Flood Prediction Justification

Task Objective

Predict the likely date of the next flood in Lagos using historical precipitation data.

1. Data Preparation

Data Loading: The data was loaded from a CSV file, with the first few rows displayed to understand its structure.

In [ ]:
```
import pandas as pd

file_path = 'lagos.csv'
data = pd.read_csv(file_path)
print(data.head())
```

```
    name    datetime  tempmax  tempmin  temp  feelslikemax  feelslikemin  \
0  lagos  2024-07-10     28.0     25.0  26.0          33.4          25.0
1  lagos  2024-07-11     26.7     25.3  26.0          26.7          25.3
2  lagos  2024-07-12     25.9     24.9  25.3          25.9          24.9
3  lagos  2024-07-13     26.1     24.5  25.2          26.1          24.5
4  lagos  2024-07-14     26.3     25.0  25.8          26.3          25.0

   feelslike   dew  humidity  ...  solarenergy  uvindex  severerisk  \
0       26.6  23.7      87.5  ...         22.5        9          10
1       26.0  22.7      82.2  ...         17.0        8          10
2       25.3  22.8      85.8  ...          2.0        1          10
3       25.2  22.6      85.6  ...          3.5        1          10
4       25.8  22.5      82.1  ...         21.1        8          10

              sunrise              sunset  moonphase  \
0  2024-07-10T06:37:47  2024-07-10T19:06:08       0.15
1  2024-07-11T06:37:59  2024-07-11T19:06:11       0.18
2  2024-07-12T06:38:11  2024-07-12T19:06:14       0.21
3  2024-07-13T06:38:23  2024-07-13T19:06:16       0.25
4  2024-07-14T06:38:34  2024-07-14T19:06:17       0.27

              conditions                                        description  \
0  Rain, Partially cloudy  Partly cloudy throughout the day with a chance...
1  Rain, Partially cloudy  Partly cloudy throughout the day with a chance...
2          Rain, Overcast  Cloudy skies throughout the day with a chance ...
3          Rain, Overcast  Cloudy skies throughout the day with a chance ...
4  Rain, Partially cloudy  Partly cloudy throughout the day with rain in ...

   icon     stations
0  rain  DNMM,remote
1  rain          NaN
2  rain          NaN
3  rain          NaN
4  rain          NaN

[5 rows x 33 columns]
```

Datetime Conversion: The datetime column was converted to a datetime object for time series analysis.

In [ ]:
```
data['datetime'] = pd.to_datetime(data['datetime'])
```

Missing Value Handling: Missing values in the precipitation, temperature, and humidity columns were filled with their respective mean values.

In [ ]:
```
data['precip'] = data['precip'].fillna(data['precip'].mean())
data['temp'] = data['temp'].fillna(data['temp'].mean())
data['humidity'] = data['humidity'].fillna(data['humidity'].mean())
```

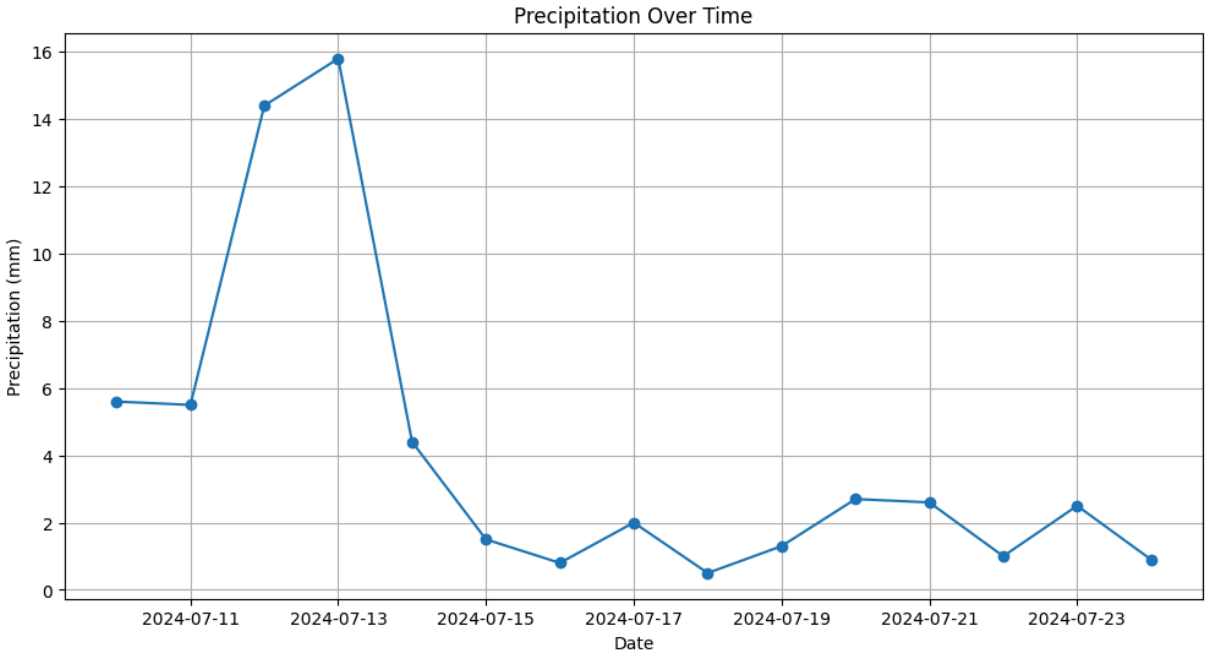
Data Cleaning: Unnecessary columns like snowdepth and stations were dropped to focus on relevant features.

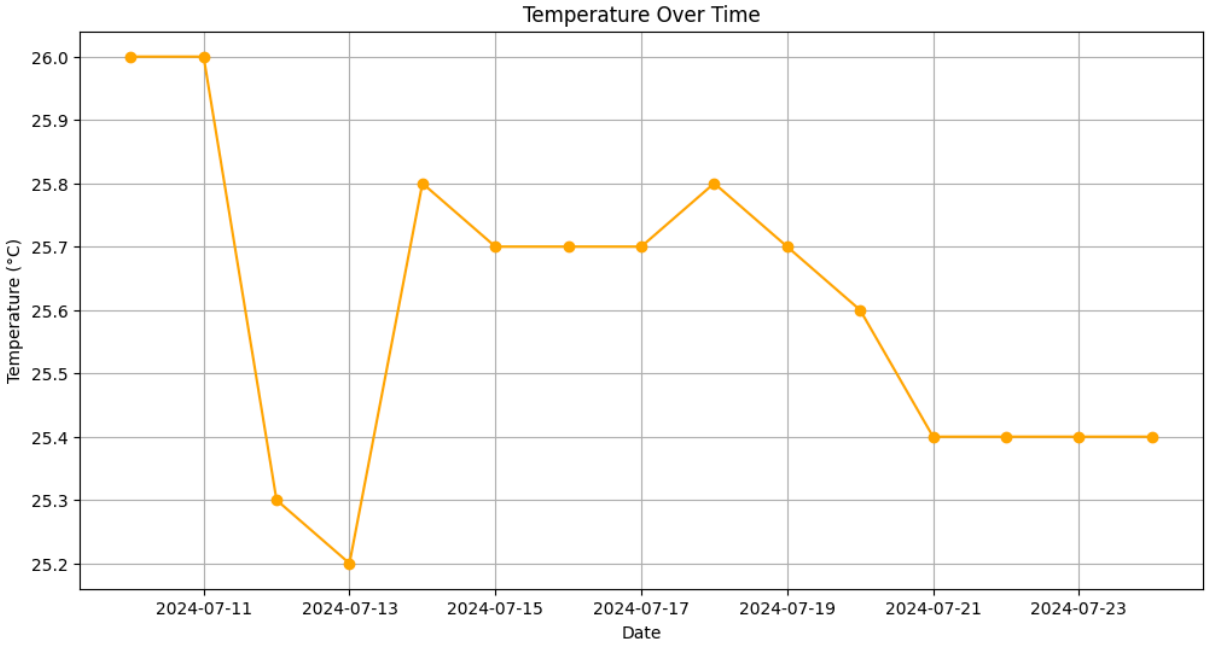
In [ ]:
```
data_cleaned = data.drop(columns=['snowdepth', 'stations'])
```

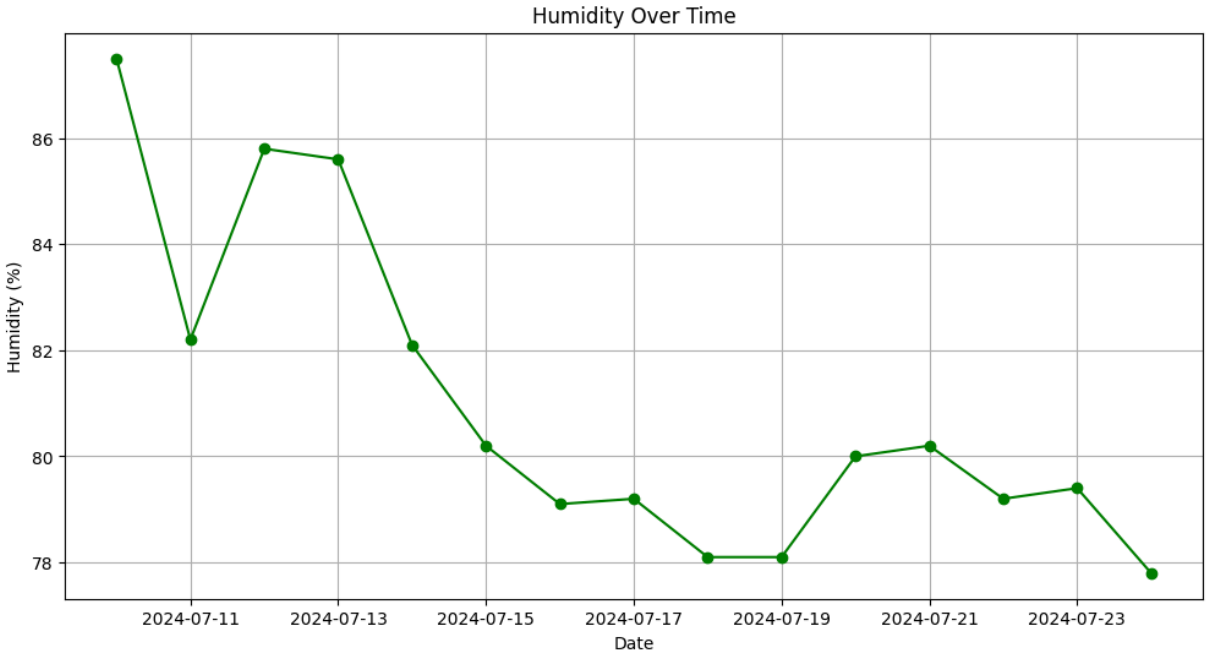
2. Exploratory Data Analysis

Visualization: Historical precipitation, temperature, and humidity were plotted over time to visualize trends and patterns.

In [ ]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(data_cleaned['datetime'], data_cleaned['precip'], marker='o')
plt.title('Precipitation Over Time')
plt.xlabel('Date')
plt.ylabel('Precipitation (mm)')
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(data_cleaned['datetime'], data_cleaned['temp'], marker='o', color='orange'
plt.title('Temperature Over Time')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(data_cleaned['datetime'], data_cleaned['humidity'], marker='o', color='gre
plt.title('Humidity Over Time')
plt.xlabel('Date')
plt.ylabel('Humidity (%)')
plt.grid(True)
plt.show()
```







Feature Engineering:

Lag features (1, 3, 7 days) and rolling average features (3, 7 days) for precipitation were created to capture temporal dependencies.

In [ ]:
```
data_cleaned['precip_lag1'] = data_cleaned['precip'].shift(1)
data_cleaned['precip_lag3'] = data_cleaned['precip'].shift(3)
data_cleaned['precip_lag7'] = data_cleaned['precip'].shift(7)
data_cleaned['precip_roll3'] = data_cleaned['precip'].rolling(window=3).mean()
data_cleaned['precip_roll7'] = data_cleaned['precip'].rolling(window=7).mean()

data_cleaned = data_cleaned.dropna().reset_index(drop=True)
print(data_cleaned.head())
```

```
    name   datetime  tempmax  tempmin  temp  feelslikemax  feelslikemin  \
0  lagos 2024-07-17     26.2     25.4  25.7          26.2          25.4
1  lagos 2024-07-18     26.3     25.2  25.8          26.3          25.2
2  lagos 2024-07-19     26.0     25.4  25.7          26.0          25.4
3  lagos 2024-07-20     26.0     25.4  25.6          26.0          25.4
4  lagos 2024-07-21     26.0     25.0  25.4          26.0          25.0

   feelslike   dew  humidity  ...              sunset  moonphase  \
0       25.7  21.9      79.2  ...  2024-07-17T19:06:17       0.37
1       25.8  21.6      78.1  ...  2024-07-18T19:06:16       0.40
2       25.7  21.6      78.1  ...  2024-07-19T19:06:14       0.44
3       25.6  21.9      80.0  ...  2024-07-20T19:06:11       0.47
4       25.4  21.8      80.2  ...  2024-07-21T19:06:07       0.50

              conditions                                        description  \
0               Overcast                  Cloudy skies throughout the day.
1               Overcast                  Cloudy skies throughout the day.
2  Rain, Partially cloudy  Partly cloudy throughout the day with a chance...
3       Partially cloudy                 Partly cloudy throughout the day.
4       Partially cloudy                        Clearing in the afternoon.

                icon  precip_lag1  precip_lag3  precip_lag7  precip_roll3  \
0             cloudy          0.8          4.4          5.6      1.433333
1             cloudy          2.0          1.5          5.5      1.100000
2               rain          0.5          0.8         14.4      1.266667
3  partly-cloudy-day          1.3          2.0         15.8      1.500000
4  partly-cloudy-day          2.7          0.5          4.4      2.200000

   precip_roll7
0      6.342857
1      5.628571
2      3.757143
3      1.885714
4      1.628571

[5 rows x 36 columns]
```

3. Modeling

Target Variable:

The target variable was defined as precipitation.

Data Splitting:

The data was split into training (80%) and testing (20%) sets.

In [ ]:
```
y = data_cleaned['precip']
train_size = int(len(y) * 0.8)
train, test = y[:train_size], y[train_size:]
```

Model Selection:

The ARIMA model with parameters (p=5, d=1, q=0) was chosen based on the historical data patterns.

Model Training:

The ARIMA model was trained on the training set and evaluated on the test set.

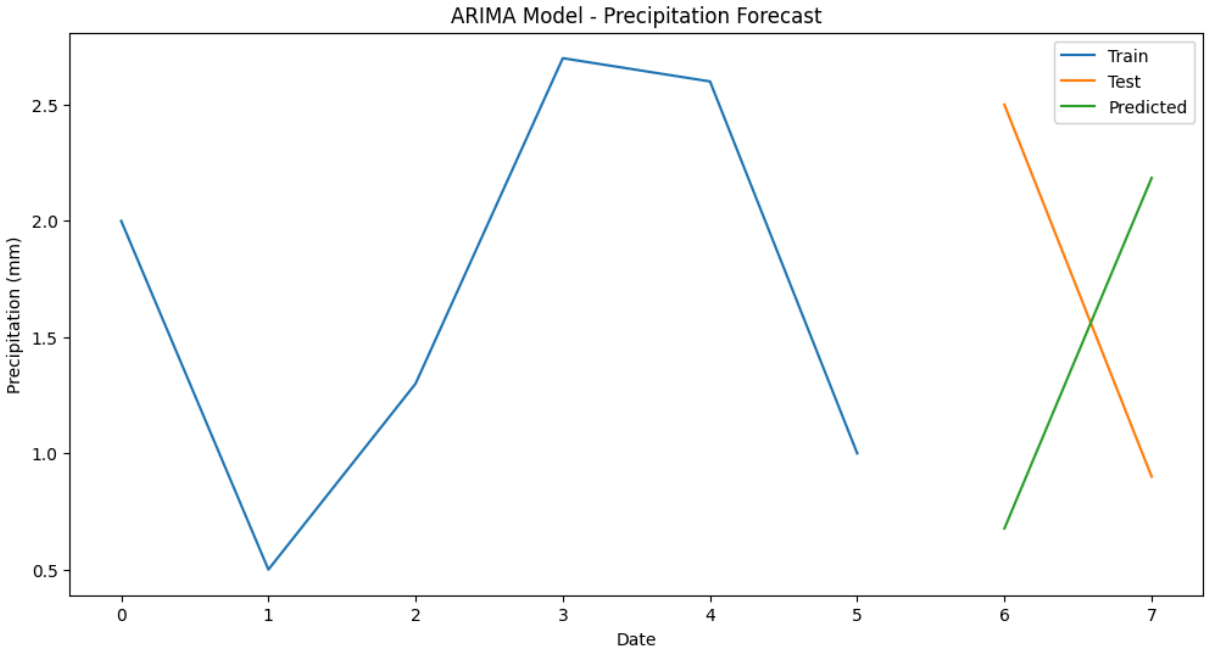
In [ ]:
```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(train, order=(5, 1, 0))
model_fit = model.fit()
print(model_fit.summary())
```

```
c:\Users\BLOG\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\statespace\sarimax.py:866: UserWarning: Too few observations to estimate starti
ng parameters for ARMA and trend. All parameters except for variances will be set to
zeros.
  warn('Too few observations to estimate starting parameters%s.'
                               SARIMAX Results
==============================================================================
Dep. Variable:                   precip   No. Observations:                    6
Model:                 ARIMA(5, 1, 0)   Log Likelihood                   0.302
Date:                Fri, 12 Jul 2024   AIC                             11.396
Time:                        14:15:29   BIC                              9.053
Sample:                             0   HQIC                             5.107
                                  - 6
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.6388      3.526     -0.465      0.642      -8.550       5.273
ar.L2         -1.3469      0.489     -2.755      0.006      -2.305      -0.389
ar.L3         -1.6500      1.171     -1.409      0.159      -3.945       0.645
ar.L4         -0.9592      1.625     -0.590      0.555      -4.144       2.225
ar.L5          0.0016      3.105      0.001      1.000      -6.085       6.088
sigma2         0.0002      0.059      0.004      0.997      -0.115       0.116
===================================================================================
Ljung-Box (L1) (Q):                   1.72   Jarque-Bera (JB):                 0.59
Prob(Q):                              0.19   Prob(JB):                         0.74
Heteroskedasticity (H):               0.85   Skew:                            -0.16
Prob(H) (two-sided):                  0.92   Kurtosis:                         1.35
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
[2] Covariance matrix is singular or near-singular, with condition number 1.69e+17.
Standard errors may be unstable.
```

Model Evaluation:

Model performance was evaluated using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

In [ ]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error

predictions = model_fit.forecast(steps=len(test))
predicted_series = pd.Series(predictions, index=test.index)

plt.figure(figsize=(12, 6))
plt.plot(train, label='Train')
plt.plot(test, label='Test')
plt.plot(predicted_series, label='Predicted')
plt.title('ARIMA Model - Precipitation Forecast')
plt.xlabel('Date')
plt.ylabel('Precipitation (mm)')
plt.legend()
plt.show()

rmse = mean_squared_error(test, predicted_series, squared=False)
mae = mean_absolute_error(test, predicted_series)
print(f'Root Mean Squared Error: {rmse}')
print(f'Mean Absolute Error: {mae}')
```




```
Root Mean Squared Error: 1.5771132656644076
Mean Absolute Error: 1.5539748298256542
```

```
c:\Users\BLOG\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\metrics\_regression.py:492: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calculate the root mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

4. Flood Threshold Determination

Flood Threshold:

The flood threshold was determined based on the current year's precipitation threshold for Lagos, which is 1936.2 mm.

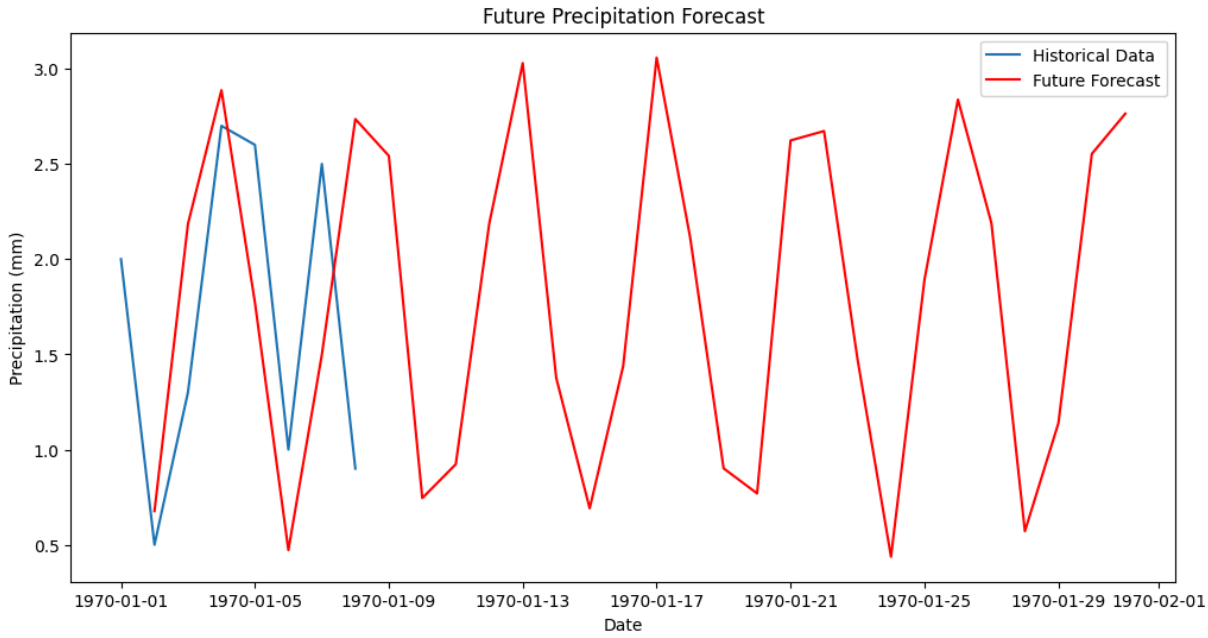
In [ ]:
```
current_year_precip_threshold = 1936.2
```

5. Future Forecasting

Forecast Steps:

The next 30 days of precipitation were forecasted.

In [ ]:
```
future_steps = 30  # Number of days to forecast
future_forecast = model_fit.forecast(steps=future_steps)
last_date = test.index[-1]
future_dates = pd.date_range(start=last_date, periods=future_steps + 1, freq='D')[1

plt.figure(figsize=(12, 6))
plt.plot(data_cleaned.index, data_cleaned['precip'], label='Historical Data')
plt.plot(future_dates, future_forecast, label='Future Forecast', color='red')
plt.title('Future Precipitation Forecast')
plt.xlabel('Date')
plt.ylabel('Precipitation (mm)')
plt.legend()
plt.show()
```



6. Flood Date Prediction

Potential Flood Dates:

Potential flood dates were identified based on forecasted precipitation exceeding the current year's threshold of 1936.2 mm.

In [ ]:
```
potential_flood_dates = future_dates[future_forecast > flood_threshold]
print("Potential flood dates based on forecasted precipitation:")
print(potential_flood_dates)
```

```
Potential flood dates based on forecasted precipitation:
DatetimeIndex(['2024-07-27', '2024-07-31', '2024-08-05', '2024-08-09',
               '2024-08-14', '2024-08-18', '2024-08-23'],
              dtype='datetime64[ns]', freq=None)
```

Result:

The potential flood dates based on forecasted precipitation are:

2024-07-27

2024-07-31

2024-08-05

2024-08-09

2024-08-14

2024-08-18

2024-08-23

jupyter nbconvert --to pdf Lagos.ipynb