Weather Temperature Prediction, Analysis and Visualization

Overview

As each day progresses, we are witnessing an alarming increase in the severity of weather events. Among the most pressing global challenges we face is the rise in extreme temperatures. In this project, I aim to illustrate the changes in temperature and precipitation patterns from 2000 to 2019 in my hometown of Pabna. By providing this data, I hope to support authorities in making informed, data-driven decisions regarding climate-related initiatives and the timely implementation of governmental projects throughout the year.

Data Preprocessing

Exploratory Data Analysis

Precipitation Trend Analysis

Model Training & Evaluation

Conclusion and Result

Initial Data import & Exploration

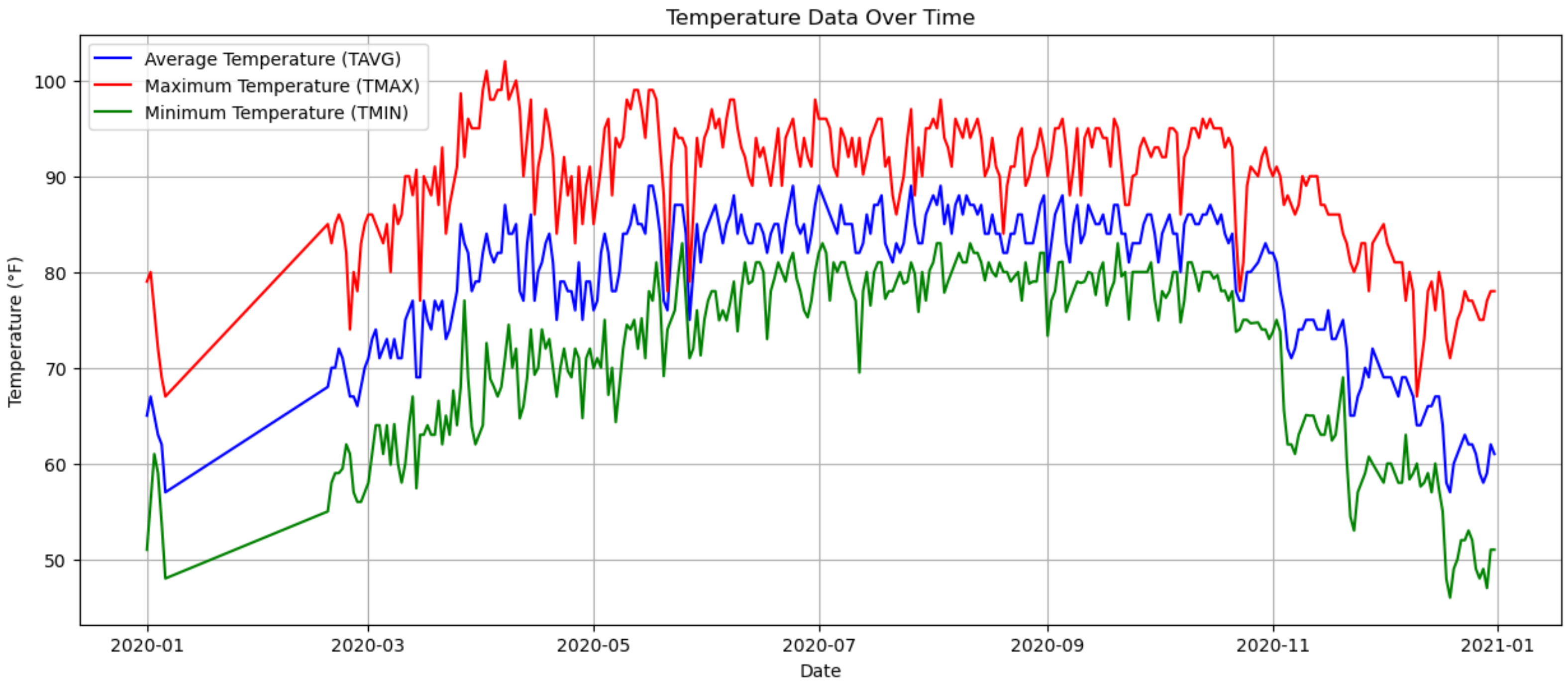
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

train_df = pd.read_csv('train.csv', index_col='DATE', parse_dates=True)
validation_df = pd.read_csv('test.csv', index_col='DATE', parse_dates=True)
train_df.head()
```

	STATION	NAME	LATITUDE	LONGITUDE	ELEVATION	PRCP	TAVG	TMAX	TMIN
DATE									
2000-01-01	BGM00041907	ISHURDI, BG	24.153	89.049	13.7	0.0	61.0	75.0	52.562500
2000-01-07	BGM00041907	ISHURDI, BG	24.153	89.049	13.7	0.0	54.0	68.0	45.935333
2000-01-08	BGM00041907	ISHURDI, BG	24.153	89.049	13.7	0.0	55.0	71.0	45.171464
2000-01-10	BGM00041907	ISHURDI, BG	24.153	89.049	13.7	0.0	60.0	78.0	50.657107
2000-01-13	BGM00041907	ISHURDI, BG	24.153	89.049	13.7	0.0	66.0	81.0	55.866583

Temperature Data Visualization

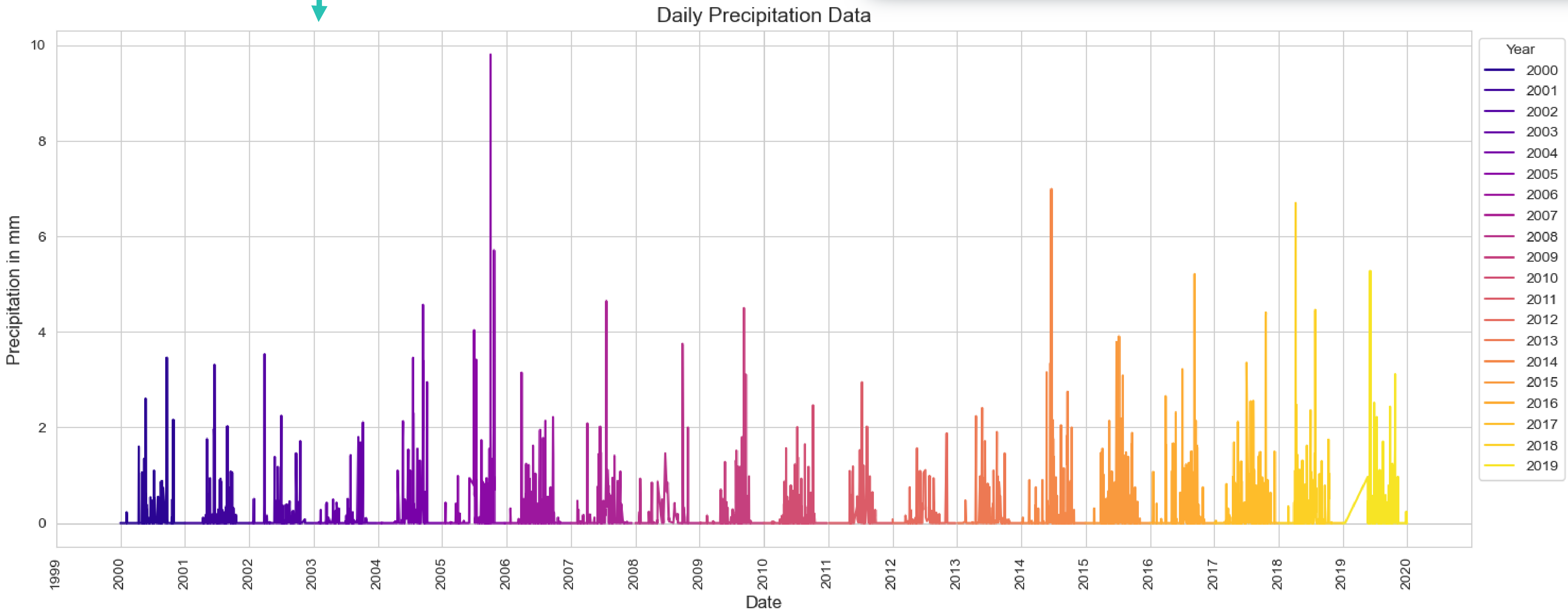
```python
plt.figure(figsize=(15, 6))
plt.plot(validation_df.index, validation_df['TAVG'], label='Average Temperature (TAVG)', color='blue')
plt.plot(validation_df.index, validation_df['TMAX'], label='Maximum Temperature (TMAX)', color='red')
plt.plot(validation_df.index, validation_df['TMIN'], label='Minimum Temperature (TMIN)', color='green')
plt.title('Temperature Data Over Time')
plt.xlabel('Date')
plt.ylabel('Temperature (°F)')
plt.legend()
plt.grid()
plt.show()
```

Temperature is in Fahrenheit scale which needs to be converted to Celsius scale



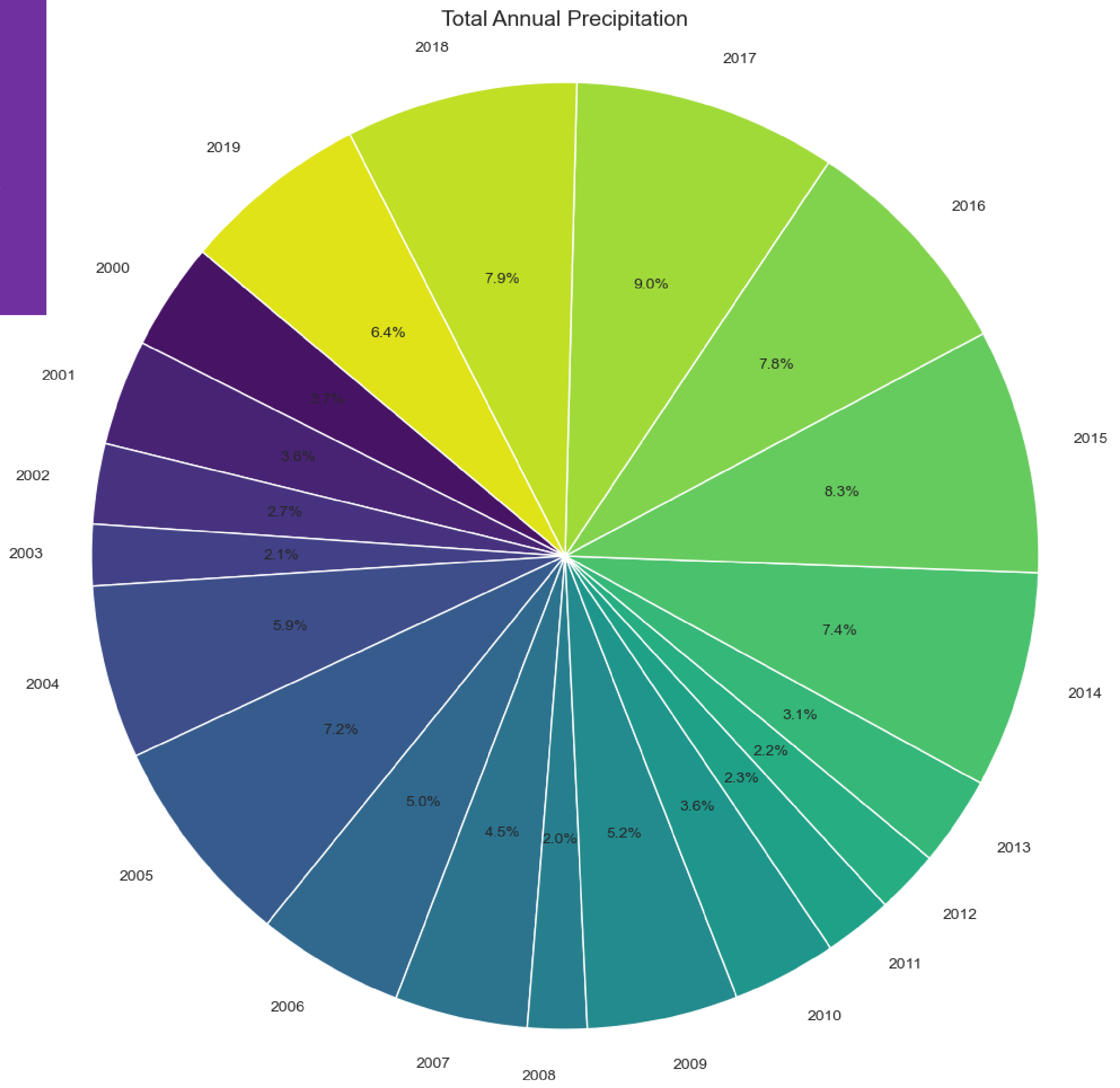
Precipitation Analysis

```python
import matplotlib.dates as mdates

sns.set_style("whitegrid")

years = train_df.index.year.unique()
colors = sns.color_palette("plasma", len(years))

plt.figure(figsize=(15, 6))

for i, year in enumerate(years):
    yearly_data = train_df[train_df.index.year == year]
    plt.plot(yearly_data.index, yearly_data['PRCP'], linestyle='-', color=colors[i], label=str(year))

plt.title('Daily Precipitation Data', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Precipitation in mm', fontsize=12)
plt.gca().xaxis.set_major_locator(mdates.YearLocator(1))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))

plt.xticks(rotation=90)
plt.legend(title="Year", loc="upper left", bbox_to_anchor=(1,1))
plt.tight_layout()
plt.show()
```



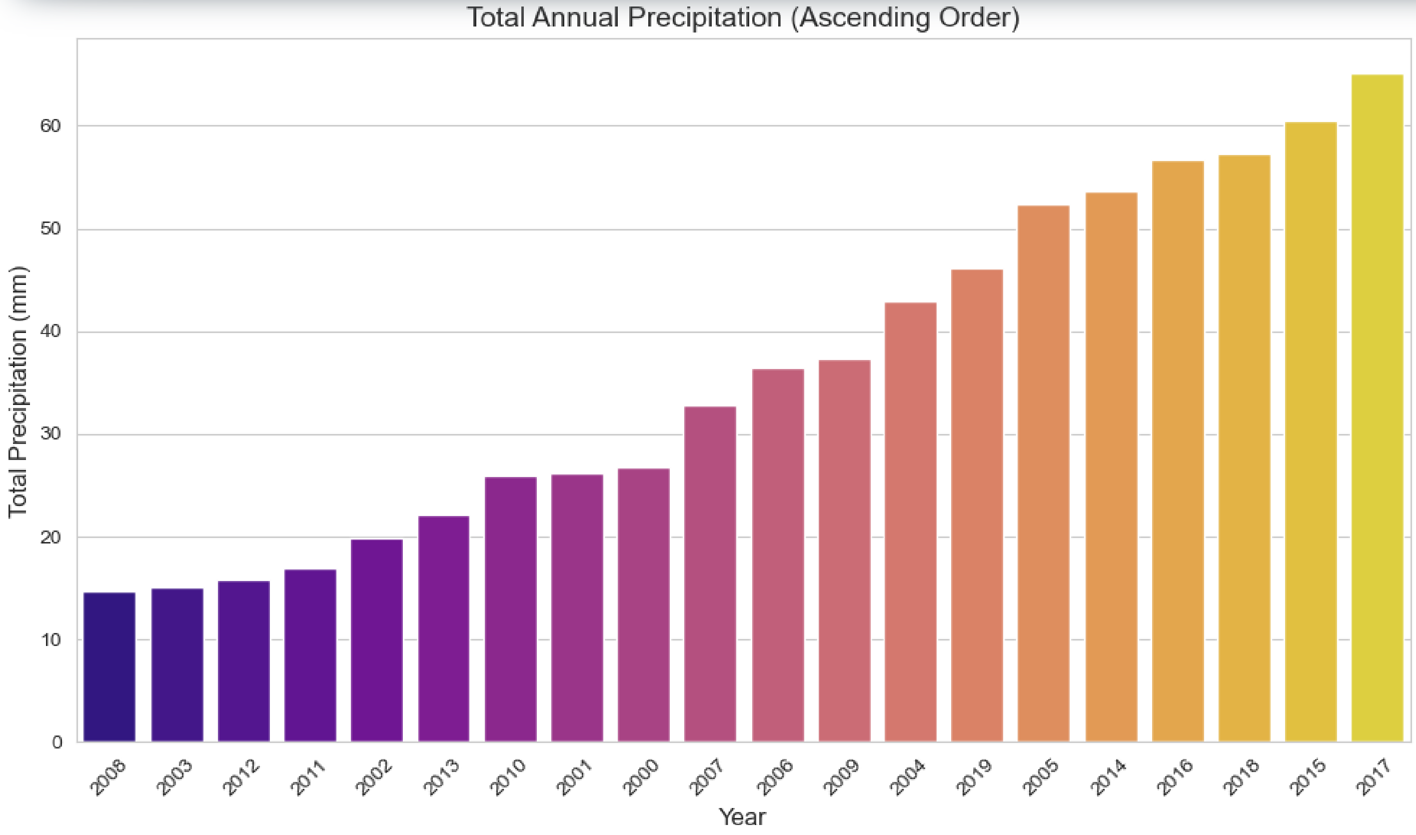
Total Annual Precipitation by Year

```python
annual_precipitation = train_df.resample('YE').sum()['PRCP']

sns.set_style("whitegrid")
colors = sns.color_palette("viridis", len(annual_precipitation))

plt.figure(figsize=(16, 12))
plt.pie(annual_precipitation, labels=annual_precipitation.index.year, autopct='%1.1f%%', startangle=140, colors=colors)

plt.title('Total Annual Precipitation', fontsize=14)
plt.axis('equal')

plt.show()

```



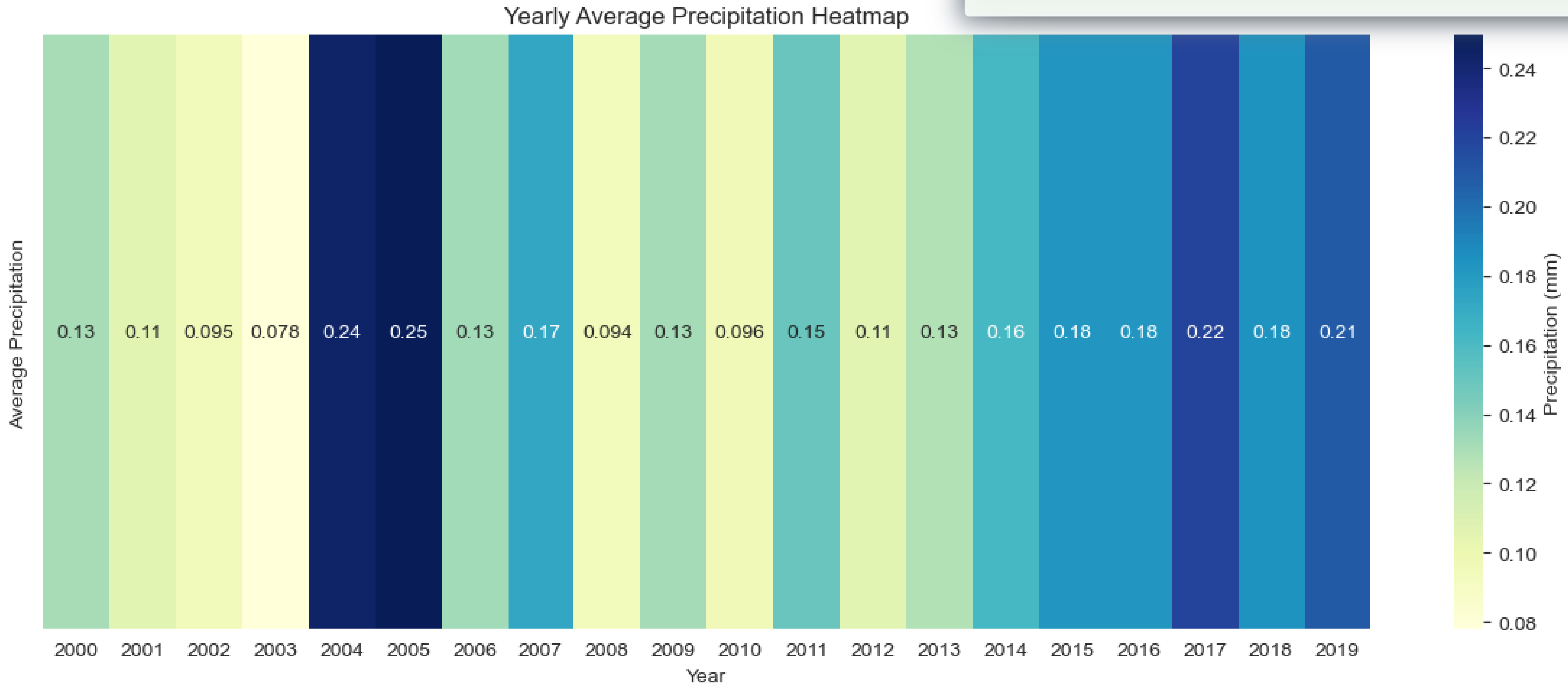
Total Annual Precipitation by Year (Ascending)

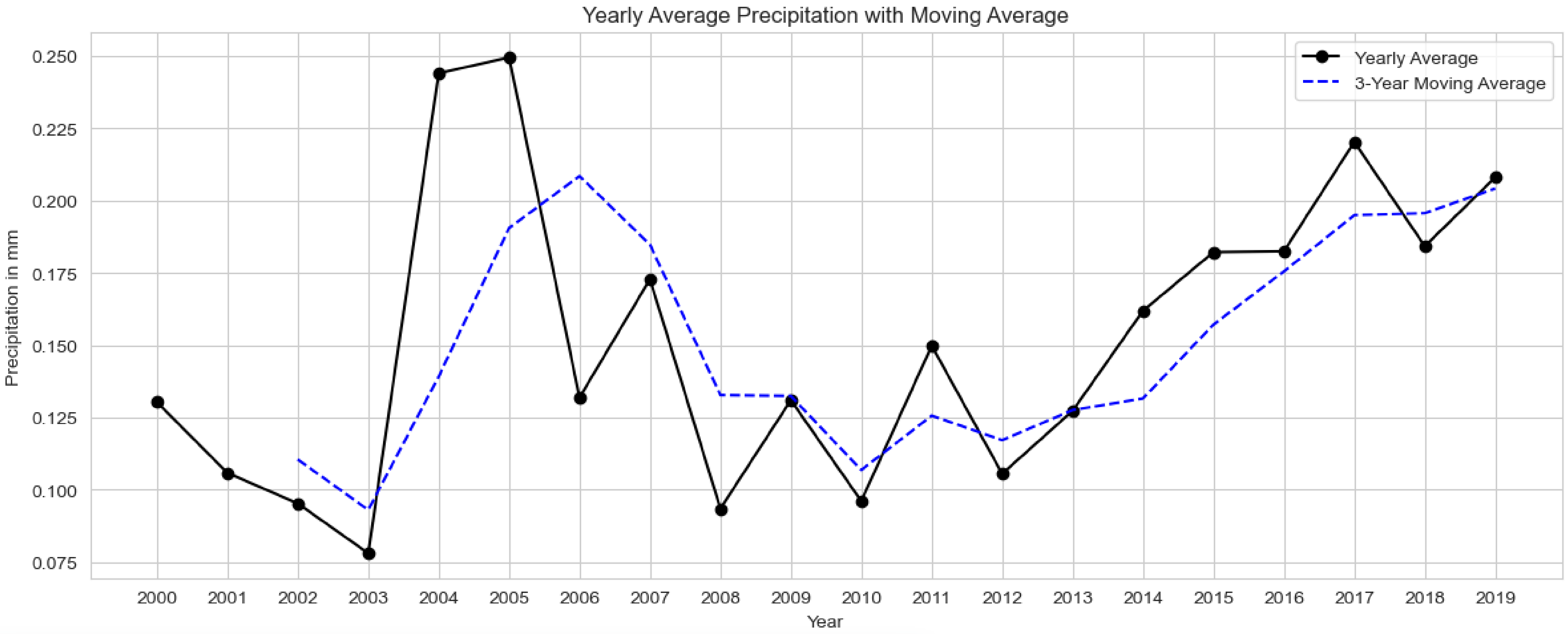
```python
annual_precipitation = train_df.resample('YE').sum()['PRCP']

annual_precipitation_sorted = annual_precipitation.sort_values()

sns.set_style("whitegrid")
plt.figure(figsize=(10, 6))

sns.barplot(x=annual_precipitation_sorted.index.year.astype(str), y=annual_precipitation_sorted.values, palette="plasma", hue=annual_precipitation_sorted.index.year.astype(str))

plt.title('Total Annual Precipitation (Ascending Order)', fontsize=14)
plt.xlabel('Year', fontsize=12)
plt.ylabel('Total Precipitation (mm)', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()

```



Correlation Heatmap for Trend Detection in Precipitation

```python
train_df['Year'] = train_df.index.year

yearly_avg = train_df.groupby('Year')['PRCP'].mean()

yearly_avg_df = pd.DataFrame(yearly_avg).reset_index()
yearly_avg_df.columns = ['Year', 'Average Precipitation']

heatmap_data = yearly_avg_df.set_index('Year')

plt.figure(figsize=(12, 5))
sns.heatmap(heatmap_data.T, annot=True, cmap='YlGnBu', cbar_kws={'label': 'Precipitation (mm)'})

plt.title('Yearly Average Precipitation Heatmap')
plt.xlabel('Year')
plt.tight_layout()
plt.show()
```

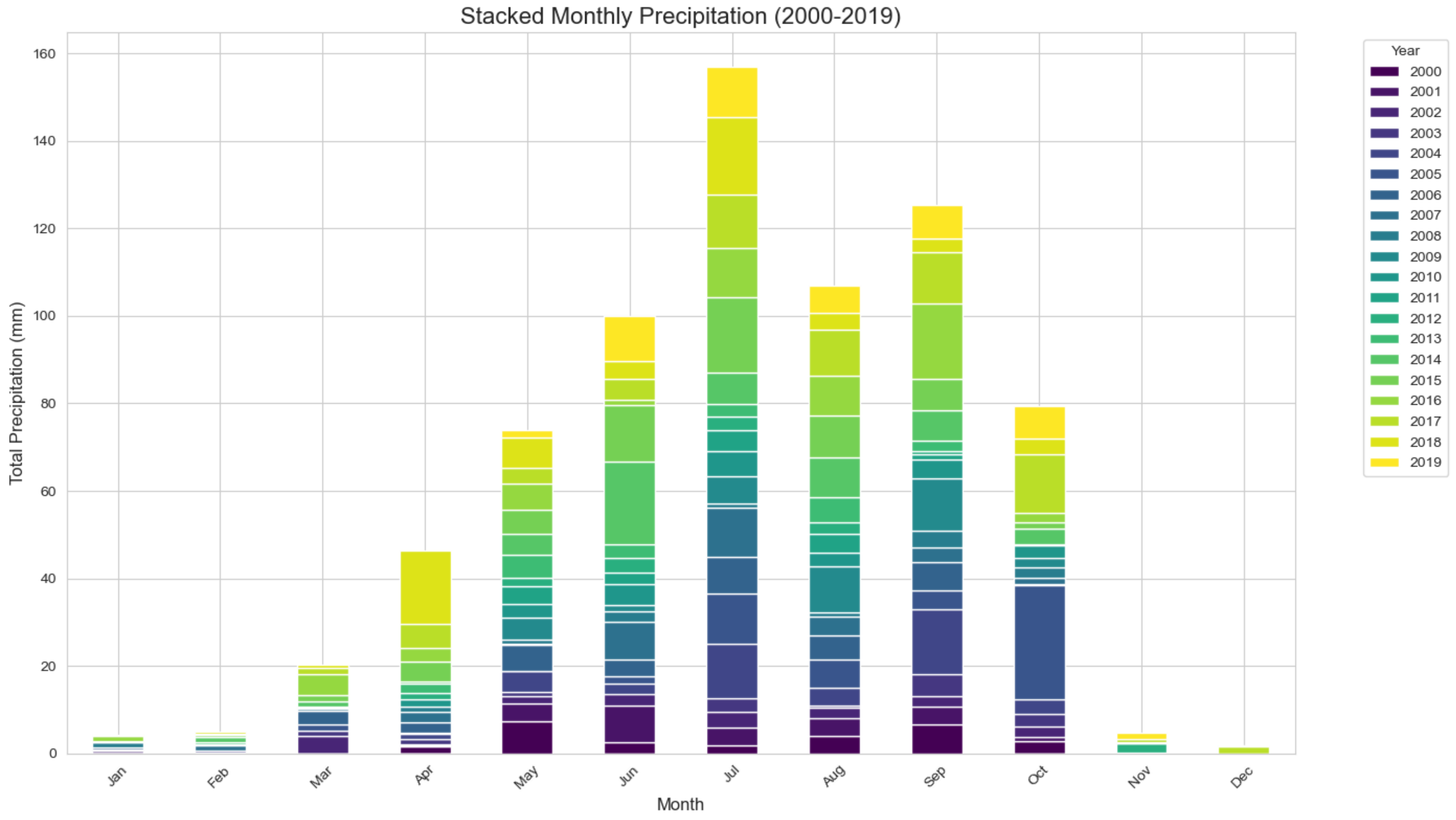




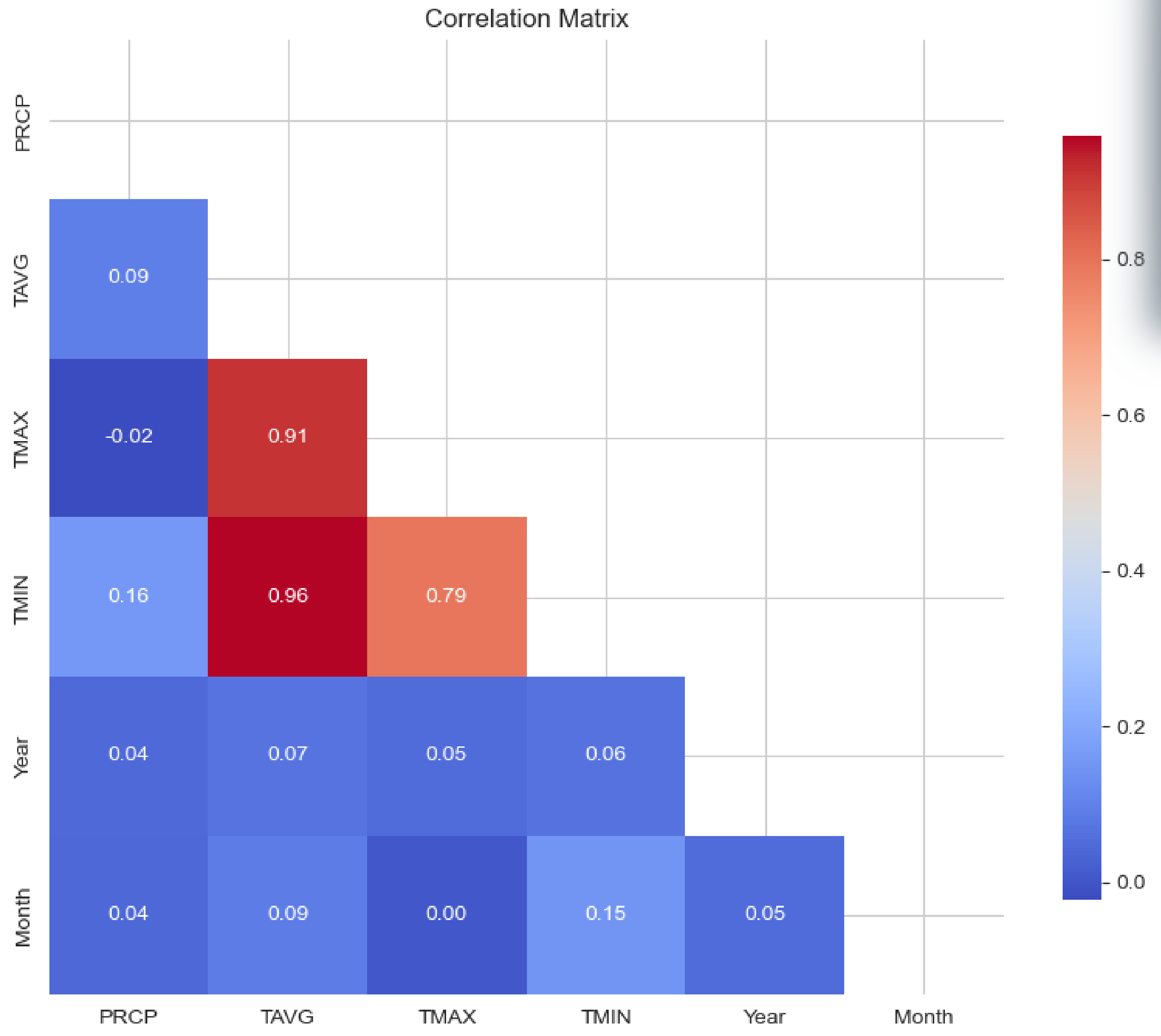

```python
yearly_avg = train_df.groupby('Year')['PRCP'].mean()
moving_avg = yearly_avg.rolling(window=3).mean()

plt.figure(figsize=(12, 5))
plt.plot(yearly_avg.index, yearly_avg.values, marker='o', linestyle='-', color='black', label='Yearly Average')
plt.plot(moving_avg.index, moving_avg.values, linestyle='--', color='blue', label='3-Year Moving Average')

plt.title('Yearly Average Precipitation with Moving Average')
plt.xlabel('Year')
plt.ylabel('Precipitation in mm')
plt.xticks(yearly_avg.index.astype(int))
plt.legend()
plt.tight_layout()
plt.show()
```

Moving Average Trend Detection for Precipitation

```python
train_df['Year'] = train_df.index.year
train_df['Month'] = train_df.index.month
monthly_total = train_df.groupby(['Year', 'Month'])['PRCP'].sum().unstack(level=0)

plt.figure(figsize=(14, 8))
monthly_total.plot(kind='bar', stacked=True, colormap="viridis", figsize=(14, 8))

plt.title('Stacked Monthly Precipitation (2000-2019)', fontsize=16)
plt.xlabel('Month', fontsize=12)
plt.ylabel('Total Precipitation (mm)', fontsize=12)
plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'], rotation=45)

plt.legend(title='Year', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()

plt.show()

```



Monthly Precipitation for weather Extremeness measurement



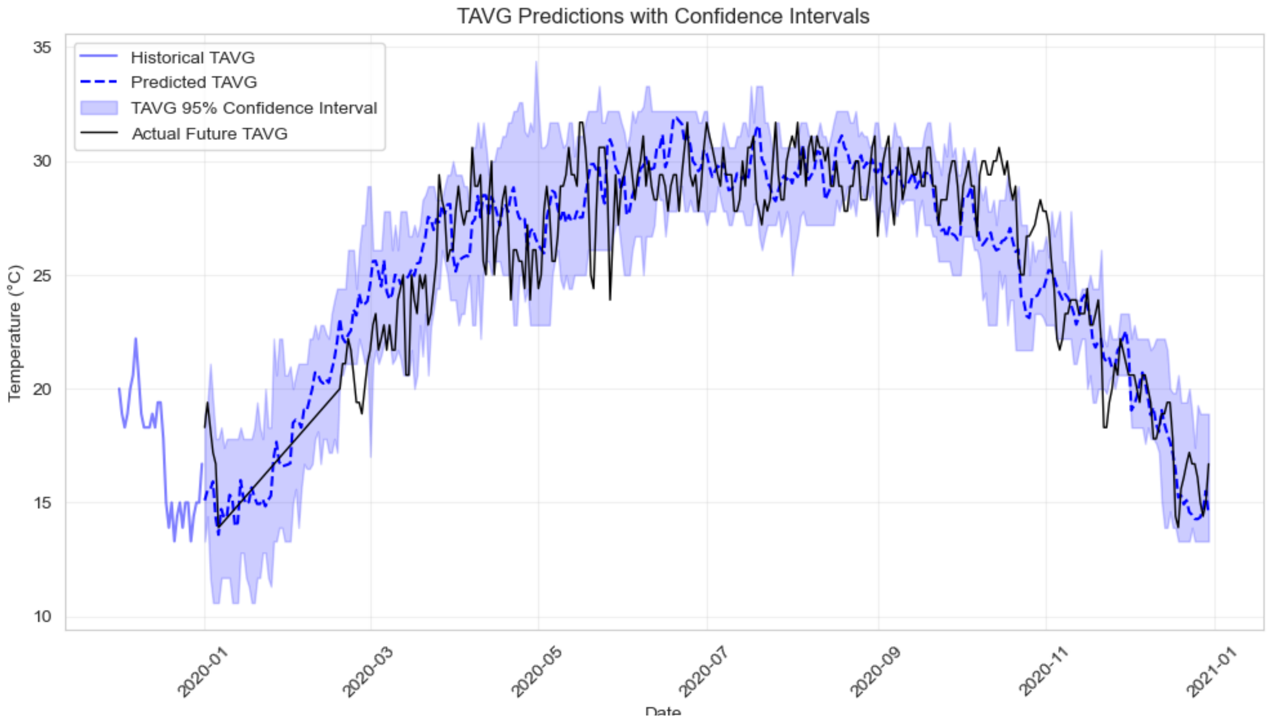

```python
corr = train_df.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))

plt.figure(figsize=(10, 8))
sns.heatmap(corr, mask=mask, cmap='coolwarm', annot=True,
fmt=".2f", square=True, cbar_kws={"shrink": .8})
plt.title('Correlation Matrix')
plt.show()
```

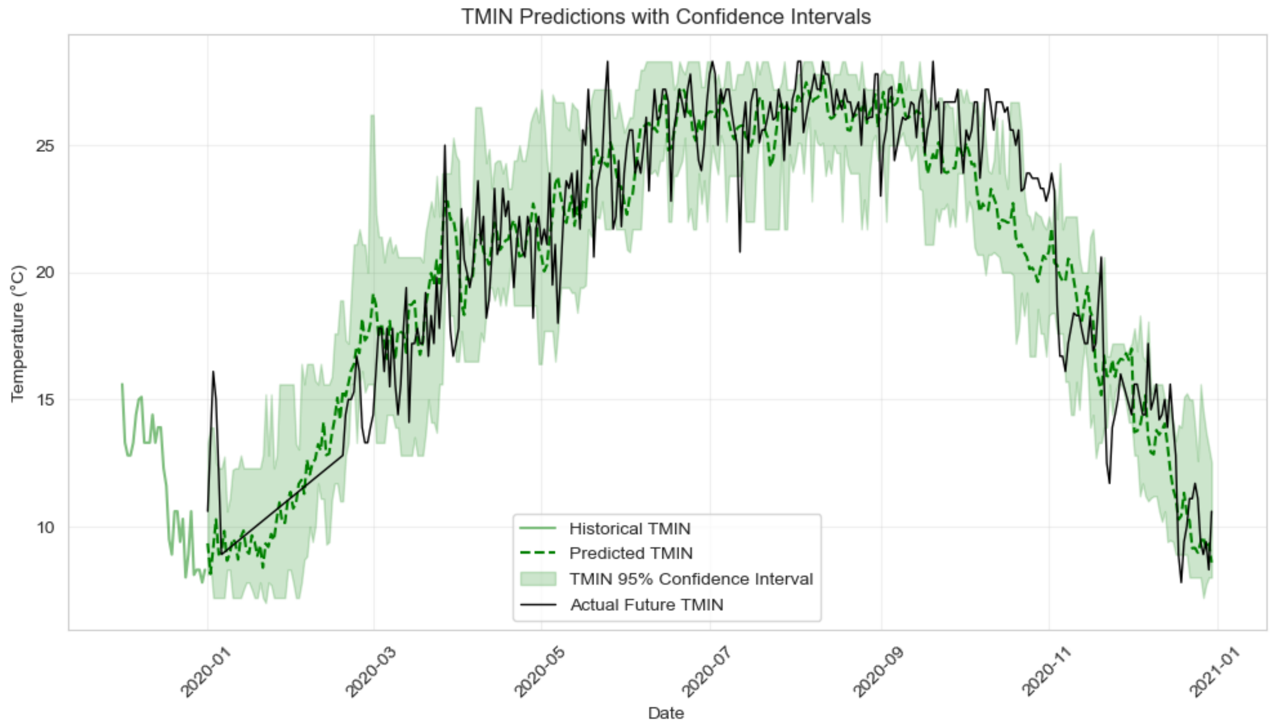
Correlation Analysis

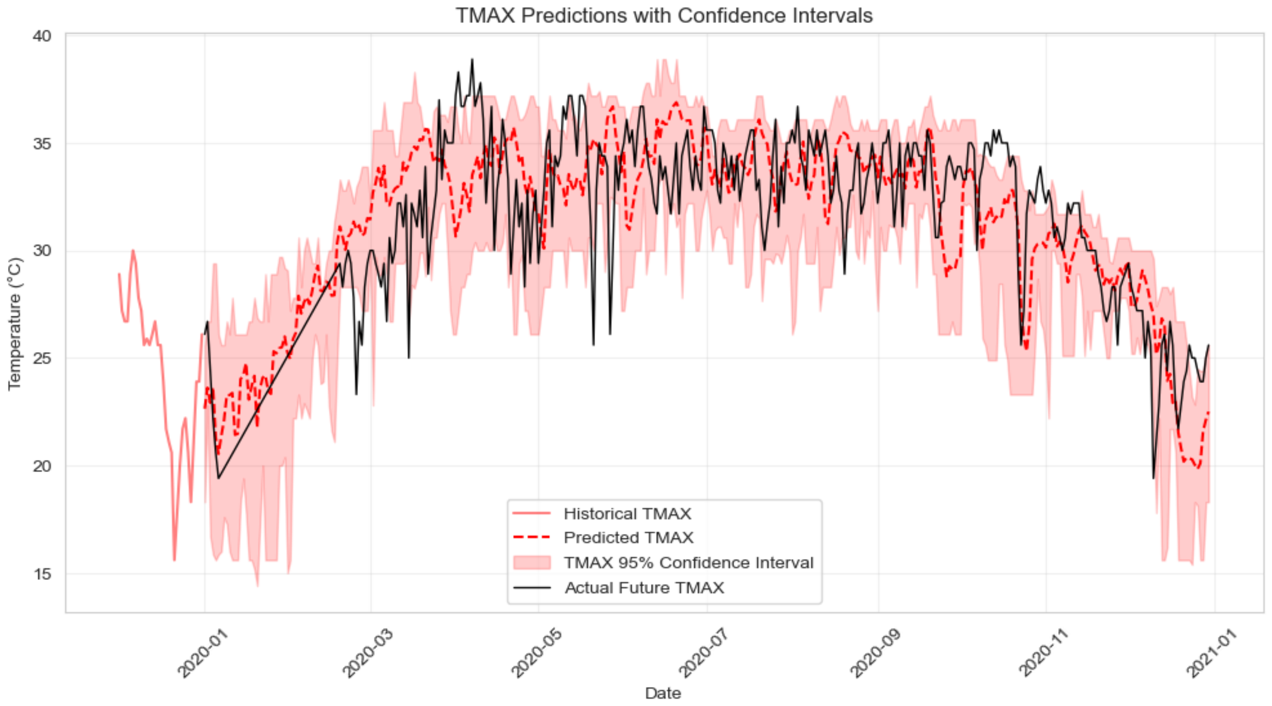
Model Training and Performance Measurement

```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, root_mean_squared_log_error

def mean_bias_error(y_true, y_pred):
    return np.mean(y_true - y_pred)

def mean_absolute_percentage_error(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100


def train(df):
    X = prepare_features(df)
    global features
    features = X.columns.tolist()

    for temp_type in ['TAVG', 'TMAX', 'TMIN']:
        y = df[temp_type]

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
        X_train_np = X_train.values
        X_test_np = X_test.values

        models[temp_type]['rf'].fit(X_train_np, y_train)
        models[temp_type]['xgb'].fit(X_train_np, y_train)

        rf_pred = models[temp_type]['rf'].predict(X_test_np)
        xgb_pred = models[temp_type]['xgb'].predict(X_test_np)

        print(f"\nModel Performance for {temp_type}:")
        print(f"Random Forest RMSLE Score: {root_mean_squared_log_error(y_test, rf_pred)}")
        print(f"Random Forest RMSE Score: {np.sqrt(mean_squared_error(y_test, rf_pred))}")
        print(f"Random Forest MAPE Score: {mean_absolute_percentage_error(y_test, rf_pred)}")

        print(50*'=')

        print(f"XGBoost RMSLE Score: {root_mean_squared_log_error(y_test, xgb_pred)}")
        print(f"XGBoost RMSE Score: {np.sqrt(mean_squared_error(y_test, xgb_pred))}")
        print(f"XGBoost MAPE Score: {mean_absolute_percentage_error(y_test, xgb_pred)}")


train(train_df)
```

```
Model Performance for TAVG:
Random Forest RMSLE Score: 0.060031115355293437
Random Forest RMSE Score: 1.450324718600018
Random Forest MAPE Score: 4.800244898556525
==================================================
XGBoost RMSLE Score: 0.06264418420002439
XGBoost RMSE Score: 1.513683072057833б
XGBoost MAPE Score: 5.026013454228542

Model Performance for TMAX:
Random Forest RMSLE Score: 0.06365790766549992
Random Forest RMSE Score: 1.8937605931519088
Random Forest MAPE Score: 4.681665135765115
==================================================
XGBoost RMSLE Score: 0.06836094378459093
XGBoost RMSE Score: 2.035190205922834
XGBoost MAPE Score: 5.074077803415624

Model Performance for TMIN:
Random Forest RMSLE Score: 0.09512417274757966
Random Forest RMSE Score: 1.7122844939396322
Random Forest MAPE Score: 7.42408032503871
==================================================
XGBoost RMSLE Score: 0.09754750395828822
XGBoost RMSE Score: 1.7476898061420194
XGBoost MAPE Score: 7.467356188027982
```







Performance measurement on test data (with Confidence Interval)

Model Performance

Metric	TAVG	TMIN	TMAX
RMSE	3.89	4.61	3.61
MAE	2.99	3.65	2.82
RMSLE	0.17	0.27	0.12
MAPE	13.14	21.23	9.33

Future Temperature Prediction with CI

```
target_dates = ['2020-06-15']
get_temperature_for_dates(results, target_dates)
```

```
Available prediction date range:
From: 2020-01-01 00:00:00
To: 2020-12-30 00:00:00
================================================================


Predictions for 2020-06-15:
Average Temperature (TAVG): 31.2°C
TAVG 95% Confidence Interval: [28.3°C to 32.2°C]

Minimum Temperature (TMIN): 27.0°C
TMIN 95% Confidence Interval: [25.3°C to 28.3°C]

Maximum Temperature (TMAX): 36.0°C
TMAX 95% Confidence Interval: [33.3°C to 38.9°C]
Temperature range for 2020-06-15 25.3°C to 38.9°C
```

The temperature is predicted for a specific date in the future using a completely separate holdout set. The actual values for that day was:

Average Temp. : 29.4°C
Minimum Temp. : 26.1°C
Maximum Temp. : 33.3°C

Which suggests that the model is performing very well on future data.

Conclusions and Results

The models show strong predictive performance with:
- Low RMSE values indicating accurate predictions
- Reasonable confidence intervals capturing temperature variations
- Good handling of seasonal patterns

Key Findings:
- Temperature predictions maintain realistic ranges between min and max values
- Models capture seasonal patterns effectively
- Ensemble approach (RF + XGBoost) provides robust predictions
- 'TMIN' gave high MAPE which is the cause of data imputation and having too much missing data

Visualization Results:
- Clear seasonal patterns in temperature data
- Strong correlation between different temperature metrics
- Effective confidence interval estimation for predictions

Model Performance:
- XGBoost generally outperforms Random Forest
- Combined ensemble approach provides more stable predictions
- Models handle both short-term and long-term predictions effectively

Thank you

Md. Rafiquzzaman Rafi

+8801846385072

rafiquzzamanrafi100@gmail.com