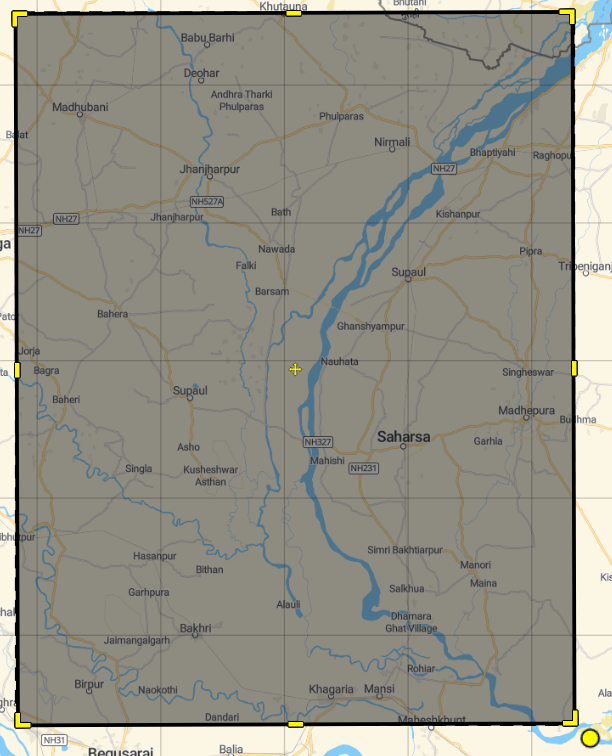
**CSV Parameterd for Kosi Region:**

**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**

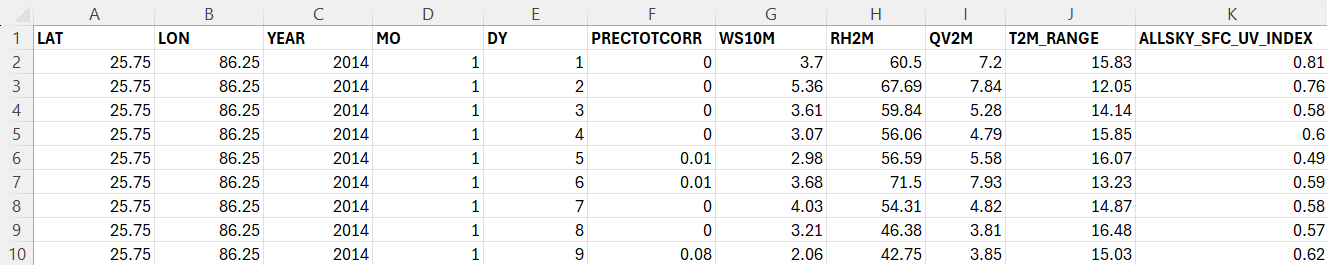
**Daily average dataset used for the aoi coordinates** westlimit=86.2461; southlimit=25.4653; eastlimit=86.7172; northlimit=26.2192 with duration= 2014 – 2023 (10 YEARS)

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:**



**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

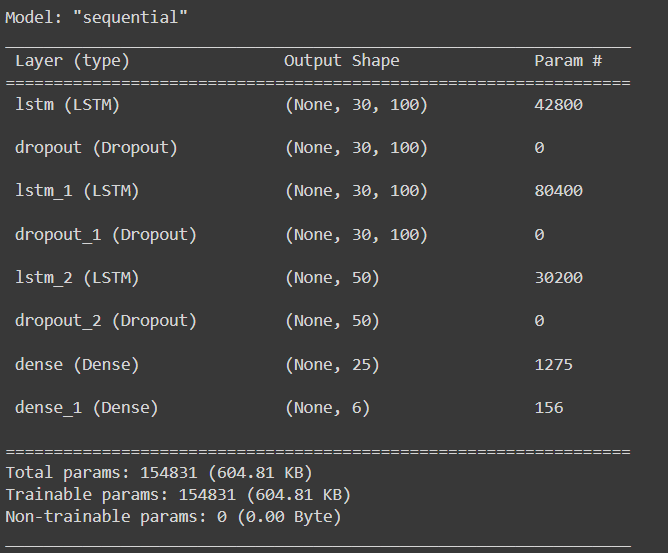
X, y = create\_sequences(scaled\_data, SEQ\_LENGTH)

**# 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**
2. # Define the LSTM model
3. model = Sequential()
4. model.add(LSTM(100, return\_sequences=True, input\_shape=(SEQ\_LENGTH, 6)))
5. model.add(Dropout(0.3))
6. model.add(LSTM(100, return\_sequences=True))
7. model.add(Dropout(0.3))
8. model.add(LSTM(50))
9. model.add(Dropout(0.3))
10. model.add(Dense(25))
11. model.add(Dense(6))  # Output layer
12. # Compile the model
13. model.compile(optimizer='adam', loss='mean\_squared\_error')
14. model.summary()



**RESULTS:**

A graph with numbers and lines

Description automatically generated

PRECTOTCORR- MSE: 0.005509878491951068, MAE: 0.033541882158388985, RMSE: 0.07422855577169118, R²: 0.20674122660768435

WS10M - MSE: 0.011977256002227762, MAE: 0.08003778268565791, RMSE: 0.10944065059303952, R²: 0.12330569498672339

RH2M - MSE: 0.004811742026239228, MAE: 0.05238782780455773, RMSE: 0.06936672131677572, R²: 0.9094222955794752

QV2M - MSE: 0.004265433783190752, MAE: 0.05170731303653358, RMSE: 0.06531028849416264, R²: 0.9431018518443797

T2M\_RANGE - MSE: 0.007029784658533242, MAE: 0.06434565153712091, RMSE: 0.08384381109260983, R²: 0.8353440252279535

ALLSKY\_SFC\_UV\_INDEX- MSE: 0.009567821386559138, MAE: 0.021783227585812363, RMSE: 0.09781524107499372, R²: -0.004220316750685127

A screenshot of a graph

Description automatically generated

**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),

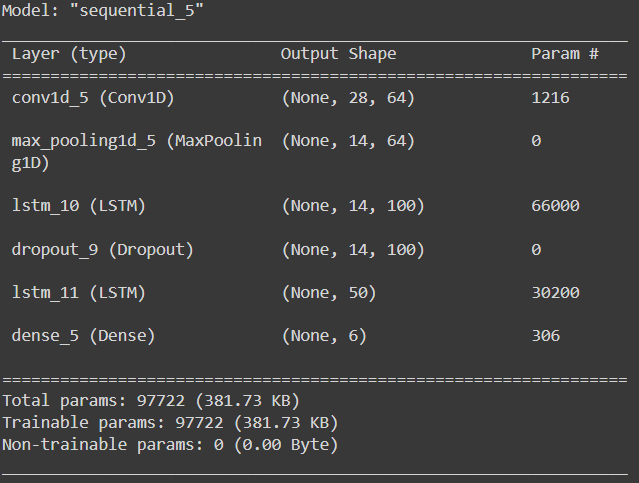
    LSTM(50),

    Dense(len(parameters))

])

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

model.summary()



**RESULTS:**

A graph with numbers and lines

Description automatically generated

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

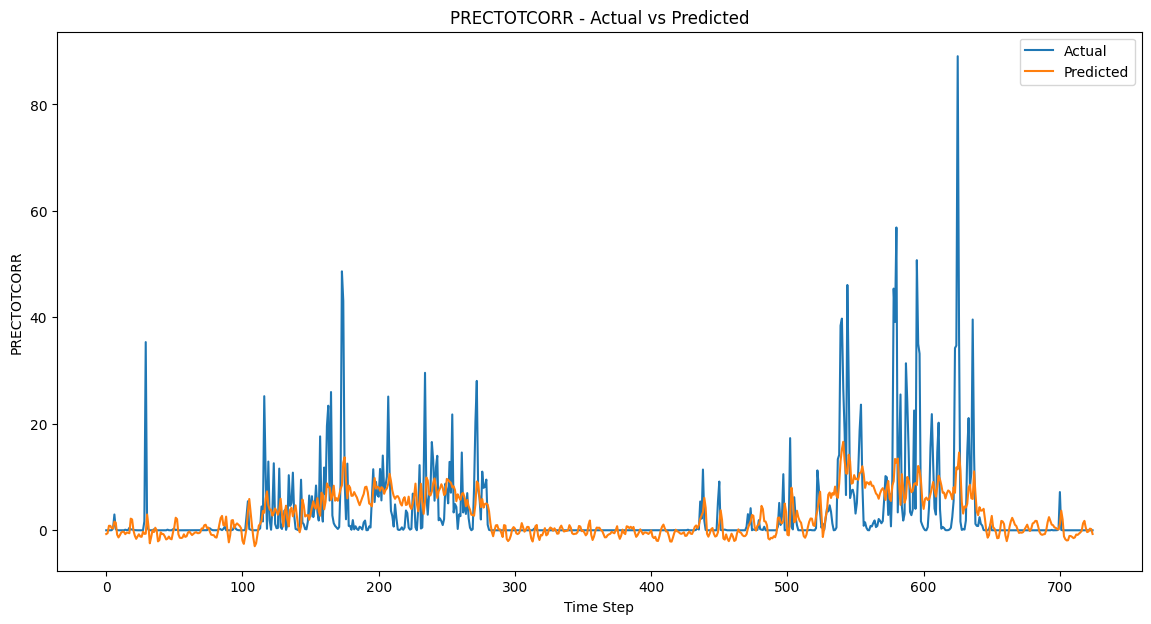
WS10M - MSE: 1.3213525861292739, MAE: 0.8622876688463934, RMSE: 1.1495010161497352, R2: 0.21783208373339302

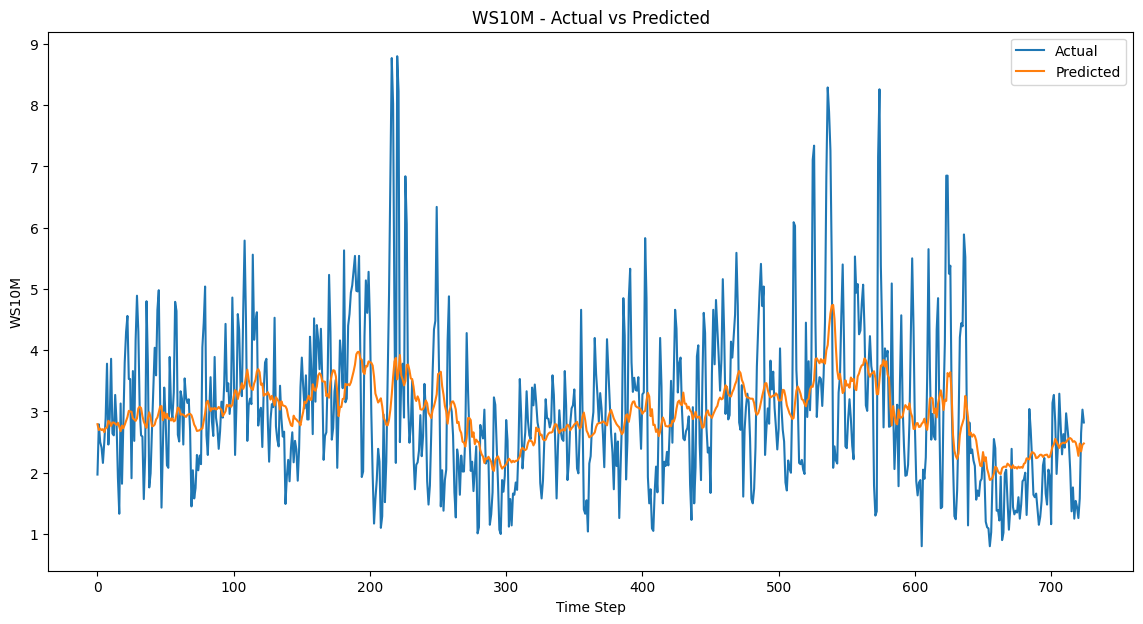
RH2M - MSE: 39.115718962057606, MAE: 4.758565859880119, RMSE: 6.254256067835535, R2: 0.9026510320820771

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





A graph showing a line of blue and orange lines

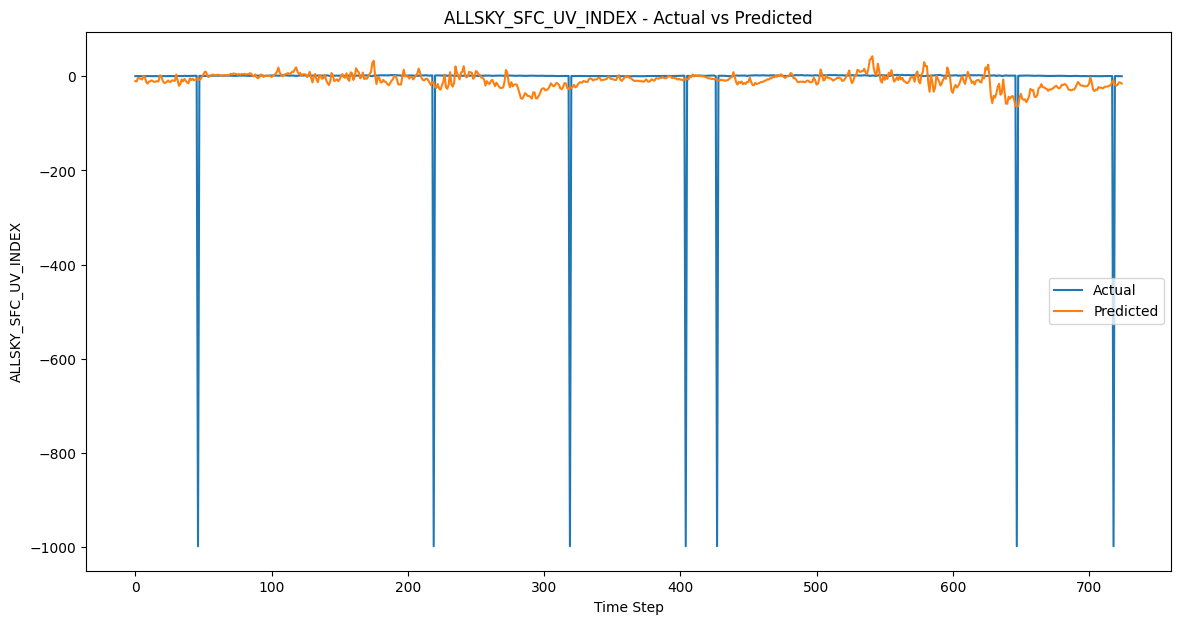
Description automatically generated

A graph of blue and orange lines

Description automatically generated

A graph of a graph

Description automatically generated with medium confidence



**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))

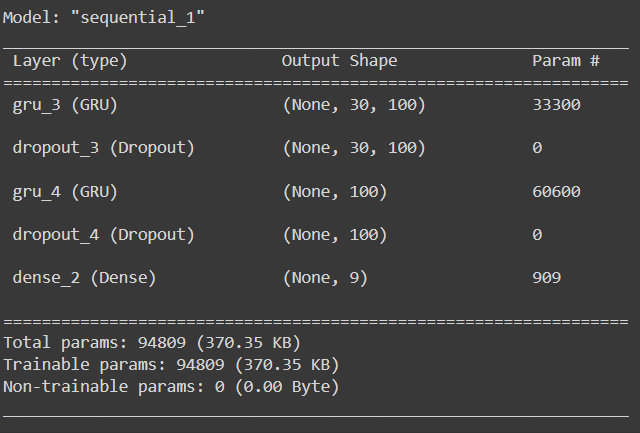
model.add(GRU(100))

model.add(Dropout(0.2))

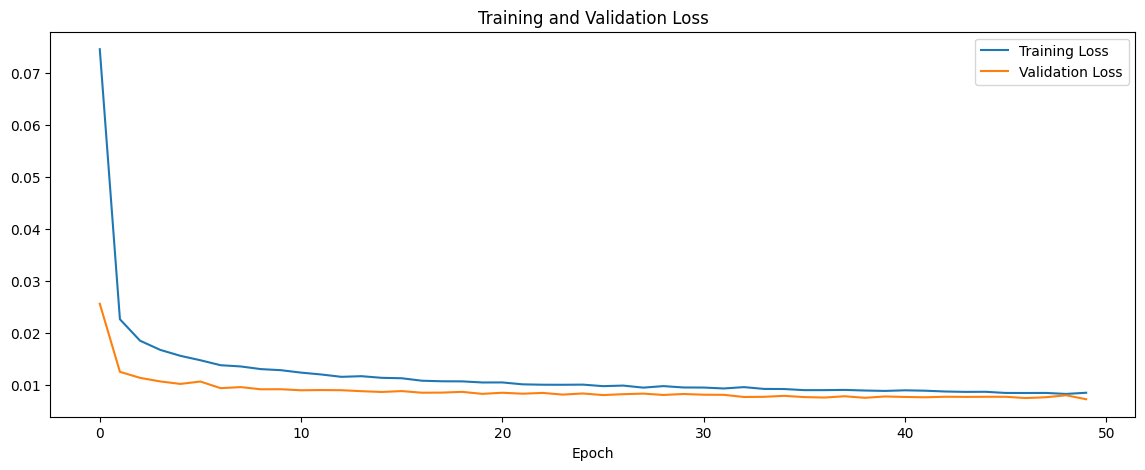
model.add(Dense(X\_train.shape[2]))

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

model.summary()



**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

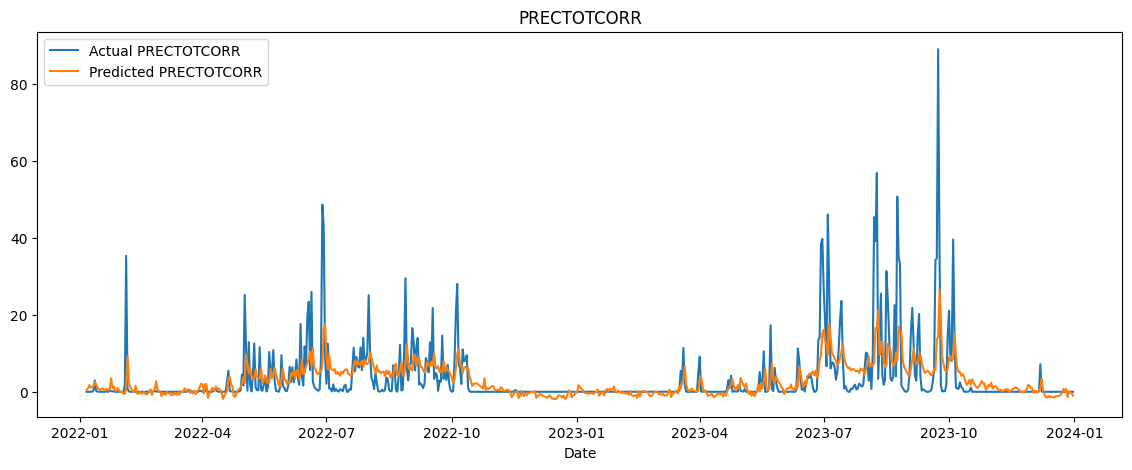
T2M\_RANGE - R-squared: 0.8629443025780161

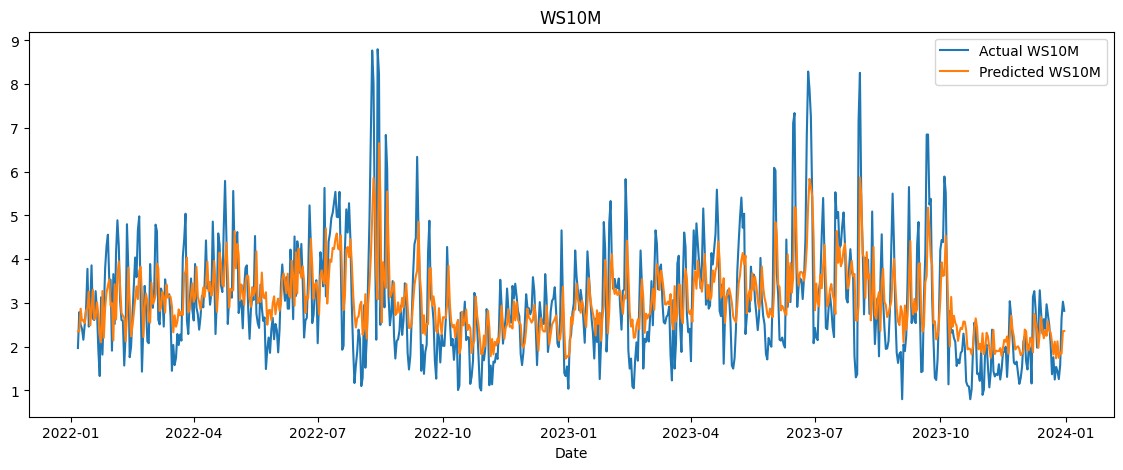
ALLSKY\_SFC\_UV\_INDEX - MSE: 10223.685548752297

ALLSKY\_SFC\_UV\_INDEX - MAE: 40.58522459114832

ALLSKY\_SFC\_UV\_INDEX - RMSE: 101.11224232877193

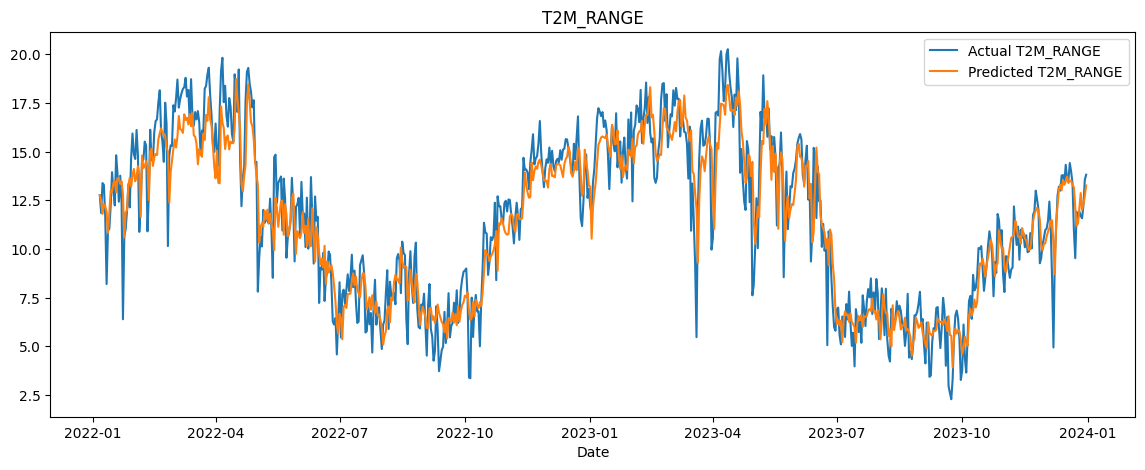
ALLSKY\_SFC\_UV\_INDEX - R-squared: -0.06824605528347116



A graph with blue and orange lines

Description automatically generated

A graph with blue and orange lines

Description automatically generatedA graph with blue and orange lines

Description automatically generated

**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()

A screenshot of a computer program

Description automatically generated

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:**

A graph of a person with a blue line

Description automatically generated with medium confidence

**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

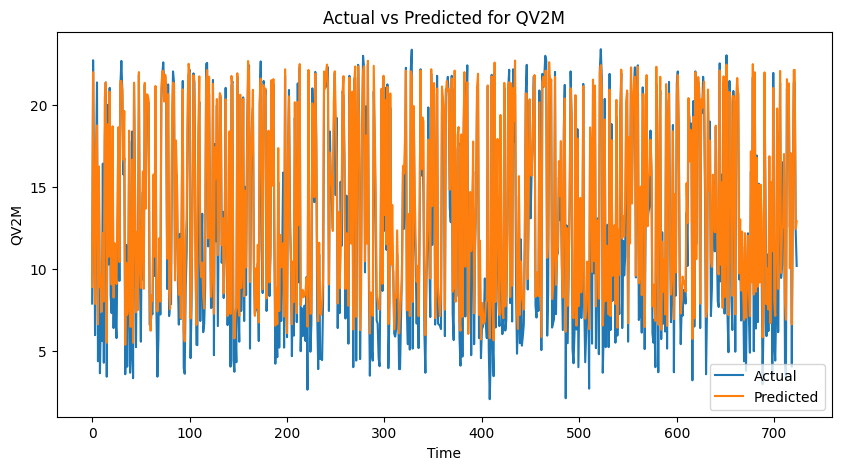
A graph with blue and orange lines

Description automatically generated

A graph with blue and orange lines

Description automatically generatedA graph showing a graph of blue and orange lines

Description automatically generated with medium confidence



A graph with blue and orange lines

Description automatically generatedA graph with blue and orange lines

Description automatically generated

**AFTER LAG:**

A graph with blue and orange lines

Description automatically generated

A graph with blue and orange lines

Description automatically generated

A graph with blue and orange lines

Description automatically generated

A graph of blue and orange lines

Description automatically generated

A graph with blue and orange lines

Description automatically generated

