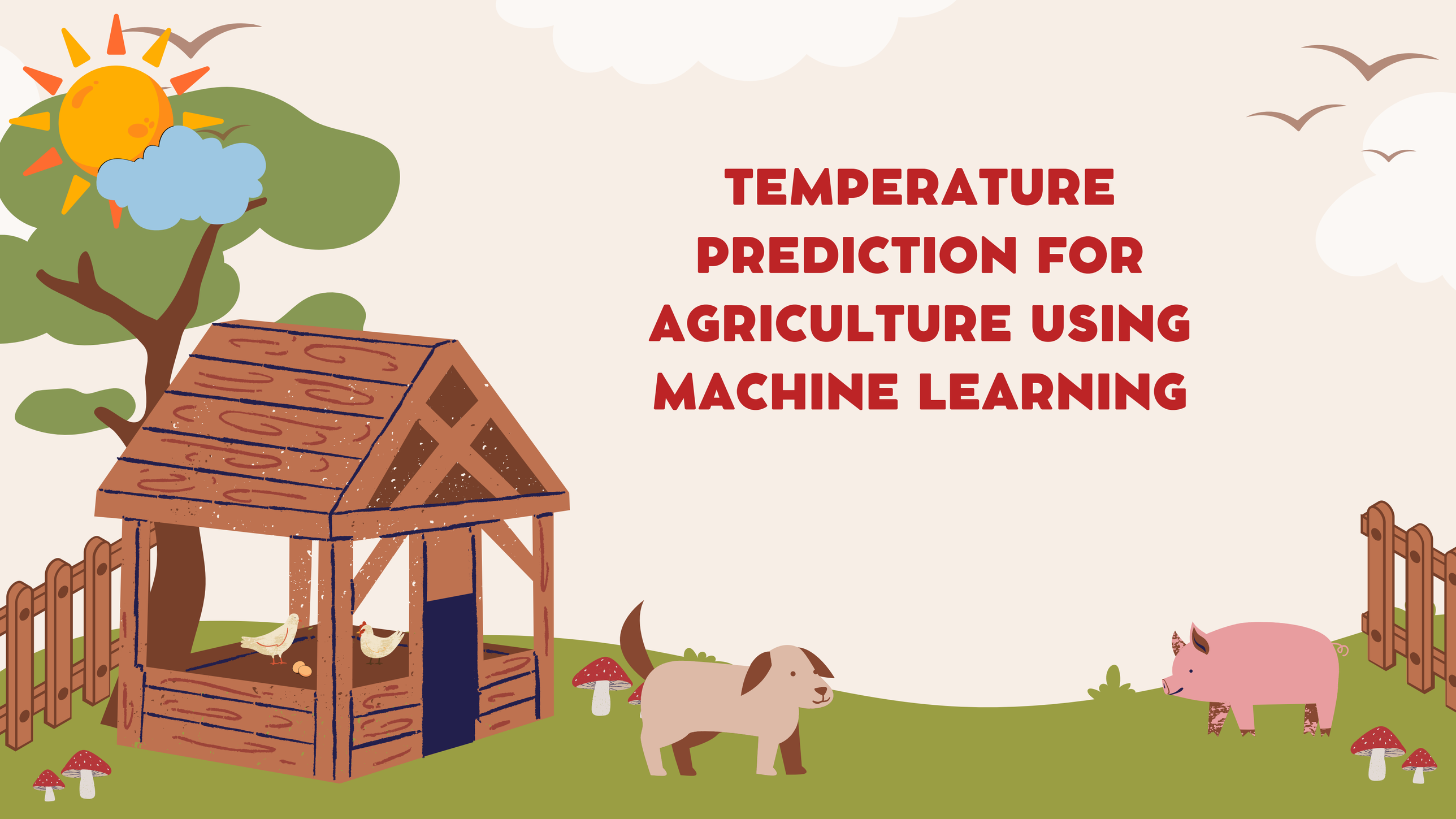
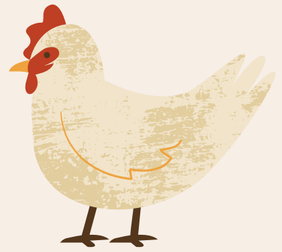


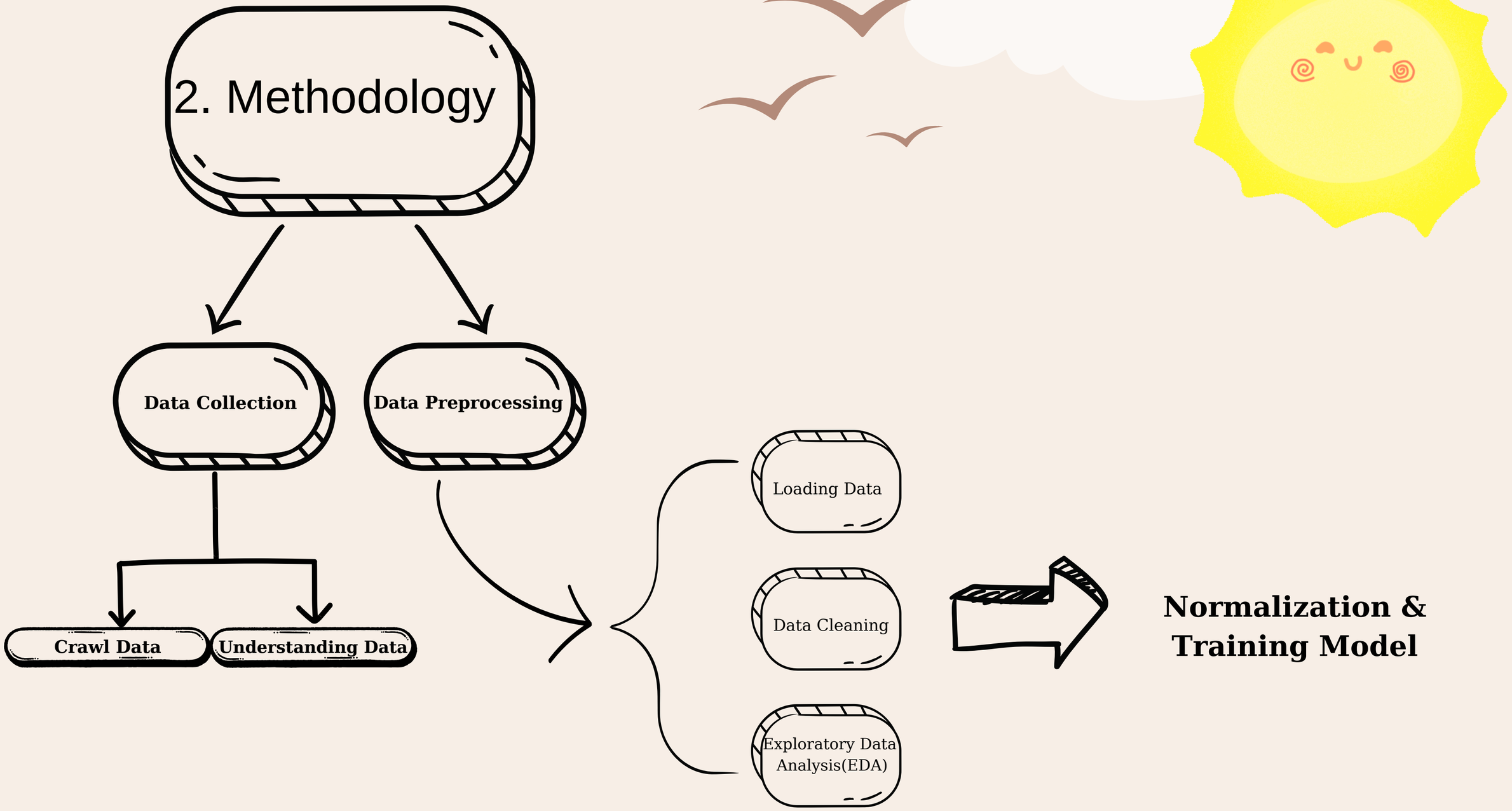
TEMPERATURE PREDICTION FOR AGRICULTURE USING MACHINE LEARNING

Topic Outline



1. Introduction

2. Methodology



2.1 Data Collection

Requests

Use the API from: visualcrossing.com

2010 to 2024 Thach Hoa area

Crawl Data

Understanding Data

- tempmax, tempmin, temp: highest lowest temperature and average temperature.
- feelslikemax, feelslikemin, feelslike: actual temperature versus felt temperature.
- humidity: Humidity during the day.
- dew: dewpoint
- precip (precipitation): This is the actual amount of rainwater that falls or reaches the ground during the day.
- precipprob (precipitation probability): This is the probability of rain during the day.
- precipcover (precipitation coverage): This is the percentage of area of a specific area covered by rain.
- preciptype: type of rain
- snow, snowdepth: snow and snow depth
- solar radiation, solar energy: solar radiation and solar energy during the day
- windspeed, winddir: Wind speed and direction can also impact rainy conditions.
- visibility: Visibility.
- uv index: UV rays
- severerisk: Severe weather conditions such as storms, flash floods
- sunrise: sunrise
- description, icon, stations: weather status of the day such as: rainy, windy, cloudy sky

2.2 Data Preprocessing

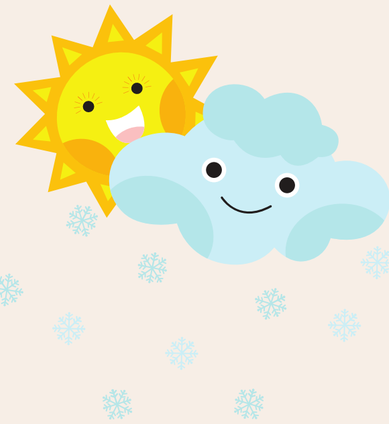
+) Loading Data

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

```
dataa = pd.read_excel('dataaaset_weather.xlsx')
```

name	datetime	tempmax	tempmin	temp	feelslikemax	feelslikemin	feelslike	dew	humidity	...	solarenergy	uvindex	severerisk	sunrise	sunset	moonphase	conditions	description	icon	stations
Xã Thạch Hòa, Huyện Thạch Thất, Việt Nam	2010-01-01	18.5	16.0	16.6	18.5	16.0	16.6	16.0	96.6	...	9.7	4	NaN	2010-01-01T06:35:03	2010-01-01T17:27:49	0.50	Rain, Overcast	Cloudy skies throughout the day with rain.	rain	48820099999,48831099999,VVNB
Xã Thạch Hòa, Huyện Thạch Thất, Việt Nam	2010-01-02	18.0	15.8	16.8	18.0	15.8	16.8	15.5	92.2	...	10.2	5	NaN	2010-01-02T06:35:22	2010-01-02T17:28:26	0.55	Rain, Overcast	Cloudy skies throughout the day with rain in t...	rain	48820099999,VVNB

+) Data Cleaning



```
dataa.columns
```

```
Index(['name', 'datetime', 'tempmax', 'tempmin', 'temp', 'feelslikemax',
       'feelslikemin', 'feelslike', 'dew', 'humidity', 'precip', 'precipprob',
       'precipcover', 'preciptype', 'snow', 'snowdepth', 'windspeed',
       'winddir', 'visibility', 'solarradiation', 'solarenergy', 'uvindex',
       'severerisk', 'sunrise', 'sunset', 'moonphase', 'conditions',
       'description', 'icon', 'stations'],
      dtype='object')
```

```
dataa.drop(['name', 'snow','temp','preciptype','feelslikemax', 'feelslikemin', 'feelslike','precipprob', 'snowdepth', 'visibility', 'uvindex', 'severerisk',
            'sunrise', 'sunset', 'moonphase', 'conditions', 'description', 'stations', 'icon'], inplace=True, axis=1)
```

```
dataa.head(10)
```

	datetime	tempmax	tempmin	dew	humidity	precip	precipcover	windspeed	winddir	solarradiation	solarenergy
0	2010-01-01	18.5	16.0	16.0	96.6	1.2	50.00	14.8	44.4	112.5	9.7
1	2010-01-02	18.0	15.8	15.5	92.2	0.2	8.33	13.0	37.3	120.3	10.2
2	2010-01-03	22.0	17.0	15.7	79.6	0.0	0.00	13.0	85.1	135.5	11.7
3	2010-01-04	22.0	18.0	17.6	87.8	0.0	0.00	14.8	104.3	196.5	17.0
4	2010-01-05	27.0	19.9	19.8	86.1	0.0	0.00	18.4	117.6	208.5	18.0
5	2010-01-06	23.0	17.8	16.7	81.5	6.9	62.50	31.3	55.8	151.9	13.0
6	2010-01-07	18.0	13.3	13.4	85.2	1.4	58.33	20.5	43.7	82.5	7.0
7	2010-01-08	19.0	13.0	10.0	68.8	0.7	20.83	13.0	55.0	188.9	16.1
8	2010-01-09	18.5	13.0	14.5	86.5	0.0	0.00	18.4	85.8	173.5	15.0
9	2010-01-10	24.0	16.9	17.0	87.0	0.0	0.00	20.5	350.5	209.3	18.2

```
dataa.shape
```

```
(5128, 11)
```

```
dataa.isnull().sum()
```

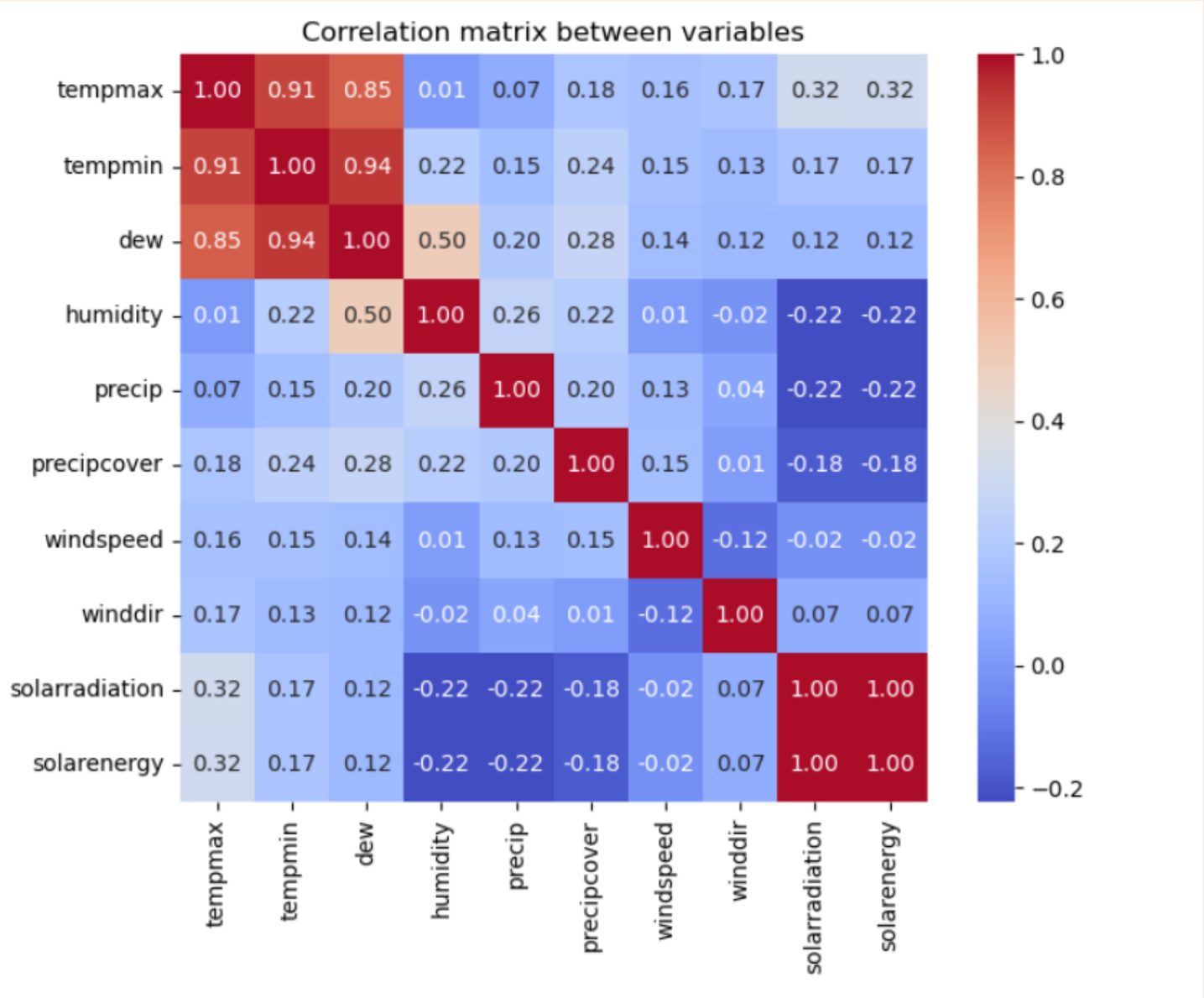
```
datetime          0
tempmax           0
tempmin           0
dew               0
humidity          0
precip            0
precipcover       0
windspeed         0
winddir           0
solarradiation    0
solarenergy       0
dtype: int64
```

Check for missing values

```
dataa.dtypes
```

```
datetime           object
tempmax           float64
tempmin           float64
dew               float64
humidity          float64
precip            float64
precipcover       float64
windspeed         float64
winddir           float64
solarradiation    float64
solarenergy       float64
dtype: object
```

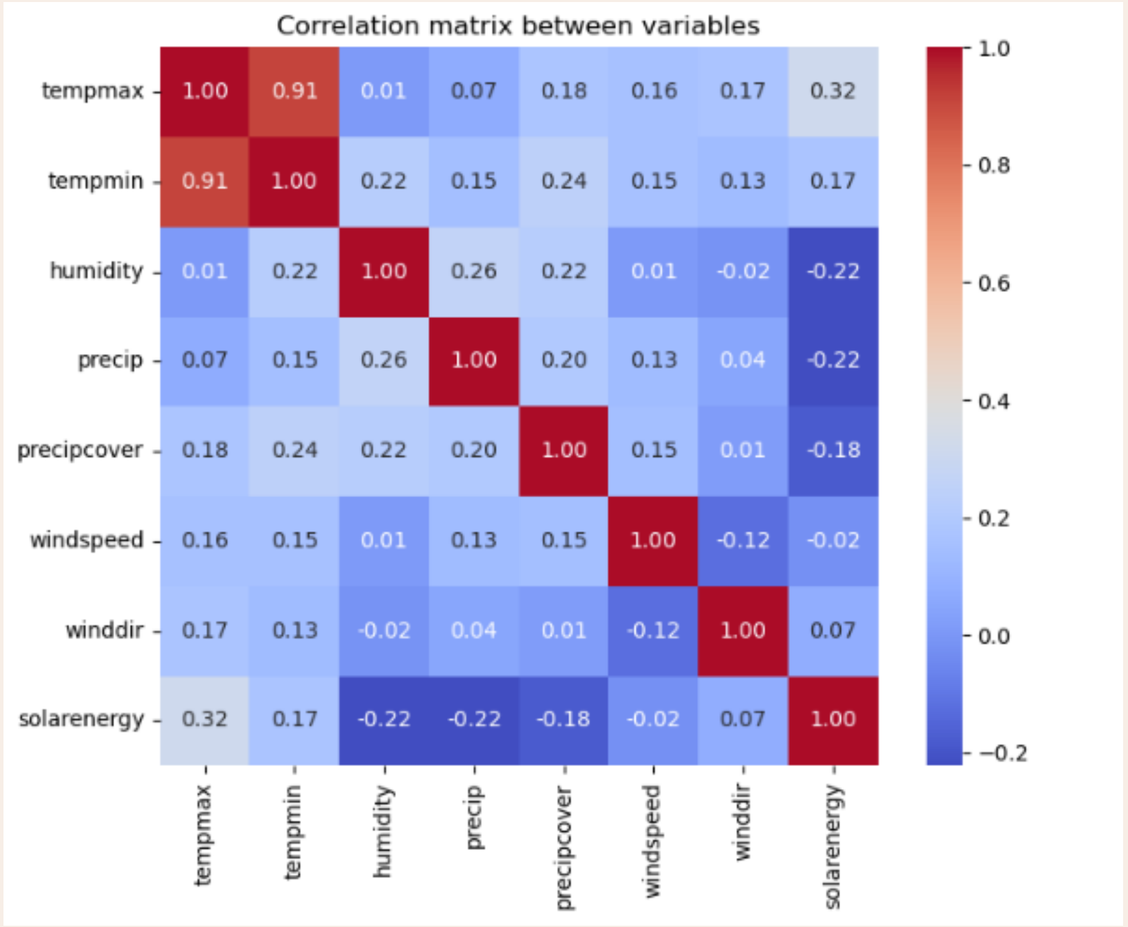
+) Exploratory Data Analysis(EDA)



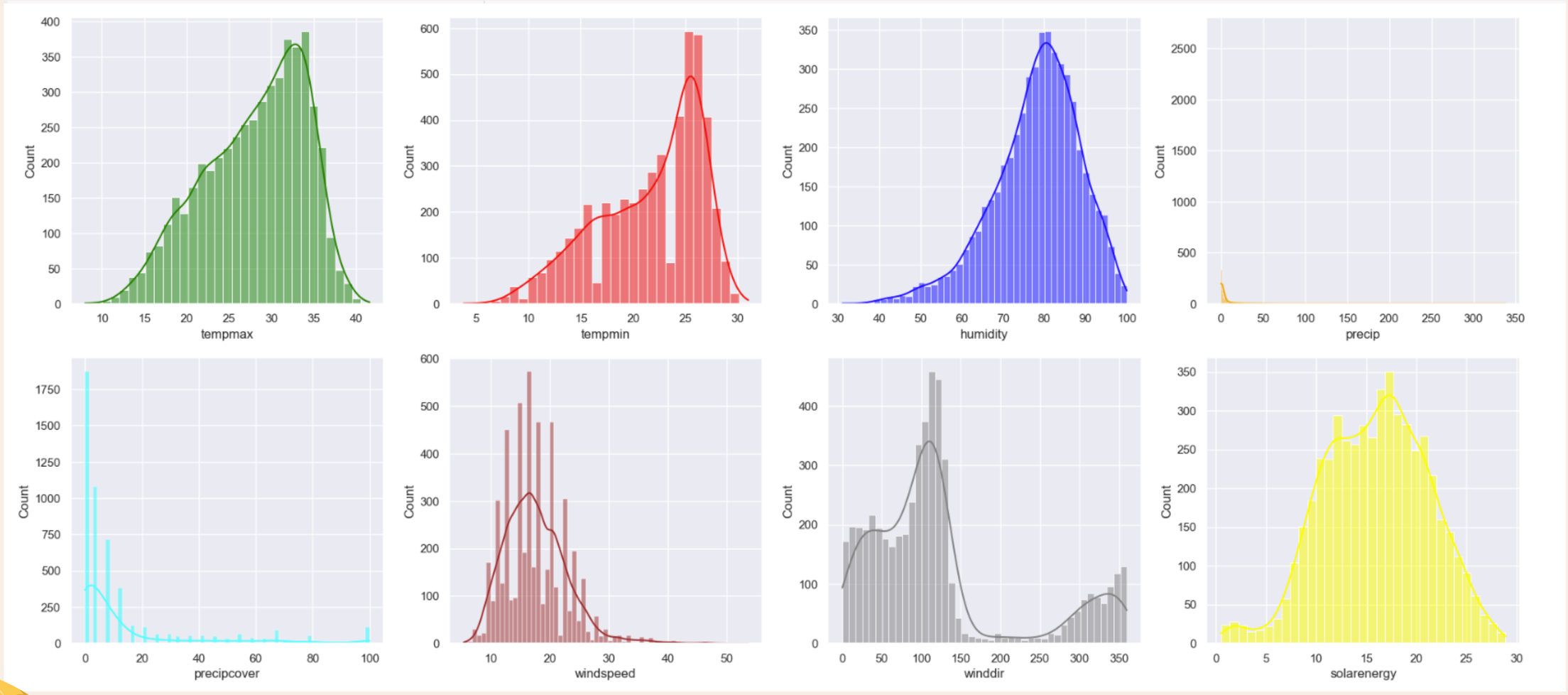
```
weather = dataa.drop(['solarradiation', 'dew'], axis=1)
```

```
weather.head(1)
```

	datetime	tempmax	tempmin	humidity	precip	precipcover	windspeed	winddir	solarenergy
0	2010-01-01	18.5	16.0	96.6	1.2	50.0	14.8	44.4	9.7



Correlation of data after deletion

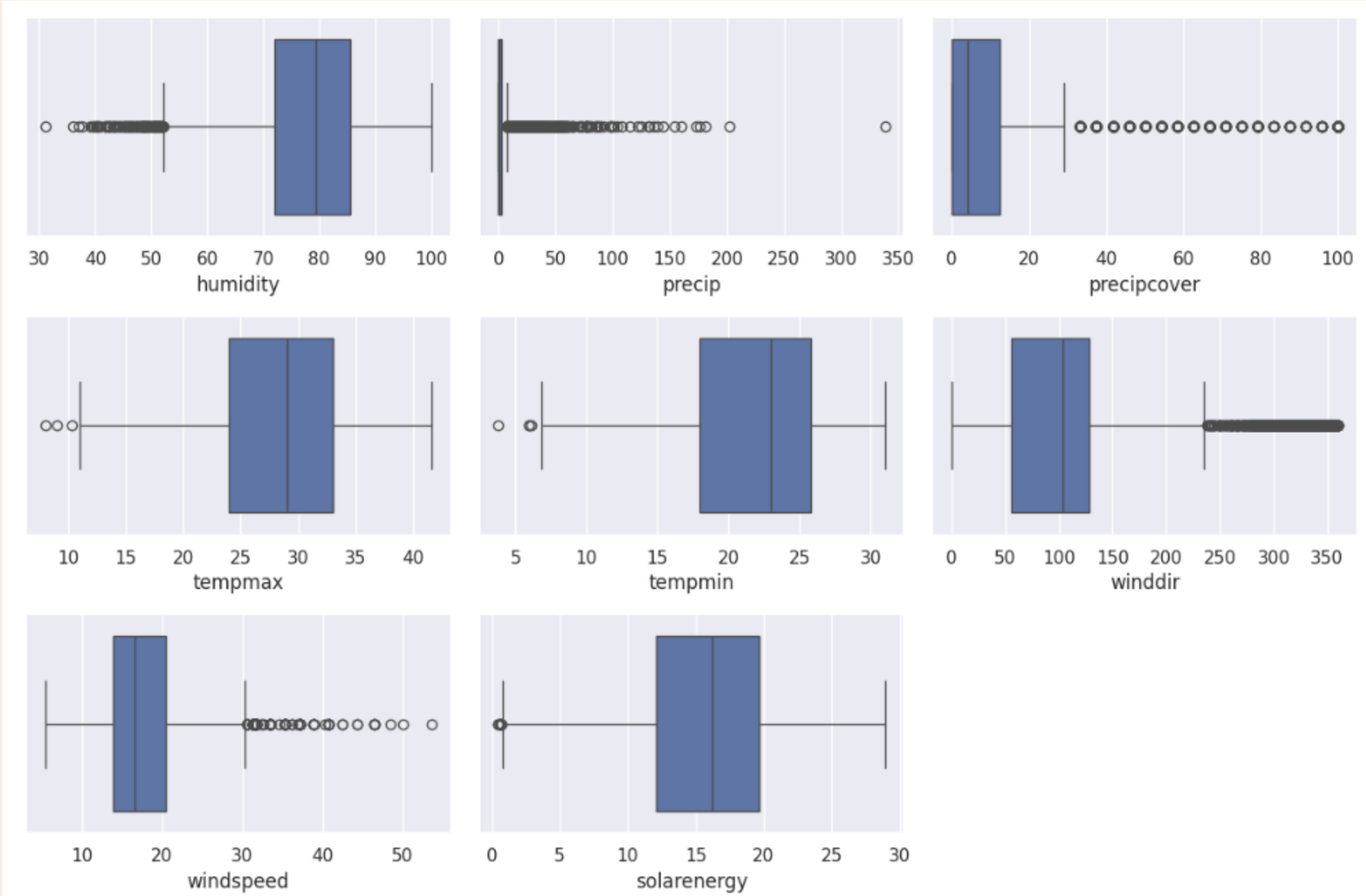


Visualizing using histplot

```
Q1 = weather.quantile(0.25)
Q3 = weather.quantile(0.75)
IQR = Q3 - Q1

# Cal outliers
outliers = ((weather < (Q1 - 1.5 * IQR)) | (weather > (Q3 + 1.5 * IQR)))

# Calculate the total outlier values for each column
outliers_sum = outliers.sum()

print("Total outlier values for each column:")
print(outliers_sum)
```

```
Total outlier values for each column:
datetime         0
humidity       118
precip         757
precipcover    712
solarenergy      8
tempmax          3
tempmin          4
winddir        836
windspeed      105
dtype: int64
```

Check for outliers



Visualizing using subplot

```python
# cal mean
mean_values = weather.mean()

# Replace outliers with the average value of the corresponding column
weather['humidity'] = np.where(outliers['humidity'], mean_values['humidity'], weather['humidity'])
weather['precip'] = np.where(outliers['precip'], mean_values['precip'], weather['precip'])
weather['precipcover'] = np.where(outliers['precipcover'], mean_values['precipcover'], weather['precipcover'])
weather['solarenergy'] = np.where(outliers['solarenergy'], mean_values['solarenergy'], weather['solarenergy'])
weather['tempmax'] = np.where(outliers['tempmax'], mean_values['tempmax'], weather['tempmax'])
weather['tempmin'] = np.where(outliers['tempmin'], mean_values['tempmin'], weather['tempmin'])
weather['winddir'] = np.where(outliers['winddir'], mean_values['winddir'], weather['winddir'])
weather['windspeed'] = np.where(outliers['windspeed'], mean_values['windspeed'], weather['windspeed'])

# Recheck for outliers after performing the replacement
outliers_after_replace = ((weather < (Q1 - 1.5 * IQR)) | (weather > (Q3 + 1.5 * IQR)))
outliers_sum_after_replace = outliers_after_replace.sum()
print("\nTotal outliers for each column after replacing with mean:")
print(outliers_sum_after_replace)
```

```
Total outliers for each column after replacing with mean:
datetime       0
humidity       0
precip         0
precipcover    0
solarenergy    0
tempmax        0
tempmin        0
winddir        0
windspeed      0
dtype: int64
```

Handle outliers values

```
weather['datetime'] = pd.to_datetime(weather['datetime'])
```

```
weather.set_index('datetime', inplace=True)
```

```
weather.head()
```

datetime	tempmax	tempmin	humidity	precip	precipcover	windspeed	winddir	solarenergy
2010-01-01	18.5	16.0	96.6	1.2	13.411195	14.8	44.4	9.7
2010-01-02	18.0	15.8	92.2	0.2	8.330000	13.0	37.3	10.2
2010-01-03	22.0	17.0	79.6	0.0	0.000000	13.0	85.1	11.7
2010-01-04	22.0	18.0	87.8	0.0	0.000000	14.8	104.3	17.0
2010-01-05	27.0	19.9	86.1	0.0	0.000000	18.4	117.6	18.0

Convert datetime column

Normalization data

```
] from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Normalize data using the Min-Max Scaling method for columns that need to be normalized
weather_scaled = scaler.fit_transform(weather[['humidity', 'precip', 'tempmax', 'tempmin', 'winddir', 'windspeed', 'precipcover', 'solarenergy']])

# Create a new DataFrame from normalized data
weather_scaled_df = pd.DataFrame(weather_scaled, columns=['humidity', 'precip', 'tempmax', 'tempmin', 'winddir', 'windspeed', 'precipcover', 'solarenergy'])

print(weather_scaled_df.head())
```

```
   humidity    precip   tempmax   tempmin   winddir  windspeed  precipcover  \
0  0.928870  0.166667  0.245098  0.380165  0.188486   0.377510     0.459760
1  0.836820  0.027778  0.228758  0.371901  0.158209   0.305221     0.285567
2  0.573222  0.000000  0.359477  0.421488  0.362047   0.305221     0.000000
3  0.744770  0.000000  0.359477  0.462810  0.443923   0.377510     0.000000
4  0.709205  0.000000  0.522876  0.541322  0.500640   0.522088     0.000000

   solarenergy
0     0.316726
1     0.334520
2     0.387900
3     0.576512
4     0.612100
```

2.3. Trainning model:

2.3.1 Linear Regression

2.3.2 Support Vector Regression (SVR)

2.3.3 XGBoost

2.3.1 Linear Regression

Tempmax

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd

# choose target for tempmax
target_column_tempmax = 'tempmax'
X_tempmax = weather.drop(columns=[target_column_tempmax])
y_tempmax = weather[target_column_tempmax]

# split train and test for tempmax
X_train_tempmax, X_test_tempmax, y_train_tempmax, y_test_tempmax = train_test_split(X_tempmax, y_tempmax, test_size=0.2, random_state=43)

# linear model for tempmax
model_tempmax = LinearRegression()
model_tempmax.fit(X_train_tempmax, y_train_tempmax)

# predict on test for tempmax
y_pred_test_tempmax = model_tempmax.predict(X_test_tempmax)

# predict on train
y_pred_train_tempmax = model_tempmax.predict(X_train_tempmax)

# Evaluate the model on the test set
mse_test_tempmax = mean_squared_error(y_test_tempmax, y_pred_test_tempmax)
rmse_test_tempmax = mean_squared_error(y_test_tempmax, y_pred_test_tempmax, squared=False)
r2_test_tempmax = r2_score(y_test_tempmax, y_pred_test_tempmax)

# Evaluate the model on the train set
mse_train_tempmax = mean_squared_error(y_train_tempmax, y_pred_train_tempmax)
rmse_train_tempmax = mean_squared_error(y_train_tempmax, y_pred_train_tempmax, squared=False)
r2_train_tempmax = r2_score(y_train_tempmax, y_pred_train_tempmax)
```

Tempmin

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd

# choose target for tempmin
target_column_tempmin = 'tempmin'
X_tempmin = weather.drop(columns=[target_column_tempmin])
y_tempmin = weather[target_column_tempmin]

# split train and test for tempmin
X_train_tempmin, X_test_tempmin, y_train_tempmin, y_test_tempmin = train_test_split(X_tempmin, y_tempmin, test_size=0.2, random_state=43)

# linear model for tempmin
model_tempmin = LinearRegression()
model_tempmin.fit(X_train_tempmin, y_train_tempmin)

# predict on test for tempmin
y_pred_test_tempmin = model_tempmin.predict(X_test_tempmin)

# predict on train for tempmin
y_pred_train_tempmin = model_tempmin.predict(X_train_tempmin)

# Evaluate the model on the test set
mse_test_tempmin = mean_squared_error(y_test_tempmin, y_pred_test_tempmin)
rmse_test_tempmin = mean_squared_error(y_test_tempmin, y_pred_test_tempmin, squared=False)
r2_test_tempmin = r2_score(y_test_tempmin, y_pred_test_tempmin)

# Evaluate the model on the train set
mse_train_tempmin = mean_squared_error(y_train_tempmin, y_pred_train_tempmin)
rmse_train_tempmin = mean_squared_error(y_train_tempmin, y_pred_train_tempmin, squared=False)
r2_train_tempmin = r2_score(y_train_tempmin, y_pred_train_tempmin)
```

2.3.2 Support Vector Regression (SVR)

Tempmax

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd

# Select target as tempmax
target_column_tempmax = 'tempmax'
X_tempmax = weather.drop(columns=[target_column_tempmax])
y_tempmax = weather[target_column_tempmax]

# Divide the data into training set and test set
X_train_tempmax_svr, X_test_tempmax_svr, y_train_tempmax_svr, y_test_tempmax_svr = train_test_split(X_tempmax, y_tempmax, test_size=0.2, random_state=43)

# Build the SVR model
svr_model_tempmax = SVR(kernel='rbf')  # Select the kernel function as radial basis function (RBF)
svr_model_tempmax.fit(X_train_tempmax_svr, y_train_tempmax_svr)

# Predict on the test set
y_pred_test_tempmax_svr = svr_model_tempmax.predict(X_test_tempmax_svr)

# Predict on the training set
y_pred_train_tempmax_svr = svr_model_tempmax.predict(X_train_tempmax_svr)

# Evaluate the SVR model on the test set
mse_test_tempmax_svr = mean_squared_error(y_test_tempmax_svr, y_pred_test_tempmax_svr)
rmse_test_tempmax_svr = np.sqrt(mse_test_tempmax_svr)
r2_test_tempmax_svr = r2_score(y_test_tempmax_svr, y_pred_test_tempmax_svr)

# Evaluate the SVR model on the train set
mse_train_tempmax_svr = mean_squared_error(y_train_tempmax_svr, y_pred_train_tempmax_svr)
rmse_train_tempmax_svr = np.sqrt(mse_train_tempmax_svr)
r2_train_tempmax_svr = r2_score(y_train_tempmax_svr, y_pred_train_tempmax_svr)
```

Tempmin

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import numpy as np

# choose target as tempmin
target_column_tempmin = 'tempmin'
X_tempmin = weather.drop(columns=[target_column_tempmin])
y_tempmin = weather[target_column_tempmin]

# Divide the data into training set and test set
X_train_tempmin_svr, X_test_tempmin_svr, y_train_tempmin_svr, y_test_tempmin_svr = train_test_split(X_tempmin, y_tempmin, test_size=0.2, random_state=43)

# Build SVR model for tempmin
svr_model_tempmin = SVR(kernel='rbf')  # Select the kernel function as radial basis function (RBF)
svr_model_tempmin.fit(X_train_tempmin_svr, y_train_tempmin_svr)

# Predict on the test set
y_pred_test_tempmin_svr = svr_model_tempmin.predict(X_test_tempmin_svr)

# Predict on the training set
y_pred_train_tempmin_svr = svr_model_tempmin.predict(X_train_tempmin_svr)

# Evaluate the SVR model on the test set
mse_test_tempmin_svr = mean_squared_error(y_test_tempmin_svr, y_pred_test_tempmin_svr)
rmse_test_tempmin_svr = np.sqrt(mse_test_tempmin_svr)
r2_test_tempmin_svr = r2_score(y_test_tempmin_svr, y_pred_test_tempmin_svr)

# Evaluate the SVR model on the training set
mse_train_tempmin_svr = mean_squared_error(y_train_tempmin_svr, y_pred_train_tempmin_svr)
rmse_train_tempmin_svr = np.sqrt(mse_train_tempmin_svr)
r2_train_tempmin_svr = r2_score(y_train_tempmin_svr, y_pred_train_tempmin_svr)
```

2.3.3 XGBoost

Tempmax

```python
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd

# Select target as tempmax
target_column_tempmax = 'tempmax'
X_tempmax = weather.drop(columns=[target_column_tempmax])
y_tempmax = weather[target_column_tempmax]

# Divide the data into training set and test set
X_train_tempmax, X_test_tempmax, y_train_tempmax, y_test_tempmax = train_test_split(X_tempmax, y_tempmax, test_size=0.2, random_state=43)

# Build the XGBoost model
xgb_model_tempmax = xgb.XGBRegressor()
xgb_model_tempmax.fit(X_train_tempmax, y_train_tempmax)

# Predict on the test set
y_pred_test_tempmax_xgb = xgb_model_tempmax.predict(X_test_tempmax)

# Predict on the training set
y_pred_train_tempmax_xgb = xgb_model_tempmax.predict(X_train_tempmax)


# Evaluate the XGBoost model on the test set
mse_test_tempmax_xgb = mean_squared_error(y_test_tempmax, y_pred_test_tempmax_xgb)
rmse_test_tempmax_xgb = np.sqrt(mse_test_tempmax_xgb)
r2_test_tempmax_xgb = r2_score(y_test_tempmax, y_pred_test_tempmax_xgb)

# Evaluate the XGBoost model on the training set
mse_train_tempmax_xgb = mean_squared_error(y_train_tempmax, y_pred_train_tempmax_xgb)
rmse_train_tempmax_xgb = np.sqrt(mse_train_tempmax_xgb)
r2_train_tempmax_xgb = r2_score(y_train_tempmax, y_pred_train_tempmax_xgb)
```

Tempmin

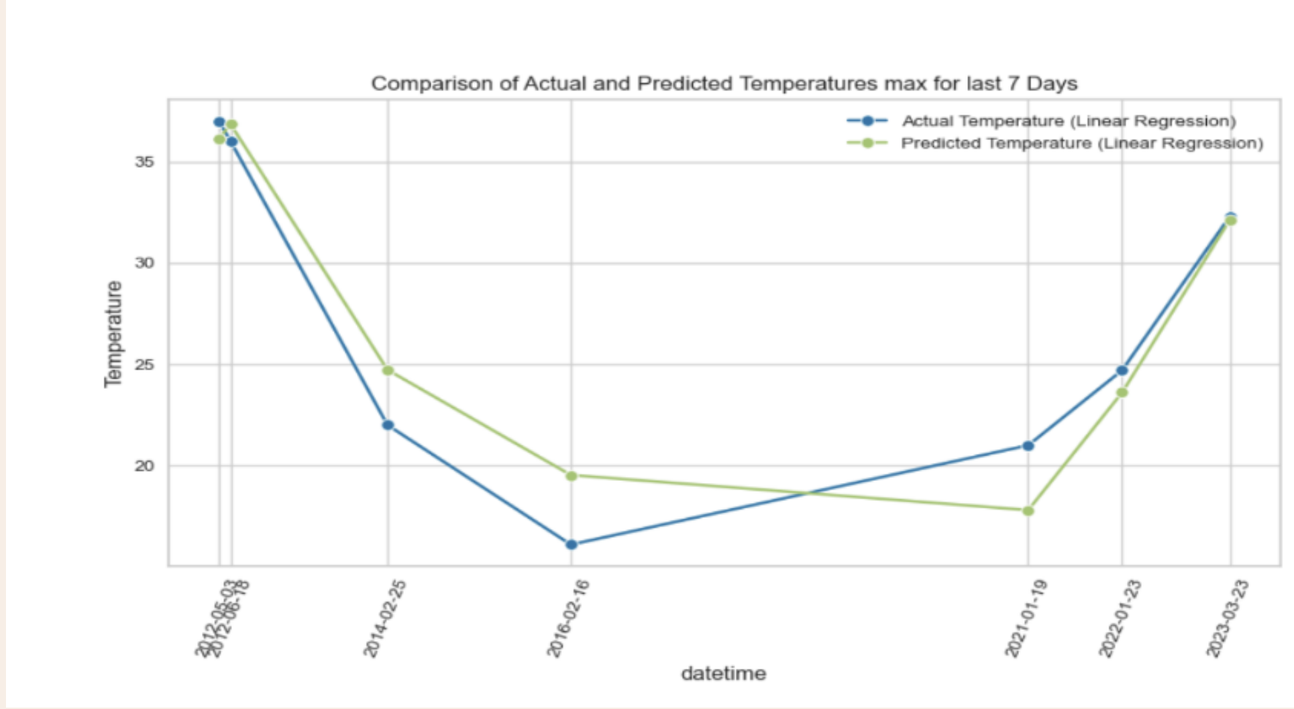
```
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
from sklearn.model_selection import train_test_split

# Select target as tempmin
target_column_tempmin = 'tempmin'
X_tempmin = weather.drop(columns=[target_column_tempmin])
y_tempmin = weather[target_column_tempmin]

# Divide the data into training set and test set
X_train_tempmin, X_test_tempmin, y_train_tempmin, y_test_tempmin = train_test_split(X_tempmin, y_tempmin, test_size=0.2, random_state=43)

# Build the XGBoost model
xgb_model_tempmin = XGBRegressor()
xgb_model_tempmin.fit(X_train_tempmin, y_train_tempmin)

# Predict on the test set
y_pred_test_tempmin_xgb = xgb_model_tempmin.predict(X_test_tempmin)

# Predict on the train set
y_pred_train_tempmin_xgb = xgb_model_tempmin.predict(X_train_tempmin)

# Evaluate the XGBoost model on the test set
mse_test_tempmin_xgb = mean_squared_error(y_test_tempmin, y_pred_test_tempmin_xgb)
rmse_test_tempmin_xgb = np.sqrt(mse_test_tempmin_xgb)
r2_test_tempmin_xgb = r2_score(y_test_tempmin, y_pred_test_tempmin_xgb)

# Evaluate the XGBoost model on the training set
mse_train_tempmin_xgb = mean_squared_error(y_train_tempmin, y_pred_train_tempmin_xgb)
r2_train_tempmin_xgb = r2_score(y_train_tempmin, y_pred_train_tempmin_xgb)
rmse_train_tempmin_xgb = np.sqrt(mse_train_tempmin_xgb)
```

3. Results

3.1 Linear Regression

```
Test Set Mean Squared Error for tempmax: 4.346374651834435
Train Set Mean Squared Error for tempmax: 4.227681560406169
Test Set Root Mean Squared Error for tempmax: 2.084796069603556
Train Set Root Mean Squared Error for tempmax: 2.056132670915515
Test Set R-squared for tempmax: 0.8814084206355618
Train Set R-squared for tempmax: 0.8809123749479472

Linear Regression Results for tempmax:
            Actual  Predicted              Model Temperature Type
datetime
2021-01-19    21.0  17.803702  Linear Regression           tempmax
2012-05-03    37.0  36.123734  Linear Regression           tempmax
2012-06-18    36.0  36.839249  Linear Regression           tempmax
2023-03-23    32.3  32.131695  Linear Regression           tempmax
2022-01-23    24.7  23.604535  Linear Regression           tempmax
2016-02-16    16.1  19.532414  Linear Regression           tempmax
2014-02-25    22.0  24.723103  Linear Regression           tempmax
```

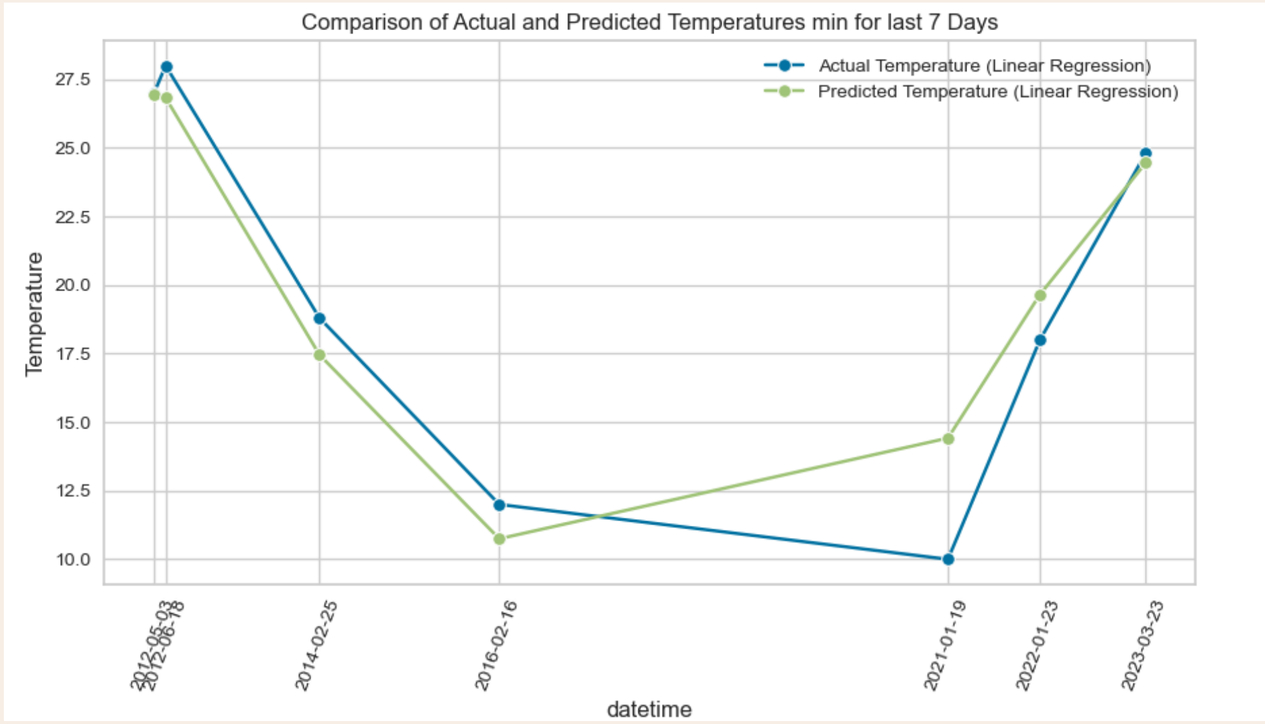
Evaluation results for tempmax



Lineplot illustration for tempmax

```
Test Set Mean Squared Error for tempmin: 3.1989451890639957
Train Set Mean Squared Error for tempmin: 3.131709188850151
Test Set Root Mean Squared Error for tempmin: 1.7885595290803142
Train Set Root Mean Squared Error for tempmin: 1.769663580698363
Test Set R-squared for tempmin: 0.8706715344713349
Train Set R-squared for tempmin: 0.8736702981127094

Linear Regression Results for tempmin:
            Actual  Predicted              Model Temperature Type
datetime
2021-01-19    10.0  14.419366  Linear Regression          tempmin
2012-05-03    27.0  26.942774  Linear Regression          tempmin
2012-06-18    28.0  26.832149  Linear Regression          tempmin
2023-03-23    24.8  24.454899  Linear Regression          tempmin
2022-01-23    18.0  19.654891  Linear Regression          tempmin
2016-02-16    12.0  10.747345  Linear Regression          tempmin
2014-02-25    18.8  17.456596  Linear Regression          tempmin
```

Evaluation results for tempmin



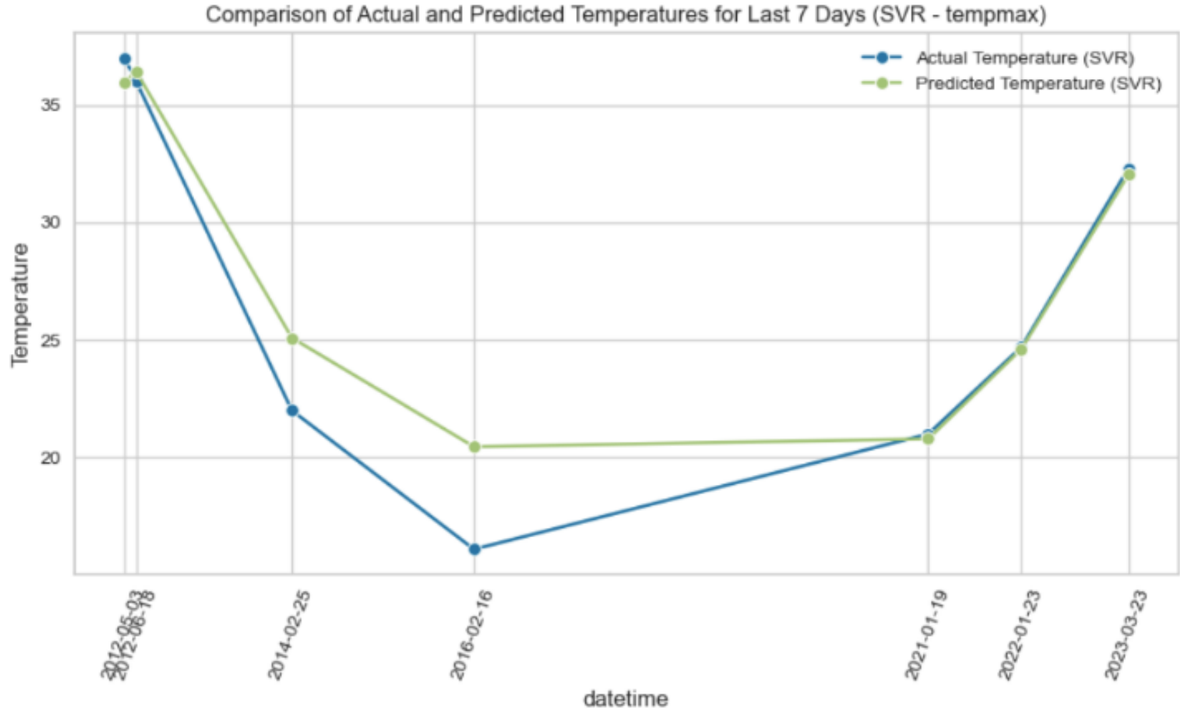
Lineplot illustration for tempmin

3. Results

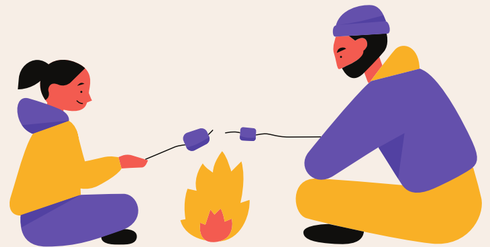
3.2 Support Vector Regression

```
Test Set Mean Squared Error for tempmax (SVR): 4.882487789519589
Train Set Mean Squared Error for tempmax (SVR): 4.943606834468043
Test Set Root Mean Squared Error for tempmax (SVR): 2.20963521639197
Train Set Root Mean Squared Error for tempmax (SVR): 2.2234223248110205
Test Set R-squared for tempmax (SVR): 0.866780481534805
Train Set R-squared for tempmax (SVR): 0.860745803888944

SVR Results for tempmax:
            Actual  Predicted Model Temperature Type
datetime
2021-01-19    21.0  20.793494   SVR          tempmax
2012-05-03    37.0  35.963231   SVR          tempmax
2012-06-18    36.0  36.417938   SVR          tempmax
2023-03-23    32.3  32.033618   SVR          tempmax
2022-01-23    24.7  24.613381   SVR          tempmax
2016-02-16    16.1  20.460785   SVR          tempmax
2014-02-25    22.0  25.087836   SVR          tempmax
```

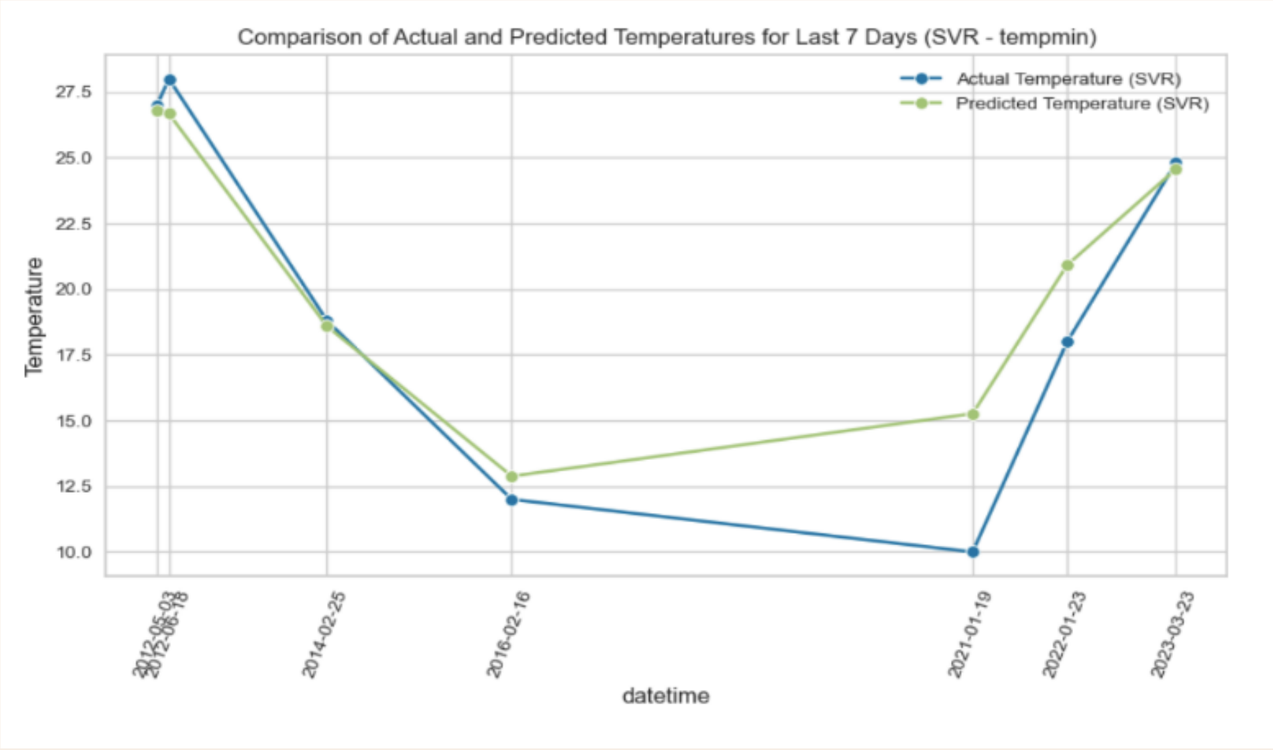
Evaluation results for tempmax



Lineplot illustration for tempmax



```
Test Set Mean Squared Error for tempmin (SVR): 3.5112543484339143
Train Set Mean Squared Error for tempmin (SVR): 3.5358816765090624
Test Set Root Mean Squared Error for tempmin (SVR): 1.8738341304485608
Train Set Root Mean Squared Error for tempmin (SVR): 1.880394021610647
Test Set R-squared for tempmin (SVR): 0.8580453524129681
Train Set R-squared for tempmin (SVR): 0.8573664247968917

SVR Results for tempmin:
            Actual  Predicted Model Temperature Type
datetime
2021-01-19    10.0  15.262272   SVR          tempmin
2012-05-03    27.0  26.802281   SVR          tempmin
2012-06-18    28.0  26.674774   SVR          tempmin
2023-03-23    24.8  24.562338   SVR          tempmin
2022-01-23    18.0  20.918021   SVR          tempmin
2016-02-16    12.0  12.884694   SVR          tempmin
2014-02-25    18.8  18.588623   SVR          tempmin
```



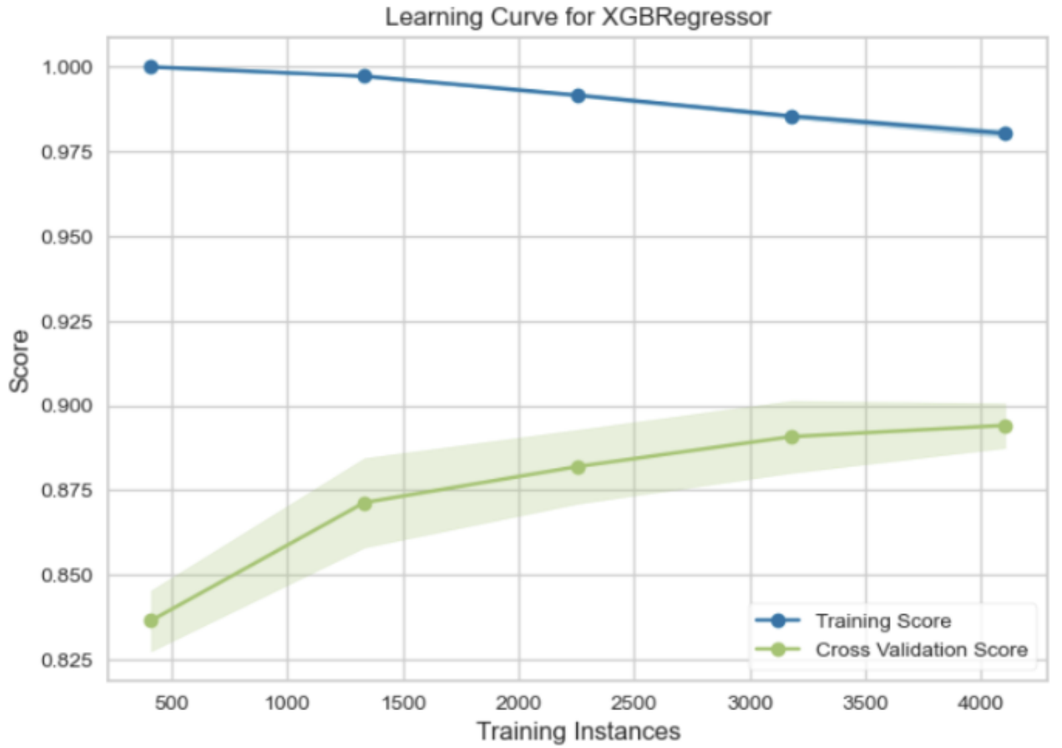
Evalution results for tempmin

Lineplot illustration for tempmin

3. Results

3.3 XGBoost

```
Test Set Mean Squared Error for tempmax (XGBoost): 3.693283223491405
Train Set Mean Squared Error for tempmax (XGBoost): 0.7455564755859758
Test Set Root Mean Squared Error for tempmax (XGBoost): 1.9217916701587103
Train Set Root Mean Squared Error for tempmax (XGBoost): 0.8634561225597834
Test Set R-squared for tempmax (XGBoost): 0.899228132501378
Train Set R-squared for tempmax (XGBoost): 0.9789987612001736

XGBoost Results for tempmax:
            Actual  Predicted    Model Temperature Type
datetime
2021-01-19    21.0  20.449900  XGBoost           tempmax
2012-05-03    37.0  37.220024  XGBoost           tempmax
2012-06-18    36.0  37.417812  XGBoost           tempmax
2023-03-23    32.3  32.811668  XGBoost           tempmax
2022-01-23    24.7  22.178982  XGBoost           tempmax
2016-02-16    16.1  19.025278  XGBoost           tempmax
2014-02-25    22.0  23.301252  XGBoost           tempmax
```

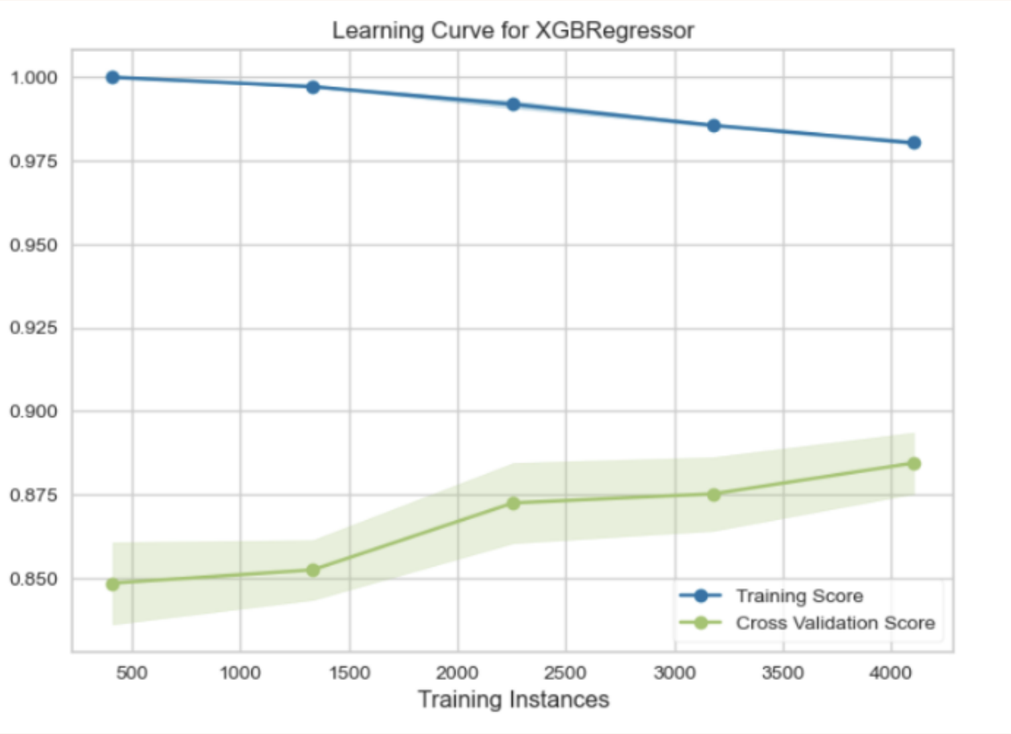


Results of tempmax assessment when overfitting

Learning curve chart evaluates overfitting

```
Test Set Mean Squared Error for tempmin (XGBoost): 2.5113715664677136
Train Set Mean Squared Error for tempmin (XGBoost): 0.509841409569159
Test Set R-squared for tempmin (XGBoost): 0.8984690853178943
Train Set R-squared for tempmin (XGBoost): 0.9794335586746111
Test Set Root Mean Squared Error for tempmin (XGBoost): 1.584730755197145
Train Set Root Mean Squared Error for tempmax (XGBoost): 0.7140317987100848

XGBoost Results for tempmin:
             Actual  Predicted    Model Temperature Type
datetime
2021-01-19    10.0  12.481799  XGBoost          tempmin
2012-05-03    27.0  27.751976  XGBoost          tempmin
2012-06-18    28.0  26.629633  XGBoost          tempmin
2023-03-23    24.8  24.643600  XGBoost          tempmin
2022-01-23    18.0  21.028711  XGBoost          tempmin
2016-02-16    12.0  13.027866  XGBoost          tempmin
2014-02-25    18.8  18.687027  XGBoost          tempmin
```

Results of tempmin assessment when overfitting

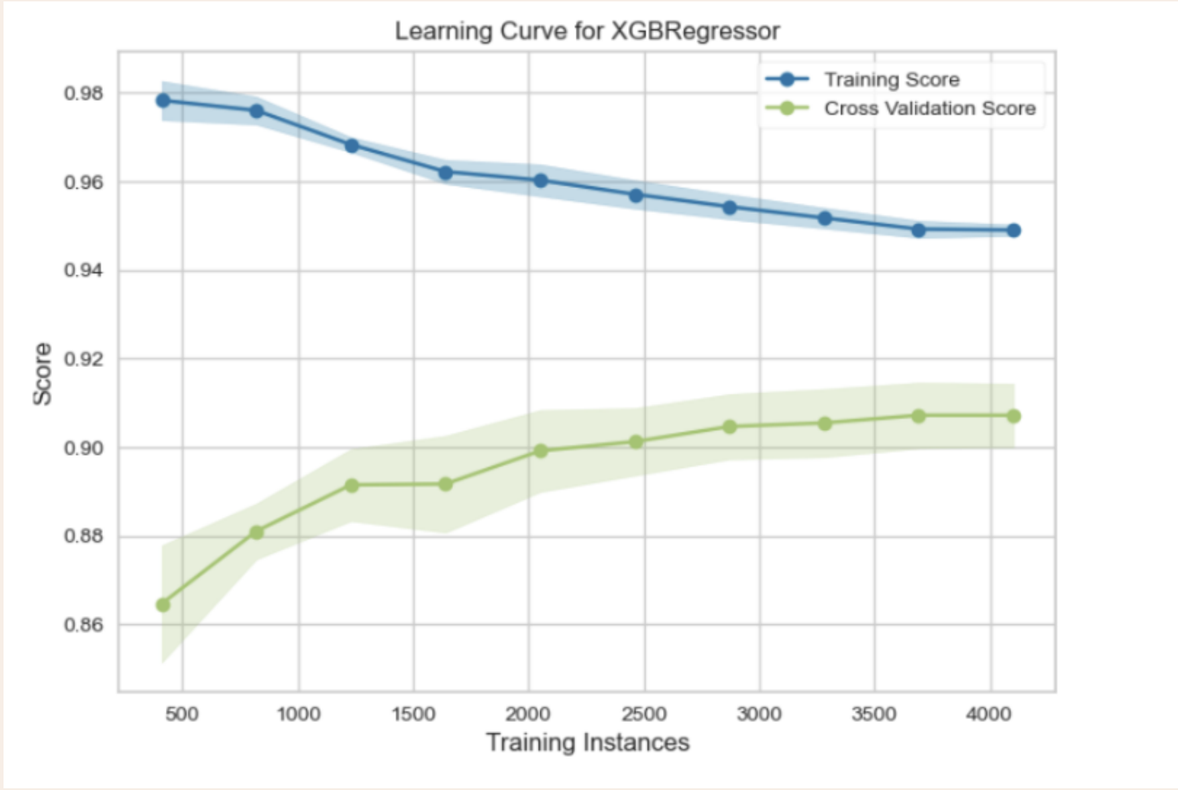


Learning curve chart evaluates overfitting

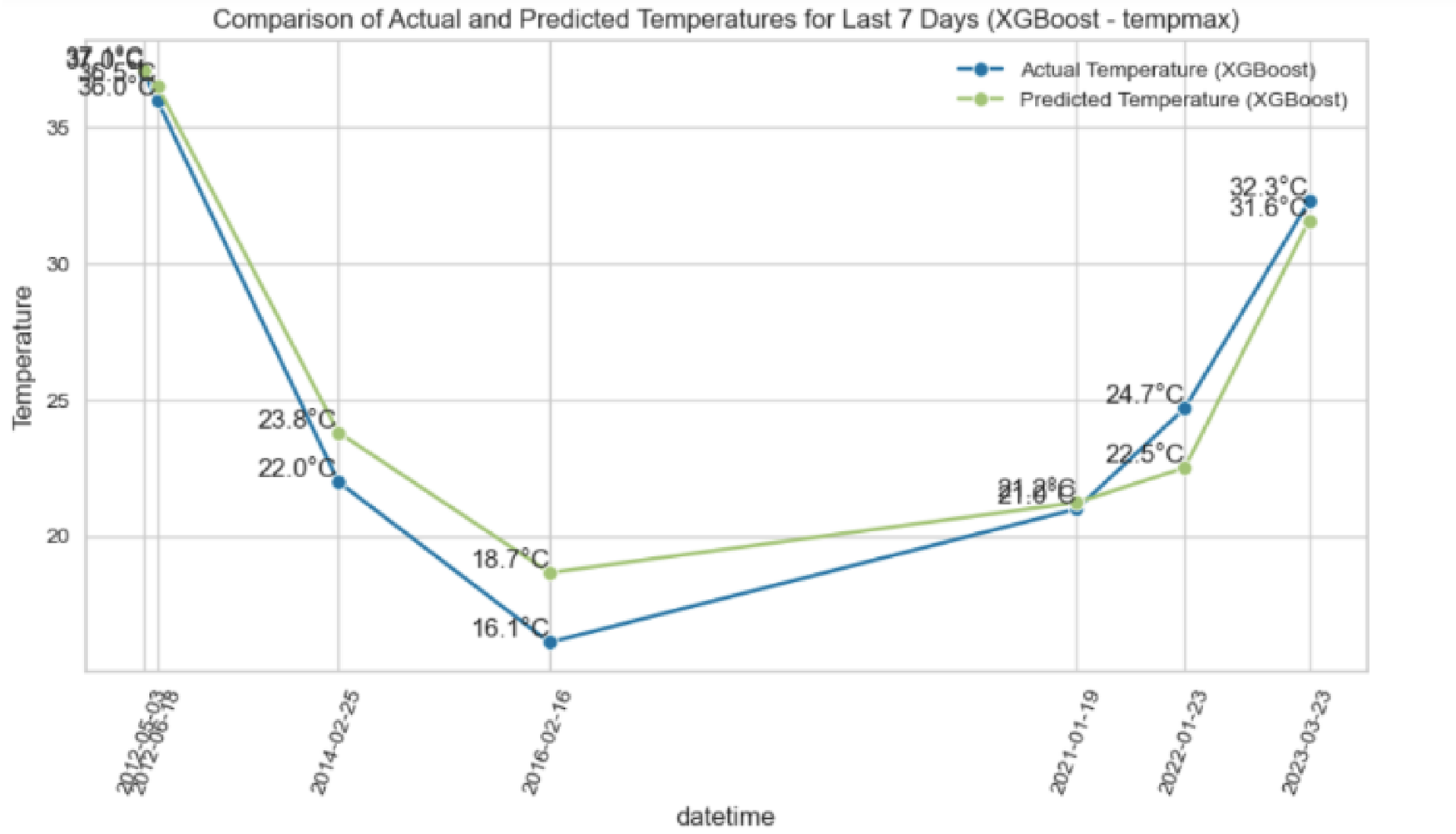
After Overfitting Mitigation

```
Test Set Mean Squared Error: 3.124920987111419
Train Set Mean Squared Error: 1.8292166271963441
Test Set Root Mean Squared Error: 1.7677446046053766
Train Set Root Mean Squared Error: 1.352485351934114
Test Set R-squared: 0.9147359938025106
Train Set R-squared: 0.9484736348454753

XGBoost Results for tempmax:
            Actual  Predicted    Model Temperature Type
datetime
2021-01-19    21.0  21.234159  XGBoost           tempmax
2012-05-03    37.0  37.121212  XGBoost           tempmax
2012-06-18    36.0  36.521019  XGBoost           tempmax
2023-03-23    32.3  31.594252  XGBoost           tempmax
2022-01-23    24.7  22.514416  XGBoost           tempmax
2016-02-16    16.1  18.661448  XGBoost           tempmax
2014-02-25    22.0  23.815035  XGBoost           tempmax
```

Results of tempmax evaluation when handling overfitting

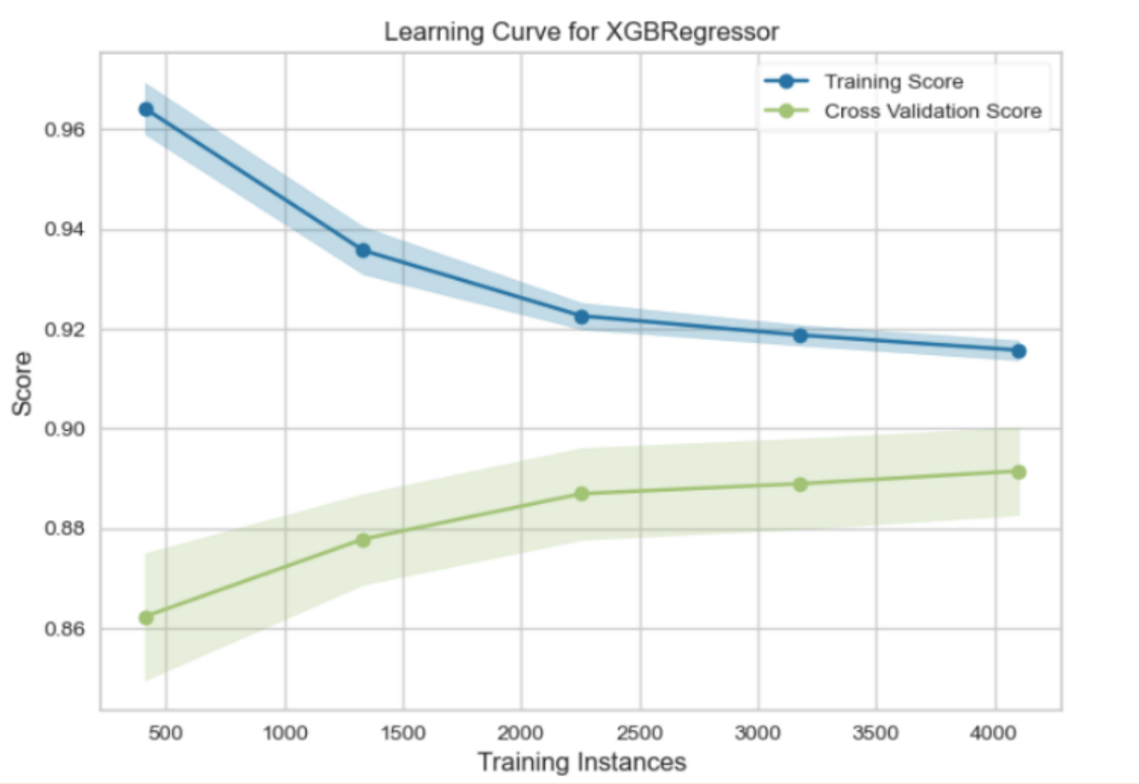


Learning curve chart after overfitting processing



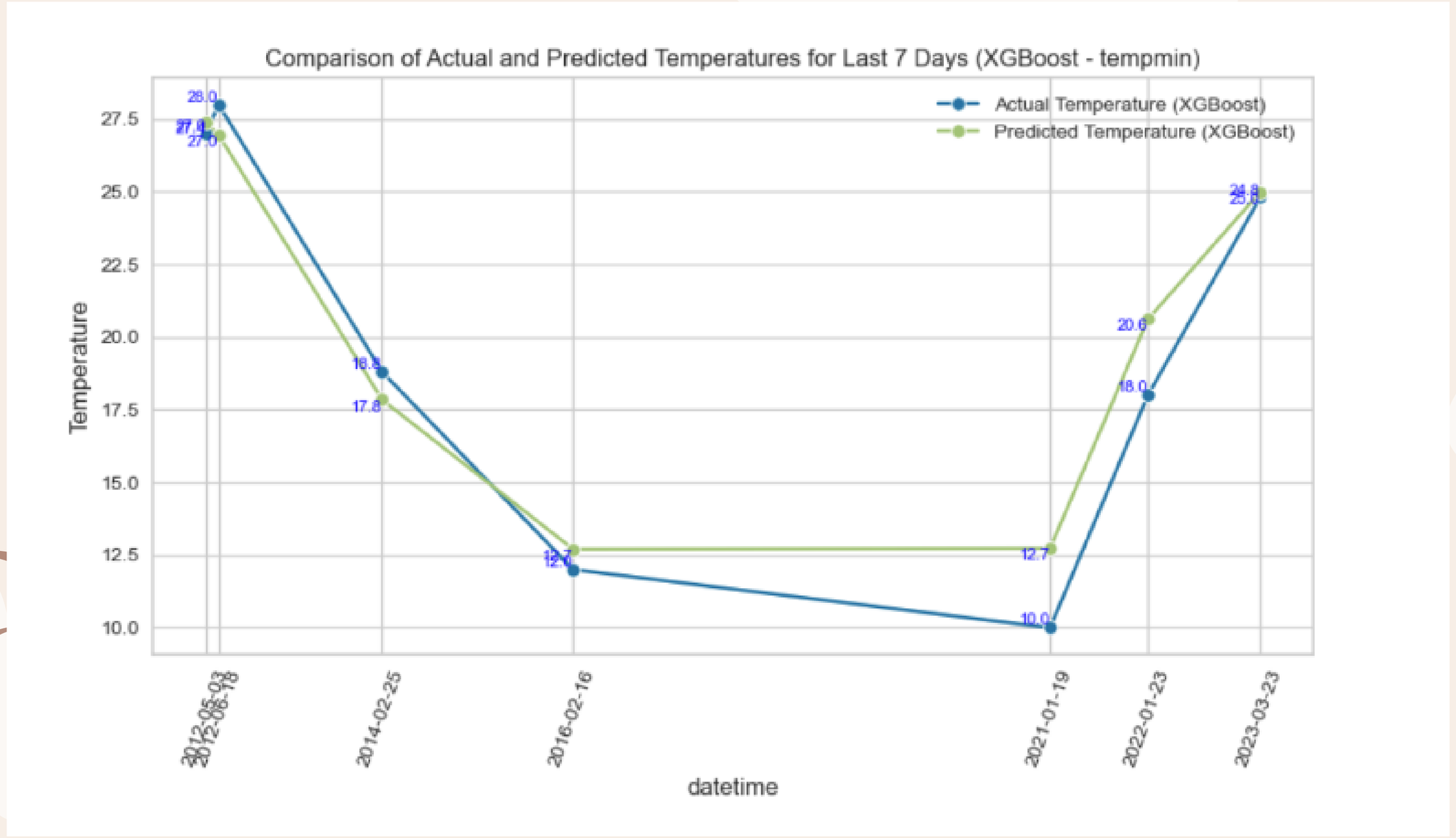
Lineplot illustration for tempmax

```
Train Set Mean Squared Error for tempmin (XGBoost): 2.0977266085081347
Train Set R-squared for tempmin (XGBoost): 0.915380017391982
Test Set Mean Squared Error for tempmin (XGBoost): 2.614768208607035
Test Set R-squared for tempmin (XGBoost): 0.8942889170816865
Train Set Root Mean Squared Error for tempmin (XGBoost): 1.4483530676282406
Test Set Root Mean Squared Error for tempmin (XGBoost): 1.6170244922718502

XGBoost Results for tempmin:
            Actual  Predicted    Model Temperature Type
datetime
2021-01-19    10.0  12.730933  XGBoost           tempmin
2012-05-03    27.0  27.372696  XGBoost           tempmin
2012-06-18    28.0  26.958458  XGBoost           tempmin
2023-03-23    24.8  24.987003  XGBoost           tempmin
2022-01-23    18.0  20.611650  XGBoost           tempmin
2016-02-16    12.0  12.698373  XGBoost           tempmin
2014-02-25    18.8  17.842281  XGBoost           tempmin
```



Results of tempmin evaluation when handing overfitting

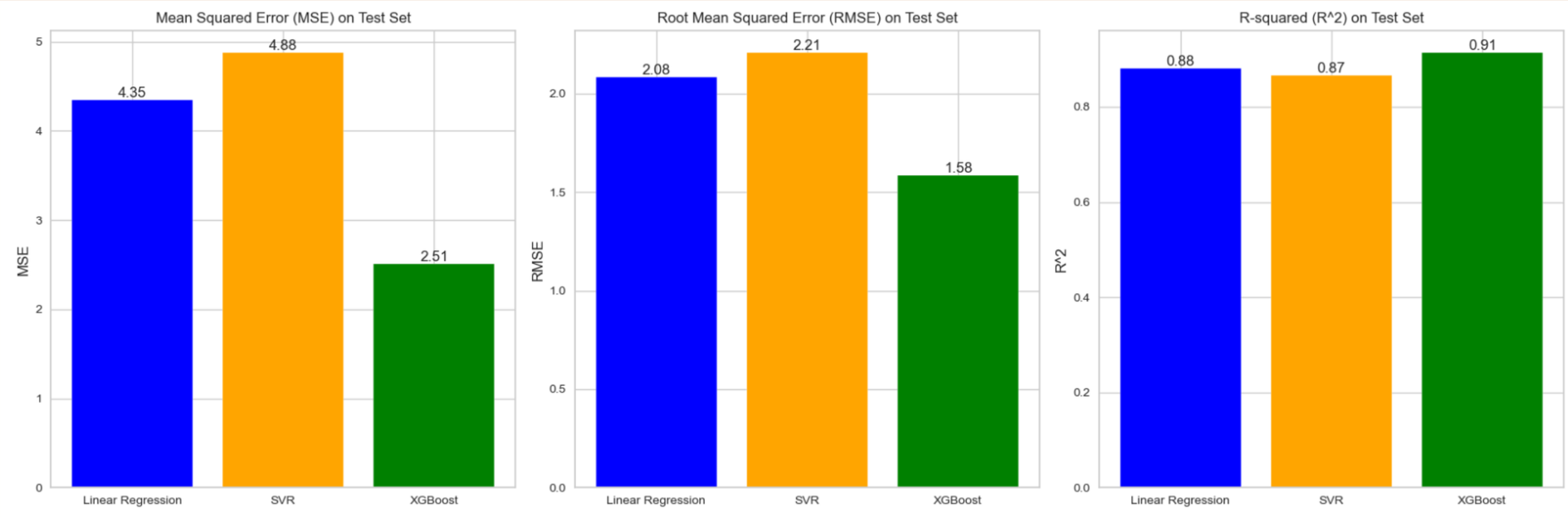
Learning curve chart after overfitting processing



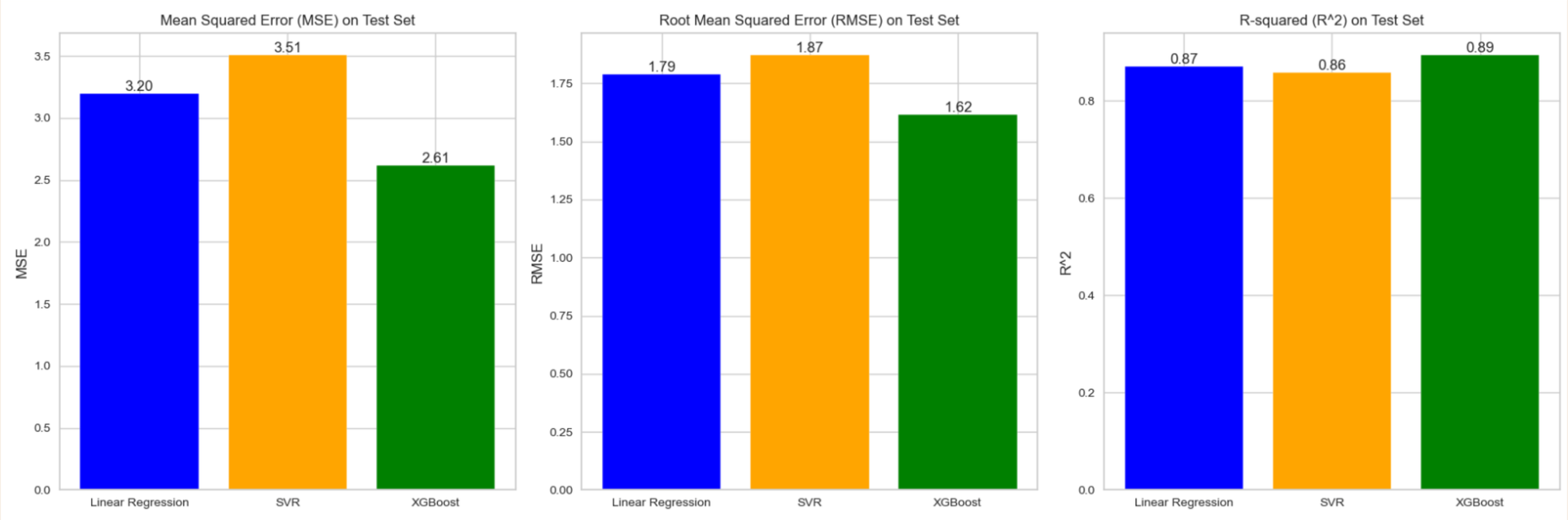
Lineplot illustration for tempmin

4. Conclusion

Tempmax



Tempmin



References

[1] M. Wang, Q. Xu, Y. Cao, S. G. Hassan, W. Liu, M. He, T. Liu, L. Xu, L. Cao, S. Liu and H. Wu, “An Ensemble Model for Water Temperature Prediction in Intensive Aquaculture,” IEEE, no. 1, 2023.

[2] A. Pelosi, G. B. Chirico, S. F. Bolognesi, C. D. Michele and G. D’Urso, “Forecasting crop evapotranspiration under standard conditions in precision farming,” IEEE, no. 2, 2019.

[3] T. Anjali, K. Chandini, K. Anoop and V. L. Lajish, “Temperature Prediction using Machine Learning,” IEEE, no. 3, 2019.

[4] A. Sharaff and S. R. Roy, “Comparative Analysis of Temperature Prediction Using Regression Methods and Back Propagation Neural Network,” IEEE, no. 4, 2018.

[5] S. Madan, P. Kumar, S. Rawat and T. Choudhury, “Analysis of Weather Prediction using,” IEEE, no. 5, 2018.

[6] I.-C. Chen and S.-C. Hu, “Realizing Specific Weather Forecast through Machine Learning Enabled Prediction Model,” ACM, no. 6, 2019.

[7] C. Z. Basha, N. Bhavana, P. Bhavya and S. V, “Rainfall Prediction Using Machine Learning & Deep,” IEEE, no. 7, 2020.

[8] A. A and A. Titus, “PREDICTION OF WEATER FORECASTCONDITIONS FORSMART AGRICULTURE SUPPORTED BY MACHINE LEARNING,” IEEE, no. 8, 2020.

[9] D. N. Fente and D. K. Singh, “Weather Forecasting Using Artificial Neural Network,” IEEE, no. 9, 2018.

[10] S. Jain and D. Ramesh, “Machine Learning convergence for weather based crop selection,” IEEE, no. 10, 2020.

References

[11] M. Schultz, S. Reitmann and S. Alam, “Classification of weather impacts on airport operations,” ACM, no. 11, 2020.

[12] S. Jiang and C. Cai, “The impacts of weather conditions on metro ridership: An empirical study from three mega cities in China,” Sciencedirect, no. 12, 2020.

[13] M. S. Tahsin, S. Abdullah, M. A. Karim, M. U. Ahmed, F. Tafannum and M. Y. Ara., “A comparative study on data mining models for weather forecasting,” Sciencedirect, no. 13, 2023.

[14] A. Muniasamy, “Machine Learning for Smart Farming: A Focus on Desert Agriculture,” IEEE, no. 14, 2020.

[15] E. Almeyda, J. Paiva and W. Ipanaqué, “Pest Incidence Prediction in Organic Banana Crops with Machine Learning Techniques,” IEEE, no. 15, 2020.

[16] G. Peng and W. Cun, “Wind Power Prediction Based on Back Propagation Algorithm with Numerical Weather Prediction,” ACM, no. 16, 2020.

[17] N.Sravanthi, M. Venka, S.Harshini and K.Ashesh, “An Ensemble Approach to Predict Weather,” IEEE, no. 17, 2020.

[18] F. Wu, S. Lu, L.-A. Armando and J. She, “Temperature Prediction Based on Long Short Term Memory Networks,” IEEE, no. 18, 2020.

[19] S. Nguyen-Xuan and N. L. Nhat, “A dynamic model for temperature prediction in glass greenhouse,” IEEE, no. 19, 2019.

[20] B. Azari, K. Hassan, J. Pierce and S. Ebrahimi, “Evaluation of Machine Learning Methods Application in Temperature Prediction,” ResearchGate, no. 20, 2022.

Thank you