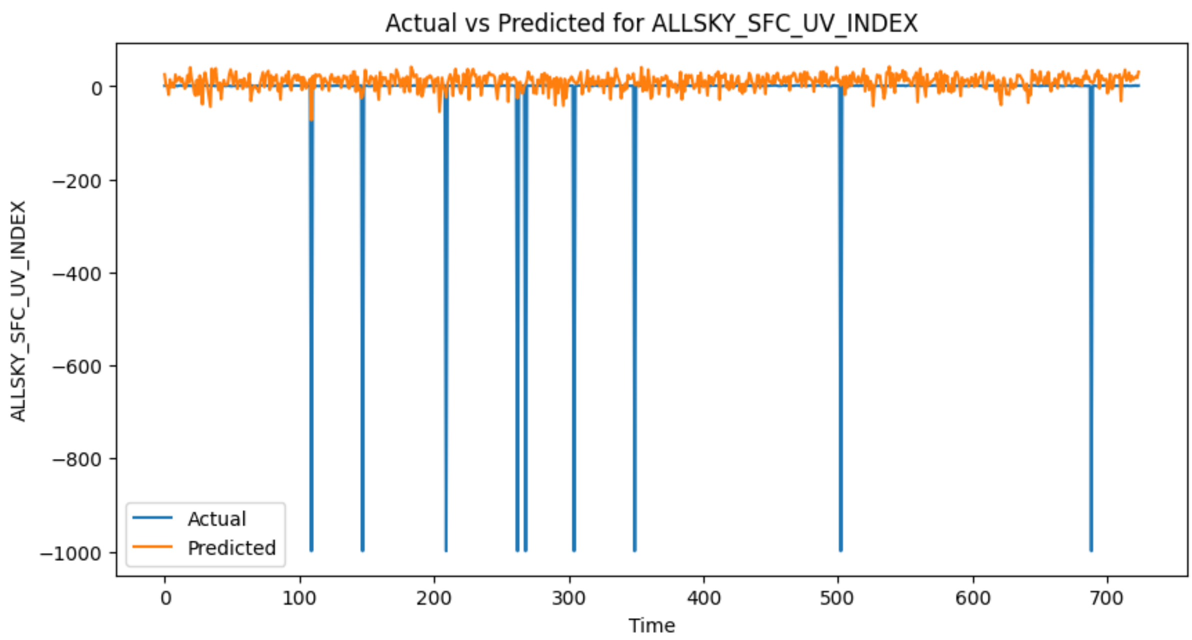

Actual vs Predicted for WS10M
Actual
Predicted
WS10M
2
4
6
8
10
0
100
200
300
400
500
600
700
Time


Actual vs Predicted for RH2M
Actual
Predicted
RH2M
20
40
60
80
0
100
200
300
400
500
600
700
Time


Actual vs Predicted for QV2M
QV2M
Time
Actual
Predicted
20
15
10
5
0
100
200
300
400
500
600
700

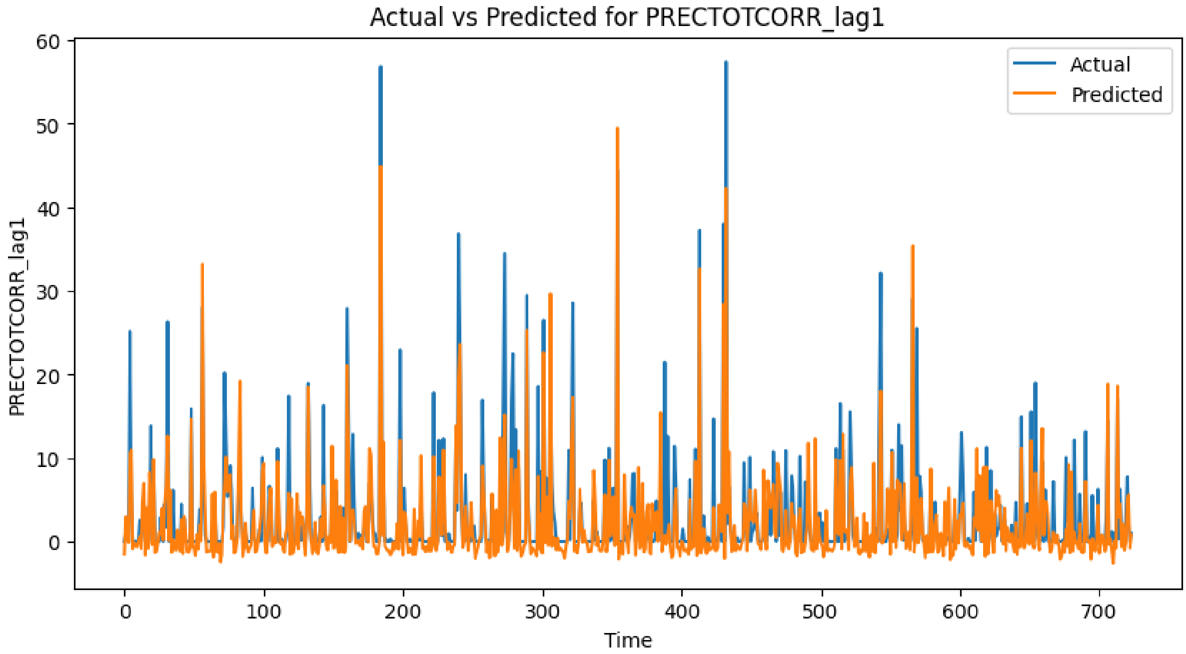


Actual vs Predicted for T2M_RANGE
Actual
Predicted
T2M_RANGE
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
100
200
300
400
500
600
700
Time

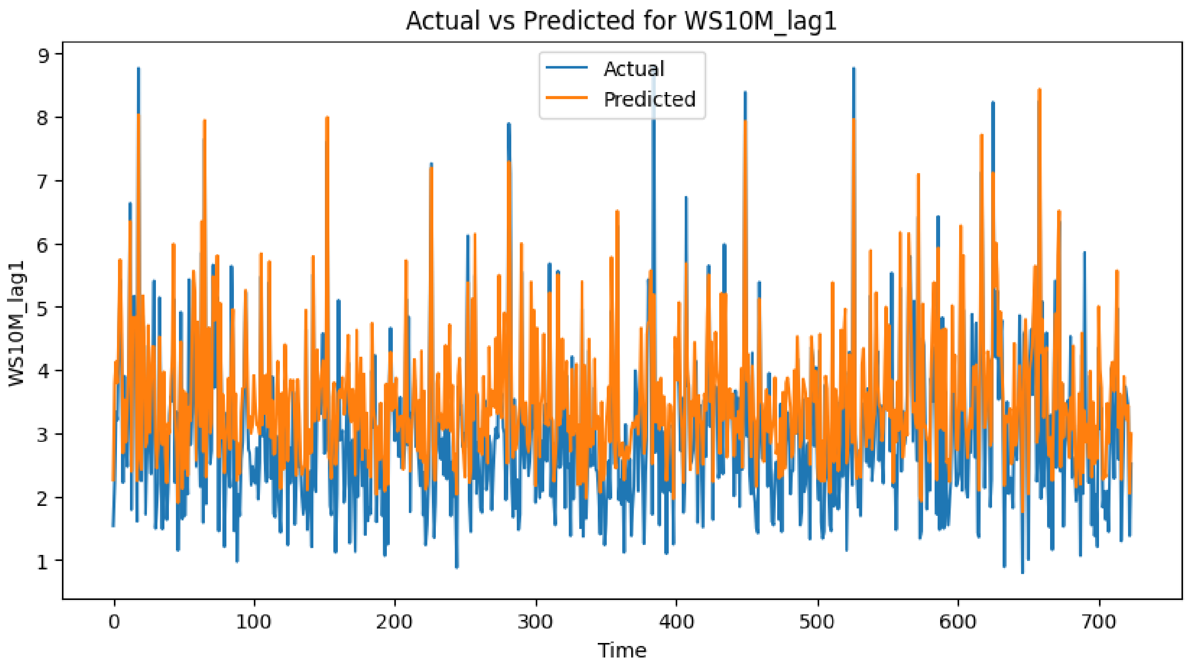


Actual vs Predicted for ALLSKY_SFC_UV_INDEX
ALLSKY_SFC_UV_INDEX
0
−200
−400
−600
−800
−1000
Actual
Predicted
0
100
200
300
400
500
600
700
Time

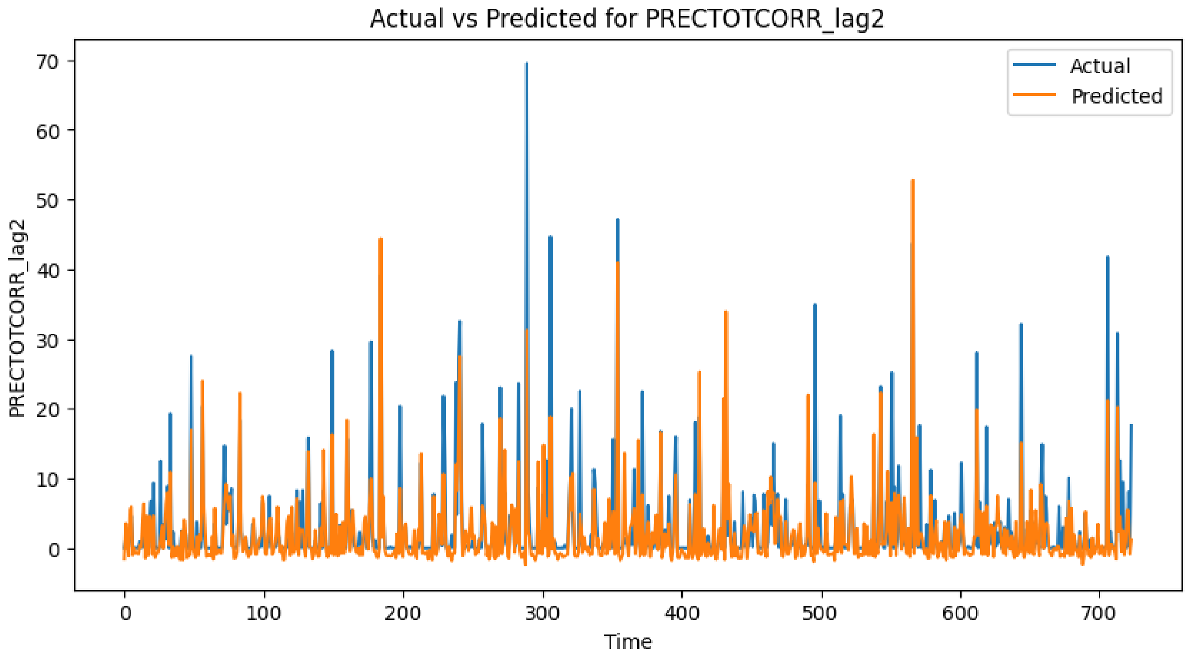
AFTER LAG:



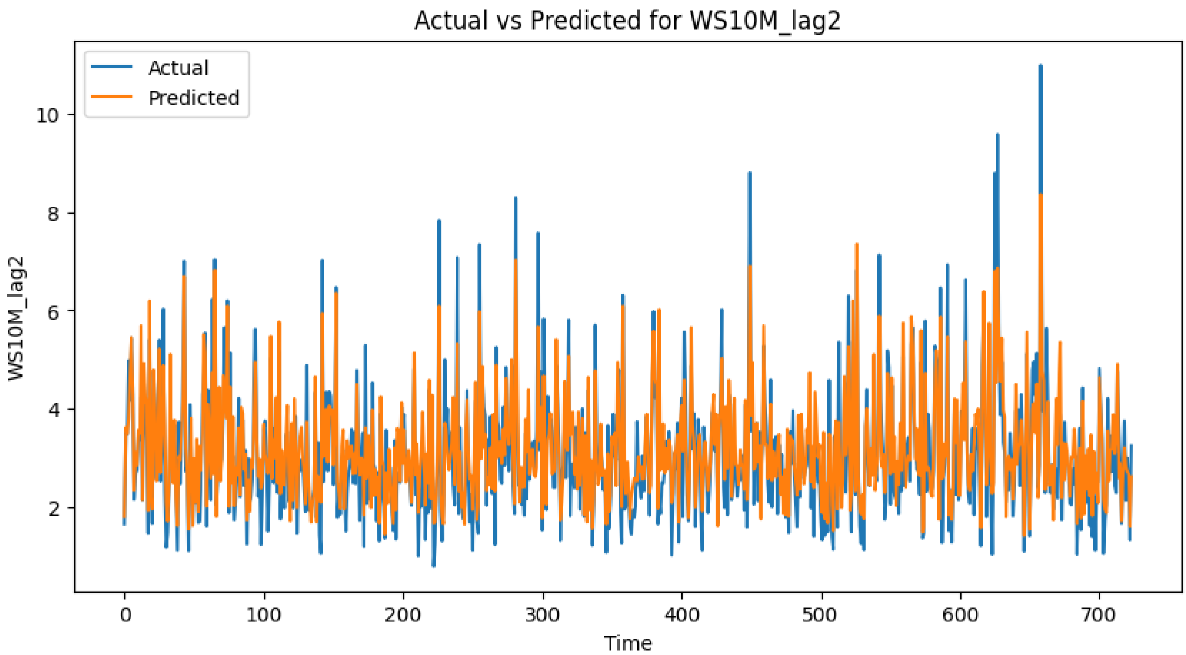
Actual vs Predicted for PRECTOTCORR_lag1
Actual
Predicted
PRECTOTCORR_lag1
Time



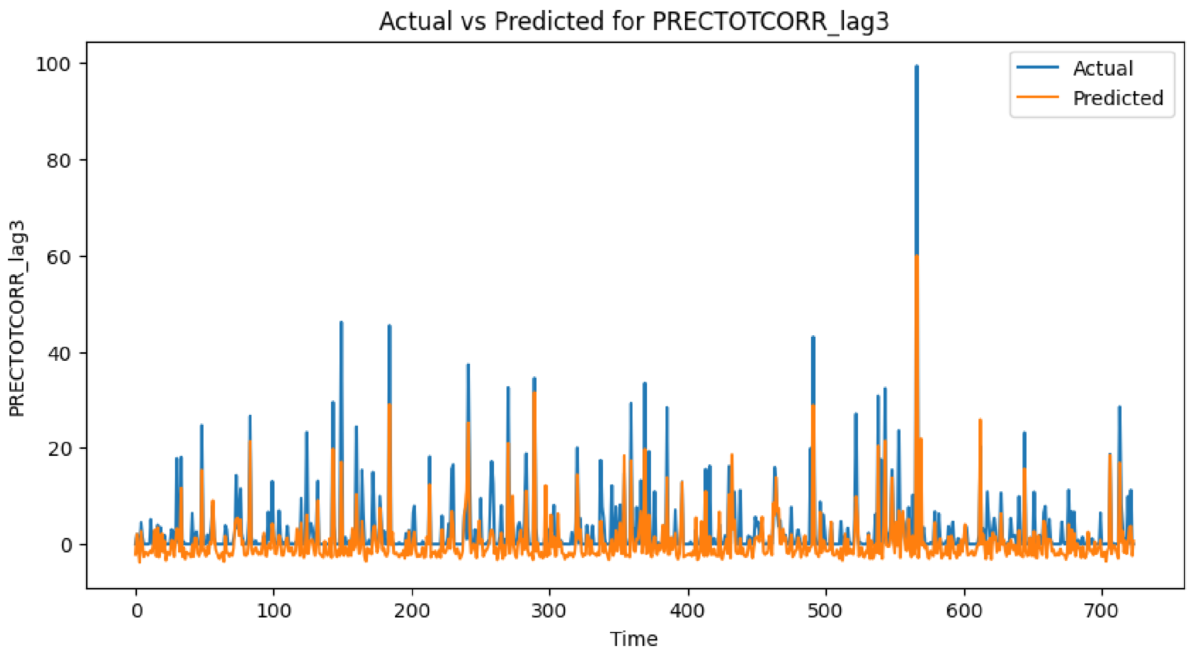
Actual vs Predicted for WS10M_lag1
Actual
Predicted
WS10M_lag1
Time

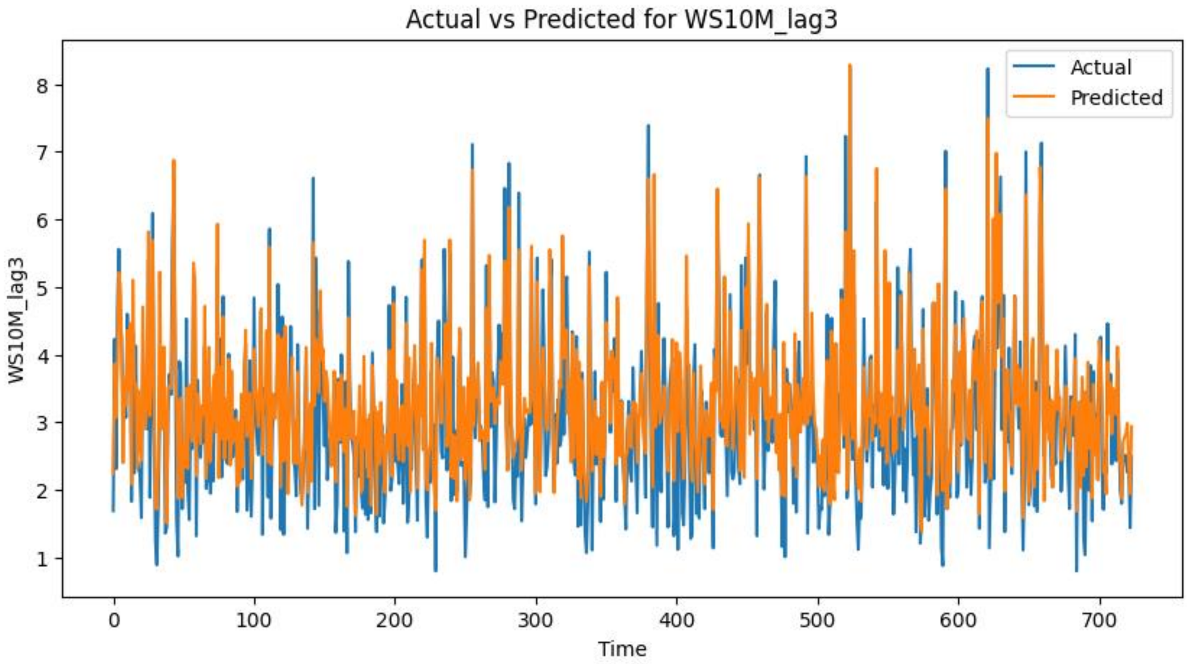


Actual vs Predicted for PRECTOTCORR_lag2
Actual
Predicted
PRECTOTCORR_lag2
Time



Actual vs Predicted for WS10M_lag2
Actual
Predicted
WS10M_lag2
Time


Actual vs Predicted for PRECTOTCORR_lag3
Actual
Predicted
PRECTOTCORR_lag3
100
80
60
40
20
0
0
100
200
300
400
500
600
700
Time


Actual vs Predicted for WS10M_lag3
Actual
Predicted
WS10M_lag3
8
7
6
5
4
3
2
1
0
100
200
300
400
500
600
700
Time