```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
df = pd.read_csv("/content/(1990 - 2020 max and min temp only) (1).csv")

df.head()
```

	Station Index	Year	Days	Jan Max	Jan Min	Feb Max	Feb Min	Mar Max	Mar Min	April Max	...	Aug Max	Aug Min	Sep Max	Sep Min	Oct Max	Oct Min	Nov Max	Nov Min	Dec Max	Dec Min
0	Puri	1990	1	27.5	20.1	29.4	22.5	30.0	26.5	30.9	...	33.7	28.5	33.2	28.4	30.8	26.0	31.1	23.1	29.6	21.2
1	Puri	1990	2	27.0	17.8	29.6	25.4	30.2	24.0	31.9	...	32.2	25.0	32.5	28.5	31.4	25.5	24.6	22.9	29.2	21.4
2	Puri	1990	3	26.7	17.3	30.2	25.4	26.2	20.0	31.4	...	33.7	29.2	32.0	26.3	29.6	26.0	26.6	22.7	29.6	20.4
3	Puri	1990	4	26.8	16.2	29.7	23.4	26.0	19.6	30.5	...	34.0	29.0	31.6	27.4	31.5	26.4	29.0	24.4	29.4	19.8
4	Puri	1990	5	27.2	15.0	30.0	25.3	29.5	21.5	30.4	...	33.0	29.0	31.8	28.4	30.2	25.5	31.0	25.6	28.9	20.0

5 rows × 27 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14415 entries, 0 to 14414
Data columns (total 27 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Station Index  14415 non-null  object
 1   Year           14415 non-null  int64
 2   Days           14415 non-null  int64
 3   Jan Max        14415 non-null  float64
 4   Jan Min        14415 non-null  float64
 5   Feb Max        14415 non-null  float64
 6   Feb Min        14415 non-null  float64
 7   Mar Max        14415 non-null  float64
 8   Mar Min        14415 non-null  float64
 9   April Max      14415 non-null  float64
 10  April Min      14415 non-null  float64
 11  May Max        14415 non-null  float64
 12  May Min        14415 non-null  float64
 13  Jun Max        14415 non-null  float64
 14  Jun Min        14415 non-null  float64
 15  July Max       14415 non-null  float64
 16  July Min       14415 non-null  float64
 17  Aug Max        14415 non-null  float64
 18  Aug Min        14415 non-null  float64
 19  Sep Max        14415 non-null  float64
 20  Sep Min        14415 non-null  float64
 21  Oct Max        14415 non-null  float64
 22  Oct Min        14415 non-null  float64
 23  Nov Max        14415 non-null  float64
 24  Nov Min        14415 non-null  float64
 25  Dec Max        14415 non-null  float64
 26  Dec Min        14415 non-null  float64
dtypes: float64(24), int64(2), object(1)
memory usage: 3.0+ MB
```

```
df.shape
```

```
(14415, 27)
```

```
df.describe()
```

	Year	Days	Jan Max	Jan Min	Feb Max	Feb Min	Mar Max	Mar Min	April Max
count	14415.000000	14415.000000	14415.000000	14415.000000	14415.000000	14415.000000	14415.000000	14415.000000	14415.000000
mean	2005.000000	16.000000	42.949483	32.201138	50.295248	40.822629	48.483580	38.709455	51.888505
std	8.944582	8.944582	29.457437	35.245357	31.133992	37.439345	27.044431	32.726400	27.007431
min	1990.000000	1.000000	12.700000	3.100000	15.900000	0.000000	21.700000	7.900000	21.900000
25%	1997.000000	8.000000	26.800000	12.200000	29.700000	16.000000	32.500000	20.000000	35.000000
50%	2005.000000	16.000000	28.600000	15.300000	32.400000	19.200000	35.800000	22.700000	38.900000
75%	2013.000000	24.000000	31.900000	19.600000	99.900000	99.900000	40.000000	26.400000	44.500000
max	2020.000000	31.000000	99.900000	99.900000	99.900000	99.900000	99.900000	99.900000	99.900000

8 rows × 26 columns

```
months = ['Jan','Feb','Mar','April','May','Jun',
          'July','Aug','Sep','Oct','Nov','Dec']

yearly_city_data = []

for city in df['Station Index'].unique():
    city_df = df[df['Station Index'] == city]

    for year in city_df['Year'].unique():
        year_df = city_df[city_df['Year'] == year]

        # yearly max of all monthly max columns
        yearly_max = year_df[[f'{m} Max' for m in months]].max().max()

        # yearly min of all monthly min columns
        yearly_min = year_df[[f'{m} Min' for m in months]].min().min()

        yearly_city_data.append([city, year, yearly_max, yearly_min])

yearly_df = pd.DataFrame(
    yearly_city_data,
    columns=['City', 'Year', 'Yearly_MaxTemp', 'Yearly_MinTemp']
)

yearly_df.head()
```

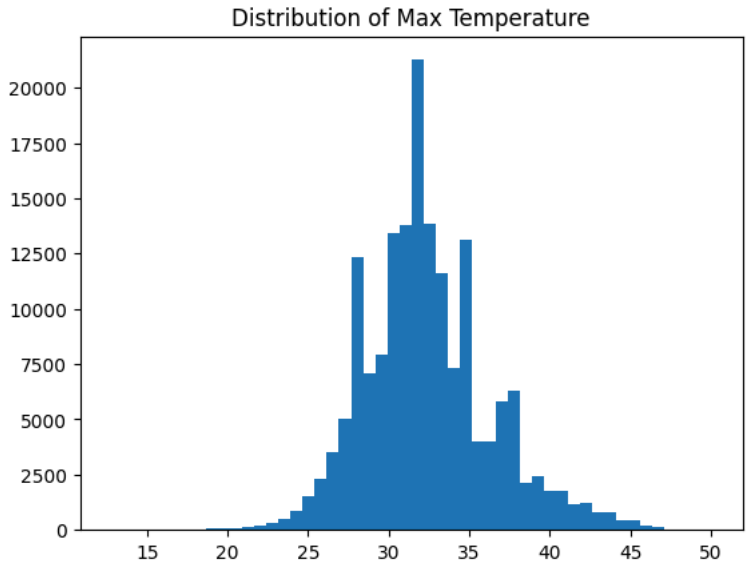
	City	Year	Yearly_MaxTemp	Yearly_MinTemp
0	Puri	1990	99.9	14.8
1	Puri	1991	99.9	14.2
2	Puri	1992	99.9	13.0
3	Puri	1993	99.9	13.9
4	Puri	1994	99.9	14.8

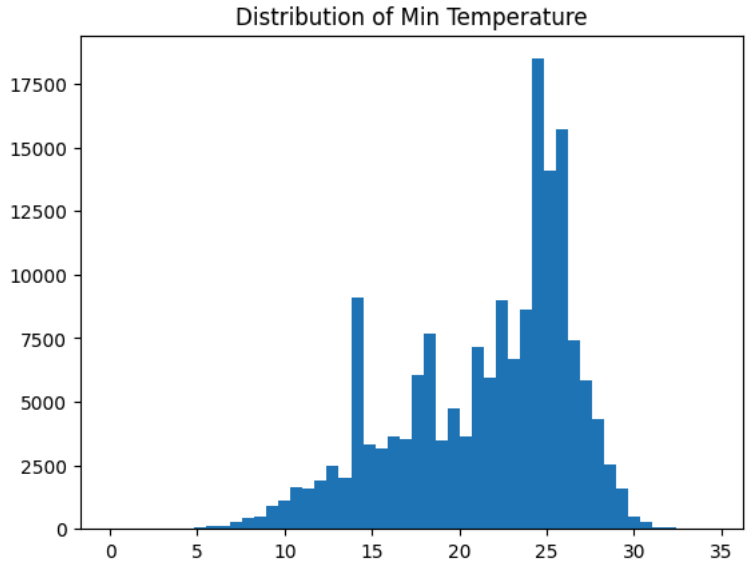
```
df.replace(99.9, np.nan, inplace=True)

df.isna().sum()
```

	0
Station Index	0
Year	0
Days	0
Jan Max	3028
Jan Min	3055
Feb Max	4053
Feb Min	4110
Mar Max	3096
Mar Min	3188
April Max	3422
April Min	3549
May Max	3029
May Min	3153
Jun Max	3453
Jun Min	3549
July Max	3161
July Min	3273
Aug Max	3127
Aug Min	3254
Sep Max	3678
Sep Min	3813
Oct Max	3180
Oct Min	3313
Nov Max	3758
Nov Min	3839
Dec Max	3286
Dec Min	3373

dtype: int64

```
# After imputation
plt.figure()
plt.hist(long_df['MaxTemp'], bins=50)
plt.title('Distribution of Max Temperature')
plt.show()

plt.figure()
plt.hist(long_df['MinTemp'], bins=50)
plt.title('Distribution of Min Temperature')
plt.show()
```






```
# List of months in your dataset
months = ['Jan','Feb','Mar','April','May','Jun','July','Aug','Sep','Oct','Nov','Dec']

# Convert Max temperatures to long format
max_df = df.melt(
    id_vars=['Station Index', 'Year', 'Days'],
    value_vars=[f'{m} Max' for m in months],
    var_name='Month',
    value_name='MaxTemp'
)

# Convert Min temperatures to long format
min_df = df.melt(
    id_vars=['Station Index', 'Year', 'Days'],
    value_vars=[f'{m} Min' for m in months],
    var_name='Month',
    value_name='MinTemp'
)

# Clean month names
max_df['Month'] = max_df['Month'].str.replace(' Max','')
min_df['Month'] = min_df['Month'].str.replace(' Min','')

# Merge Max & Min
long_df = pd.merge(
    max_df,
    min_df,
    on=['Station Index','Year','Days','Month']
)
```

```
long_df.head()
```

	Station Index	Year	Days	Month	MaxTemp	MinTemp
0	Puri	1990	1	Jan	27.5	20.1
1	Puri	1990	2	Jan	27.0	17.8
2	Puri	1990	3	Jan	26.7	17.3
3	Puri	1990	4	Jan	26.8	16.2
4	Puri	1990	5	Jan	27.2	15.0

```
long_df['MaxTemp'] = long_df.groupby('Month')['MaxTemp'].transform(
    lambda x: x.fillna(x.mean())
)

long_df['MinTemp'] = long_df.groupby('Month')['MinTemp'].transform(
    lambda x: x.fillna(x.mean())
)
```

```
long_df = long_df.sort_values(['Year','Month','Days'])

long_df['MaxTemp'] = long_df['MaxTemp'].interpolate()
long_df['MinTemp'] = long_df['MinTemp'].interpolate()
```

```
long_df.isna().sum()
```

	0
Station Index	0
Year	0
Days	0
Month	0
MaxTemp	0
MinTemp	0

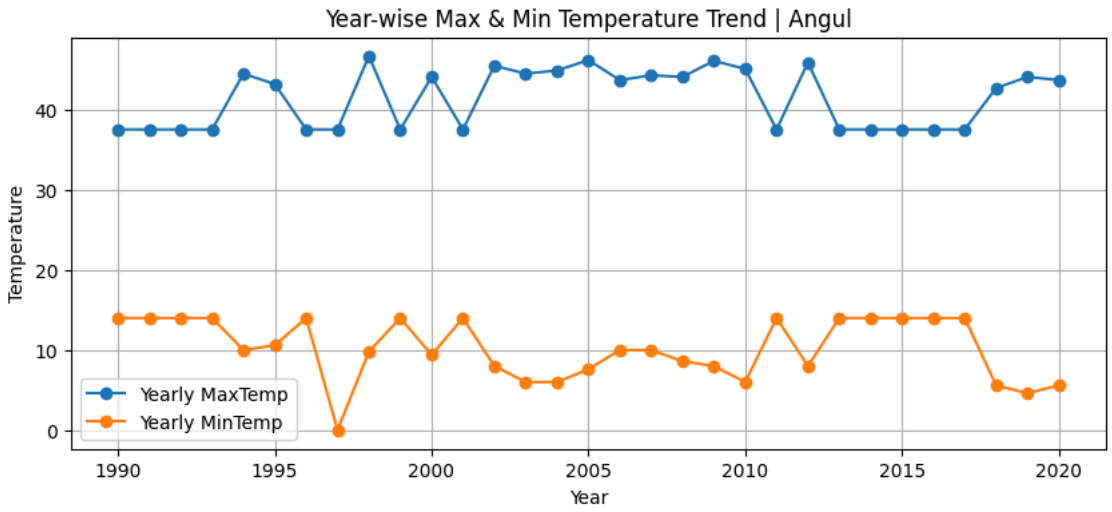
dtype: int64

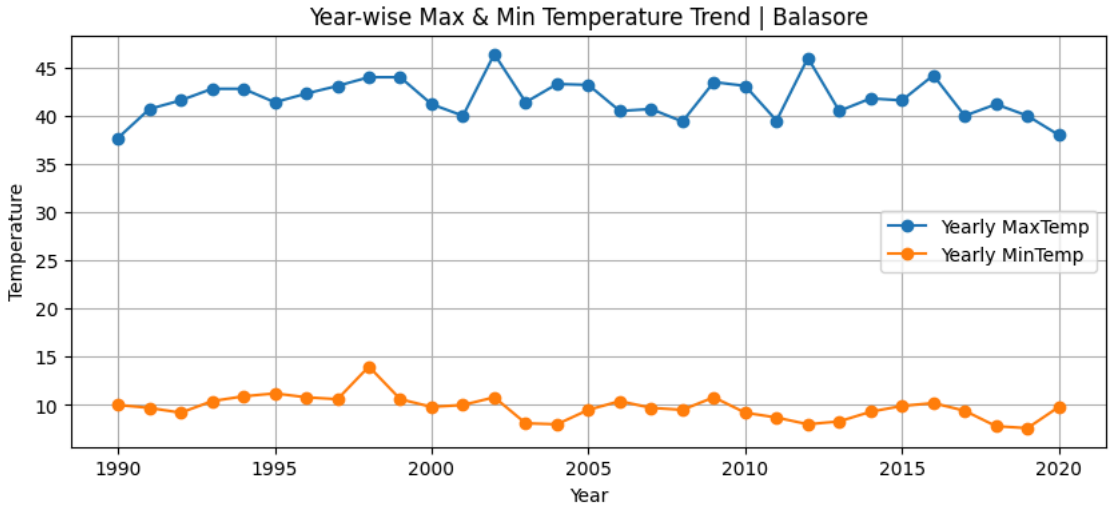
```
yearly_df = (
    long_df
    .groupby(['Station Index', 'Year'])
    .agg(
        Yearly_MaxTemp=('MaxTemp', 'max'),
        Yearly_MinTemp=('MinTemp', 'min')
    )
    .reset_index()
)

yearly_df.head()
```

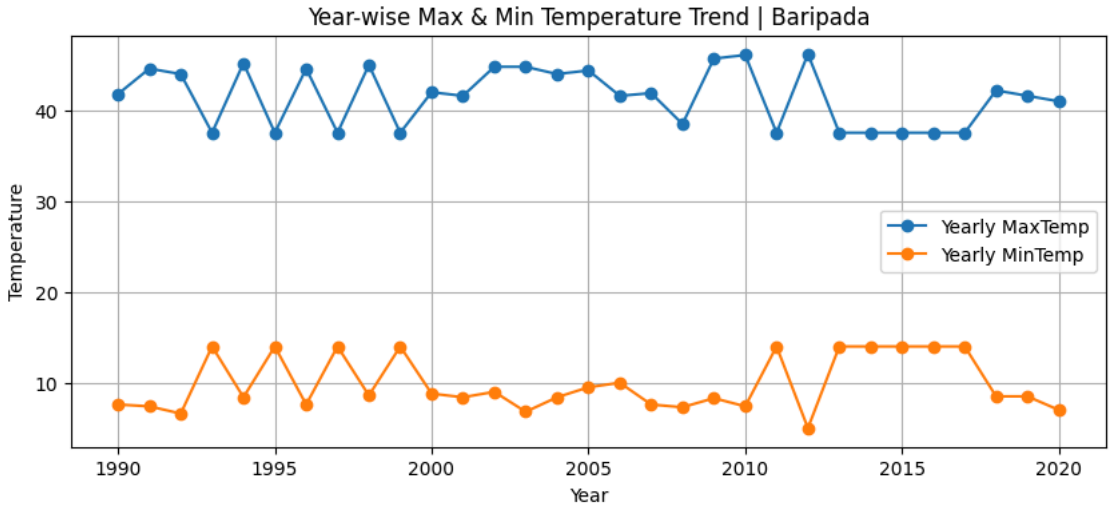
	Station Index	Year	Yearly_MaxTemp	Yearly_MinTemp
0	Angul	1990	37.535061	13.99515
1	Angul	1991	37.535061	13.99515
2	Angul	1992	37.535061	13.99515
3	Angul	1993	37.535061	13.99515
4	Angul	1994	44.500000	10.00000

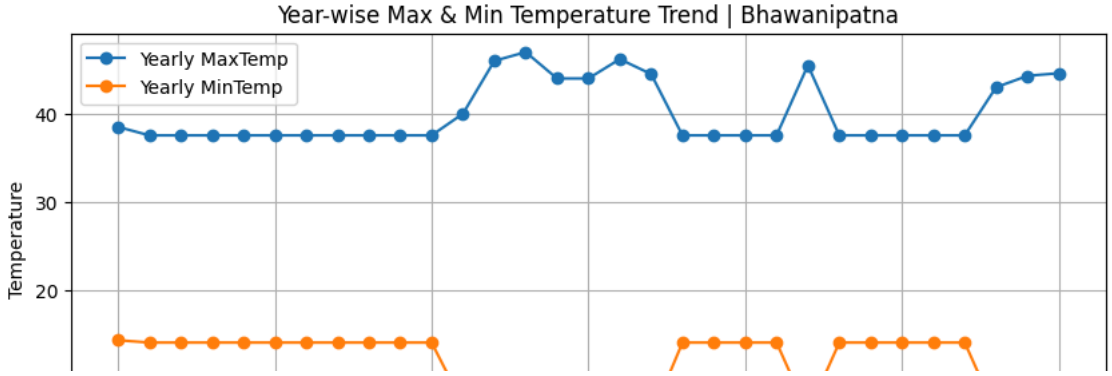
```
import matplotlib.pyplot as plt

cities = yearly_df['Station Index'].unique()
```

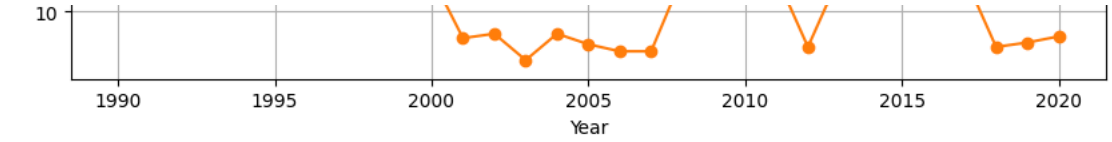
```
for city in cities:
    city_year_df = yearly_df[yearly_df['Station Index'] == city]

    plt.figure(figsize=(10,4))

    plt.plot(
        city_year_df['Year'],
        city_year_df['Yearly_MaxTemp'],
        marker='o',
        label='Yearly MaxTemp'
    )

    plt.plot(
        city_year_df['Year'],
        city_year_df['Yearly_MinTemp'],
        marker='o',
        label='Yearly MinTemp'
    )

    plt.title(f'Year-wise Max & Min Temperature Trend | {city}')
    plt.xlabel('Year')
    plt.ylabel('Temperature')
    plt.legend()
    plt.grid(True)
    plt.show()
```

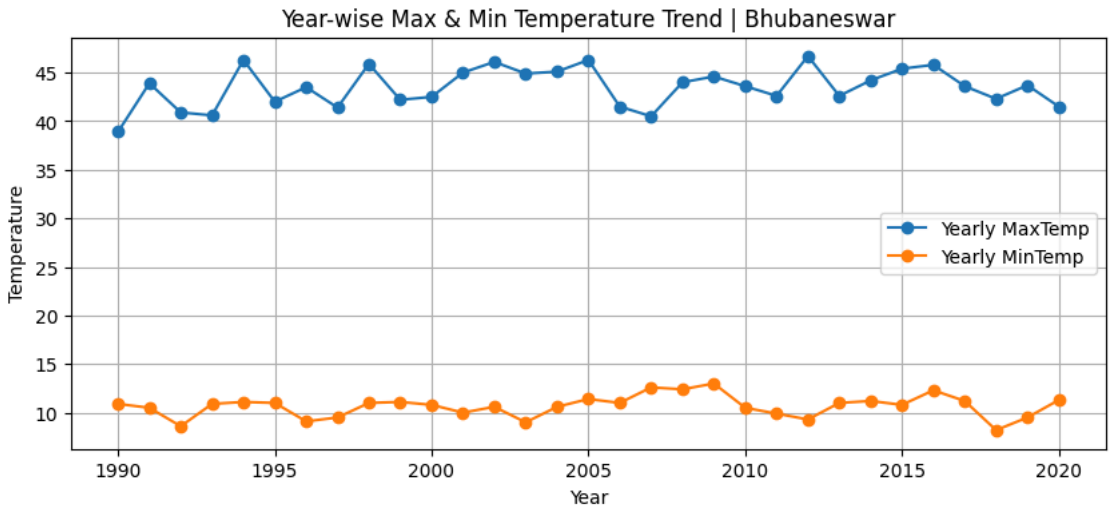

Year-wise Max & Min Temperature Trend | Angul
Temperature
Year
Yearly MaxTemp
Yearly MinTemp

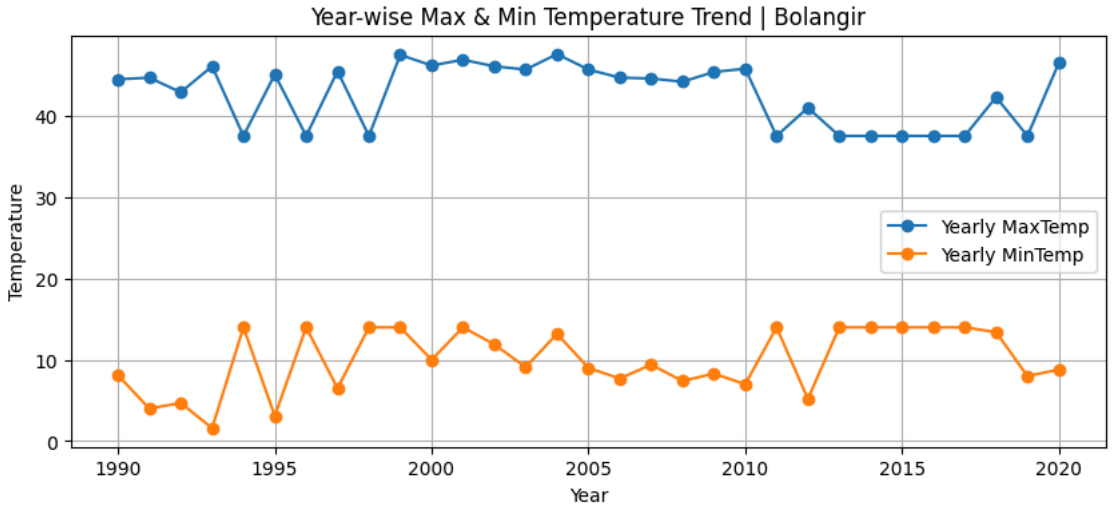

Year-wise Max & Min Temperature Trend | Balasore
Temperature
Year
Yearly MaxTemp
Yearly MinTemp

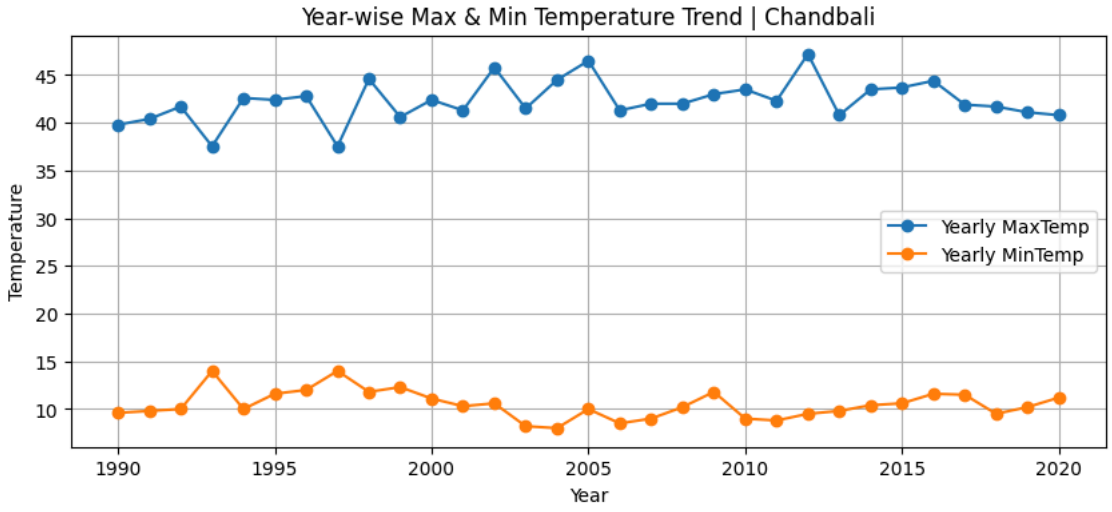

Year-wise Max & Min Temperature Trend | Baripada
Temperature
Year
Yearly MaxTemp
Yearly MinTemp

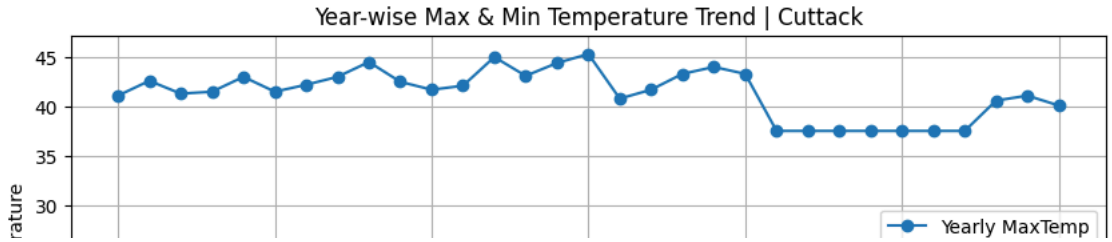

Year-wise Max & Min Temperature Trend | Bhawanipatna
Temperature
Yearly MaxTemp
Yearly MinTemp

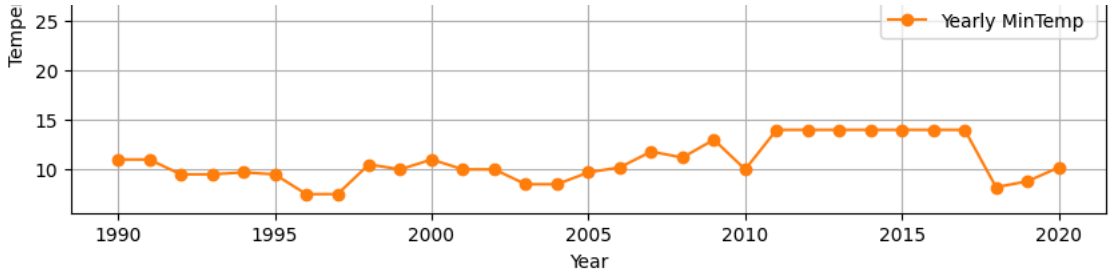

10
1990
1995
2000
2005
2010
2015
2020
Year

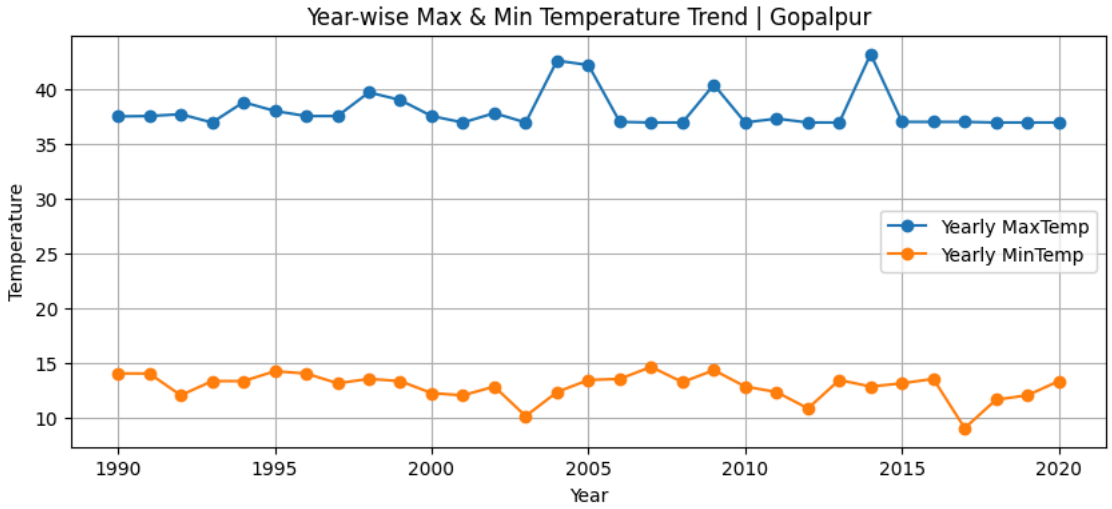

Year-wise Max & Min Temperature Trend | Bhubaneswar
Temperature
Year
Yearly MaxTemp
Yearly MinTemp

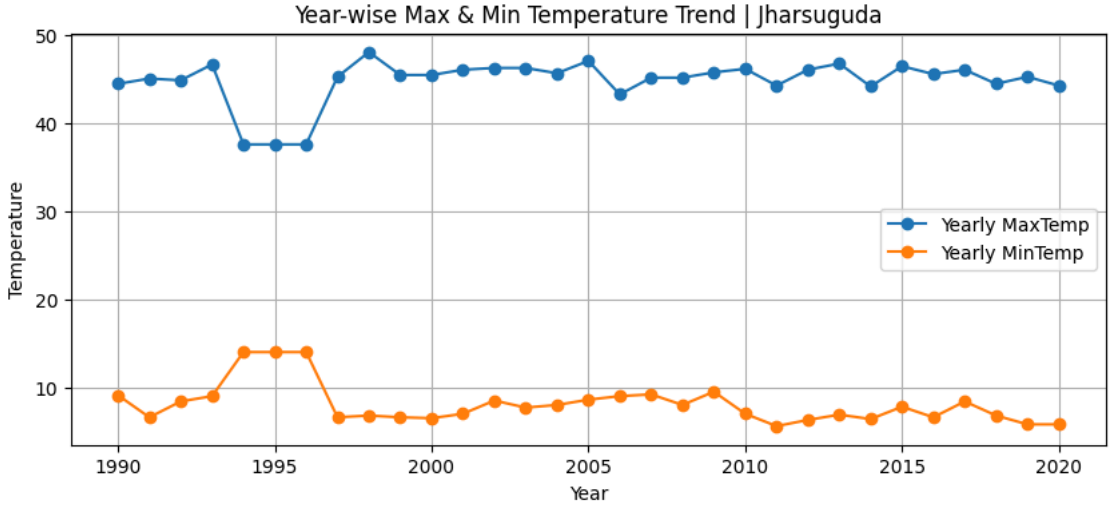

Year-wise Max & Min Temperature Trend | Bolangir
Temperature
Year
Yearly MaxTemp
Yearly MinTemp

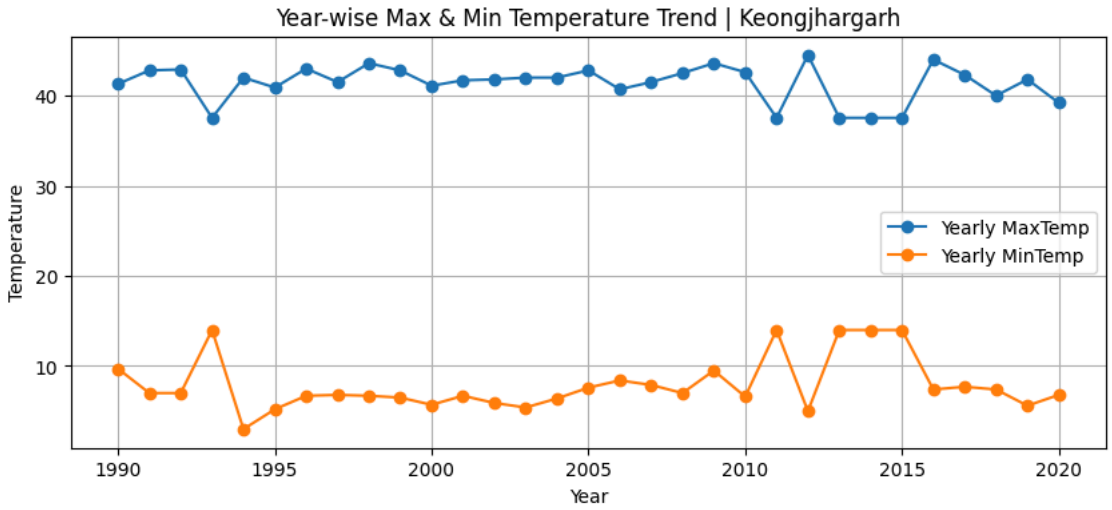

Year-wise Max & Min Temperature Trend | Chandbali
Temperature
Year
Yearly MaxTemp
Yearly MinTemp

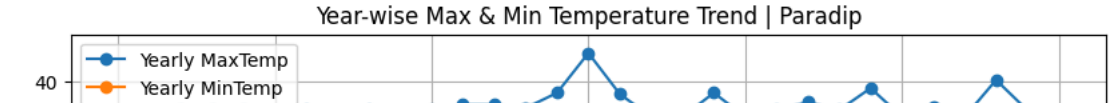

Year-wise Max & Min Temperature Trend | Cuttack
Yearly MaxTemp

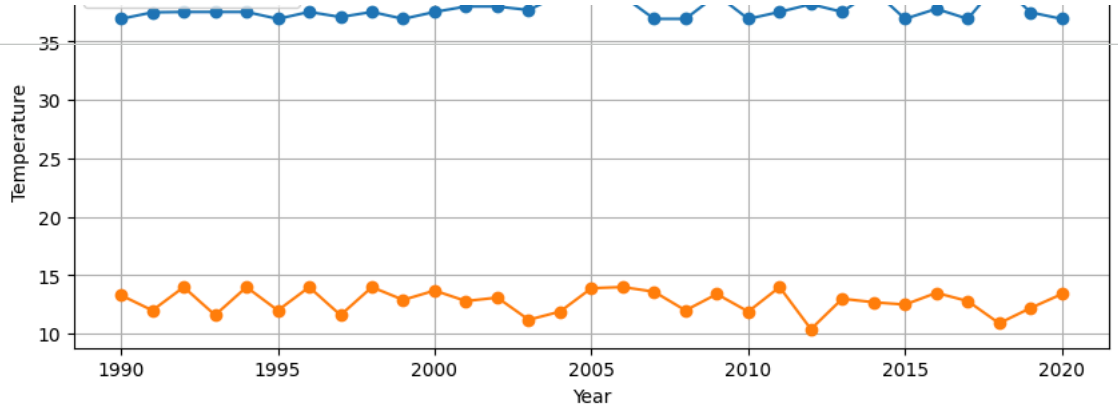

Yearly MinTemp
Year

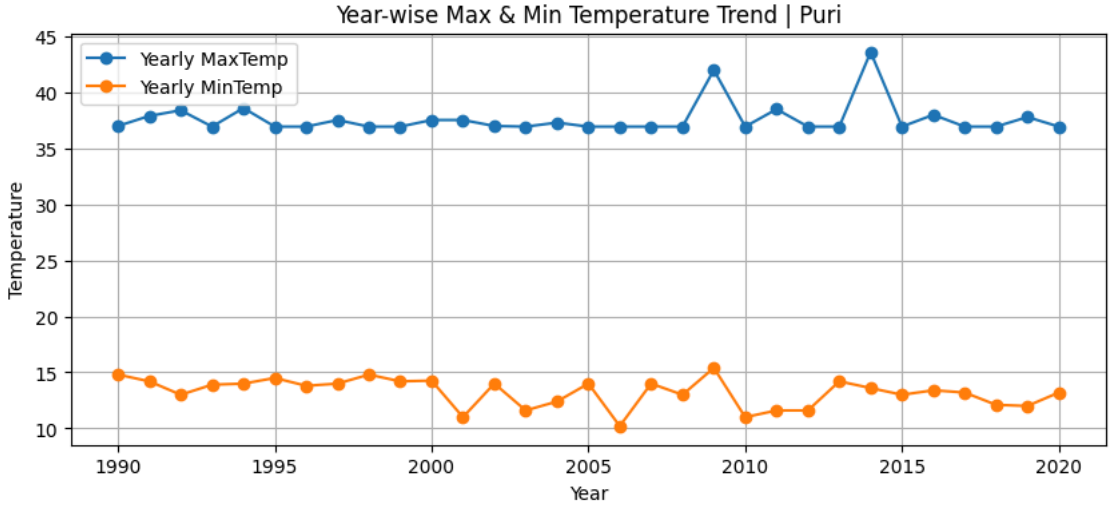

Year-wise Max & Min Temperature Trend | Gopalpur
Temperature
Yearly MaxTemp
Yearly MinTemp
Year

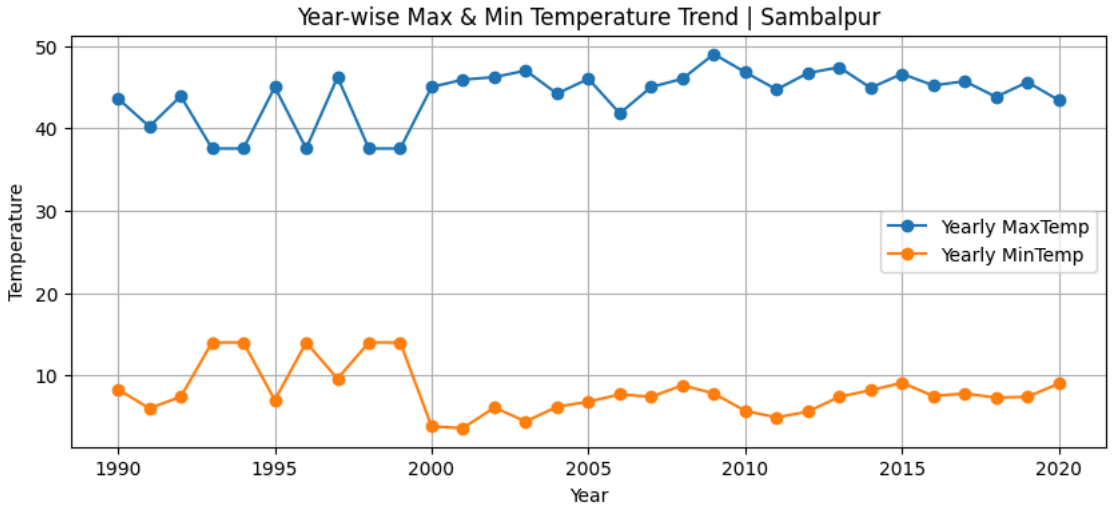

Year-wise Max & Min Temperature Trend | Jharsuguda
Temperature
Yearly MaxTemp
Yearly MinTemp
Year

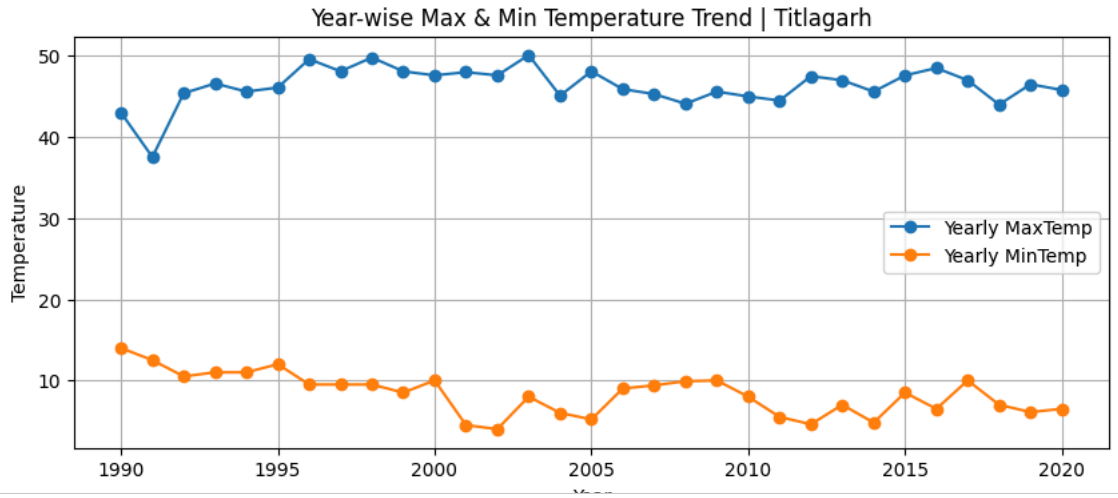

Year-wise Max & Min Temperature Trend | Keongjhargarh
Temperature
Yearly MaxTemp
Yearly MinTemp
Year


Year-wise Max & Min Temperature Trend | Paradip
Yearly MaxTemp
Yearly MinTemp


Temperature
Year
1990
1995
2000
2005
2010
2015
2020


Year-wise Max & Min Temperature Trend | Puri
Yearly MaxTemp
Yearly MinTemp
Temperature
Year


Year-wise Max & Min Temperature Trend | Sambalpur
Yearly MaxTemp
Yearly MinTemp
Temperature
Year


Year-wise Max & Min Temperature Trend | Titlagarh
Yearly MaxTemp
Yearly MinTemp
Temperature

```
long_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 172980 entries, 43245 to 129734
Data columns (total 6 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Station Index  172980 non-null  object
 1   Year           172980 non-null  int64
 2   Days           172980 non-null  int64
 3   Month          172980 non-null  object
 4   MaxTemp        172980 non-null  float64
 5   MinTemp        172980 non-null  float64
dtypes: float64(2), int64(2), object(2)
memory usage: 9.2+ MB
```

```
long_df.shape
```

```
(172980, 6)
```

```
# Month name to number mapping
month_map = {
    'Jan': 1, 'Feb': 2, 'Mar': 3, 'April': 4, 'May': 5, 'Jun': 6,
    'July': 7, 'Aug': 8, 'Sep': 9, 'Oct': 10, 'Nov': 11, 'Dec': 12
}

long_df['MonthNum'] = long_df['Month'].map(month_map)

# Create Date column
long_df['Date'] = pd.to_datetime(
    dict(year=long_df['Year'], month=long_df['MonthNum'], day=long_df['Days']),
    errors='coerce'
)

long_df = long_df.sort_values('Date').reset_index(drop=True)

long_df.head()
```

	Station Index	Year	Days	Month	MaxTemp	MinTemp	MonthNum	Date
0	Cuttack	1990	1	Jan	28.100000	17.50000	1	1990-01-01
1	Jharsuguda	1990	1	Jan	27.805357	13.99515	1	1990-01-01
2	Sambalpur	1990	1	Jan	23.800000	13.60000	1	1990-01-01
3	Balasore	1990	1	Jan	25.600000	15.00000	1	1990-01-01
4	Titlagarh	1990	1	Jan	27.805357	13.99515	1	1990-01-01

```
long_df = long_df.dropna(subset=['Date'])
```

```
long_df['Max_lag_1'] = long_df['MaxTemp'].shift(1)
long_df['Max_lag_7'] = long_df['MaxTemp'].shift(7)

long_df['Min_lag_1'] = long_df['MinTemp'].shift(1)
long_df['Min_lag_7'] = long_df['MinTemp'].shift(7)
```

```
long_df['Max_roll_7'] = long_df['MaxTemp'].rolling(7).mean()
long_df['Min_roll_7'] = long_df['MinTemp'].rolling(7).mean()
```

```
long_df.dropna(inplace=True)
```

```
train = long_df[long_df['Year'] <= 2019]
valid = long_df[long_df['Year'] == 2020]
```

```
features = [
    'MonthNum', 'Days',
```

```
    'Max_lag_1', 'Max_lag_7', 'Max_roll_7',
    'Min_lag_1', 'Min_lag_7', 'Min_roll_7'
]

X_train = train[features]
X_valid = valid[features]

y_train_max = train['MaxTemp']
y_valid_max = valid['MaxTemp']

y_train_min = train['MinTemp']
y_valid_min = valid['MinTemp']
```

```
pred_2021 = pd.DataFrame(
    future_preds,
    columns=['Pred_MaxTemp_2021', 'Pred_MinTemp_2021']
)

pred_2021.head()
```

	Pred_MaxTemp_2021	Pred_MinTemp_2021
0	27.591141	12.978446
1	27.106405	12.474208
2	26.908798	12.347047
3	27.046356	12.645483
4	27.393062	13.043955

```
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
```

```
xgb_max = XGBRegressor(
    n_estimators=500,
    learning_rate=0.05,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    objective='reg:squarederror',
    random_state=42
)

xgb_min = XGBRegressor(
    n_estimators=500,
    learning_rate=0.05,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    objective='reg:squarederror',
    random_state=42
)

xgb_max.fit(X_train, y_train_max)
xgb_min.fit(X_train, y_train_min)
```

```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.8, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=0.05, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=6,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=500,
             n_jobs=None, num_parallel_tree=None, ...)
```

```
pred_max_2020 = xgb_max.predict(X_valid)
pred_min_2020 = xgb_min.predict(X_valid)

print("MAE MaxTemp:", mean_absolute_error(y_valid_max, pred_max_2020))
print("MAE MinTemp:", mean_absolute_error(y_valid_min, pred_min_2020))

print("RMSE MaxTemp:", np.sqrt(mean_squared_error(y_valid_max, pred_max_2020)))
print("RMSE MinTemp:", np.sqrt(mean_squared_error(y_valid_min, pred_min_2020)))

print("R² MaxTemp:", r2_score(y_valid_max, pred_max_2020))
print("R² MinTemp:", r2_score(y_valid_min, pred_min_2020))
```

```
MAE MaxTemp: 1.3612978010954115
MAE MinTemp: 1.4953952176802738
RMSE MaxTemp: 1.832300932802682
RMSE MinTemp: 1.9086483957088005
R² MaxTemp: 0.7692167085172043
R² MinTemp: 0.8541655395843426
```

```
# FULL training data (1990–2020)
full_train = long_df[long_df['Year'] <= 2020]

X_full = full_train[features]
y_full_max = full_train['MaxTemp']
y_full_min = full_train['MinTemp']
```

```
xgb_max.fit(X_full, y_full_max)
xgb_min.fit(X_full, y_full_min)
```

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.8, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=0.05, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=6,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=500,
             n_jobs=None, num_parallel_tree=None, ...)
```

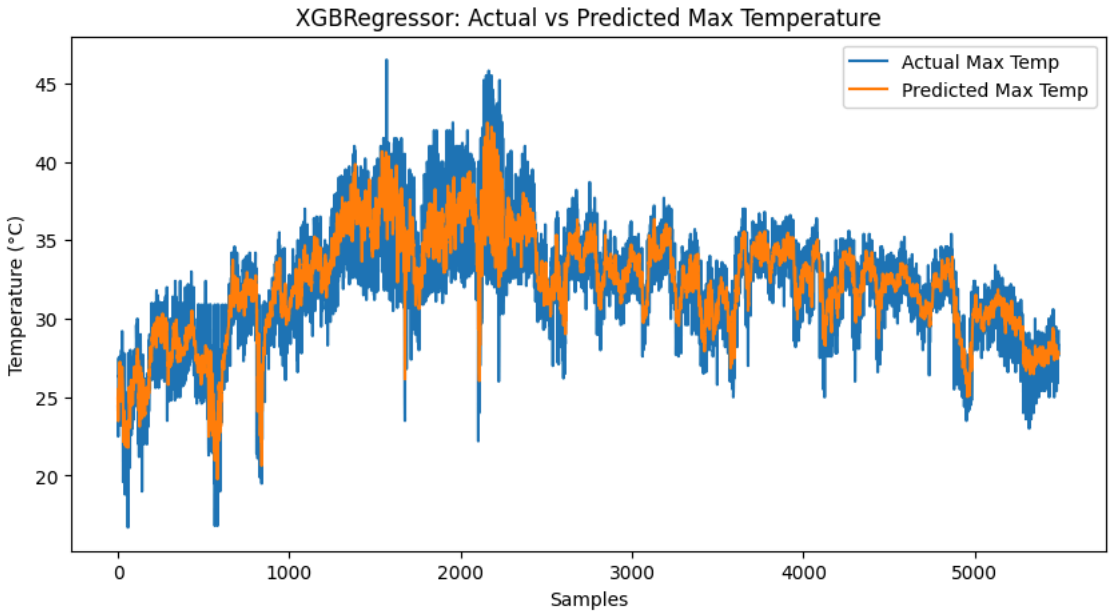
```
last_data = long_df[long_df['Year'] == 2020].copy()
future_preds = []

for i in range(365):
    row = last_data.iloc[-1]

    X_next = pd.DataFrame([{
        'MonthNum': row['MonthNum'],
        'Days': row['Days'],
        'Max_lag_1': row['MaxTemp'],
        'Max_lag_7': last_data.iloc[-7]['MaxTemp'],
        'Max_roll_7': last_data.tail(7)['MaxTemp'].mean(),
        'Min_lag_1': row['MinTemp'],
        'Min_lag_7': last_data.iloc[-7]['MinTemp'],
        'Min_roll_7': last_data.tail(7)['MinTemp'].mean()
    }])

    max_pred = xgb_max.predict(X_next)[0]
    min_pred = xgb_min.predict(X_next)[0]

    new_row = row.copy()
    new_row['MaxTemp'] = max_pred
    new_row['MinTemp'] = min_pred
    new_row['Year'] = 2021

    last_data = pd.concat([last_data, pd.DataFrame([new_row])])
    future_preds.append([max_pred, min_pred])
```

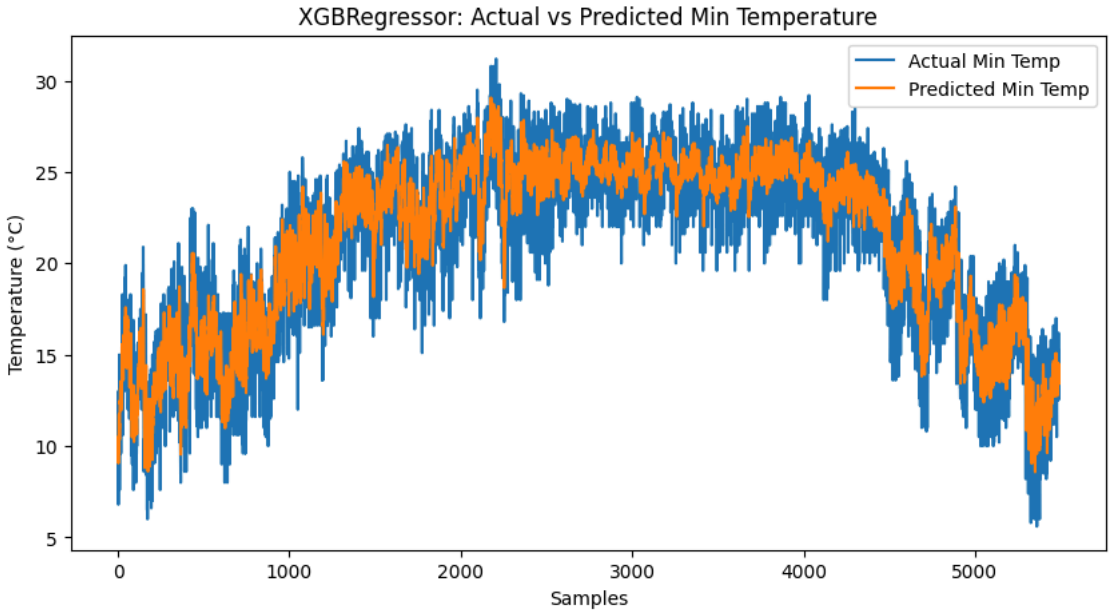
```
pred_2021 = pd.DataFrame(
    future_preds,
    columns=['Pred_MaxTemp_2021', 'Pred_MinTemp_2021']
)
```

```
pred_2021.head()
```

	Pred_MaxTemp_2021	Pred_MinTemp_2021
0	27.700073	13.984303
1	28.050776	13.657402
2	28.257248	14.314151
3	28.116571	13.371630
4	28.071854	13.437564

```
# XGBRegressor Predictions
xgb_pred_max = xgb_max.predict(X_valid)
xgb_pred_min = xgb_min.predict(X_valid)

# Plot Actual vs Predicted (Max Temperature)
plt.figure(figsize=(10,5))
plt.plot(y_valid_max.values, label="Actual Max Temp")
plt.plot(xgb_pred_max, label="Predicted Max Temp")
plt.title("XGBRegressor: Actual vs Predicted Max Temperature")
plt.xlabel("Samples")
plt.ylabel("Temperature (°C)")
plt.legend()
plt.show()

# Plot Actual vs Predicted (Min Temperature)
plt.figure(figsize=(10,5))
plt.plot(y_valid_min.values, label="Actual Min Temp")
plt.plot(xgb_pred_min, label="Predicted Min Temp")
plt.title("XGBRegressor: Actual vs Predicted Min Temperature")
plt.xlabel("Samples")
plt.ylabel("Temperature (°C)")
plt.legend()
plt.show()
```






```
!pip install catboost
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.12/dist-packages (1.2.8)
Requirement already satisfied: graphviz in /usr/local/lib/python3.12/dist-packages (from catboost) (0.21)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<3.0,>=1.16.0 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.0.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.12/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from catboost) (1.16.3)
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.12/dist-packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=0.24->catboost) (2025.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (4.61.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->catboost) (3.3.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/dist-packages (from plotly->catboost) (9.1.2)
```

```
from catboost import CatBoostRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
```