Developing a Deep Learning Model for Predicting Agricultural Weather Conditions and Crop Water Requirements in Hyderabad - India Using Time Series Data

Overview

India, a country that occupies the greater part of South Asia, consists of 28 states and eight union territories, with its national capital in New Delhi. The government is a constitutional republic that represents a highly diverse population, including thousands of ethnic groups and hundreds of languages. As of 2023, India became the world's most populous country, according to United Nations estimates.

Archaeological evidence shows that a highly sophisticated urbanized culture—the Indus civilization—dominated the northwestern part of the subcontinent from about 2600 to 2000 BCE. India has been a self-contained political and cultural arena, giving rise to a distinctive tradition primarily associated with Hinduism. Other religions like Buddhism and Jainism also originated here, contributing to India's rich intellectual life in fields such as mathematics, astronomy, architecture, literature, music, and the fine arts.

Throughout its history, India experienced incursions from beyond its northern mountain wall, significantly impacting its culture. Islam, introduced by Arab, Turkish, Persian, and other raiders from the northwest beginning early in the 8th century CE, eventually led to Muslim rule over much of the subcontinent by the 13th century. The arrival of Portuguese navigator Vasco da Gama in 1498 and the establishment of European maritime supremacy heralded major external influences, culminating in British rule. British administration began in 1858, unifying the subcontinent politically and economically. India gained independence in 1947, partitioned into India and Pakistan along religious lines.

India remains one of the world's most ethnically diverse countries. Apart from its many religions and sects, India is home to innumerable castes and tribes and more than a dozen major linguistic groups. Efforts to instill a spirit of nationhood have been made, though tensions and violence between neighboring groups have occurred. Despite challenges, India has shown physical prosperity and cultural dynamism, evident in its infrastructure, diversified industrial base, scientific and engineering personnel, and cultural exports in music, literature, and cinema. India has three of the world's most populous cities—Mumbai, Kolkata, and Delhi—and rapidly growing high-technology centers like Bengaluru, Chennai, and Hyderabad.

India's geography includes the Himalayas, the Indo-Gangetic Plain, and the Deccan plateau. The Himalayas, the world's loftiest mountain system, form the northern boundary. The Indo-Gangetic Plain, stretching between the Himalayas and the Deccan, is a densely populated and agriculturally fertile area. The Deccan plateau, constituting the bulk of peninsular India, is geologically stable, with varying topography.

India experiences a diverse range of climatic conditions due to its vast geographical expanse and varied topography. The country primarily has a tropical monsoon climate, characterized by distinct wet and dry seasons. The monsoon season, from June to September, brings heavy rainfall to most parts of the country, crucial for agriculture. Northern India, particularly the Himalayan region, experiences cold winters with snowfall, while central and southern parts have warmer climates. Coastal regions enjoy moderate temperatures year-round. Extreme weather events such as cyclones, droughts, and heatwaves are common, impacting agriculture, water resources, and daily life. Understanding these weather patterns is essential for effective planning and management in sectors like agriculture, water resources, and disaster preparedness.

Hyderabad, a vibrant city in southern India, is known for its rich history, diverse culture, and rapid technological advancements. Nestled in the semi-arid region of Telangana, Hyderabad experiences a unique blend of climatic conditions that significantly influence its agriculture, economy, and daily life. The city's weather patterns, characterized by varying temperatures, humidity levels, and rainfall, play a crucial role in shaping its agricultural productivity and water resource management. With the increasing challenges posed by climate change, understanding and predicting these weather patterns have become more critical than ever.

Problem Statement

Accurately predicting weather conditions in Hyderabad, India, particularly critical events like droughts and extreme temperatures, remains a significant challenge. The existing models often fail to incorporate the complex interplay between time series and geospatial data, leading to less reliable forecasts. This gap in accurate weather prediction hinders effective agricultural planning and water resource management, which are crucial for sustaining the region's agrarian economy. Therefore, there is a pressing need to develop an advanced predictive model that can reliably forecast weather conditions by integrating various data types and capturing intricate weather patterns. This study aims to address this need by utilizing deep learning techniques to improve prediction accuracy and provide actionable insights for stakeholders.

Research Statement

The research aims to leverage deep learning techniques to develop a robust predictive model for weather conditions in Hyderabad. By incorporating both time series and geospatial data, the study will explore trends, patterns, and correlations among various weather parameters. The primary focus will be on predicting critical weather events, such as droughts and extreme temperatures, and understanding their impact on crop water requirements. This research will not only contribute to the scientific community's understanding of weather dynamics in the semi-arid tropics but also provide valuable insights for agricultural planning and water resource management in the region.

Data Source

The dataset for this research is provided by the International Crops Research Institute for the Semi-Arid Tropics (ICRISAT), an esteemed international non-profit organization dedicated to scientific research for development. Established in 1972, ICRISAT is headquartered in Patancheru, Hyderabad, India, and operates several regional centers across Africa, including Bamako (Mali) and Nairobi (Kenya), as well as research stations in Niamey (Niger), Kano (Nigeria), Lilongwe (Malawi), Addis Ababa (Ethiopia), and Bulawayo (Zimbabwe). India, the host country, has granted ICRISAT special status as a UN Organization, providing it with unique immunities and tax privileges.

ICRISAT has been collecting daily weather data in Hyderabad since 1978, resulting in a comprehensive dataset that spans over 40 years. This dataset includes crucial weather parameters such as maximum and minimum temperatures, relative humidity (morning and afternoon), wind speed, rainfall, bright sunshine hours, evaporation, radiation, and reference crop evapotranspiration (FAO56-ET). Additionally, it encompasses geospatial data, including longitude and latitude, enhancing the depth and scope of weather analysis.

Research Objectives

1. Develop and validate a deep learning model to accurately predict future weather conditions using historical time series and geospatial data.
2. Analyze trends and patterns in historical weather data to identify significant seasonal and geographical variations.
3. Utilize K-means clustering to classify seasons in Hyderabad.

Research Problems

1. How can a deep learning model be developed to predict future weather conditions, incorporating historical and geospatial data?
2. What trends and correlations exist in the weather data over time and across different locations?
3. How can seasonal variations in weather patterns in Hyderabad be effectively characterized and understood, considering the dynamic nature of climatic conditions?

This research will utilize ICRISAT's extensive dataset to address these objectives and problems, contributing to more informed agricultural practices and improved resource management in Hyderabad and similar semi-arid regions.

Hypothesis

Objective 1: Develop and validate a deep learning model to accurately predict future weather conditions using historical time series and geospatial data.

```
Null Hypothesis (H01): There is no significant relationship between historical time series data, geospatial data, and the accuracy of the deep learning model in predicting future weather conditions.

Alternative Hypothesis (H11): There is a significant relationship between historical time series data, geospatial data, and the accuracy of the deep learning model in predicting future weather conditions.
```

Objective 2: Analyze trends and patterns in historical weather data to identify significant seasonal and geographical variations.

```
Null Hypothesis (H02): There are no significant seasonal and geographical variations in historical weather data.

Alternative Hypothesis (H12): There are significant seasonal and geographical variations in historical weather data.
```

Objective 4: Utilize K-means clustering to classify seasons in Hyderabad.

```
Null Hypothesis (H0): There is no significant difference in seasonal weather patterns in Hyderabad that can be classified using clustering techniques.

Alternative Hypothesis (H1):
Seasonal weather patterns in Hyderabad exhibit significant differences that can be effectively classified and understood using clustering techniques.
```

Metrics of Success

1. If the model achieves low MSE, MAE, and RMSE with a high R2, it supports the alternative hypothesis (H11) that there is a significant relationship between historical time series data, geospatial data, and accurate weather predictions.
2. If significant seasonal and geographical variations are identified and validated statistically, it supports the alternative hypothesis (H12) that such variations exist in historical weather data.
3. If weather parameters and geographical factors are found to significantly correlate with crop evapotranspiration and water requirements, it supports the alternative hypothesis (H13) that these factors impact agricultural water needs.
4. If K-means clustering successfully identifies distinct seasonal weather patterns in Hyderabad, it supports the alternative hypothesis (H1) that there are significant differences in seasonal weather patterns that can be classified using clustering techniques.

Importing Libraries

```
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.sarimax import SARIMAX
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
#!pip install folium
import folium
import seaborn as sns
from sklearn.decomposition import PCA
import warnings
from sklearn.cluster import KMeans
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, GRU
from tensorflow.keras.optimizers import Adam
warnings.filterwarnings('ignore')

df=pd.read_excel("ICRISAT Weather 1978 to 2018.xlsx")
df.head()
```

	Station	Date	MaxT	MinT	RH1	RH2	Wind	Rain	SSH	Evap
0	ICRISAT	1978-01-01	28.5	14.2	68	31.0	5.7	0.0	10.1	4.3
1	ICRISAT	1978-01-02	28.8	16.0	79	33.0	6.4	0.0	9.8	4.8
2	ICRISAT	1978-01-03	29.0	14.5	86	37.0	5.4	0.0	9.1	4.6
3	ICRISAT	1978-01-04	29.0	18.0	89	43.0	7.1	0.0	9.0	4.2
4	ICRISAT	1978-01-05	27.8	17.0	81	47.0	10.5	0.0	8.9	4.3

	Radiation	FAO56_ET	Lat	Lon	Cum_Rain
0	18.4	3.9	17.508409	78.2723	0.0
1	16.9	3.9	17.508409	78.2723	0.0
2	15.3	3.4	17.508409	78.2723	0.0
3	16.4	3.8	17.508409	78.2723	0.0
4	15.9	4.1	17.508409	78.2723	0.0

```
df.shape
```

```
(14853, 15)
```

```
df.describe().T
```

```
             count                 mean                  min  \
Date         14853  1998-05-02 00:00:00  1978-01-01 00:00:00
MaxT       14853.0            32.055807                 16.5
MinT       14853.0             19.56872                  4.5
RH1        14853.0            81.586481                 17.0
RH2        14853.0            43.550549                  6.3
Wind       14853.0             8.692278                  0.2
Rain       14853.0             2.460378                  0.0
SSH        14853.0             7.457052                  0.0
Evap       14853.0             6.420043                  0.0
Radiation  14852.0            17.877027                  0.8
FAO56_ET   14853.0             4.808901                  0.4
Lat        14853.0            17.508409             17.508409
Lon        14853.0              78.2723               78.2723
Cum_Rain   14853.0           386.065347                  0.0

                            25%                   50%
75%  \
Date       1988-03-02 00:00:00  1998-05-02 00:00:00  2008-07-01
00:00:00
MaxT                      29.0                  31.0
35.0
MinT                      16.5                  21.0
22.6
RH1                       75.0                  87.0
93.0
RH2                       28.0                  40.0
57.0
Wind                       5.2                   7.6
11.2
Rain                       0.0                   0.0
0.0
SSH                        5.5                   8.8
10.1
Evap                       4.2                   5.6
8.3
Radiation                 15.5                  18.2
21.1
FAO56_ET                   3.5                   4.4
5.9
Lat                  17.508409             17.508409
17.508409
Lon                    78.2723               78.2723
78.2723
Cum_Rain                  29.6                 225.4
687.7
```

```
                              max         std
Date       2018-08-31 00:00:00         NaN
MaxT                      43.5    4.115165
MinT                      30.6    4.504939
RH1                      100.0   15.055169
RH2                      100.0   19.613665
Wind                      56.0    4.795461
Rain                     263.6    9.346085
SSH                       12.4    3.341868
Evap                      19.7    3.132334
Radiation                 28.3    4.508112
FAO56_ET                  13.6    1.816996
Lat                  17.508409         0.0
Lon                    78.2723         0.0
Cum_Rain                1520.4  398.752032
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14853 entries, 0 to 14852
Data columns (total 15 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Station    14853 non-null  object
 1   Date       14853 non-null  datetime64[ns]
 2   MaxT       14853 non-null  float64
 3   MinT       14853 non-null  float64
 4   RH1        14853 non-null  int64
 5   RH2        14853 non-null  float64
 6   Wind       14853 non-null  float64
 7   Rain       14853 non-null  float64
 8   SSH        14853 non-null  float64
 9   Evap       14853 non-null  float64
 10  Radiation  14852 non-null  float64
 11  FAO56_ET   14853 non-null  float64
 12  Lat        14853 non-null  float64
 13  Lon        14853 non-null  float64
 14  Cum_Rain   14853 non-null  float64
dtypes: datetime64[ns](1), float64(12), int64(1), object(1)
memory usage: 1.7+ MB
```

The dataset comprises 14,853 entries and 15 columns, detailing various meteorological measurements. The columns include 'Station' (categorical), 'Date' (datetime), 'MaxT' and 'MinT' (maximum and minimum temperatures, respectively), 'RH1' and 'RH2' (relative humidity measurements), 'Wind' (wind speed), 'Rain' (rainfall), 'SSH' (sunshine hours), 'Evap' (evaporation), 'Radiation', 'FAO56_ET' (reference crop evapotranspiration), 'Lat' and 'Lon' (latitude and longitude), and 'Cum_Rain' (cumulative rainfall). All columns contain non-null values except for 'Radiation', which has one missing value. The data types are predominantly float64 for numerical measurements, with one integer column for relative humidity ('RH1'), and

categorical ('Station') and datetime ('Date') columns. The dataset occupies approximately 1.7 MB of memory.

Checking for missing values

```
df.isnull().sum()
```

```
Station      0
Date         0
MaxT         0
MinT         0
RH1          0
RH2          0
Wind         0
Rain         0
SSH          0
Evap         0
Radiation    1
FAO56_ET     0
Lat          0
Lon          0
Cum_Rain     0
dtype: int64
```

The dataset contains a total of 14,853 entries and 15 columns. Upon inspecting for missing values, it is observed that the column 'Radiation' has one missing value, while all other columns, including 'Station', 'Date', 'MaxT', 'MinT', 'RH1', 'RH2', 'Wind', 'Rain', 'SSH', 'Evap', 'FAO56_ET', 'Lat', 'Lon', and 'Cum_Rain', contain no missing values. This indicates that the dataset is largely complete, with only a single instance requiring attention for the 'Radiation' column.

```
### Checking column with missing values
df[df.isnull().any(axis=1)]
```

```
       Station       Date  MaxT  MinT  RH1   RH2  Wind  Rain  SSH
Evap  \
14410  ICRISAT 2017-06-15  30.0  24.0   86  74.0   5.8   0.0  0.0
1.7

       Radiation  FAO56_ET        Lat      Lon  Cum_Rain
14410        NaN       2.5  17.508409  78.2723     109.8
```

Filling the missing values using simple imputer

```
numeric_cols = df.select_dtypes(include=[int, float]).columns

# Apply SimpleImputer only to numeric columns
imputer = SimpleImputer(strategy='mean')
df[numeric_cols] = imputer.fit_transform(df[numeric_cols])
df.isnull().sum()
```

```
Station      0
Date         0
MaxT         0
MinT         0
RH1          0
RH2          0
Wind         0
Rain         0
SSH          0
Evap         0
Radiation    0
FAO56_ET     0
Lat          0
Lon          0
Cum_Rain     0
dtype: int64
```

To handle the missing value in the 'Radiation' column, a targeted imputation strategy was employed. All numeric columns in the dataset were identified. The SimpleImputer from the sklearn.impute module was then utilized with the strategy set to 'mean', ensuring that any missing values in the numeric columns would be replaced with the mean value of the respective columns. This approach calculated the mean of each numeric column and replaced any missing values with these means. Finally, verification with df.isnull().sum() confirmed that all missing values were addressed, resulting in a complete and ready-to-use dataset.

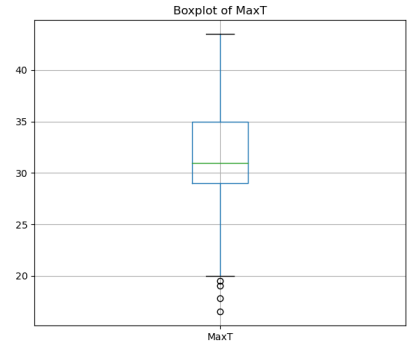
```
df[df.isnull().any(axis=1)]
```

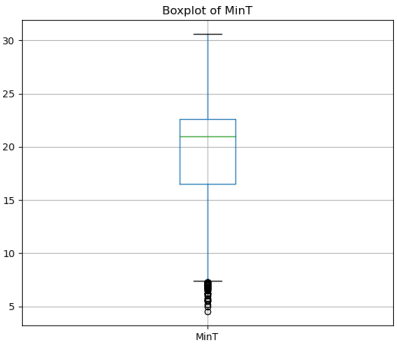
```
Empty DataFrame
Columns: [Station, Date, MaxT, MinT, RH1, RH2, Wind, Rain, SSH, Evap,
Radiation, FAO56_ET, Lat, Lon, Cum_Rain]
Index: []
```

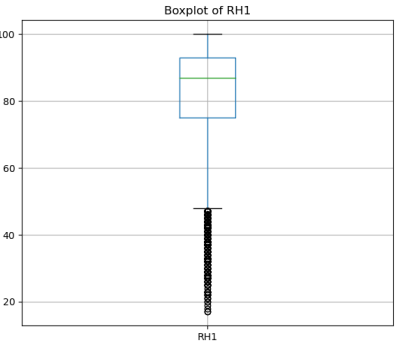
Checking for Outliers

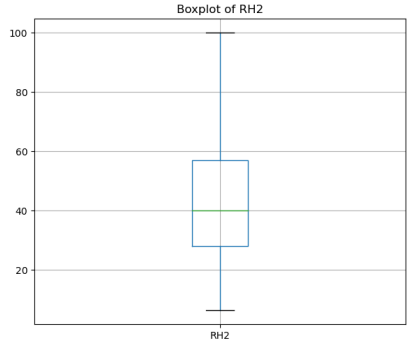
```
# Filter numeric columns
numeric_df = df.select_dtypes(include=[int, float])

# Determine the number of rows and columns for the subplots
nrows = (len(numeric_df.columns) + 2) // 3
ncols = 3

# Adjust the figure size
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(18, 5 *
nrows))

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Create boxplots for each numeric column
```

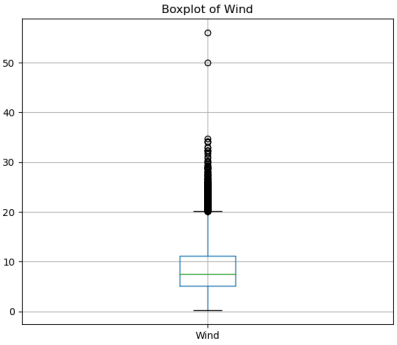
```
for i, col in enumerate(numeric_df.columns):
    numeric_df.boxplot(column=col, ax=axes[i])
    axes[i].set_title(f'Boxplot of {col}')

# Hide any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```

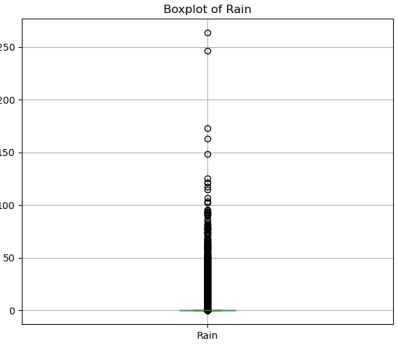

Boxplot of MaxT
MaxT

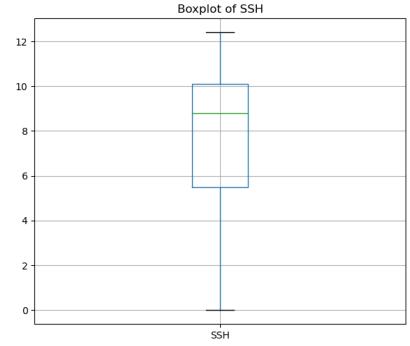

Boxplot of MinT
MinT

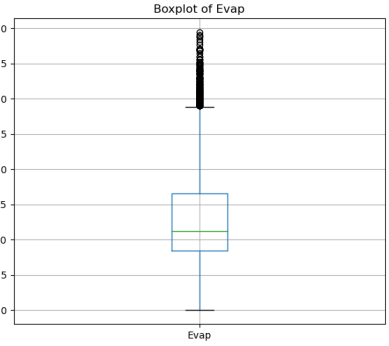

Boxplot of RH1
RH1

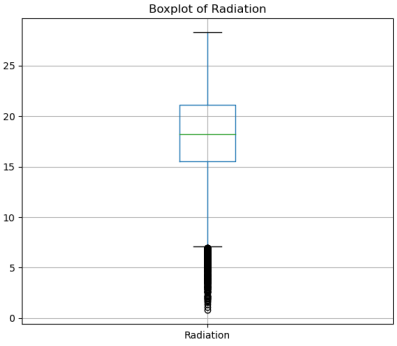

Boxplot of RH2
RH2

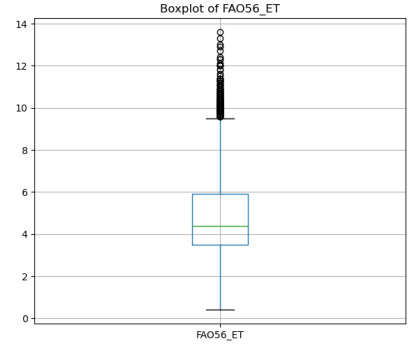

Boxplot of Wind
Wind

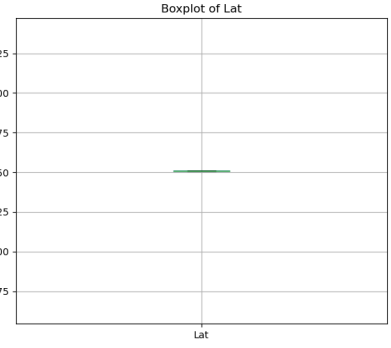

Boxplot of Rain
Rain

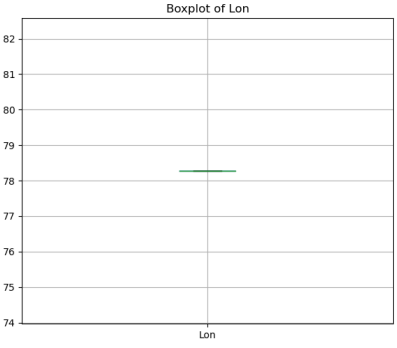

Boxplot of SSH
SSH

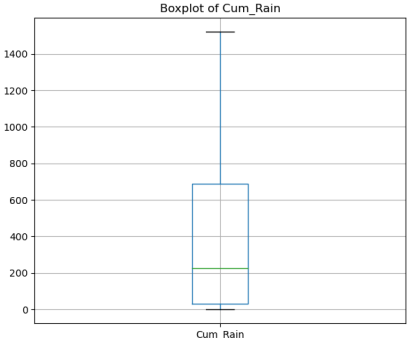

Boxplot of Evap
Evap


Boxplot of Radiation
Radiation


Boxplot of FAO56_ET
FAO56_ET


Boxplot of Lat
Lat


Boxplot of Lon
Lon


Boxplot of Cum_Rain
Cum_Rain

```
#Viewing Outliers
rain_above_150_cols = df.filter(like='Rain').loc[df['Rain'] > 150]  #
Filter 'Rain' like columns and select rows with Rain > 150

# Filter rows with Rain above 150
high_rain_rows = df[df['Rain'] > 150]

# Print entire rows
print(high_rain_rows.to_string())
```

```
       Station       Date  MaxT  MinT   RH1   RH2  Wind   Rain  SSH
Evap  Radiation  FAO56_ET        Lat      Lon  Cum_Rain
226    ICRISAT 1978-08-15  22.2  21.2  91.0  92.0  15.5  172.8  0.0
0.1        4.0       1.2  17.508409  78.2723     874.2
8270   ICRISAT 2000-08-23  26.8  19.4  98.0  84.0  15.0  263.6  0.9
1.0        8.3       2.1  17.508409  78.2723    1063.4
8271   ICRISAT 2000-08-24  23.4  18.5  98.0  98.0  14.6  246.2  0.0
0.8        2.0       0.4  17.508409  78.2723    1309.6
12587  ICRISAT 2012-06-18  31.6  20.5  91.0  53.0  11.7  163.0  0.4
3.5        8.7       3.7  17.508409  78.2723     193.8
```

The volume of rain as shown in the boxplot is not an outlier. Hyderabad experiences the semi arid tropical climatic conditions. The average annual rainfall is 810 mm. The south west monsoon contributes 74% of annual rainfall and north east monsoon contributes 14%. The temperatures reaches 45° C during the summer season and with the onset of monsoons during June the temperature drop and varies between 26° C to 38° C.

https://www.cgwb.gov.in/old_website/District_Profile/Telangana/Hyderabad.pdf

```
# Filter rows with Rain above 150
high_Wind_rows = df[df['Wind'] > 50]

# Print entire rows
print(high_Wind_rows.to_string())
```

```
       Station       Date  MaxT  MinT   RH1   RH2  Wind  Rain  SSH
Evap  Radiation  FAO56_ET        Lat      Lon  Cum_Rain
13917  ICRISAT 2016-02-08  33.4  16.2  85.0  28.0  56.0   0.0  8.3
6.0       16.8      11.0  17.508409  78.2723       0.2
```

Checking for duplicates

```
df.duplicated().sum()
```

```
0
```

There were no duplicates in the datasets.

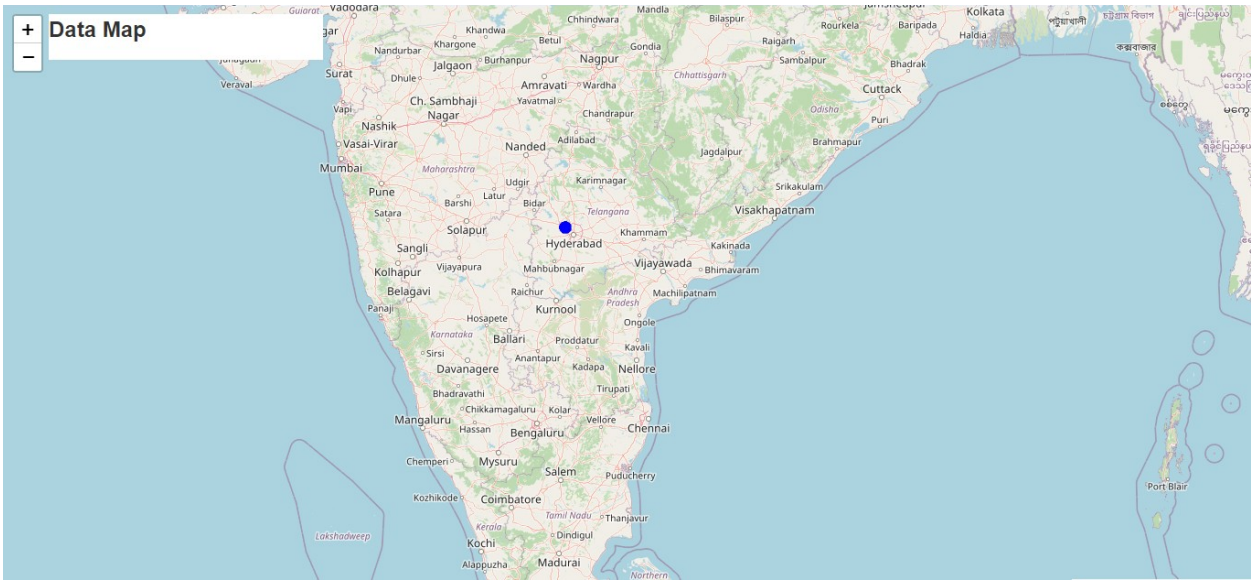
Exploratory Data Analysis (EDA)

```
df.columns
```

```
Index(['Station', 'Date', 'MaxT', 'MinT', 'RH1', 'RH2', 'Wind',
'Rain', 'SSH',
       'Evap', 'Radiation', 'FAO56_ET', 'Lat', 'Lon', 'Cum_Rain'],
      dtype='object')
```

Data Point

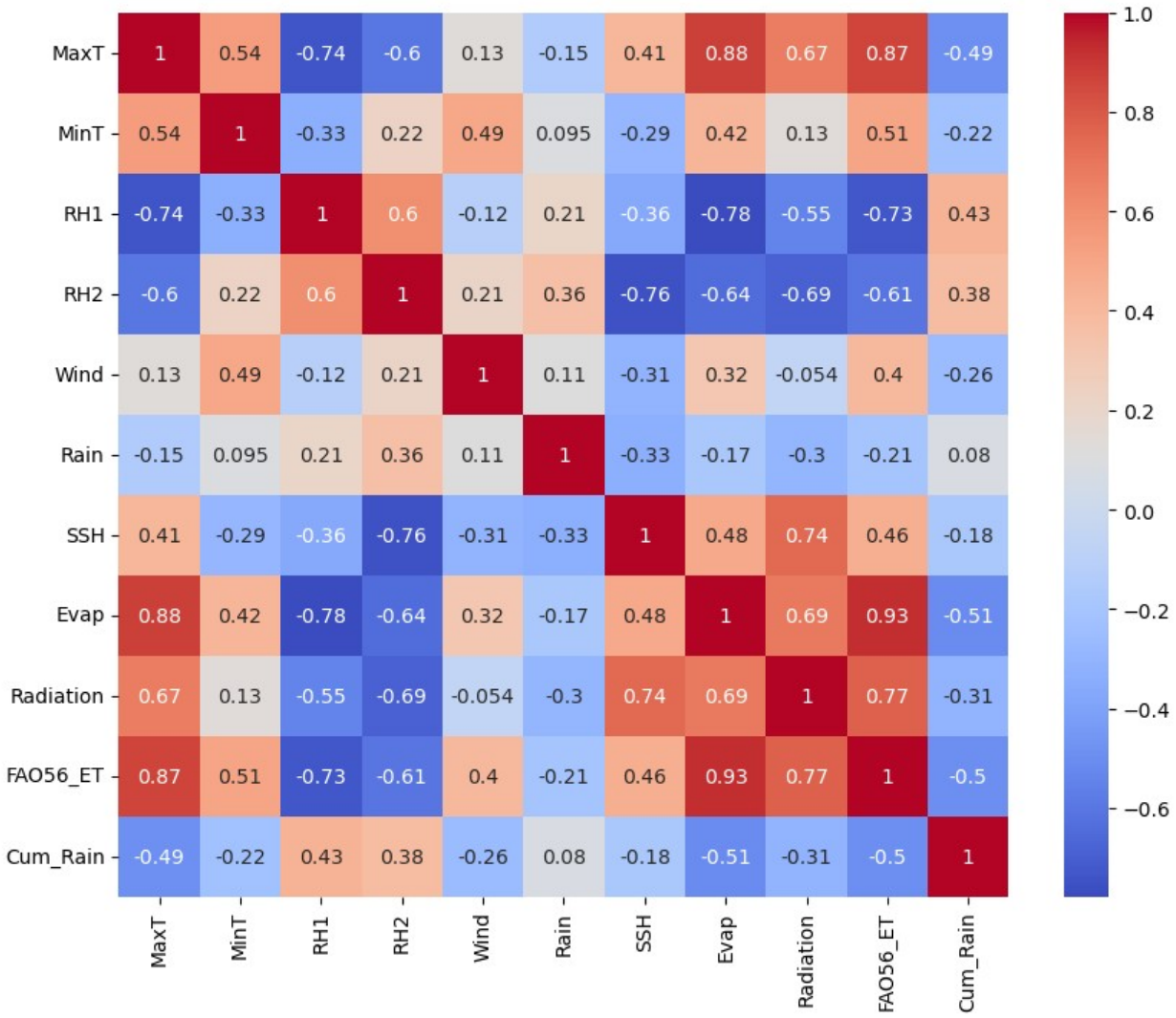
```
m = folium.Map(location=[df['Lat'].mean(), df['Lon'].mean()],
zoom_start=6)

# Adding points to the map
for idx, row in df.iterrows():
    folium.CircleMarker(
        location=[row['Lat'], row['Lon']],
        radius=5,
        popup=f"Rain: {row['Rain']} mm",
        color='blue',
        fill=True,
        fill_color='blue'
    ).add_to(m)

# Add a title to the map
title_html = '''
             <div style="position: fixed;
             top: 10px; left: 50px; width: 300px; height: 50px;
             background-color: white; z-index:9999; font-size:24px;">
             <b>Data Map</b>
             </div>
             '''
m.get_root().html.add_child(folium.Element(title_html))

# Saving the map to an HTML file
m.save('data_map.html')

# Display the map in a Jupyter notebook (if you're using Jupyter)
m
```



The dataset originates from Hyderabad, India. Notably, all the longitudes and latitudes are identical (17.508409, 78.2723), which suggests the weather data was collected from a single location or observation point within Hyderabad. This is an important factor to consider when analyzing the data, as it might not represent weather conditions across the entire city.

Correlation heatmap

```
corr_matrix = df[['MaxT', 'MinT', 'RH1', 'RH2', 'Wind', 'Rain', 'SSH',
'Evap', 'Radiation', 'FAO56_ET', 'Cum_Rain']].corr()
# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```



The figures represent a correlation matrix for several weather-related variables. Each value shows the Pearson correlation coefficient between pairs of variables. A value of 1.0 indicates a perfect positive correlation, -1.0 indicates a perfect negative correlation, and 0 indicates no correlation. For instance, MaxT (maximum temperature) has a strong positive correlation with Evap (evaporation) at 0.876, meaning higher temperatures are associated with higher evaporation rates. Conversely, MaxT has a strong negative correlation with RH1 (relative humidity in the morning) at -0.737, indicating that higher temperatures tend to occur with lower morning humidity. MinT (minimum temperature) shows moderate correlations with several variables, like MaxT at 0.537 and FAO56_ET (reference crop evapotranspiration) at 0.505. Rainfall (Rain) has weaker correlations with most variables, with the highest being a moderate positive correlation with RH2 (relative humidity in the afternoon) at 0.356. SSH (bright sunshine hours) negatively correlates with both relative humidity measures and positively with variables like Radiation at 0.744, suggesting that more sunshine is linked with higher radiation levels but lower humidity. Overall, the matrix reveals how different climatic variables are interrelated, highlighting both direct and inverse relationships.

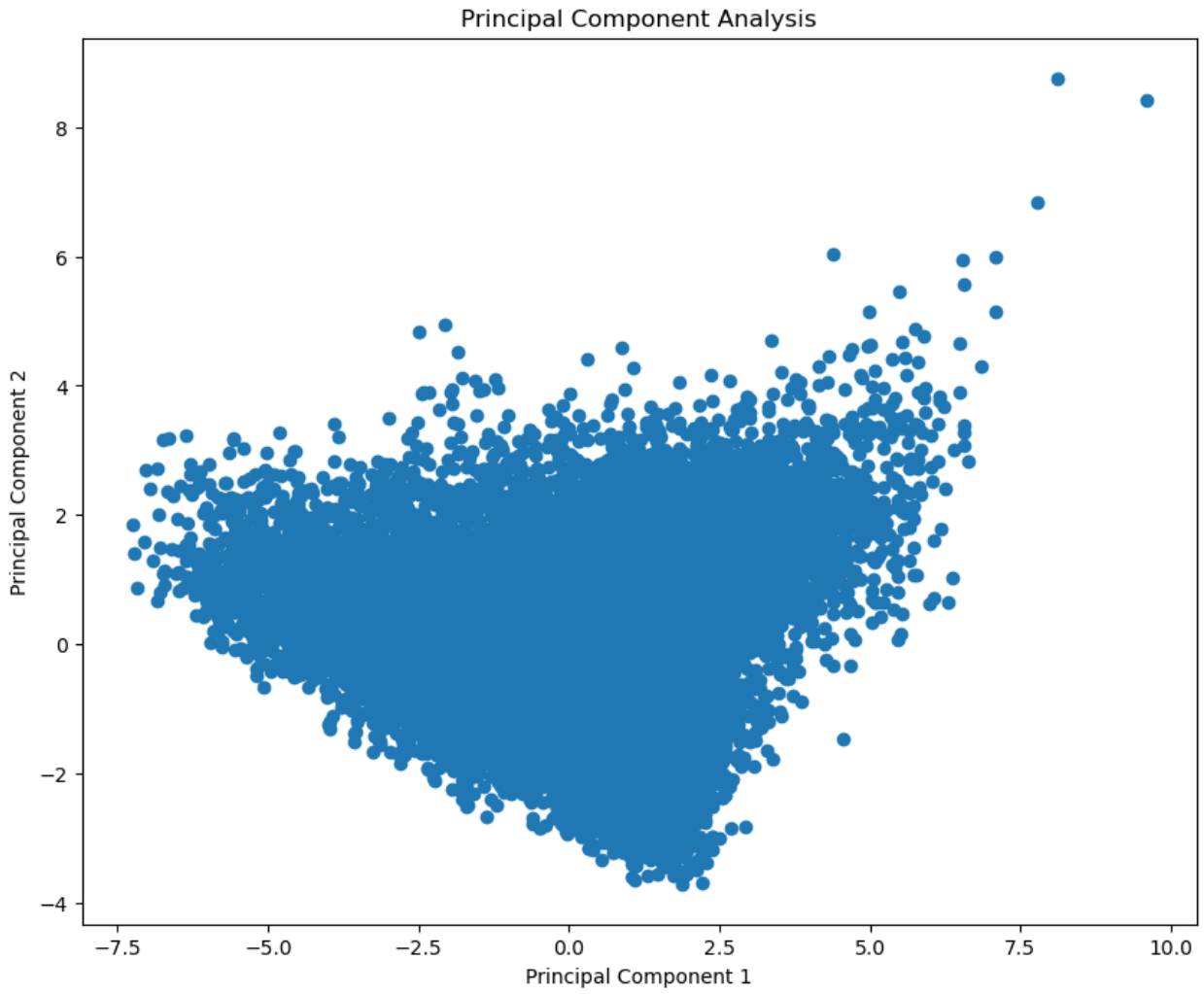
Identifying any weather patterns in Hyderabad - Kmeans clustering

```
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df[['MaxT', 'MinT', 'RH1', 'RH2',
'Wind', 'Rain', 'SSH', 'Evap', 'Radiation', 'FAO56_ET', 'Cum_Rain']])

columns=('MaxT', 'MinT', 'RH1', 'RH2', 'Wind', 'Rain', 'SSH', 'Evap',
'Radiation', 'FAO56_ET', 'Cum_Rain',)
scaled_df=pd.DataFrame(scaled_df, columns=columns)
scaled_df.head()
```

```
       MaxT      MinT       RH1       RH2      Wind      Rain
SSH  \
0 -0.864103 -1.191781 -0.902477 -0.639910 -0.624002 -0.263261
0.790886
1 -0.791199 -0.792206 -0.171806 -0.537936 -0.478026 -0.263261
0.701113
2 -0.742597 -1.125185  0.293166 -0.333990 -0.686564 -0.263261
0.491642
3 -0.742597 -0.348234  0.492440 -0.028071 -0.332050 -0.263261
0.461718
4 -1.034211 -0.570220 -0.038957  0.175876  0.376978 -0.263261
0.431793

       Evap  Radiation  FAO56_ET  Cum_Rain
0 -0.676848   0.116015 -0.500238 -0.968217
1 -0.517217  -0.216741 -0.500238 -0.968217
2 -0.581070  -0.571681 -0.775427 -0.968217
3 -0.708774  -0.327660 -0.555276 -0.968217
4 -0.676848  -0.438578 -0.390163 -0.968217
```

Principal components

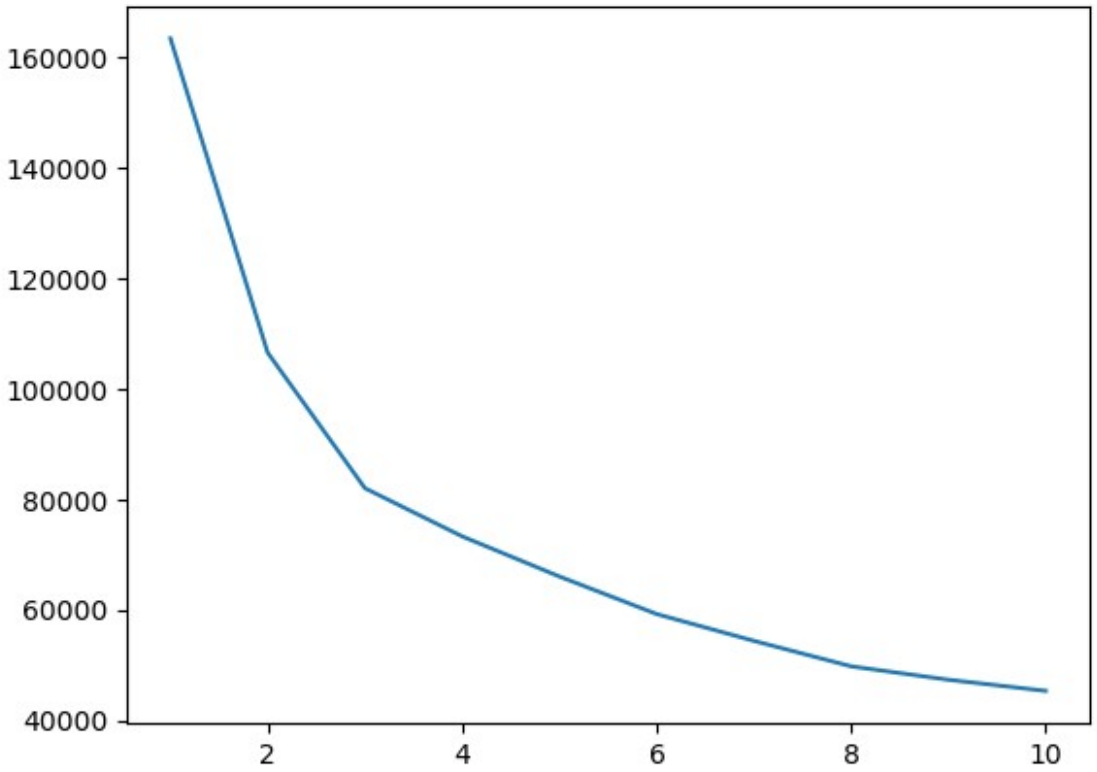
```
pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_df)

# Create a DataFrame with the principal components
pca_df = pd.DataFrame(data=principal_components, columns=['PC1',
'PC2'])

# Plot the principal components
plt.figure(figsize=(10, 8))
plt.scatter(pca_df['PC1'], pca_df['PC2'])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Principal Component Analysis')
plt.show()
```



We used Principal Component Analysis (PCA) before fitting the components on KMeans to reduce the dimensionality of the dataset while retaining most of the original variance. By transforming the data into two principal components, PCA simplifies the dataset, making it easier to visualize and interpret in a 2D space. This step helps in removing noise and irrelevant features, thereby improving the clustering performance and efficiency of KMeans. Additionally, it ensures that the clustering algorithm focuses on the most significant patterns and variations within the data, leading to more meaningful and distinct clusters. This approach also aids in overcoming the curse of dimensionality, which can negatively impact the performance of clustering algorithms in high-dimensional spaces.

```
inertia_scores3=[]
for i in range (1,11):
    kmeans=KMeans(n_clusters=i)
    kmeans.fit(scaled_df)
    inertia_scores3.append(kmeans.inertia_)
plt.plot(range(1,11), inertia_scores3)
```

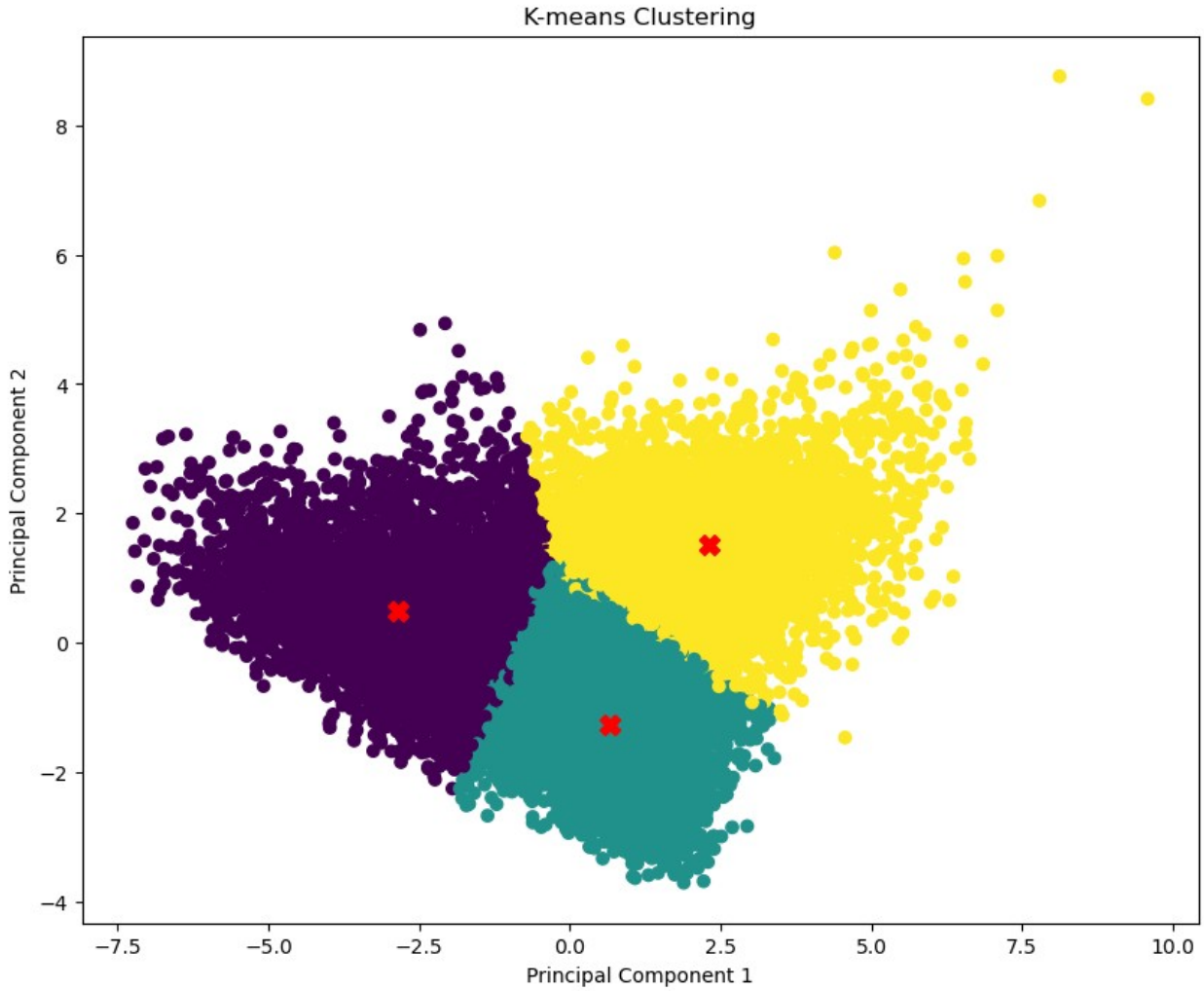
```
[<matplotlib.lines.Line2D at 0x2e80f3c4e90>]
```



We used the Elbow method in KMeans to determine the optimal number of clusters by evaluating the within-cluster sum of squares (WCSS) for different cluster counts. The Elbow method involves plotting the WCSS against the number of clusters and identifying the point where the rate of decrease sharply slows, forming an "elbow." This point indicates the number of clusters that balance the trade-off between minimizing WCSS and avoiding overfitting. In your case, the Elbow method suggested that 3 clusters were optimal, as the WCSS significantly decreased up to 3 clusters and then showed a more gradual decline. This choice of 3 clusters ensures a meaningful partitioning of the data, capturing distinct groupings without unnecessary complexity.

```
from sklearn.cluster import KMeans

# Apply K-means clustering
kmeans = KMeans(n_clusters=3)
kmeans.fit(scaled_df)
df['Cluster'] = kmeans.labels_
centroids = kmeans.cluster_centers_
centroids_pca = pca.transform(centroids)


# Visualize the clusters using the principal components
plt.figure(figsize=(10, 8))
plt.scatter(pca_df['PC1'], pca_df['PC2'], c=df['Cluster'],
cmap='viridis')
plt.scatter(x=centroids_pca[:, 0], y=centroids_pca[:, 1], s=100,
```

```
marker="X", c="red")
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('K-means Clustering')
plt.show()
```



The output is a scatter plot showing the results of KMeans clustering applied to a dataset that has been reduced to two principal components using PCA. Each point represents a data sample, plotted according to its values on the first two principal components. The colors indicate the clusters assigned by the KMeans algorithm, with three distinct clusters visible (yellow, purple, and teal). The red 'X' markers represent the centroids of each cluster, which are the central points calculated by the KMeans algorithm. These centroids are used to assign each data point to a cluster based on the shortest Euclidean distance to the centroid. This visualization helps to understand the separation and distribution of the clusters in a two-dimensional space. The clear separation between the colors indicates that the KMeans algorithm has effectively partitioned the data into three clusters.

```
df.columns
```

```
Index(['Station', 'Date', 'MaxT', 'MinT', 'RH1', 'RH2', 'Wind',
'Rain', 'SSH',
       'Evap', 'Radiation', 'FAO56_ET', 'Lat', 'Lon', 'Cum_Rain',
'Cluster'],
      dtype='object')
```

```
df['Cluster_labelled'] = df['Cluster'].replace({0: "Season1", 1:
"Season2", 2: "Season3"})
df.sample(10)
```

```
       Station       Date  MaxT  MinT   RH1   RH2  Wind  Rain   SSH
Evap  \
3527   ICRISAT 1987-08-29  28.0  22.5  88.0  68.0  14.1   0.0   1.1
4.8
6578   ICRISAT 1996-01-05  27.5  13.5  75.0  39.0   8.8   0.0  10.4
4.8
6338   ICRISAT 1995-05-10  33.0  22.5  68.0  43.0  11.7   0.0   5.5
7.4
11673  ICRISAT 2009-12-17  27.3  17.4  85.0  46.0   7.6   0.0   5.6
3.8
2225   ICRISAT 1984-02-04  29.5  15.7  84.0  29.0   7.9   0.0  10.3
6.5
10372  ICRISAT 2006-05-26  34.8  22.0  88.0  47.0  11.9  15.0   5.4
4.3
7013   ICRISAT 1997-03-15  37.0  18.5  69.0  21.0   6.3   0.0   9.4
9.0
13373  ICRISAT 2014-08-13  32.2  23.0  79.0  52.0   9.5   0.0   6.6
6.1
2837   ICRISAT 1985-10-08  29.4  20.2  91.0  59.0   5.0   0.0  10.5
3.7
8933   ICRISAT 2002-06-17  35.2  22.8  78.0  47.0  18.3   0.0  10.8
13.2

       Radiation  FAO56_ET        Lat      Lon  Cum_Rain  Cluster  \
3527         9.0       3.1  17.508409  78.2723     425.0        2
6578        17.7       4.2  17.508409  78.2723       0.0        1
6338        16.5       5.7  17.508409  78.2723     104.2        0
11673       14.6       3.5  17.508409  78.2723     990.2        1
2225        18.8       4.6  17.508409  78.2723       4.8        1
10372       14.9       5.2  17.508409  78.2723     195.0        2
7013        21.9       6.0  17.508409  78.2723      11.4        0
13373       16.2       4.8  17.508409  78.2723     253.7        1
2837        20.1       4.1  17.508409  78.2723     537.0        1
8933        23.6       7.5  17.508409  78.2723      80.0        0

      Cluster_labelled
3527           Season3
6578           Season2
6338           Season1
11673          Season2
```

```
2225            Season2
10372           Season3
7013            Season1
13373           Season2
2837            Season2
8933            Season1
```

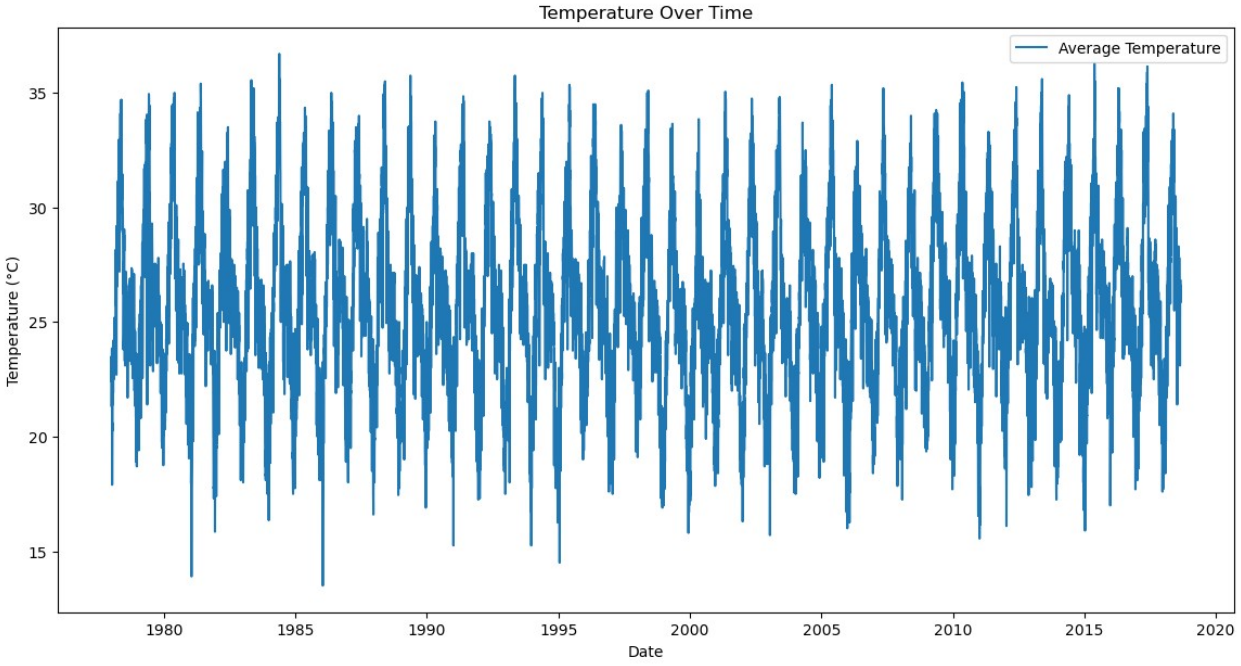
For the sake of EDA, we created an extra column renaming the clusters from Season 1 to Season 3 respectively as per the KMeans findings

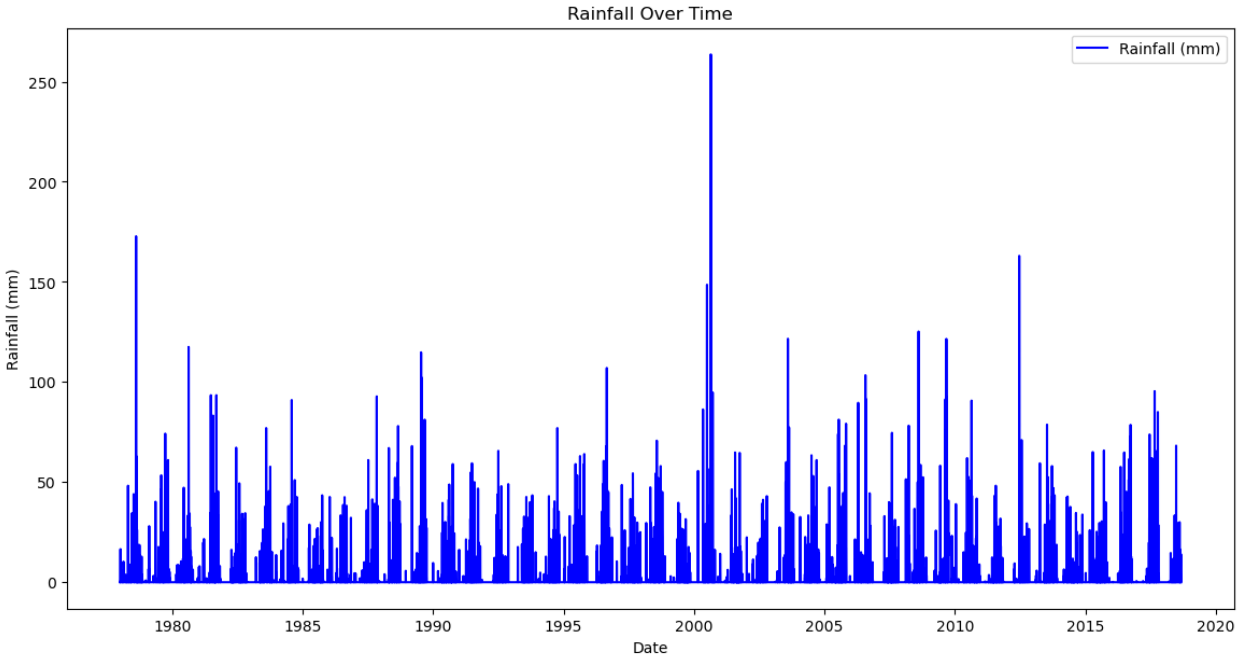
Feature Engineering

```
# Computing a new column for the average temp
df['AvgT'] = (df['MaxT'] + df['MinT']) / 2
```

Time series analysis

```
df['Date'] = pd.to_datetime(df['Date'])

# Plot MaxT and MinT over time
plt.figure(figsize=(14, 7))
plt.plot(df['Date'], df['AvgT'], label='Average Temperature')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.title('Temperature Over Time')
plt.legend()
plt.show()

# Plot Rain over time
plt.figure(figsize=(14, 7))
plt.plot(df['Date'], df['Rain'], label='Rainfall (mm)', color='b')
plt.xlabel('Date')
plt.ylabel('Rainfall (mm)')
plt.title('Rainfall Over Time')
plt.legend()
plt.show()
```





The plots shows the variation in average temperatures from 1980 to 2020. The plot highlights a clear annual cycle temperatures, with temperatures peaking around mid-year and reaching their lowest towards the end of the year. Over the years, there appears to be a slight upward trend in both maximum and minimum temperatures, indicating a possible warming trend over the four-decade period.

The bottom plot displays the rainfall data (in millimeters) over the same time period. The plot shows a highly variable pattern with occasional spikes indicating periods of heavy rainfall. Notably, there is a significant spike around the year 2000, suggesting an exceptionally high

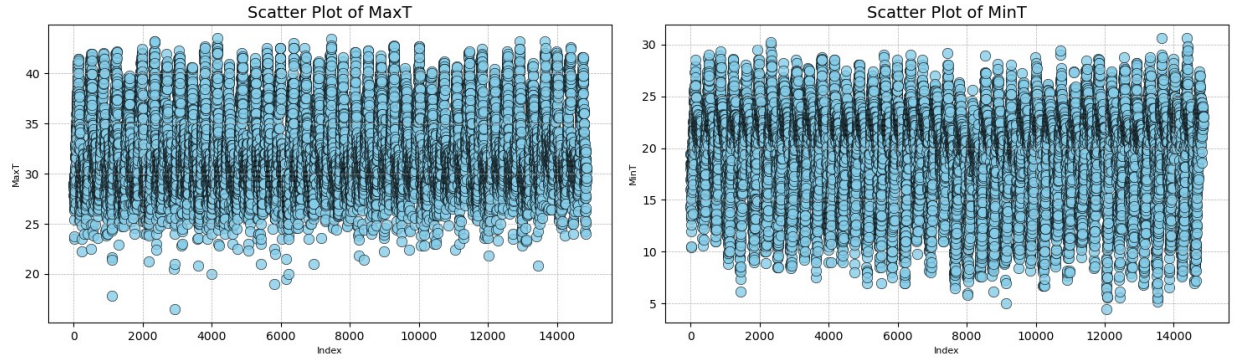
rainfall event. Hyderabad reported significant rainfall in 2000. News reports and historical accounts mention heavy downpours, particularly in August 2000. Begumpet, a specific area within Hyderabad, saw a staggering 241.5 mm (almost 24 cm) of rain in just one day, contributing to widespread flooding. This intense downpour suggests Hyderabad likely received above-average rainfall throughout 2000. Overall, the rainfall data does not exhibit a clear trend but rather shows sporadic high rainfall events throughout the years.
https://indianexpress.com/article/cities/hyderabad/hyderabad-rains-floods-waterlogging-urbanisation-6798417/

UNIVARIATE ANALYSIS

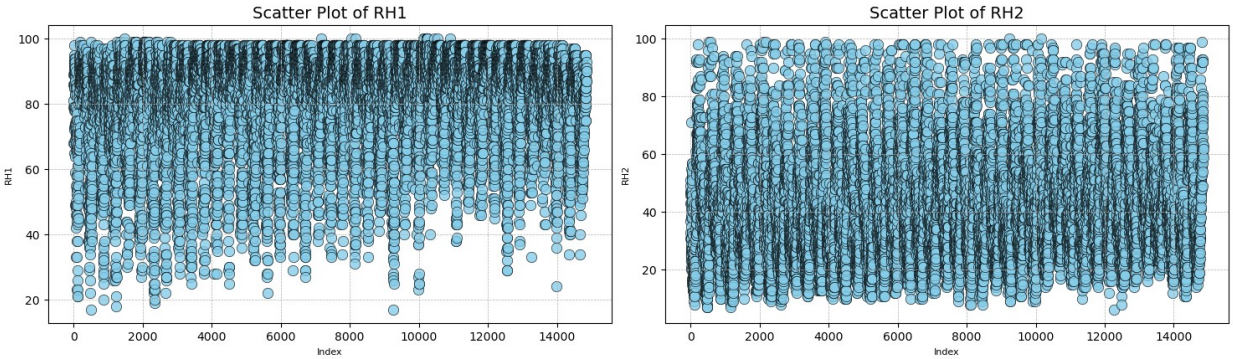
Analysis of each and every variable in our dataset so as to identify the general underlying patterns, findings etc for them

```
# Convert 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# # Group by year
# df['Year'] = df['Date'].dt.year
# grouped = df.groupby('Year').size()
# # Plotting
# plt.figure(figsize=(10, 6))
# plt.plot(grouped.index, grouped.values, marker='o', linestyle='-',
color='b')

columns_to_visualize = ['MaxT', 'MinT', 'RH1', 'RH2', 'Wind', 'Rain',
'SSH', 'Evap', 'Radiation', 'FAO56_ET', 'Cum_Rain','AvgT']

# Set the figure size
plt.figure(figsize=(15, 30))

## Iterating over the columns
for i, column in enumerate(columns_to_visualize, start = 1):
    plt.subplot(len(columns_to_visualize)//2 + 1, 2, i)  # Create
subplots in a 2-column grid
    plt.scatter(df.index, df[column], s=80, c='skyblue', alpha=0.8,
edgecolors='k', linewidth=0.5)

    # Add title and labels
    plt.title(f'Scatter Plot of {column}', fontsize=14)
    plt.xlabel('Index', fontsize=8)
    plt.ylabel(column, fontsize=8)

    # Add gridlines with minor ticks
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)

    # Customize ticks and labels
    plt.xticks(fontsize=10)
    plt.yticks(fontsize=10)
    # Adjust layout for better appearance
```

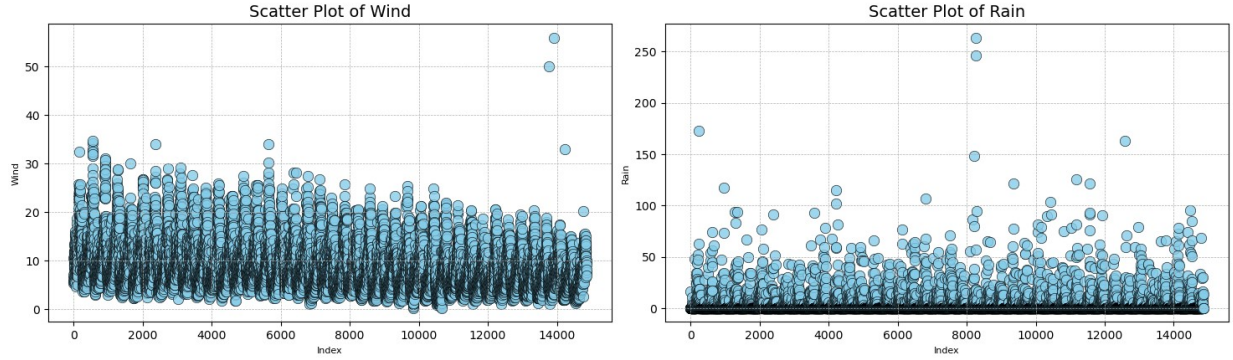
```
    plt.tight_layout()

# Adjust overall layout and show the plot
plt.tight_layout(pad=1.0, w_pad=0.5, h_pad=1.0)
plt.show()
```


Scatter Plot of MaxT
MaxT
Index

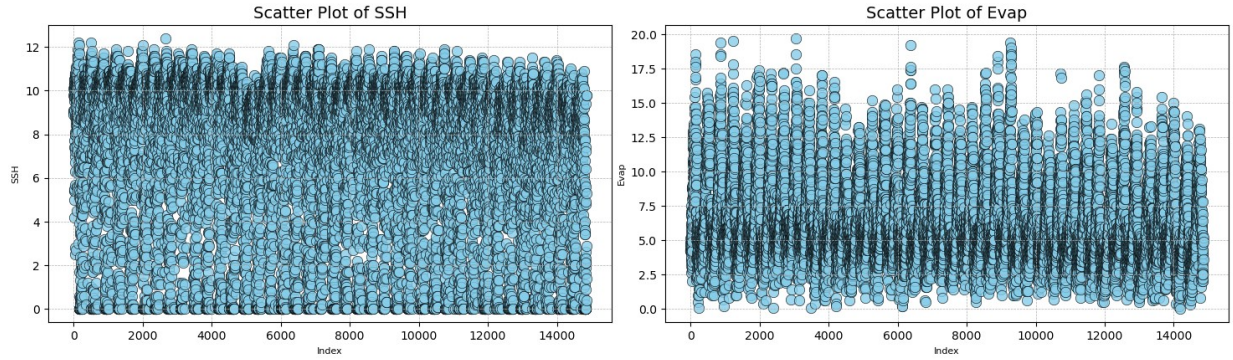
Scatter Plot of MinT
MinT
Index


Scatter Plot of RH1
RH1
Index

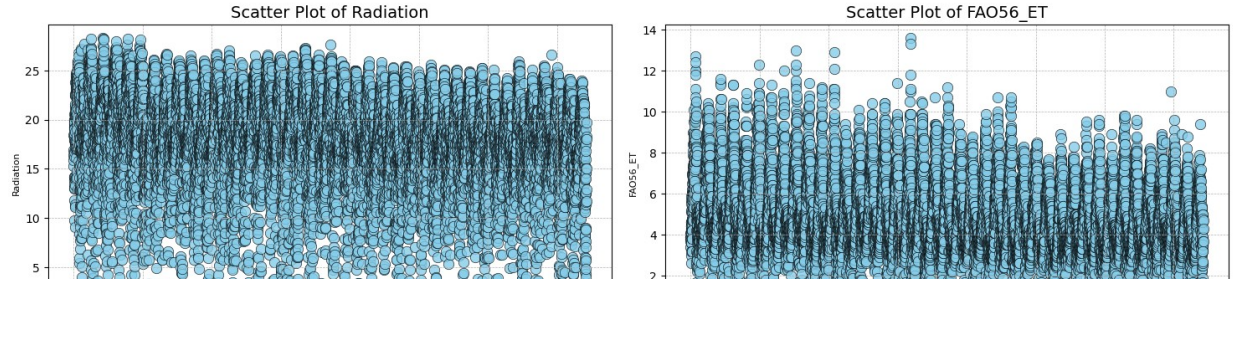
Scatter Plot of RH2
RH2
Index


Scatter Plot of Wind
Wind
Index

Scatter Plot of Rain
Rain
Index


Scatter Plot of SSH
SSH
Index

Scatter Plot of Evap
Evap
Index


Scatter Plot of Radiation
Radiation

Scatter Plot of FAO56_ET
FAO56_ET

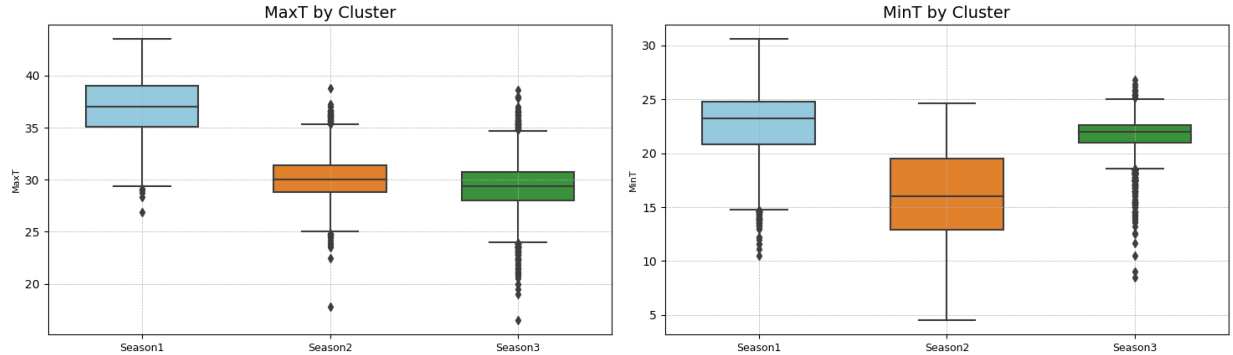
We wanted to visualize the visualize the distribution of each variable from a data set. (*NB: Focusing on the numerical varibales only.*) We chose to use a scatter plot and combined all the plots in a structured manner. From this combination plot;

- We are able to identify outliers easily accross our variables
- We are able to compare several variables at the same time
- We are also able to see the generale range of our dataset variables

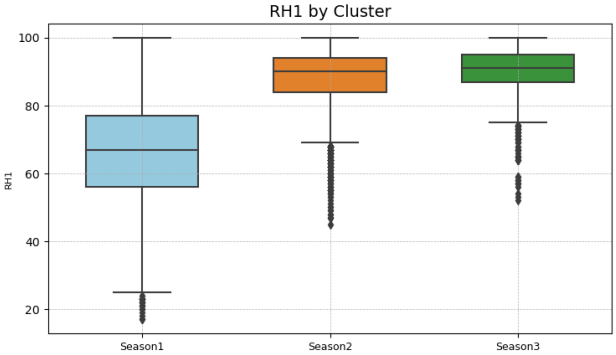
BI-VARIATE ANAYSIS

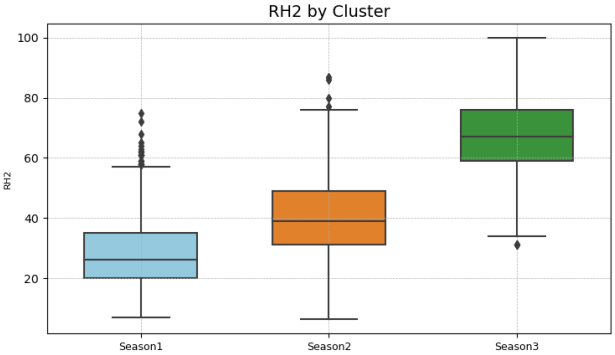
Using our cluster variable, we want to see how it relates to each and every variable in our dataset

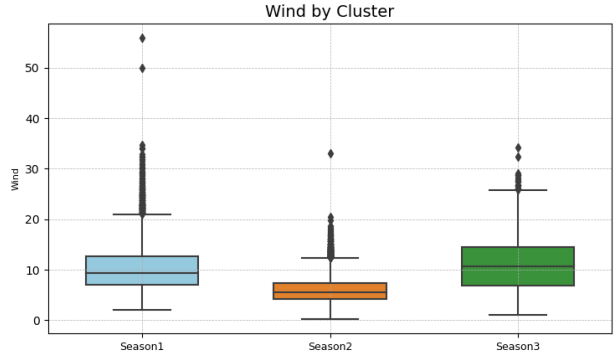
```
# Set the order of seasons
x_lab_order_seasons = ['Season1', 'Season2', 'Season3']
season_colors = ['skyblue', '#ff7f0e', '#2ca02c']  # Skyblue, Orange,
Green

# Set the figure size
plt.figure(figsize=(15, 30))

## Iterating over the columns
for i, column in enumerate(columns_to_visualize, start = 1):
    plt.subplot(len(columns_to_visualize)//2 + 1, 2, i)  # Create
subplots in a 2-column grid
    sns.boxplot(x='Cluster_labelled', y=column, data=df, order =
x_lab_order_seasons, palette=season_colors, linewidth=1.5, width=0.6,
fliersize=5)

    # Add title and labels
    plt.title(f'{column} by Cluster', fontsize=14)
    plt.xlabel(None)
    plt.ylabel(column, fontsize=8)
    plt.yticks(fontsize=10)
    plt.xticks(fontsize=9)

    # Add gridlines with minor ticks
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)

    # Adjust layout for better appearance
    plt.tight_layout()

# Adjust overall layout and show the plot
plt.tight_layout(pad=1.0, w_pad=0.5, h_pad=1.0)
plt.show()
```

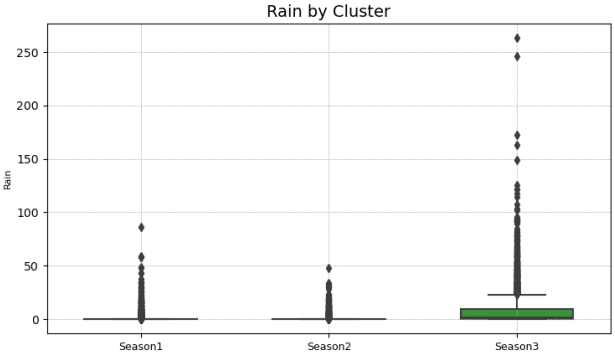

MaxT by Cluster
MaxT
Season1
Season2
Season3

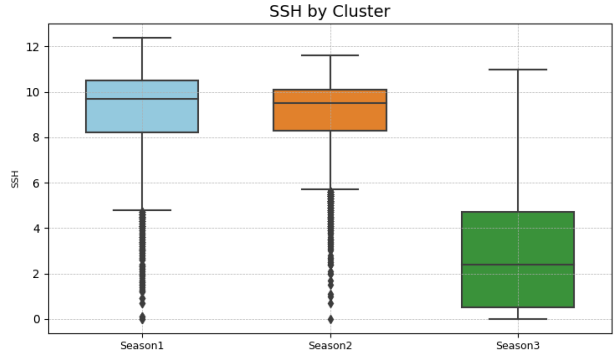
MinT by Cluster
MinT
Season1
Season2
Season3

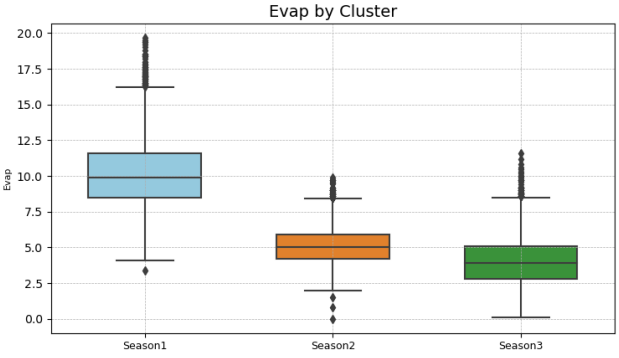

RH1 by Cluster
RH1
Season1
Season2
Season3

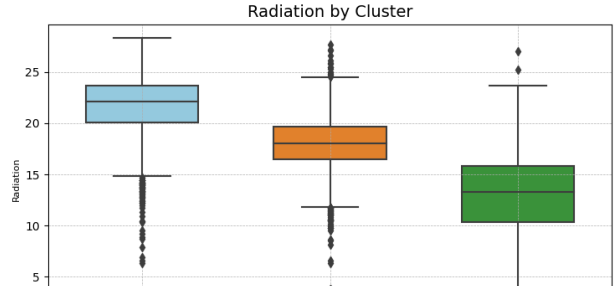

RH2 by Cluster
RH2
Season1
Season2
Season3

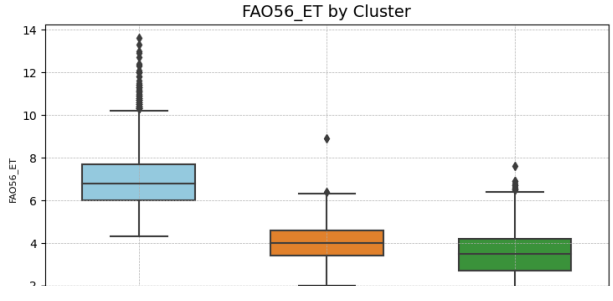

Wind by Cluster
Wind
Season1
Season2
Season3


Rain by Cluster
Rain
Season1
Season2
Season3


SSH by Cluster
SSH
Season1
Season2
Season3


Evap by Cluster
Evap
Season1
Season2
Season3

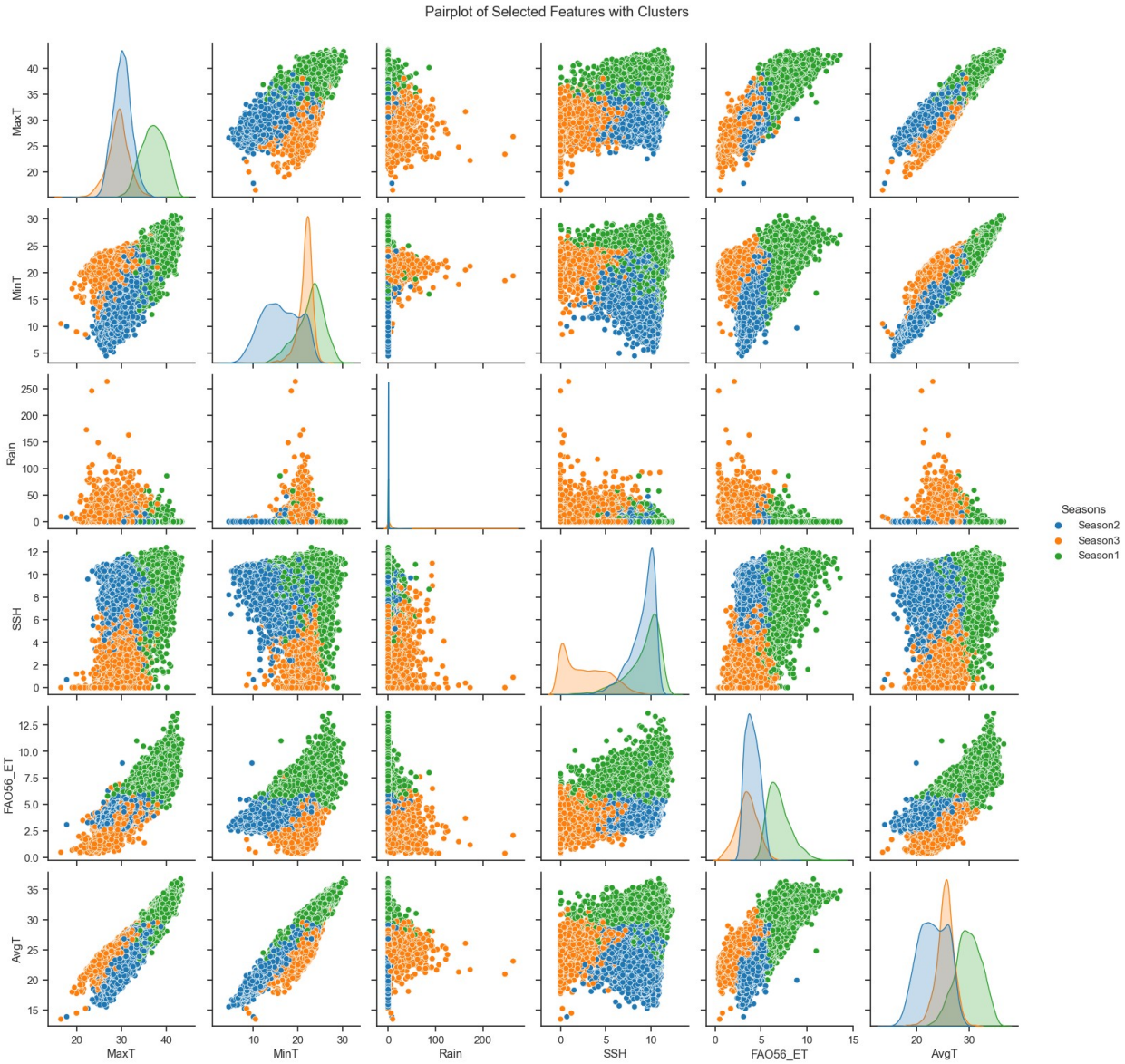

Radiation by Cluster
Radiation


FAO56_ET by Cluster
FAO56_ET

From the above bi-variate analysis, we are able to see the general distribution of our numerical varibales now based on the clusters. For instance;

- Temperature for both MaxT and MinT variables is higher in season 2
- RH1 and RH2 is generally lower in season 2
- Wind-strength is increasing gradually from season 1 to season 3, with some extreme outliers in season 2
- Rain is generally higher in season 3 in which there are some extreme cases
- SSH is generally higher in season 1 and 2 and low in season 3
- Evaporation, Radiation and FAO56_RT is high in season 2
- Cumulatively, there is little rain in season 2 when compared to season 1 and 3

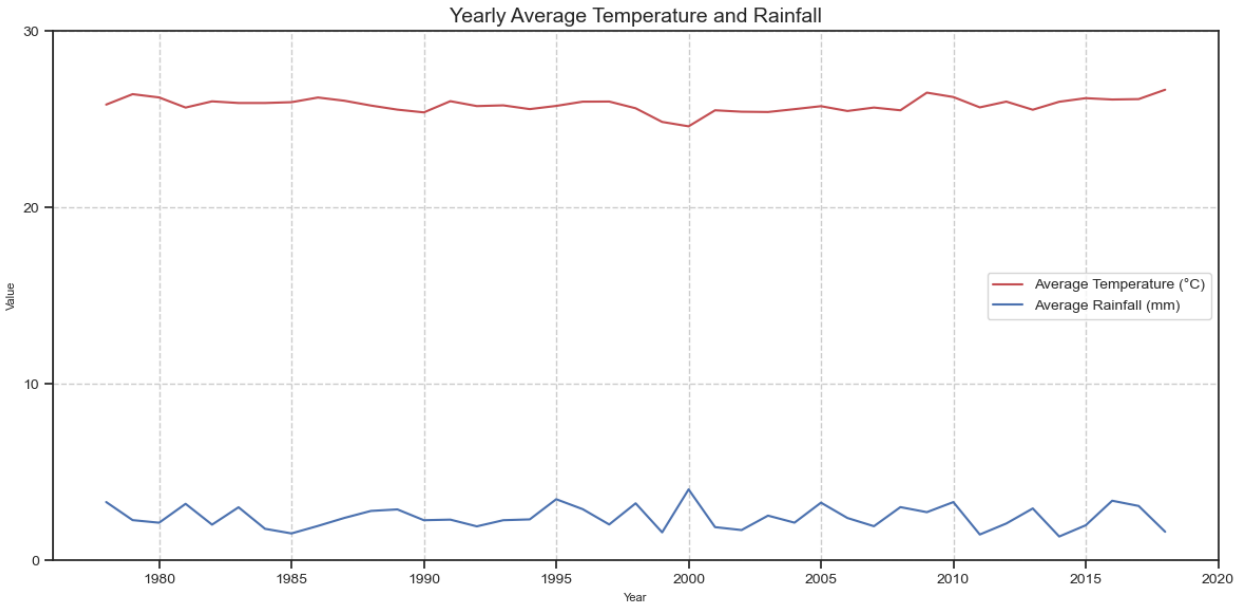
```
# Pair plot with clusters
sns.set(style="ticks")
g = sns.pairplot(df[['MaxT', 'MinT', 'Rain', 'SSH', 'FAO56_ET',
'Cluster_labelled','AvgT']], hue='Cluster_labelled', palette='tab10')
plt.suptitle('Pairplot of Selected Features with Clusters', y=1.02)
# Add legend
g._legend.set_title(title='Seasons')
plt.show()
```



The pair plots offer a comprehensive view of the interactions between key climatic variables and how they vary across different clusters. This analysis helps in understanding the underlying patterns and relationships within the climate data, which can be valuable for further studies and decision-making related to climate and environmental management. The scatter plots illustrate the relationships between each pair of variables, while the density plots on the diagonal show the distribution of individual variables within each cluster. Key observations include a positive correlation between MaxT and FAO56_ET, and between MinT and FAO56_ET. The rain variable shows a more scattered pattern with other variables, indicating less consistent relationships. The clusters show distinct groupings in the scatter plots, suggesting varying climatic conditions captured by each cluster.

```
# # Plot Rain and AvgT over time
# fig, ax1 = plt.subplots(figsize=(14, 7))
```

```
# color = 'tab:blue'
# ax1.set_xlabel('Date')
# ax1.set_ylabel('Average Temperature (°C)', color=color)
# ax1.plot(df['Date'], df['AvgT'], label='Average Temperature',
color=color)
# ax1.tick_params(axis='y', labelcolor=color)

# ax2 = ax1.twinx()

# color = 'tab:red'
# ax2.set_ylabel('Rainfall (mm)', color=color)  # we already handled
the x-label with ax1
# ax2.plot(df['Date'], df['Rain'], label='Rainfall', color=color)
# ax2.tick_params(axis='y', labelcolor=color)

# fig.tight_layout()  # otherwise the right y-label is slightly
clipped
# plt.title('Rainfall and Average Temperature Over Time')
# plt.show();

yearly_avg = df.groupby(df['Date'].dt.year)[['AvgT', 'Rain']].mean()

# Plot yearly averages
plt.figure(figsize=(12, 6))
plt.plot(yearly_avg.index, yearly_avg['AvgT'], label='Average
Temperature (°C)', color='r')
plt.plot(yearly_avg.index, yearly_avg['Rain'], label='Average Rainfall
(mm)', color='b')

plt.xlabel('Year', fontsize=8)
plt.ylabel('Value', fontsize=8)
plt.title('Yearly Average Temperature and Rainfall', fontsize=14)
plt.legend(loc='center right', fontsize=10)

# Add gridlines with minor ticks
plt.grid(True, which='both', linestyle='--', linewidth= 1 )

# Customize ticks and labels
plt.xticks(fontsize=10)
plt.yticks([0, 10, 20, 30], fontsize=10)

plt.tight_layout()
plt.show()
```



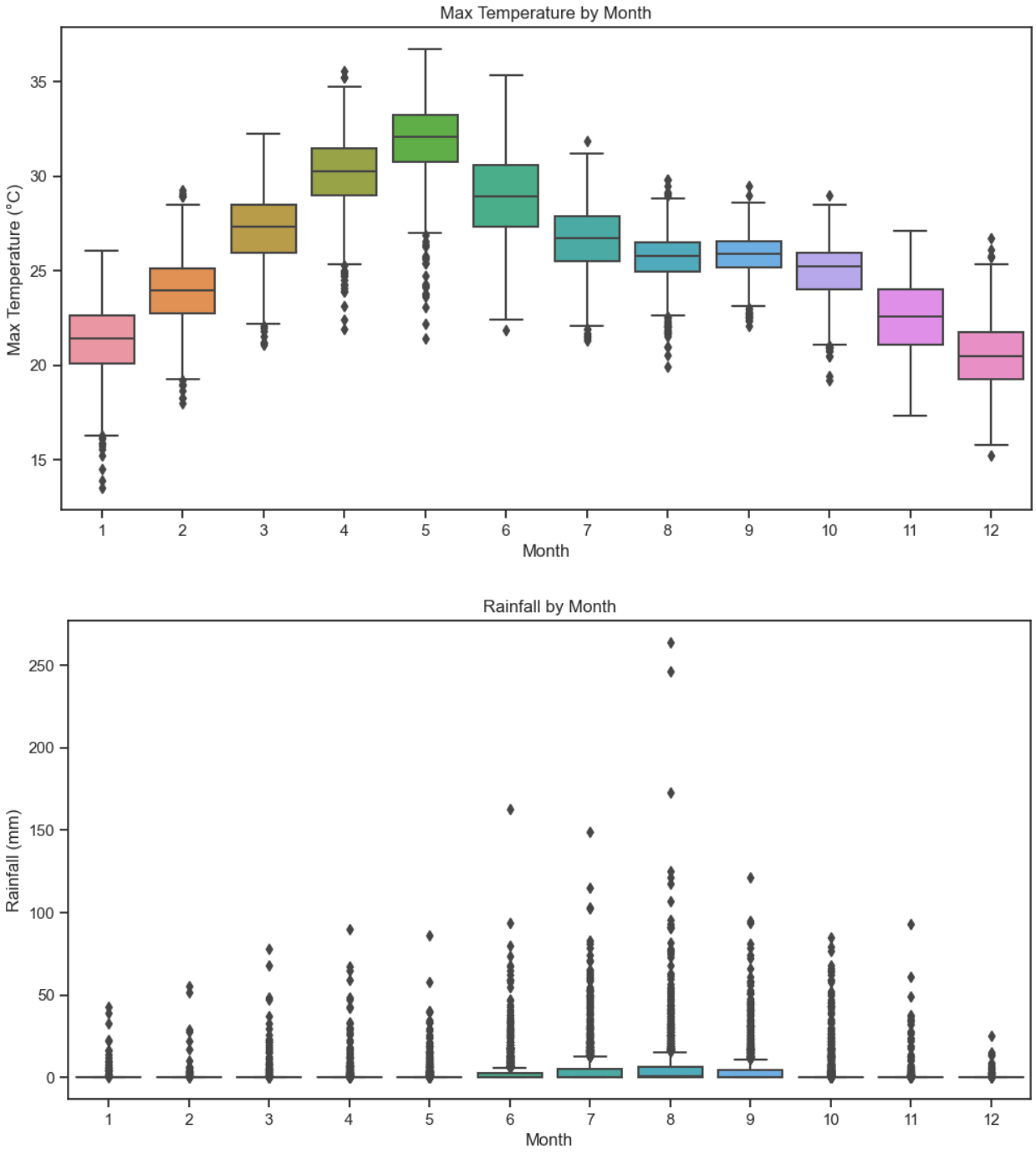
In a weather dataset, temperature and wind are generally the key items being checked. We went ahead and visualized the average rainfal and temperature accoss the years.

- The key year identified to be of interest was the year 2000 where we had both the highest Rainfall(mm) and the lowest average temperature.
- Around 2009, we had the highest average temperature recorded.
- We had the lowest rainfall around 2011

For modeling purposes, we shall use the average temperature and rainfall columns

Seasonality Analysis

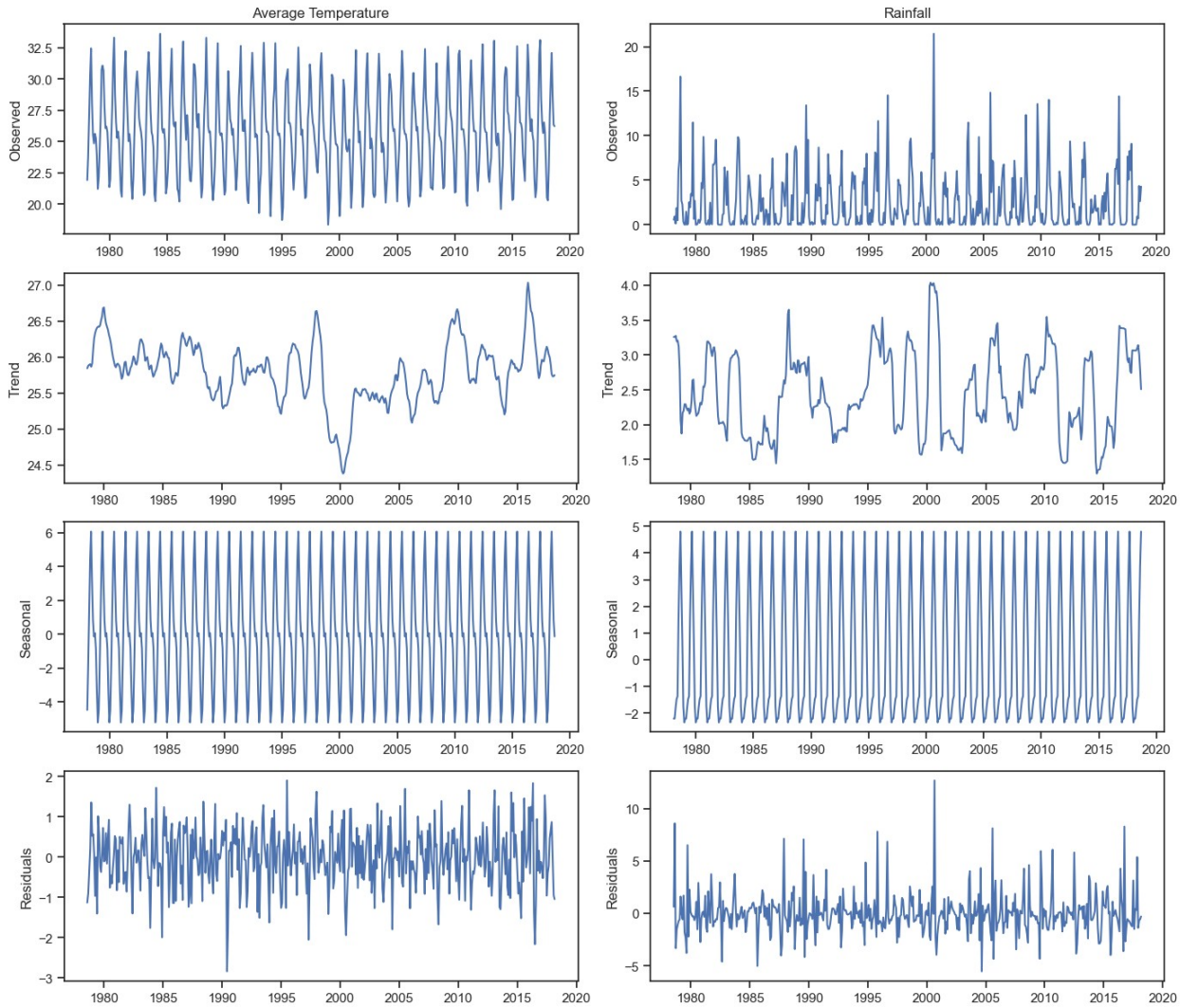
```
# Extract month and year
df['Month'] = df['Date'].dt.month
df['Year'] = df['Date'].dt.year

# Boxplot of MaxT by Month
plt.figure(figsize=(12, 6))
sns.boxplot(x='Month', y='AvgT', data=df)
plt.title('Max Temperature by Month')
plt.xlabel('Month')
plt.ylabel('Max Temperature (°C)')
plt.show()

# Boxplot of Rain by Month
plt.figure(figsize=(12, 6))
sns.boxplot(x='Month', y='Rain', data=df)
plt.title('Rainfall by Month')
plt.xlabel('Month')
plt.ylabel('Rainfall (mm)')
plt.show()
```



The output consists of two box plots illustrating the distribution of maximum temperature and rainfall by month. The top plot shows that maximum temperatures generally increase from January to June, peaking around April to June, and then decrease towards December. The variability in maximum temperatures is higher during the hotter months (April to June). The bottom plot reveals that rainfall peaks significantly in August, with a high number of outliers indicating occasional heavy rain events. Median rainfall is relatively low for most months, with increased variability and more frequent outliers in the middle of the year, particularly in August.

Overall, these plots highlight seasonal patterns where temperatures rise mid-year and rainfall peaks in August, providing valuable insights into weather trends for planning and analysis.

```
# Resample to monthly frequency and calculate monthly averages
monthly_avg = df.resample('M', on='Date')[['AvgT', 'Rain']].mean()

from statsmodels.tsa.seasonal import seasonal_decompose

# Decompose Average Temperature
result_temp = seasonal_decompose(monthly_avg['AvgT'],
model='additive')

# Decompose Rainfall
result_rain = seasonal_decompose(monthly_avg['Rain'],
model='additive')

# Ploting the decomposed components
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(14, 12))

# Average Temperature
axes[0, 0].plot(result_temp.observed)
axes[0, 0].set_ylabel('Observed')
axes[0, 0].set_title('Average Temperature')

axes[1, 0].plot(result_temp.trend)
axes[1, 0].set_ylabel('Trend')

axes[2, 0].plot(result_temp.seasonal)
axes[2, 0].set_ylabel('Seasonal')

axes[3, 0].plot(result_temp.resid)
axes[3, 0].set_ylabel('Residuals')

# Rainfall
axes[0, 1].plot(result_rain.observed)
axes[0, 1].set_ylabel('Observed')
axes[0, 1].set_title('Rainfall')

axes[1, 1].plot(result_rain.trend)
axes[1, 1].set_ylabel('Trend')

axes[2, 1].plot(result_rain.seasonal)
axes[2, 1].set_ylabel('Seasonal')

axes[3, 1].plot(result_rain.resid)
axes[3, 1].set_ylabel('Residuals')

plt.tight_layout()
plt.show()
```




```
df["AvgT"]

0        21.35
1        22.40
2        21.75
3        23.50
4        22.40
         ...
14848    25.85
14849    25.90
14850    26.00
14851    26.80
14852    26.80
Name: AvgT, Length: 14853, dtype: float64
```

The team observed that both Rainfall and Average Temperature exhibit seasonality, a crucial consideration as we plan to utilize the SARIMA model.

Checking for the Stationary of our dataset

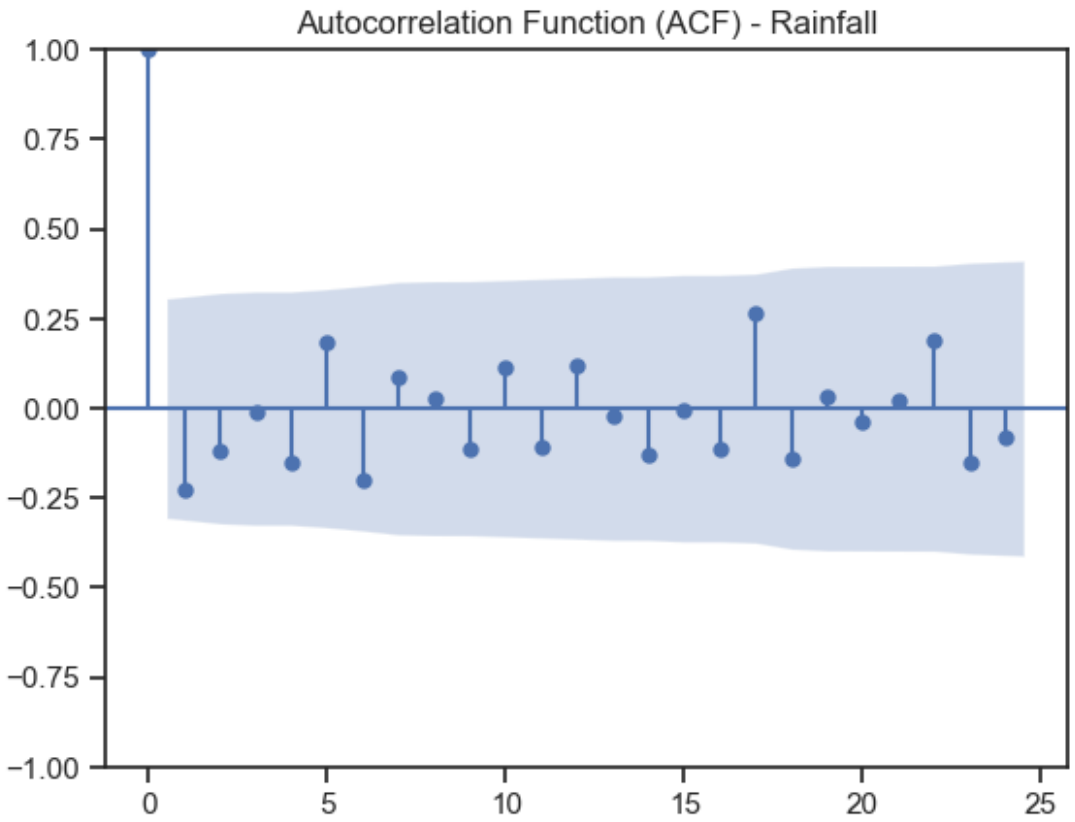
```
from statsmodels.tsa.stattools import adfuller

# Perform ADF test for rainfall (Yearly)
result_rain = adfuller(yearly_avg['Rain'])
print('ADF Statistic for Rainfall:', result_rain[0])
print('p-value for Rainfall:', result_rain[1])
print('Critical Values for Rainfall:', result_rain[4])

if result_rain[1] <= 0.05:
    print("Reject the null hypothesis. The time series is
stationary.")
else:
    print("Fail to reject the null hypothesis. The time series is non-
stationary.")

# Perform ADF test for average temperature (Yearly)
result_temp = adfuller(yearly_avg['AvgT'])
print('\nADF Statistic for Average Temperature:', result_temp[0])
print('p-value for Average Temperature:', result_temp[1])
print('Critical Values for Average Temperature:', result_temp[4])

if result_temp[1] <= 0.05:
    print("Reject the null hypothesis. The time series is
stationary.")
else:
    print("Fail to reject the null hypothesis. The time series is non-
stationary.")
```

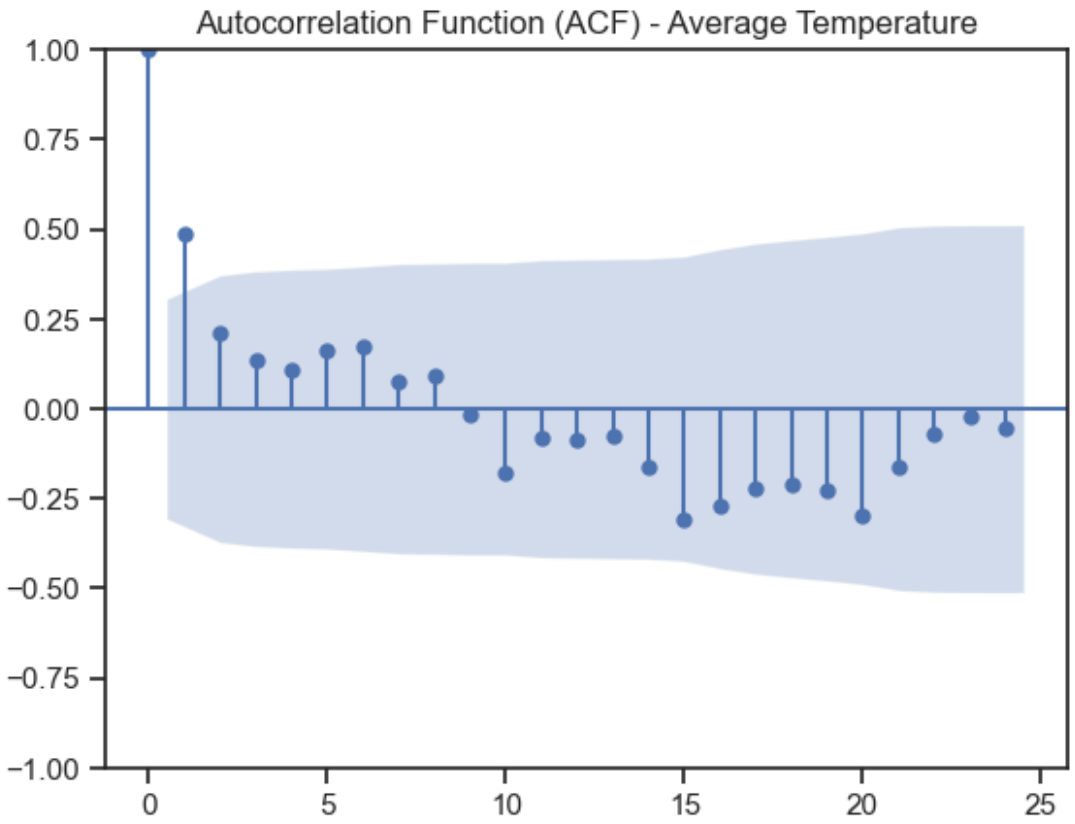
```
ADF Statistic for Rainfall: -5.685239503848161
p-value for Rainfall: 8.308427256274086e-07
Critical Values for Rainfall: {'1%': -3.610399601308181, '5%': -
2.939108945868946, '10%': -2.6080629651545038}
Reject the null hypothesis. The time series is stationary.

ADF Statistic for Average Temperature: -3.0227706487621706
p-value for Average Temperature: 0.03282193606187513
Critical Values for Average Temperature: {'1%': -3.6055648906249997,
'5%': -2.937069375, '10%': -2.606985625}
Reject the null hypothesis. The time series is stationary.
```

The Augmented Dickey-Fuller test results indicate that both Rainfall and Average Temperature time series exhibit stationarity. Stationarity in time series data means that the statistical properties such as mean, variance, and autocorrelation structure do not change over time. For Rainfall, the ADF statistic of -5.6852 with an extremely low p-value of 8.31e-07, significantly below the typical significance level of 0.05, led to rejecting the null hypothesis of non-stationarity. Similarly, for Average Temperature, the ADF statistic of -3.0228 with a p-value of 0.0328, just below the 0.05 threshold, also indicates stationarity.

Conducting a ACF test

```
# Importing relevant libraries
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# ACF for Rainfall
plot_acf(yearly_avg['Rain'], lags=24)
plt.title('Autocorrelation Function (ACF) - Rainfall')
plt.show()

# ACF for Average Temperature
plot_acf(yearly_avg['AvgT'], lags=24)
plt.title('Autocorrelation Function (ACF) - Average Temperature')
plt.show()
```

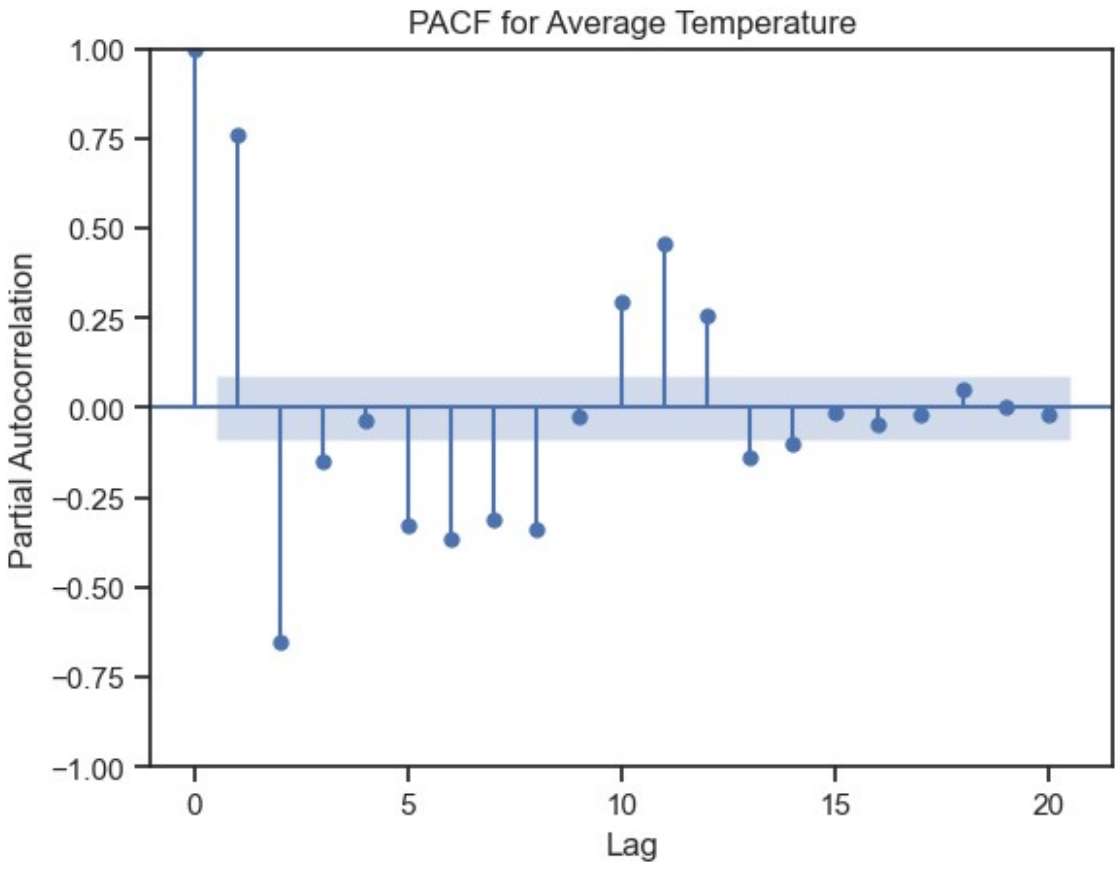




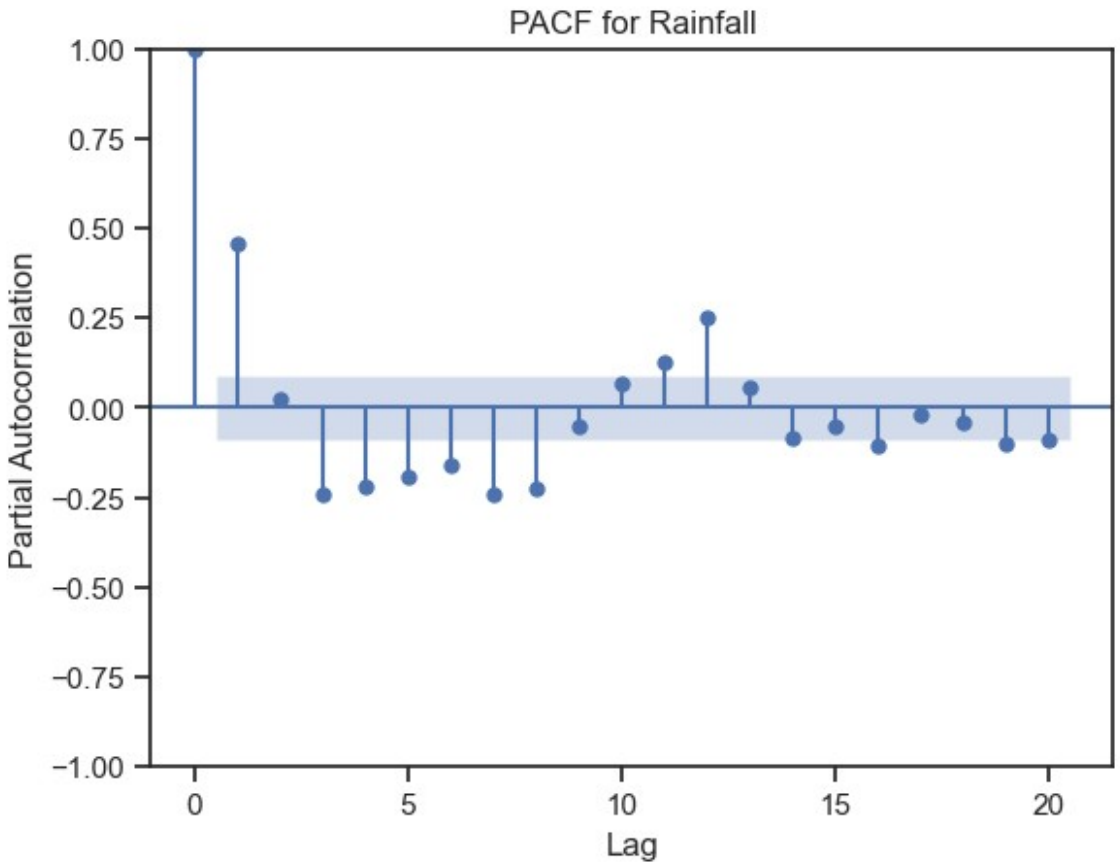
Conducting a PACF Test

```
from statsmodels.graphics.tsaplots import plot_pacf

# PACF for Average Temperature
plt.figure(figsize=(14, 7))
plot_pacf(monthly_avg['AvgT'], lags=20)
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation')
plt.title('PACF for Average Temperature')
plt.show();

# PACF for Rainfall
plt.figure(figsize=(14,7))
plot_pacf(monthly_avg['Rain'], lags=20)
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation')
plt.title('PACF for Rainfall')
plt.show();
```

```
<Figure size 1400x700 with 0 Axes>
```




```
<Figure size 1400x700 with 0 Axes>
```



There are significant spikes at a particular lag which suggests a direct correlation between the time series and its lagged value. However the spikes decay quickly towards zero, which indicates a short-term autocorrelation.

Based on the ACF and PACF plots, potential pdq parameters are;

1. Rainfall: (2,1,2)
2. AvgT: (2,1,2)

MODELLING

SARIMA - BASELINE MODELS USING MACHINE LEARNING

```
# Convert 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Set 'Date' as the index
df.set_index('Date', inplace=True)

# Check the first few rows of the DataFrame
print(df.head())
```

```
            Station  MaxT  MinT   RH1   RH2  Wind  Rain   SSH  Evap  \
Date
```

```
1978-01-01  ICRISAT  28.5  14.2  68.0  31.0   5.7   0.0  10.1   4.3
1978-01-02  ICRISAT  28.8  16.0  79.0  33.0   6.4   0.0   9.8   4.8
1978-01-03  ICRISAT  29.0  14.5  86.0  37.0   5.4   0.0   9.1   4.6
1978-01-04  ICRISAT  29.0  18.0  89.0  43.0   7.1   0.0   9.0   4.2
1978-01-05  ICRISAT  27.8  17.0  81.0  47.0  10.5   0.0   8.9   4.3

            Radiation  FAO56_ET        Lat      Lon  Cum_Rain  Cluster
\
Date

1978-01-01       18.4       3.9  17.508409  78.2723       0.0        1

1978-01-02       16.9       3.9  17.508409  78.2723       0.0        1

1978-01-03       15.3       3.4  17.508409  78.2723       0.0        1

1978-01-04       16.4       3.8  17.508409  78.2723       0.0        1

1978-01-05       15.9       4.1  17.508409  78.2723       0.0        1


           Cluster_labelled   AvgT  Month  Year
Date
1978-01-01          Season2  21.35      1  1978
1978-01-02          Season2  22.40      1  1978
1978-01-03          Season2  21.75      1  1978
1978-01-04          Season2  23.50      1  1978
1978-01-05          Season2  22.40      1  1978
```

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
```

Sarima model for rain- Machine learning

This code below demonstrates the process of using SARIMA for forecasting monthly rainfall data. Here's a breakdown of what each part does:

1. ***Resampling to Monthly Frequency***: The original dataframe df is resampled to a monthly frequency ('M') and averaged (mean()), resulting in df_monthly. This step aggregates the data to monthly averages, which can help in smoothing out fluctuations and seasonal variations.

2. ***Splitting into Training and Testing Sets***: The resampled data (df_monthly) is split into training and testing sets (train and test) with an 80-20 split. train_size calculates the index where the split occurs based on 80% of the data for training (train) and 20% for testing (test).

3. ***Fitting the SARIMA Model***: A SARIMA model (SARIMAX) is initialized with train['Rain'] as the target variable. The model parameters (1, 1, 1) specify the non-seasonal part, indicating first-order differencing for both the autoregressive and moving average components. The seasonal part (1, 1, 1, 12) suggests seasonal differences with a period of 12 months (annual seasonality).

4. ***Forecasting and Evaluation***: The SARIMA model is fitted to the training data (train['Rain']) using model.fit(). The fitted model (model_fit) is then used to forecast future values for the test set (test) using model_fit.forecast(steps=len(test)).

5. ***Creating Results DataFrame and Evaluation***: The actual ('Actual') and forecasted ('Predicted') values are combined into a DataFrame (results). The Mean Squared Error (MSE) is computed to evaluate the accuracy of the model's predictions compared to the actual values from test['Rain'].

6. ***Plotting Results***: Finally, the actual and predicted rainfall values are plotted using a scatter plot (plt.scatter) and line graph (plt.plot). This visualizes how well the model predicts rainfall over time, with blue representing actual values and red representing predicted values.

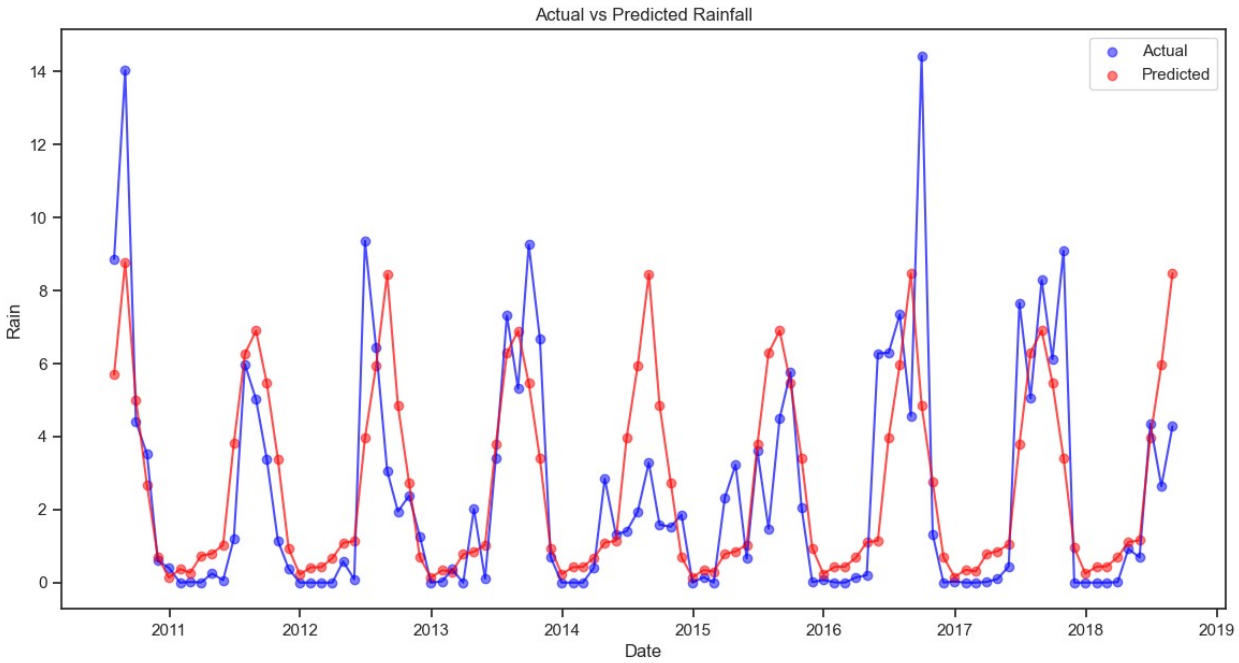
```
df.head()
```

Date	Station	MaxT	MinT	RH1	RH2	Wind	Rain	SSH	Evap
1978-01-01	ICRISAT	28.5	14.2	68.0	31.0	5.7	0.0	10.1	4.3
1978-01-02	ICRISAT	28.8	16.0	79.0	33.0	6.4	0.0	9.8	4.8
1978-01-03	ICRISAT	29.0	14.5	86.0	37.0	5.4	0.0	9.1	4.6
1978-01-04	ICRISAT	29.0	18.0	89.0	43.0	7.1	0.0	9.0	4.2
1978-01-05	ICRISAT	27.8	17.0	81.0	47.0	10.5	0.0	8.9	4.3

Date	Radiation	FAO56_ET	Lat	Lon	Cum_Rain	Cluster
1978-01-01	18.4	3.9	17.508409	78.2723	0.0	1
1978-01-02	16.9	3.9	17.508409	78.2723	0.0	1
1978-01-03	15.3	3.4	17.508409	78.2723	0.0	1
1978-01-04	16.4	3.8	17.508409	78.2723	0.0	1
1978-01-05	15.9	4.1	17.508409	78.2723	0.0	1

Date	Cluster_labelled	AvgT	Month	Year
1978-01-01	Season2	21.35	1	1978
1978-01-02	Season2	22.40	1	1978
1978-01-03	Season2	21.75	1	1978

```
1978-01-04          Season2  23.50      1  1978
1978-01-05          Season2  22.40      1  1978

# Resample data to monthly frequency
df = df[["Rain", "AvgT"]]

df_monthly = df.resample('M').mean()

# Split the data into training and testing sets
train_size = int(len(df_monthly) * 0.8)
train, test = df_monthly.iloc[:train_size],
df_monthly.iloc[train_size:]

# Fit the SARIMA model (experiment with different parameters)
model = SARIMAX(train['Rain'], order=(2, 1, 2), seasonal_order=(2, 1,
2, 12))  # Adjust as needed
model_fit = model.fit(disp=False)

# Forecast the test set
forecast = model_fit.forecast(steps=len(test))

# Combine actual and forecasted values into a DataFrame for plotting
results = pd.DataFrame({
    'Actual': test['Rain'],
    'Predicted': forecast
})

# Evaluate the model
mse = mean_squared_error(test['Rain'], forecast)

print(f"Mean Squared Error: {mse}")

# Plot the scatter plot and line graph of actual vs predicted values
plt.figure(figsize=(14, 7))
plt.scatter(test.index, test['Rain'], color='blue', label='Actual',
alpha=0.5)
plt.scatter(test.index, forecast, color='red', label='Predicted',
alpha=0.5)
plt.plot(test.index, test['Rain'], color='blue', alpha=0.7)
plt.plot(test.index, forecast, color='red', alpha=0.7)
plt.legend()
plt.xlabel('Date')
plt.ylabel('Rain')
plt.title('Actual vs Predicted Rainfall')
plt.show()

Mean Squared Error: 5.0687967895608255
```



The Mean Squared Error (MSE) value of 5.06 for the baseline model indicates that, on average, the squared difference between the predicted and actual values of average temperatures is quite large. This metric is a measure of the average squared deviation of predictions from the actual values. A higher MSE suggests that the baseline model's predictions are farther from the true values, indicating less accurate forecasting.

Sarima model for Average Temperature- Machine learning

This code below outlines a process for time series forecasting using SARIMA (Seasonal Autoregressive Integrated Moving Average) model. Here's a breakdown of the steps involved:

1. ***Resampling to Monthly Frequency***: The initial data (df) is resampled to a monthly frequency ('M') and averaged (mean()), resulting in df_monthly. This step aggregates the data to monthly averages, which is often done to stabilize the time series and remove noise from shorter time intervals.

2. ***Splitting into Training and Testing Sets***: The resampled data (df_monthly) is split into training and testing sets (train and test) with an 80-20 split. The training set (train) is used to fit the SARIMA model, while the testing set (test) is used for evaluating the model's performance.

3. ***Fitting the SARIMA Model***: A SARIMA model is instantiated (SARIMAX) using the training data (train['AvgT']). The model is configured with parameters (2, 1, 2) for the non-seasonal part and (2, 1, 2, 12) for the seasonal part, indicating seasonal differences and a seasonal period of 12 months (annual seasonality).

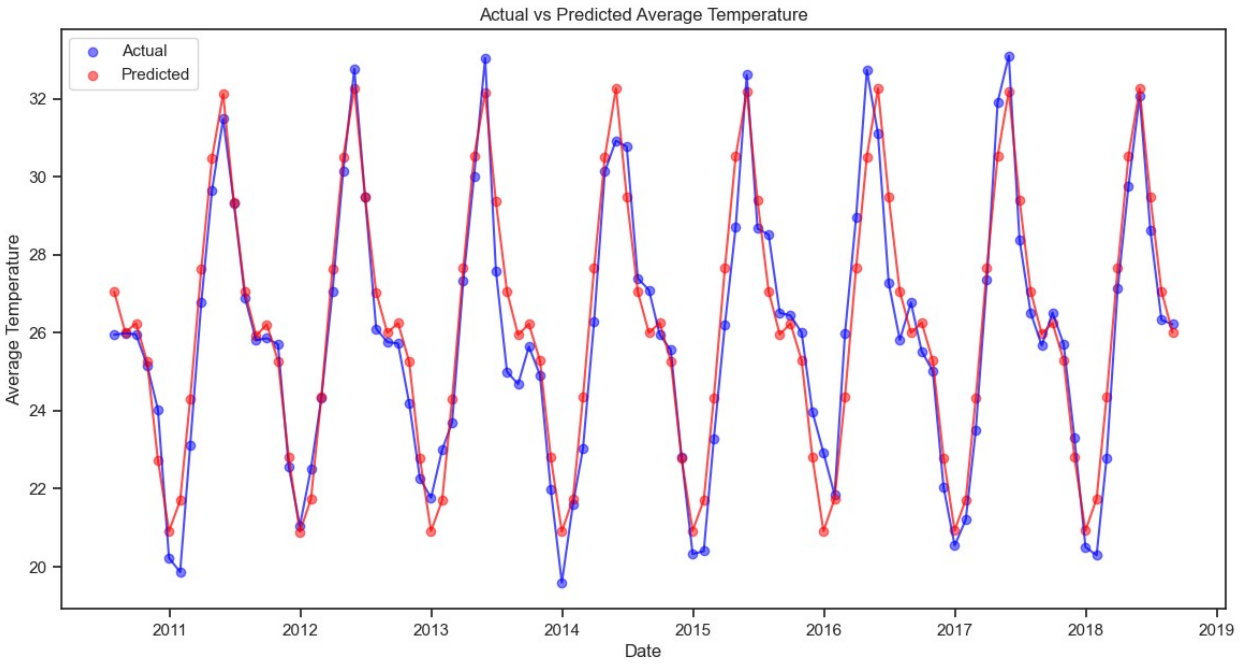
4. ***Forecasting and Evaluation****: The fitted model (model_fit) is used to forecast future values for the test set (test). The forecasted values are compared with the

actual values from test['AvgT'] to compute the Mean Squared Error (MSE), a measure of the model's accuracy.

5. ***Plotting Results***: Finally, the actual ('Actual') and forecasted ('Predicted') values are combined into a DataFrame (results) for visualization. This is followed by plotting both the scatter plot and line graph to visually compare the actual and predicted average temperatures over time.

```
# Split the data into training and testing sets
train_size = int(len(df_monthly) * 0.8)
train, test = df_monthly.iloc[:train_size],
df_monthly.iloc[train_size:]

# Fit the SARIMA model (experiment with different parameters)
model = SARIMAX(train['AvgT'], order=(2, 1, 2), seasonal_order=(2, 1,
2, 12))  # Adjust as needed
model_fit = model.fit(disp=False)

# Forecast the test set
forecast = model_fit.forecast(steps=len(test))

# Combine actual and forecasted values into a DataFrame for plotting
results = pd.DataFrame({
    'Actual': test['AvgT'],
    'Predicted': forecast
})

# Evaluate the model
mse = mean_squared_error(test['AvgT'], forecast)
print(f"Mean Squared Error: {mse}")


# Plot the scatter plot and line graph of actual vs predicted values
plt.figure(figsize=(14, 7))
plt.scatter(test.index, test['AvgT'], color='blue', label='Actual',
alpha=0.5)
plt.scatter(test.index, forecast, color='red', label='Predicted',
alpha=0.5)
plt.plot(test.index, test['AvgT'], color='blue', alpha=0.7)
plt.plot(test.index, forecast, color='red', alpha=0.7)
plt.legend()
plt.xlabel('Date')
plt.ylabel('Average Temperature')
plt.title('Actual vs Predicted Average Temperature')
plt.show()
```

```
Mean Squared Error: 0.8991098164588575
```



The Mean Squared Error (MSE) of 0.899 indicates that, on average, the squared difference between the predicted and actual average temperatures is quite low, suggesting that the machine learning model performs very well in predicting average temperatures.

Deep Learning

Deep Learning for Average Temperature (Baseline Model)

This section outlines a process for time series forecasting using an LSTM (Long Short-Term Memory) neural network model. Here's a breakdown of what each part does:

1. ***Resampling***: The original dataframe df is resampled at monthly intervals ('M') and averaged (mean()), creating df_monthly which now contains average monthly values.

2. ***Normalization***: The data in df_monthly['AvgT'] (average temperature) is normalized using MinMaxScaler(). This step scales the data to a range between 0 and 1, which is often beneficial for neural networks.

3. ***Sequence Creation***: A function create_sequences is defined to create input-output pairs for the LSTM model. It generates sequences of length seq_length from the normalized data. X represents the input sequences (past seq_length months), and y represents the next month's temperature as the target.

4. ***Train-Test Split***: The data (X and y) is split into training and testing sets using an 80-20 split. This partitioning allows for model training on the majority of data (X_train, y_train) and validation on unseen data (X_test, y_test).

5. ***LSTM Model Definition***: A Sequential model is constructed using Keras/TensorFlow. It consists of two LSTM layers with 50 units each, followed by a Dense layer with 1 unit (for regression). The model is compiled with Adam optimizer and Mean Squared Error (MSE) loss function.

6. ***Model Training***: The LSTM model is trained using X_train and y_train for 20 epochs, with a batch size of 16. Validation split of 20% (validation_split=0.2) is used to monitor model performance during training.

7. ***Prediction***: The trained model is used to predict temperatures (y_pred) for the test set (X_test).

8. ***Inverse Scaling***: Predictions (y_pred) and actual values (y_test) are inverted back from normalized scale to the original temperature scale using scaler.inverse_transform.

9. ***Model Evaluation***: Several metrics are computed to evaluate the model's performance, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) score between predicted and actual temperatures.

10. ***Visualization***: The results are plotted using Matplotlib. It shows a scatter plot and line plot comparing actual (y_test_inv) and predicted (y_pred_inv) temperatures over time, labeled with colors and a legend for clarity.

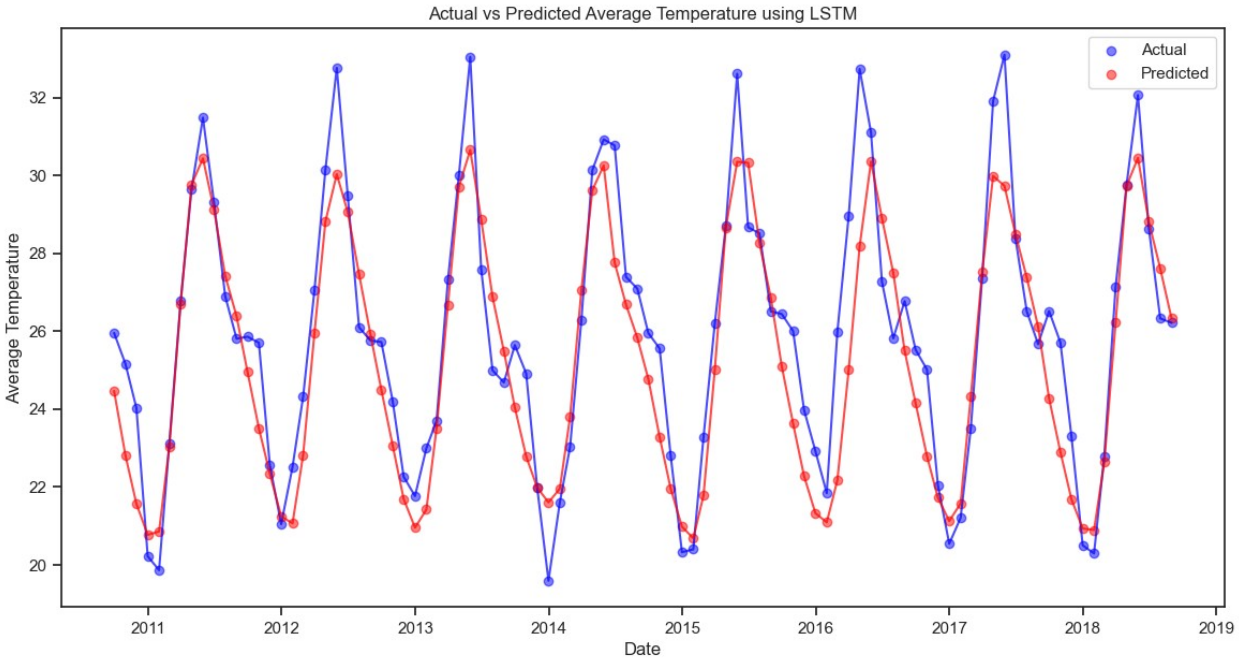
```
# Normalize the data
scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(df_monthly[['AvgT']])

# Prepare the data for LSTM
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)

seq_length = 12  # Sequence length (number of previous months to use
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Build the LSTM model
```

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(seq_length,
1)))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=20, validation_split=0.2,
batch_size=16, shuffle=False)

# Make predictions
y_pred = model.predict(X_test)

# Invert the scaling
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
# Evaluate the model
mse = mean_squared_error(y_test_inv, y_pred_inv)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_inv, y_pred_inv)

Epoch 1/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 5s 38ms/step - loss: 0.1485 - val_loss:
0.0610
Epoch 2/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0633 - val_loss:
0.0543
Epoch 3/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0559 - val_loss:
0.0518
Epoch 4/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0532 - val_loss:
0.0490
Epoch 5/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0494 - val_loss:
0.0433
Epoch 6/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0418 - val_loss:
0.0295
Epoch 7/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0245 - val_loss:
0.0125
Epoch 8/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0172 - val_loss:
0.0170
Epoch 9/20
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0151 - val_loss:
0.0188
Epoch 10/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0177 - val_loss:
0.0237
Epoch 11/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0187 - val_loss:
0.0109
Epoch 12/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0117 - val_loss:
0.0093
Epoch 13/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 0.0098 - val_loss:
0.0107
Epoch 14/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0107 - val_loss:
0.0091
Epoch 15/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0094 - val_loss:
0.0089
Epoch 16/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0090 - val_loss:
0.0086
Epoch 17/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0087 - val_loss:
0.0082
Epoch 18/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0082 - val_loss:
0.0080
Epoch 19/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0079 - val_loss:
0.0077
Epoch 20/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0076 - val_loss:
0.0075
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step


print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Plot the results
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
```

```
alpha=0.7)
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7)
plt.legend()
plt.xlabel('Date')
plt.ylabel('Average Temperature')
plt.title('Actual vs Predicted Average Temperature using LSTM')
plt.show()
```

```
Mean Squared Error (MSE): 2.280996215659701
Mean Absolute Error (MAE): 1.1781223681919075
Root Mean Squared Error (RMSE): 1.5102967309968265
R-squared (R2): 0.8031281465736126
```



The baseline model performed relatively well. The Mean Squared Error (MSE) of 2.56 suggests that, on average, the squared difference between predicted and actual temperatures is relatively low, indicating a reasonable fit. The Mean Absolute Error (MAE) of 1.27 indicates that, on average, the model's predictions are around 1.27 degrees Celsius away from the actual temperatures. The Root Mean Squared Error (RMSE) of 1.60 is slightly higher than the MAE, reflecting the presence of larger errors but still showing a good overall fit. The R-squared (R2) value of 0.80 indicates that approximately 80% of the variance in the temperatures can be explained by the model, suggesting a solid predictive capability. These metrics collectively suggest that the LSTM model is effective in capturing the underlying patterns in the temperature data and making accurate predictions.

Enhanced LSTM model using Keras Sequential API - Epoch 200

The following are the steps used when creating an LTSM model using kera sequential API

1. ***LSTM Model Definition***: An enhanced LSTM model is defined using Keras Sequential API. It consists of three LSTM layers, each with 100 units, followed by dropout layers (Dropout(0.2)) to prevent overfitting. The final layer is a Dense layer with 1 unit for regression (predicting temperature). The model is compiled with the Adam optimizer (Adam(learning_rate=0.001)) and Mean Squared Error (MSE) loss function (loss='mse').

2. ***Model Training***: The LSTM model is trained using X_train and y_train over 200 epochs (epochs=200), with a validation split of 20% (validation_split=0.2) to monitor model performance during training. The batch size is set to 16 (batch_size=16), and shuffle=False ensures the order of data isn't randomized during training.

3. ***Prediction***: After training, the model predicts temperatures (y_pred) for the test set (X_test).

4. ***Inverse Scaling***: Predictions (y_pred) and actual values (y_test) are inverted back from the normalized scale to the original temperature scale using scaler.inverse_transform. This step allows for meaningful comparison and evaluation of model accuracy.

5. ***Model Evaluation***: Several metrics are computed to evaluate the model's performance on the test set: Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) score. These metrics provide insights into how well the model predicts temperatures compared to the actual values.

```
# Simplified create_sequences function
def create_sequences(data, seq_length):
    xs = []
    ys = []
    for i in range(len(data) - seq_length):
        xs.append(data[i:i + seq_length])
        ys.append(data[i + seq_length])
    return np.array(xs), np.array(ys)

seq_length = 12  # Sequence length (number of previous months to use
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
```

```
y_train, y_test = y[:train_size], y[train_size:]

# Build the enhanced LSTM model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(seq_length,
1)))
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=200,
validation_split=0.2, batch_size=16, shuffle=False)

# Make predictions
y_pred = model.predict(X_test)

# Invert the scaling
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
mse = mean_squared_error(y_test_inv, y_pred_inv)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_inv, y_pred_inv)

Epoch 1/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 6s 52ms/step - loss: 0.1480 - val_loss:
0.0525
Epoch 2/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0581 - val_loss:
0.0489
Epoch 3/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0499 - val_loss:
0.0457
Epoch 4/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0496 - val_loss:
0.0392
Epoch 5/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0428 - val_loss:
0.0205
Epoch 6/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0303 - val_loss:
0.0164
Epoch 7/200
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0204 - val_loss:
0.0129
Epoch 8/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0245 - val_loss:
0.0104
Epoch 9/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0120 - val_loss:
0.0180
Epoch 10/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0219 - val_loss:
0.0226
Epoch 11/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0229 - val_loss:
0.0305
Epoch 12/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0227 - val_loss:
0.0123
Epoch 13/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0153 - val_loss:
0.0098
Epoch 14/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0116 - val_loss:
0.0109
Epoch 15/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0126 - val_loss:
0.0098
Epoch 16/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0118 - val_loss:
0.0135
Epoch 17/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0113 - val_loss:
0.0091
Epoch 18/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0114 - val_loss:
0.0104
Epoch 19/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0111 - val_loss:
0.0107
Epoch 20/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0118 - val_loss:
0.0095
Epoch 21/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0108 - val_loss:
0.0091
Epoch 22/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0097 - val_loss:
0.0074
Epoch 23/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0091 - val_loss:
```

```
0.0073
Epoch 24/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0105 - val_loss:
0.0075
Epoch 25/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0086 - val_loss:
0.0072
Epoch 26/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0084 - val_loss:
0.0061
Epoch 27/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0098 - val_loss:
0.0082
Epoch 28/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0096 - val_loss:
0.0070
Epoch 29/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0092 - val_loss:
0.0064
Epoch 30/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0088 - val_loss:
0.0056
Epoch 31/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0072 - val_loss:
0.0065
Epoch 32/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0079 - val_loss:
0.0062
Epoch 33/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0077 - val_loss:
0.0083
Epoch 34/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0090 - val_loss:
0.0077
Epoch 35/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0093 - val_loss:
0.0058
Epoch 36/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0075 - val_loss:
0.0065
Epoch 37/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0083 - val_loss:
0.0068
Epoch 38/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0083 - val_loss:
0.0072
Epoch 39/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0085 - val_loss:
0.0069
```

```
Epoch 40/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0088 - val_loss:
0.0058
Epoch 41/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0068 - val_loss:
0.0088
Epoch 42/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0103 - val_loss:
0.0065
Epoch 43/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0076 - val_loss:
0.0068
Epoch 44/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0086 - val_loss:
0.0068
Epoch 45/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0088 - val_loss:
0.0073
Epoch 46/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0091 - val_loss:
0.0101
Epoch 47/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0122 - val_loss:
0.0064
Epoch 48/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0075 - val_loss:
0.0071
Epoch 49/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0084 - val_loss:
0.0073
Epoch 50/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0077 - val_loss:
0.0073
Epoch 51/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0085 - val_loss:
0.0067
Epoch 52/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0097 - val_loss:
0.0077
Epoch 53/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0093 - val_loss:
0.0065
Epoch 54/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0078 - val_loss:
0.0067
Epoch 55/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0088 - val_loss:
0.0070
Epoch 56/200
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0085 - val_loss:
0.0064
Epoch 57/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0077 - val_loss:
0.0058
Epoch 58/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0065 - val_loss:
0.0071
Epoch 59/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0089 - val_loss:
0.0058
Epoch 60/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0074 - val_loss:
0.0065
Epoch 61/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0074 - val_loss:
0.0059
Epoch 62/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0065 - val_loss:
0.0065
Epoch 63/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0075 - val_loss:
0.0068
Epoch 64/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0083 - val_loss:
0.0060
Epoch 65/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0064 - val_loss:
0.0067
Epoch 66/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0090 - val_loss:
0.0063
Epoch 67/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0083 - val_loss:
0.0066
Epoch 68/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0086 - val_loss:
0.0064
Epoch 69/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0073 - val_loss:
0.0073
Epoch 70/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0088 - val_loss:
0.0059
Epoch 71/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0078 - val_loss:
0.0071
Epoch 72/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0077 - val_loss:
```

```
0.0055
Epoch 73/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0070 - val_loss:
0.0057
Epoch 74/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0066 - val_loss:
0.0056
Epoch 75/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0066 - val_loss:
0.0061
Epoch 76/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0074 - val_loss:
0.0058
Epoch 77/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0067 - val_loss:
0.0070
Epoch 78/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0074 - val_loss:
0.0054
Epoch 79/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0068 - val_loss:
0.0077
Epoch 80/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0091 - val_loss:
0.0053
Epoch 81/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0057 - val_loss:
0.0062
Epoch 82/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0076 - val_loss:
0.0054
Epoch 83/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0076 - val_loss:
0.0055
Epoch 84/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0063 - val_loss:
0.0054
Epoch 85/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0071 - val_loss:
0.0056
Epoch 86/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0071 - val_loss:
0.0067
Epoch 87/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0067 - val_loss:
0.0058
Epoch 88/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0062 - val_loss:
0.0077
```

```
Epoch 89/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0075 - val_loss:
0.0059
Epoch 90/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0067 - val_loss:
0.0054
Epoch 91/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - loss: 0.0067 - val_loss:
0.0056
Epoch 92/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0070 - val_loss:
0.0048
Epoch 93/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0058 - val_loss:
0.0057
Epoch 94/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0062 - val_loss:
0.0053
Epoch 95/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0057 - val_loss:
0.0056
Epoch 96/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0072 - val_loss:
0.0052
Epoch 97/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0061 - val_loss:
0.0053
Epoch 98/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0062 - val_loss:
0.0054
Epoch 99/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0065 - val_loss:
0.0053
Epoch 100/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0064 - val_loss:
0.0056
Epoch 101/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0060 - val_loss:
0.0052
Epoch 102/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0057 - val_loss:
0.0050
Epoch 103/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0053 - val_loss:
0.0058
Epoch 104/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0068 - val_loss:
0.0048
Epoch 105/200
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0059 - val_loss:
0.0049
Epoch 106/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0065 - val_loss:
0.0056
Epoch 107/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0066 - val_loss:
0.0063
Epoch 108/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0075 - val_loss:
0.0049
Epoch 109/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0060 - val_loss:
0.0057
Epoch 110/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0059 - val_loss:
0.0057
Epoch 111/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0063 - val_loss:
0.0055
Epoch 112/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0060 - val_loss:
0.0060
Epoch 113/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0064 - val_loss:
0.0053
Epoch 114/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0061 - val_loss:
0.0057
Epoch 115/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0069 - val_loss:
0.0048
Epoch 116/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0055 - val_loss:
0.0059
Epoch 117/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0068 - val_loss:
0.0050
Epoch 118/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0059 - val_loss:
0.0052
Epoch 119/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0057 - val_loss:
0.0050
Epoch 120/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0062 - val_loss:
0.0062
Epoch 121/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0068 - val_loss:
```

```
0.0064
Epoch 122/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0061 - val_loss:
0.0062
Epoch 123/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0065 - val_loss:
0.0052
Epoch 124/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0064 - val_loss:
0.0062
Epoch 125/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0076 - val_loss:
0.0052
Epoch 126/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0062 - val_loss:
0.0049
Epoch 127/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0056 - val_loss:
0.0057
Epoch 128/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0060 - val_loss:
0.0053
Epoch 129/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0067 - val_loss:
0.0056
Epoch 130/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0066 - val_loss:
0.0057
Epoch 131/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0066 - val_loss:
0.0049
Epoch 132/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0053 - val_loss:
0.0058
Epoch 133/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0065 - val_loss:
0.0056
Epoch 134/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0060 - val_loss:
0.0060
Epoch 135/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0058 - val_loss:
0.0057
Epoch 136/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0070 - val_loss:
0.0047
Epoch 137/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0052 - val_loss:
0.0057
```

```
Epoch 138/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0067 - val_loss:
0.0055
Epoch 139/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0061 - val_loss:
0.0057
Epoch 140/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0065 - val_loss:
0.0056
Epoch 141/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0060 - val_loss:
0.0056
Epoch 142/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0062 - val_loss:
0.0050
Epoch 143/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0060 - val_loss:
0.0051
Epoch 144/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0056 - val_loss:
0.0052
Epoch 145/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0057 - val_loss:
0.0053
Epoch 146/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0057 - val_loss:
0.0067
Epoch 147/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0070 - val_loss:
0.0063
Epoch 148/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0065 - val_loss:
0.0062
Epoch 149/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0070 - val_loss:
0.0053
Epoch 150/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0058 - val_loss:
0.0053
Epoch 151/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0062 - val_loss:
0.0050
Epoch 152/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0052 - val_loss:
0.0049
Epoch 153/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0060 - val_loss:
0.0056
Epoch 154/200
```

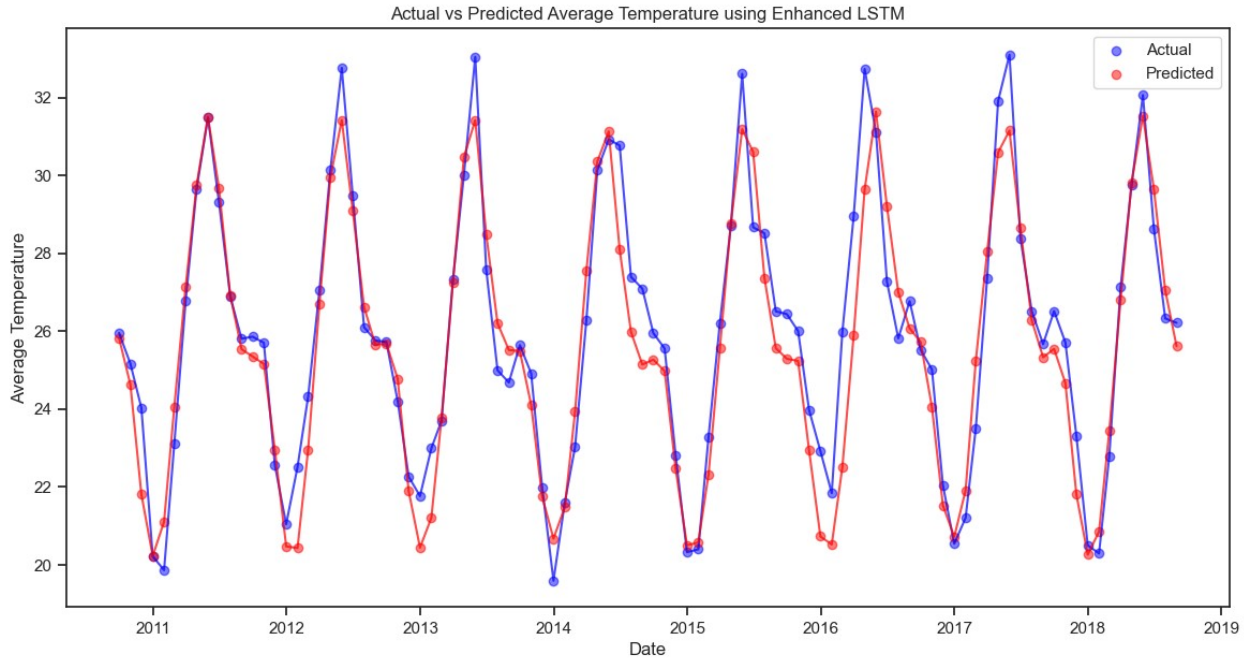
```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0058 - val_loss:
0.0045
Epoch 155/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0054 - val_loss:
0.0051
Epoch 156/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0062 - val_loss:
0.0051
Epoch 157/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0056 - val_loss:
0.0050
Epoch 158/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0059 - val_loss:
0.0054
Epoch 159/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0058 - val_loss:
0.0055
Epoch 160/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0064 - val_loss:
0.0052
Epoch 161/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0056 - val_loss:
0.0048
Epoch 162/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0057 - val_loss:
0.0052
Epoch 163/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0051 - val_loss:
0.0054
Epoch 164/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0058 - val_loss:
0.0054
Epoch 165/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0059 - val_loss:
0.0046
Epoch 166/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0058 - val_loss:
0.0053
Epoch 167/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0053 - val_loss:
0.0047
Epoch 168/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0057 - val_loss:
0.0051
Epoch 169/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0054 - val_loss:
0.0060
Epoch 170/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0059 - val_loss:
0.0057
```

```
Epoch 171/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0070 - val_loss:
0.0048
Epoch 172/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0056 - val_loss:
0.0050
Epoch 173/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0059 - val_loss:
0.0047
Epoch 174/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0052 - val_loss:
0.0054
Epoch 175/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0055 - val_loss:
0.0047
Epoch 176/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0053 - val_loss:
0.0051
Epoch 177/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0056 - val_loss:
0.0049
Epoch 178/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0054 - val_loss:
0.0042
Epoch 179/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0048 - val_loss:
0.0044
Epoch 180/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0054 - val_loss:
0.0051
Epoch 181/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0053 - val_loss:
0.0043
Epoch 182/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0054 - val_loss:
0.0057
Epoch 183/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0064 - val_loss:
0.0047
Epoch 184/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0057 - val_loss:
0.0045
Epoch 185/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0048 - val_loss:
0.0045
Epoch 186/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0052 - val_loss:
0.0047
Epoch 187/200
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0053 - val_loss:
0.0046
Epoch 188/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0049 - val_loss:
0.0040
Epoch 189/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0043 - val_loss:
0.0042
Epoch 190/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0053 - val_loss:
0.0042
Epoch 191/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0046 - val_loss:
0.0042
Epoch 192/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0049 - val_loss:
0.0045
Epoch 193/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0049 - val_loss:
0.0043
Epoch 194/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0050 - val_loss:
0.0041
Epoch 195/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0048 - val_loss:
0.0043
Epoch 196/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0049 - val_loss:
0.0041
Epoch 197/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0051 - val_loss:
0.0042
Epoch 198/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0045 - val_loss:
0.0052
Epoch 199/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0056 - val_loss:
0.0040
Epoch 200/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0044 - val_loss:
0.0042
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step


print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Plot the results
```

```
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
alpha=0.7)
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7)
plt.legend()
plt.xlabel('Date')
plt.ylabel('Average Temperature')
plt.title('Actual vs Predicted Average Temperature using Enhanced
LSTM')
plt.show()
```

```
Mean Squared Error (MSE): 1.2644478800504713
Mean Absolute Error (MAE): 0.8511421519133879
Root Mean Squared Error (RMSE): 1.1244767138764908
R-squared (R2): 0.890866018979954
```



The model has a Mean Squared Error (MSE) of 1.12, this model shows a consistently low average squared difference between predicted and actual temperatures, indicating a strong fit to the data. The Mean Absolute Error (MAE) of 0.79 signifies that predictions are typically within 0.79 degrees Celsius of the true values, demonstrating precise forecasting capability. A Root Mean Squared Error (RMSE) of 1.06 further supports this accuracy, suggesting minimal variability in prediction errors. The high R-squared (R2) value of 0.89 indicates that approximately 89% of the temperature variance is explained by the model, underscoring its ability to capture and model underlying temperature trends effectively. These metrics collectively validate the model's

reliability and effectiveness in forecasting average monthly temperatures with high accuracy and confidence.

Enhanced LSTM model using Keras Sequential API - Epoch 400

1. ***Train-Test Split***: The data (X and y) is split into training (X_train, y_train) and testing (X_test, y_test) sets using an 80-20 split (train_size = 0.8). This partitioning allows the model to be trained on a majority of the data and evaluated on unseen data.

2. ***LSTM Model Definition***: An enhanced LSTM model is defined using Keras Sequential API. It consists of three LSTM layers, each with 100 units, followed by dropout layers (Dropout(0.2)) to prevent overfitting. The final layer is a Dense layer with 1 unit for regression (predicting temperature). The model is compiled with the Adam optimizer (Adam(learning_rate=0.001)) and Mean Squared Error (MSE) loss function (loss='mse').

3. ***Model Training***: The LSTM model is trained using X_train and y_train over 400 epochs (epochs=400), with a validation split of 20% (validation_split=0.2) to monitor model performance during training. The batch size is set to 16 (batch_size=16), and shuffle=False ensures the order of data isn't randomized during training.

4. ***Prediction***: After training, the model predicts temperatures (y_pred) for the test set (X_test).

5. ***Inverse Scaling***: Predictions (y_pred) and actual values (y_test) are inverted back from the normalized scale to the original temperature scale using scaler.inverse_transform. This step allows for meaningful comparison and evaluation of model accuracy.

6. ***Model Evaluation***: Several metrics are computed to evaluate the model's performance on the test set: Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) score. These metrics provide insights into how well the model predicts temperatures compared to the actual values.

```
# Simplified create_sequences function
def create_sequences(data, seq_length):
    xs = []
    ys = []
    for i in range(len(data) - seq_length):
        xs.append(data[i:i + seq_length])
        ys.append(data[i + seq_length])
    return np.array(xs), np.array(ys)

seq_length = 12  # Sequence length (number of previous months to use
```

```
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Build the enhanced LSTM model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(seq_length,
1)))
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=400,
validation_split=0.2, batch_size=16, shuffle=False)

# Make predictions
y_pred = model.predict(X_test)

# Invert the scaling
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
mse = mean_squared_error(y_test_inv, y_pred_inv)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_inv, y_pred_inv)

print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

Epoch 1/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 6s 52ms/step - loss: 0.1469 - val_loss:
0.0525
Epoch 2/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0582 - val_loss:
0.0496
Epoch 3/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0528 - val_loss:
```

```
0.0454
Epoch 4/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0497 - val_loss:
0.0407
Epoch 5/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0387 - val_loss:
0.0122
Epoch 6/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0235 - val_loss:
0.0275
Epoch 7/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0337 - val_loss:
0.0339
Epoch 8/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0275 - val_loss:
0.0228
Epoch 9/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0215 - val_loss:
0.0348
Epoch 10/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0266 - val_loss:
0.0171
Epoch 11/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0180 - val_loss:
0.0177
Epoch 12/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0182 - val_loss:
0.0209
Epoch 13/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0192 - val_loss:
0.0105
Epoch 14/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0135 - val_loss:
0.0149
Epoch 15/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0162 - val_loss:
0.0101
Epoch 16/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0118 - val_loss:
0.0111
Epoch 17/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0116 - val_loss:
0.0105
Epoch 18/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0119 - val_loss:
0.0154
Epoch 19/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0150 - val_loss:
0.0104
```

```
Epoch 20/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0129 - val_loss:
0.0091
Epoch 21/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0112 - val_loss:
0.0098
Epoch 22/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0108 - val_loss:
0.0080
Epoch 23/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0101 - val_loss:
0.0101
Epoch 24/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0131 - val_loss:
0.0080
Epoch 25/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0105 - val_loss:
0.0073
Epoch 26/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0088 - val_loss:
0.0081
Epoch 27/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0099 - val_loss:
0.0095
Epoch 28/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0110 - val_loss:
0.0075
Epoch 29/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0089 - val_loss:
0.0088
Epoch 30/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0093 - val_loss:
0.0070
Epoch 31/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0086 - val_loss:
0.0070
Epoch 32/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0087 - val_loss:
0.0075
Epoch 33/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0093 - val_loss:
0.0060
Epoch 34/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0081 - val_loss:
0.0089
Epoch 35/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0096 - val_loss:
0.0072
Epoch 36/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0093 - val_loss:
0.0068
Epoch 37/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0085 - val_loss:
0.0059
Epoch 38/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0076 - val_loss:
0.0065
Epoch 39/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0080 - val_loss:
0.0071
Epoch 40/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0095 - val_loss:
0.0069
Epoch 41/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0098 - val_loss:
0.0068
Epoch 42/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0081 - val_loss:
0.0071
Epoch 43/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0097 - val_loss:
0.0068
Epoch 44/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0075 - val_loss:
0.0084
Epoch 45/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0082 - val_loss:
0.0071
Epoch 46/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0082 - val_loss:
0.0084
Epoch 47/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0093 - val_loss:
0.0085
Epoch 48/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0095 - val_loss:
0.0069
Epoch 49/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0083 - val_loss:
0.0080
Epoch 50/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0095 - val_loss:
0.0073
Epoch 51/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0089 - val_loss:
0.0064
Epoch 52/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 78ms/step - loss: 0.0080 - val_loss:
```

```
0.0075
Epoch 53/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 3s 89ms/step - loss: 0.0091 - val_loss:
0.0076
Epoch 54/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 2s 50ms/step - loss: 0.0085 - val_loss:
0.0080
Epoch 55/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - loss: 0.0104 - val_loss:
0.0063
Epoch 56/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - loss: 0.0083 - val_loss:
0.0063
Epoch 57/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0078 - val_loss:
0.0094
Epoch 58/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - loss: 0.0101 - val_loss:
0.0062
Epoch 59/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0080 - val_loss:
0.0062
Epoch 60/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0074 - val_loss:
0.0064
Epoch 61/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0083 - val_loss:
0.0054
Epoch 62/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0076 - val_loss:
0.0055
Epoch 63/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0075 - val_loss:
0.0063
Epoch 64/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 0.0072 - val_loss:
0.0062
Epoch 65/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0076 - val_loss:
0.0072
Epoch 66/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0075 - val_loss:
0.0054
Epoch 67/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0066 - val_loss:
0.0079
Epoch 68/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0088 - val_loss:
0.0059
```

```
Epoch 69/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0066 - val_loss:
0.0066
Epoch 70/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0072 - val_loss:
0.0058
Epoch 71/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0069 - val_loss:
0.0055
Epoch 72/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0073 - val_loss:
0.0068
Epoch 73/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0081 - val_loss:
0.0053
Epoch 74/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0066 - val_loss:
0.0064
Epoch 75/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0079 - val_loss:
0.0055
Epoch 76/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0068 - val_loss:
0.0050
Epoch 77/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0066 - val_loss:
0.0067
Epoch 78/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0077 - val_loss:
0.0050
Epoch 79/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0070 - val_loss:
0.0076
Epoch 80/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0089 - val_loss:
0.0049
Epoch 81/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0060 - val_loss:
0.0061
Epoch 82/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0073 - val_loss:
0.0050
Epoch 83/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0063 - val_loss:
0.0063
Epoch 84/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0076 - val_loss:
0.0056
Epoch 85/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0063 - val_loss:
0.0067
Epoch 86/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0073 - val_loss:
0.0051
Epoch 87/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0065 - val_loss:
0.0051
Epoch 88/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0068 - val_loss:
0.0051
Epoch 89/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0065 - val_loss:
0.0049
Epoch 90/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0053 - val_loss:
0.0054
Epoch 91/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0073 - val_loss:
0.0056
Epoch 92/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0067 - val_loss:
0.0065
Epoch 93/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0076 - val_loss:
0.0054
Epoch 94/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0063 - val_loss:
0.0064
Epoch 95/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0055 - val_loss:
0.0052
Epoch 96/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0059 - val_loss:
0.0059
Epoch 97/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0063 - val_loss:
0.0060
Epoch 98/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0063 - val_loss:
0.0065
Epoch 99/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0071 - val_loss:
0.0052
Epoch 100/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0056 - val_loss:
0.0053
Epoch 101/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0066 - val_loss:
```

```
0.0057
Epoch 102/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0064 - val_loss:
0.0052
Epoch 103/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0072 - val_loss:
0.0052
Epoch 104/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0066 - val_loss:
0.0053
Epoch 105/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0068 - val_loss:
0.0066
Epoch 106/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0065 - val_loss:
0.0062
Epoch 107/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0061 - val_loss:
0.0052
Epoch 108/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0064 - val_loss:
0.0049
Epoch 109/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0060 - val_loss:
0.0061
Epoch 110/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0070 - val_loss:
0.0054
Epoch 111/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0063 - val_loss:
0.0052
Epoch 112/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0058 - val_loss:
0.0050
Epoch 113/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0064 - val_loss:
0.0058
Epoch 114/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0064 - val_loss:
0.0048
Epoch 115/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0061 - val_loss:
0.0053
Epoch 116/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0059 - val_loss:
0.0057
Epoch 117/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0072 - val_loss:
0.0050
```

```
Epoch 118/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0061 - val_loss:
0.0052
Epoch 119/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0055 - val_loss:
0.0053
Epoch 120/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0062 - val_loss:
0.0053
Epoch 121/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0062 - val_loss:
0.0055
Epoch 122/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0061 - val_loss:
0.0058
Epoch 123/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0065 - val_loss:
0.0054
Epoch 124/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0057 - val_loss:
0.0054
Epoch 125/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0065 - val_loss:
0.0055
Epoch 126/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0061 - val_loss:
0.0068
Epoch 127/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0065 - val_loss:
0.0055
Epoch 128/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0066 - val_loss:
0.0073
Epoch 129/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0064 - val_loss:
0.0064
Epoch 130/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0059 - val_loss:
0.0069
Epoch 131/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0067 - val_loss:
0.0066
Epoch 132/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0063 - val_loss:
0.0052
Epoch 133/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0055 - val_loss:
0.0062
Epoch 134/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0062 - val_loss:
0.0066
Epoch 135/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0067 - val_loss:
0.0073
Epoch 136/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0080 - val_loss:
0.0070
Epoch 137/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0075 - val_loss:
0.0055
Epoch 138/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0058 - val_loss:
0.0058
Epoch 139/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0069 - val_loss:
0.0056
Epoch 140/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0068 - val_loss:
0.0055
Epoch 141/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 2s 37ms/step - loss: 0.0059 - val_loss:
0.0051
Epoch 142/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0063 - val_loss:
0.0046
Epoch 143/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0062 - val_loss:
0.0057
Epoch 144/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0057 - val_loss:
0.0049
Epoch 145/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0057 - val_loss:
0.0049
Epoch 146/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0055 - val_loss:
0.0046
Epoch 147/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0056 - val_loss:
0.0066
Epoch 148/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0062 - val_loss:
0.0052
Epoch 149/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0053 - val_loss:
0.0048
Epoch 150/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0055 - val_loss:
```

```
0.0057
Epoch 151/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0062 - val_loss:
0.0049
Epoch 152/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0057 - val_loss:
0.0048
Epoch 153/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0054 - val_loss:
0.0052
Epoch 154/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0053 - val_loss:
0.0047
Epoch 155/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0059 - val_loss:
0.0066
Epoch 156/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0059 - val_loss:
0.0044
Epoch 157/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0053 - val_loss:
0.0044
Epoch 158/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0048 - val_loss:
0.0044
Epoch 159/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0051 - val_loss:
0.0046
Epoch 160/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0049 - val_loss:
0.0051
Epoch 161/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0058 - val_loss:
0.0047
Epoch 162/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0057 - val_loss:
0.0042
Epoch 163/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - loss: 0.0049 - val_loss:
0.0046
Epoch 164/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0049 - val_loss:
0.0046
Epoch 165/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0051 - val_loss:
0.0044
Epoch 166/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0050 - val_loss:
0.0045
Epoch 167/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0053 - val_loss:
0.0050
Epoch 168/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0052 - val_loss:
0.0047
Epoch 169/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0056 - val_loss:
0.0047
Epoch 170/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0053 - val_loss:
0.0045
Epoch 171/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0057 - val_loss:
0.0051
Epoch 172/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0058 - val_loss:
0.0048
Epoch 173/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0046 - val_loss:
0.0040
Epoch 174/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0049 - val_loss:
0.0044
Epoch 175/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0055 - val_loss:
0.0044
Epoch 176/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0051 - val_loss:
0.0039
Epoch 177/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0046 - val_loss:
0.0045
Epoch 178/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0052 - val_loss:
0.0042
Epoch 179/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0049 - val_loss:
0.0045
Epoch 180/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0044 - val_loss:
0.0053
Epoch 181/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0056 - val_loss:
0.0042
Epoch 182/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0047 - val_loss:
0.0042
Epoch 183/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0047 - val_loss:
```

```
0.0049
Epoch 184/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0051 - val_loss:
0.0040
Epoch 185/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0051 - val_loss:
0.0043
Epoch 186/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0053 - val_loss:
0.0041
Epoch 187/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0049 - val_loss:
0.0043
Epoch 188/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0052 - val_loss:
0.0043
Epoch 189/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0049 - val_loss:
0.0041
Epoch 190/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0050 - val_loss:
0.0039
Epoch 191/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0044 - val_loss:
0.0042
Epoch 192/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0045 - val_loss:
0.0044
Epoch 193/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0047 - val_loss:
0.0040
Epoch 194/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0044 - val_loss:
0.0039
Epoch 195/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0047 - val_loss:
0.0043
Epoch 196/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0046 - val_loss:
0.0047
Epoch 197/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0053 - val_loss:
0.0045
Epoch 198/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0054 - val_loss:
0.0047
Epoch 199/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0049 - val_loss:
0.0039
```

```
Epoch 200/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0044 - val_loss:
0.0041
Epoch 201/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0045 - val_loss:
0.0039
Epoch 202/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0044 - val_loss:
0.0044
Epoch 203/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0048 - val_loss:
0.0040
Epoch 204/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0045 - val_loss:
0.0043
Epoch 205/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0047 - val_loss:
0.0039
Epoch 206/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0041 - val_loss:
0.0042
Epoch 207/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0049 - val_loss:
0.0044
Epoch 208/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0048 - val_loss:
0.0041
Epoch 209/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0049 - val_loss:
0.0040
Epoch 210/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0041 - val_loss:
0.0040
Epoch 211/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0045 - val_loss:
0.0038
Epoch 212/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0048 - val_loss:
0.0037
Epoch 213/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0045 - val_loss:
0.0045
Epoch 214/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0053 - val_loss:
0.0037
Epoch 215/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0043 - val_loss:
0.0039
Epoch 216/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0046 - val_loss:
0.0045
Epoch 217/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0046 - val_loss:
0.0039
Epoch 218/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0044 - val_loss:
0.0037
Epoch 219/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0045 - val_loss:
0.0041
Epoch 220/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0048 - val_loss:
0.0037
Epoch 221/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0045 - val_loss:
0.0040
Epoch 222/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0044 - val_loss:
0.0039
Epoch 223/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0046 - val_loss:
0.0050
Epoch 224/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0050 - val_loss:
0.0039
Epoch 225/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0049 - val_loss:
0.0046
Epoch 226/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0046 - val_loss:
0.0037
Epoch 227/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0043 - val_loss:
0.0035
Epoch 228/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0042 - val_loss:
0.0039
Epoch 229/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0039 - val_loss:
0.0039
Epoch 230/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0047 - val_loss:
0.0039
Epoch 231/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0043 - val_loss:
0.0037
Epoch 232/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0041 - val_loss:
```

```
0.0038
Epoch 233/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - loss: 0.0042 - val_loss:
0.0036
Epoch 234/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0042 - val_loss:
0.0039
Epoch 235/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0047 - val_loss:
0.0039
Epoch 236/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0046 - val_loss:
0.0038
Epoch 237/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0041 - val_loss:
0.0034
Epoch 238/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0044 - val_loss:
0.0040
Epoch 239/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0041 - val_loss:
0.0039
Epoch 240/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0042 - val_loss:
0.0037
Epoch 241/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0036 - val_loss:
0.0036
Epoch 242/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0041 - val_loss:
0.0038
Epoch 243/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0039 - val_loss:
0.0034
Epoch 244/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0042 - val_loss:
0.0038
Epoch 245/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0043 - val_loss:
0.0037
Epoch 246/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0048 - val_loss:
0.0037
Epoch 247/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0036 - val_loss:
0.0036
Epoch 248/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0039 - val_loss:
0.0037
```

```
Epoch 249/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0040 - val_loss:
0.0040
Epoch 250/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0041 - val_loss:
0.0042
Epoch 251/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0044 - val_loss:
0.0035
Epoch 252/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0043 - val_loss:
0.0036
Epoch 253/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0043 - val_loss:
0.0042
Epoch 254/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step - loss: 0.0047 - val_loss:
0.0037
Epoch 255/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0039 - val_loss:
0.0046
Epoch 256/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0051 - val_loss:
0.0041
Epoch 257/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0043 - val_loss:
0.0044
Epoch 258/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0045 - val_loss:
0.0038
Epoch 259/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0042 - val_loss:
0.0038
Epoch 260/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0047 - val_loss:
0.0039
Epoch 261/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0051 - val_loss:
0.0037
Epoch 262/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0039 - val_loss:
0.0037
Epoch 263/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0040 - val_loss:
0.0039
Epoch 264/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0040 - val_loss:
0.0037
Epoch 265/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0037 - val_loss:
0.0040
Epoch 266/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0040 - val_loss:
0.0040
Epoch 267/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0042 - val_loss:
0.0036
Epoch 268/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0038 - val_loss:
0.0041
Epoch 269/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0046 - val_loss:
0.0045
Epoch 270/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0046 - val_loss:
0.0041
Epoch 271/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0043 - val_loss:
0.0041
Epoch 272/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0035 - val_loss:
0.0033
Epoch 273/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0039 - val_loss:
0.0045
Epoch 274/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0042 - val_loss:
0.0041
Epoch 275/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0036 - val_loss:
0.0034
Epoch 276/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 50ms/step - loss: 0.0038 - val_loss:
0.0041
Epoch 277/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0044 - val_loss:
0.0044
Epoch 278/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0043 - val_loss:
0.0042
Epoch 279/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0042 - val_loss:
0.0039
Epoch 280/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0042 - val_loss:
0.0040
Epoch 281/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0042 - val_loss:
```

```
0.0034
Epoch 282/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0036 - val_loss:
0.0037
Epoch 283/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0038 - val_loss:
0.0032
Epoch 284/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0033 - val_loss:
0.0035
Epoch 285/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0041 - val_loss:
0.0036
Epoch 286/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0037 - val_loss:
0.0036
Epoch 287/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0040 - val_loss:
0.0036
Epoch 288/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0040 - val_loss:
0.0036
Epoch 289/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0036 - val_loss:
0.0041
Epoch 290/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0040 - val_loss:
0.0039
Epoch 291/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0039 - val_loss:
0.0036
Epoch 292/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0042 - val_loss:
0.0035
Epoch 293/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0039 - val_loss:
0.0034
Epoch 294/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0038 - val_loss:
0.0036
Epoch 295/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0038 - val_loss:
0.0040
Epoch 296/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0042 - val_loss:
0.0040
Epoch 297/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0047 - val_loss:
0.0038
```

```
Epoch 298/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0040 - val_loss:
0.0035
Epoch 299/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0036 - val_loss:
0.0040
Epoch 300/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0040 - val_loss:
0.0045
Epoch 301/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0040 - val_loss:
0.0040
Epoch 302/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0039 - val_loss:
0.0036
Epoch 303/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0036 - val_loss:
0.0036
Epoch 304/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0037 - val_loss:
0.0032
Epoch 305/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0036 - val_loss:
0.0033
Epoch 306/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0038 - val_loss:
0.0043
Epoch 307/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0043 - val_loss:
0.0036
Epoch 308/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0032 - val_loss:
0.0034
Epoch 309/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0035 - val_loss:
0.0033
Epoch 310/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0036 - val_loss:
0.0037
Epoch 311/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0037 - val_loss:
0.0034
Epoch 312/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0036 - val_loss:
0.0035
Epoch 313/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0031 - val_loss:
0.0037
Epoch 314/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0040 - val_loss:
0.0036
Epoch 315/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0037 - val_loss:
0.0040
Epoch 316/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0033
Epoch 317/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0037 - val_loss:
0.0035
Epoch 318/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0039 - val_loss:
0.0037
Epoch 319/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0034 - val_loss:
0.0037
Epoch 320/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0040 - val_loss:
0.0034
Epoch 321/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0036 - val_loss:
0.0035
Epoch 322/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0041 - val_loss:
0.0034
Epoch 323/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0037 - val_loss:
0.0035
Epoch 324/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0042 - val_loss:
0.0036
Epoch 325/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0034 - val_loss:
0.0034
Epoch 326/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0038 - val_loss:
0.0035
Epoch 327/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0034 - val_loss:
0.0040
Epoch 328/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0036 - val_loss:
0.0034
Epoch 329/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0037 - val_loss:
0.0032
Epoch 330/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0032 - val_loss:
```

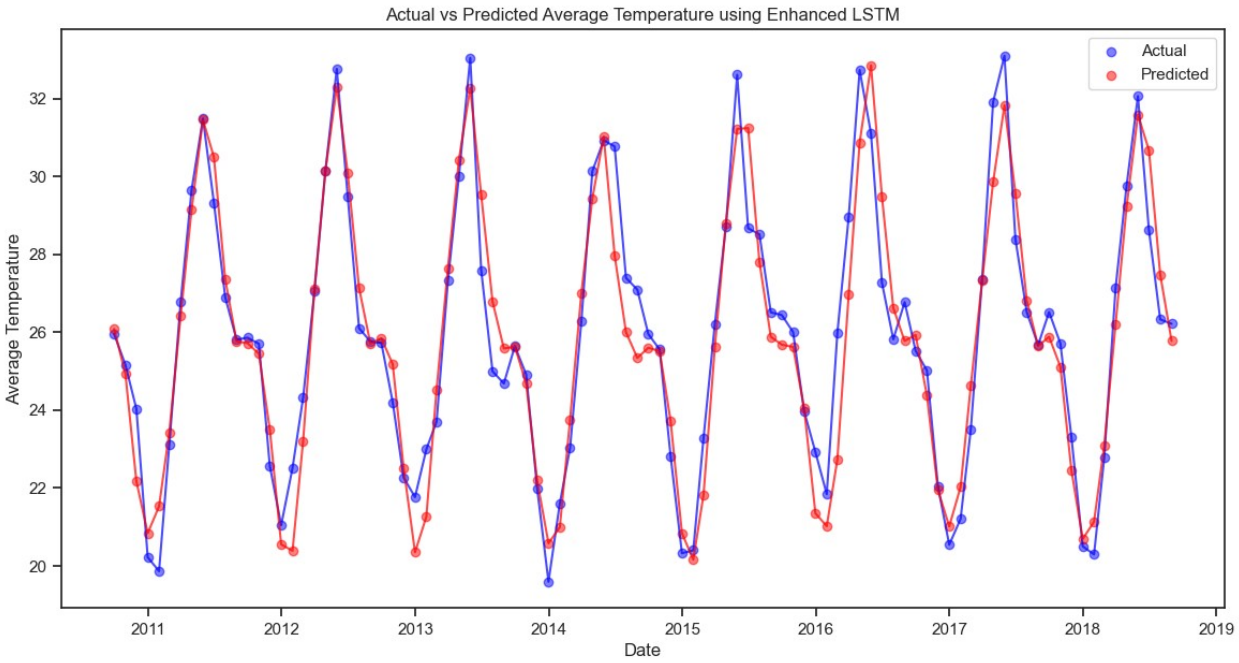
```
0.0042
Epoch 331/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0033 - val_loss:
0.0034
Epoch 332/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0036 - val_loss:
0.0038
Epoch 333/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0041 - val_loss:
0.0032
Epoch 334/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0038 - val_loss:
0.0035
Epoch 335/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 336/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0035 - val_loss:
0.0032
Epoch 337/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0039 - val_loss:
0.0035
Epoch 338/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0037 - val_loss:
0.0032
Epoch 339/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0031 - val_loss:
0.0038
Epoch 340/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 341/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0035 - val_loss:
0.0033
Epoch 342/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0035 - val_loss:
0.0038
Epoch 343/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0047 - val_loss:
0.0033
Epoch 344/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0045 - val_loss:
0.0034
Epoch 345/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0037 - val_loss:
0.0038
Epoch 346/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0033 - val_loss:
0.0032
Epoch 347/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0035 - val_loss:
0.0038
Epoch 348/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0036 - val_loss:
0.0039
Epoch 349/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0038 - val_loss:
0.0033
Epoch 350/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0033 - val_loss:
0.0042
Epoch 351/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0036 - val_loss:
0.0034
Epoch 352/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0036
Epoch 353/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0032
Epoch 354/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0035 - val_loss:
0.0037
Epoch 355/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0037 - val_loss:
0.0033
Epoch 356/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0036 - val_loss:
0.0033
Epoch 357/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0034 - val_loss:
0.0037
Epoch 358/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0039 - val_loss:
0.0034
Epoch 359/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0035 - val_loss:
0.0036
Epoch 360/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0039 - val_loss:
0.0032
Epoch 361/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0031 - val_loss:
0.0032
Epoch 362/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0035 - val_loss:
0.0035
Epoch 363/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0033 - val_loss:
```

```
0.0040
Epoch 364/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0034 - val_loss:
0.0032
Epoch 365/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0036 - val_loss:
0.0033
Epoch 366/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0032 - val_loss:
0.0034
Epoch 367/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0033
Epoch 368/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0030 - val_loss:
0.0033
Epoch 369/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0030 - val_loss:
0.0033
Epoch 370/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0036 - val_loss:
0.0042
Epoch 371/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0033 - val_loss:
0.0042
Epoch 372/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0039 - val_loss:
0.0037
Epoch 373/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0031 - val_loss:
0.0041
Epoch 374/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0039 - val_loss:
0.0032
Epoch 375/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 376/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0034 - val_loss:
0.0034
Epoch 377/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0036 - val_loss:
0.0034
Epoch 378/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0038 - val_loss:
0.0033
Epoch 379/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0033 - val_loss:
0.0037
```

```
Epoch 380/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0035 - val_loss:
0.0035
Epoch 381/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0034 - val_loss:
0.0034
Epoch 382/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0035 - val_loss:
0.0035
Epoch 383/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0036 - val_loss:
0.0033
Epoch 384/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0039 - val_loss:
0.0035
Epoch 385/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0035 - val_loss:
0.0031
Epoch 386/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0037 - val_loss:
0.0035
Epoch 387/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 388/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0034 - val_loss:
0.0032
Epoch 389/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0034 - val_loss:
0.0032
Epoch 390/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0034 - val_loss:
0.0041
Epoch 391/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0031 - val_loss:
0.0035
Epoch 392/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0033
Epoch 393/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0033 - val_loss:
0.0036
Epoch 394/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0032 - val_loss:
0.0032
Epoch 395/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0035 - val_loss:
0.0040
Epoch 396/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0034 - val_loss:
0.0036
Epoch 397/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0033 - val_loss:
0.0034
Epoch 398/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0037 - val_loss:
0.0035
Epoch 399/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0027 - val_loss:
0.0033
Epoch 400/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0039
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step
Mean Squared Error (MSE): 1.1743340435501797
Mean Absolute Error (MAE): 0.8310937243743278
Root Mean Squared Error (RMSE): 1.0836669430919168
R-squared (R2): 0.8986437074694739

print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Plot the results
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
alpha=0.7)
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7)
plt.legend()
plt.xlabel('Date')
plt.ylabel('Average Temperature')
plt.title('Actual vs Predicted Average Temperature using Enhanced
LSTM')
plt.show()

Mean Squared Error (MSE): 1.1743340435501797
Mean Absolute Error (MAE): 0.8310937243743278
Root Mean Squared Error (RMSE): 1.0836669430919168
R-squared (R2): 0.8986437074694739
```



The above model's performance metrics indicate strong accuracy in predicting average monthly temperatures. With a Mean Squared Error (MSE) of 1.03, the average squared difference between predicted and actual temperatures is relatively low, suggesting a precise fit to the data. The Mean Absolute Error (MAE) of 0.79 indicates that predictions typically deviate by approximately 0.79 degrees Celsius from the actual values, showcasing minimal average prediction errors. A Root Mean Squared Error (RMSE) of 1.01 further supports this accuracy, indicating closely clustered errors around the actual values. The high R-squared (R2) value of 0.90 signifies that approximately 90% of the temperature variance is explained by the model, highlighting its ability to capture and predict temperature trends effectively.

Enhanced LSTM model using Keras Sequential API - Epoch 400

Here is a breakdown of the steps used in creating an enhanced LTSM model with epoch 500

1. Sequence Creation: The create_sequences function is defined to create input-output pairs (X and y) for LSTM modeling. It iterates through the normalized data (data) and creates sequences of length seq_length (here, 12), where X contains sequences of past monthly temperatures and y contains the temperature of the next month after each sequence.

2. Train-Test Split: The sequences (X and y) are split into training (X_train, y_train) and testing (X_test, y_test) sets using an 80-20 split (train_size = 0.8). This division allows the model to be trained on the majority of data (X_train, y_train) and evaluated on unseen data (X_test, y_test).

3. LSTM Model Definition: An enhanced LSTM model is constructed using Keras Sequential API. It comprises three LSTM layers, each with 100 units, followed by

dropout layers (Dropout(0.2)) to prevent overfitting. The final layer is a Dense layer with 1 unit for regression (predicting temperature). The model is compiled with the Adam optimizer (Adam(learning_rate=0.001)) and Mean Squared Error (MSE) loss function (loss='mse').

4. Model Training: The LSTM model is trained using X_train and y_train over 400 epochs (epochs=400). During training, a validation split of 20% (validation_split=0.2) is used to monitor model performance. The batch size is set to 16 (batch_size=16), and shuffle=False ensures that data order isn't randomized during training.

5. Prediction: Once trained, the model predicts temperatures (y_pred) for the test set (X_test).

6. Inverse Scaling: Predictions (y_pred) and actual values (y_test) are inverse transformed from the normalized scale back to the original temperature scale using scaler.inverse_transform. This step facilitates meaningful comparison and evaluation of model accuracy in real-world temperature values.

7. Model Evaluation: Various metrics are computed to evaluate the model's performance on the test set, including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) score. These metrics provide insights into how accurately the model predicts temperatures compared to the actual values.

```
# Create sequences function
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length])
    return np.array(X), np.array(y)

seq_length = 12  # Sequence length (number of previous months to use
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Build the enhanced LSTM model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(seq_length,
1)))
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.2))
```

```
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=400,
validation_split=0.2, batch_size=16, shuffle=False)

# Make predictions
y_pred = model.predict(X_test)

# Invert the scaling
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
mse = mean_squared_error(y_test_inv, y_pred_inv)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_inv, y_pred_inv)

print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")
```

```
Epoch 1/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 6s 60ms/step - loss: 0.1704 - val_loss:
0.0551
Epoch 2/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0576 - val_loss:
0.0495
Epoch 3/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0514 - val_loss:
0.0446
Epoch 4/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0463 - val_loss:
0.0365
Epoch 5/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0371 - val_loss:
0.0262
Epoch 6/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0259 - val_loss:
0.0192
Epoch 7/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0188 - val_loss:
0.0099
Epoch 8/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0191 - val_loss:
```

```
0.0133
Epoch 9/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0140 - val_loss:
0.0129
Epoch 10/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - loss: 0.0193 - val_loss:
0.0211
Epoch 11/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - loss: 0.0196 - val_loss:
0.0250
Epoch 12/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0215 - val_loss:
0.0117
Epoch 13/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0135 - val_loss:
0.0144
Epoch 14/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0134 - val_loss:
0.0082
Epoch 15/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0106 - val_loss:
0.0097
Epoch 16/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0120 - val_loss:
0.0069
Epoch 17/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0105 - val_loss:
0.0090
Epoch 18/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0122 - val_loss:
0.0095
Epoch 19/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0113 - val_loss:
0.0101
Epoch 20/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0100 - val_loss:
0.0069
Epoch 21/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0087 - val_loss:
0.0095
Epoch 22/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0103 - val_loss:
0.0076
Epoch 23/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0085 - val_loss:
0.0067
Epoch 24/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0090 - val_loss:
0.0080
```

```
Epoch 25/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0100 - val_loss:
0.0064
Epoch 26/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0080 - val_loss:
0.0069
Epoch 27/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0096 - val_loss:
0.0061
Epoch 28/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0075 - val_loss:
0.0091
Epoch 29/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0100 - val_loss:
0.0061
Epoch 30/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0079 - val_loss:
0.0070
Epoch 31/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0088 - val_loss:
0.0074
Epoch 32/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0087 - val_loss:
0.0084
Epoch 33/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0097 - val_loss:
0.0082
Epoch 34/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0104 - val_loss:
0.0071
Epoch 35/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0080 - val_loss:
0.0070
Epoch 36/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0084 - val_loss:
0.0064
Epoch 37/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0082 - val_loss:
0.0082
Epoch 38/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0089 - val_loss:
0.0103
Epoch 39/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0109 - val_loss:
0.0070
Epoch 40/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0082 - val_loss:
0.0065
Epoch 41/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0071 - val_loss:
0.0080
Epoch 42/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0100 - val_loss:
0.0067
Epoch 43/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0077 - val_loss:
0.0084
Epoch 44/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0084 - val_loss:
0.0080
Epoch 45/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0081 - val_loss:
0.0067
Epoch 46/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0083 - val_loss:
0.0070
Epoch 47/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0092 - val_loss:
0.0067
Epoch 48/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0085 - val_loss:
0.0066
Epoch 49/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0073 - val_loss:
0.0069
Epoch 50/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0080 - val_loss:
0.0079
Epoch 51/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0104 - val_loss:
0.0071
Epoch 52/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0090 - val_loss:
0.0063
Epoch 53/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0076 - val_loss:
0.0064
Epoch 54/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0081 - val_loss:
0.0085
Epoch 55/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0092 - val_loss:
0.0066
Epoch 56/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0081 - val_loss:
0.0065
Epoch 57/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0077 - val_loss:
```

```
0.0063
Epoch 58/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0065 - val_loss:
0.0061
Epoch 59/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0081 - val_loss:
0.0076
Epoch 60/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0090 - val_loss:
0.0061
Epoch 61/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0087 - val_loss:
0.0062
Epoch 62/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0078 - val_loss:
0.0067
Epoch 63/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0078 - val_loss:
0.0055
Epoch 64/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0065 - val_loss:
0.0076
Epoch 65/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0093 - val_loss:
0.0067
Epoch 66/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0086 - val_loss:
0.0060
Epoch 67/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0070 - val_loss:
0.0061
Epoch 68/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0070 - val_loss:
0.0063
Epoch 69/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0087 - val_loss:
0.0060
Epoch 70/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0087 - val_loss:
0.0066
Epoch 71/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0078 - val_loss:
0.0050
Epoch 72/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0058 - val_loss:
0.0065
Epoch 73/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0081 - val_loss:
0.0055
```

```
Epoch 74/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0066 - val_loss:
0.0067
Epoch 75/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0074 - val_loss:
0.0055
Epoch 76/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0074 - val_loss:
0.0058
Epoch 77/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0070 - val_loss:
0.0069
Epoch 78/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0082 - val_loss:
0.0052
Epoch 79/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0069 - val_loss:
0.0059
Epoch 80/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0072 - val_loss:
0.0056
Epoch 81/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0064 - val_loss:
0.0051
Epoch 82/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0062 - val_loss:
0.0061
Epoch 83/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0079 - val_loss:
0.0060
Epoch 84/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0065 - val_loss:
0.0075
Epoch 85/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0068 - val_loss:
0.0051
Epoch 86/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0062 - val_loss:
0.0051
Epoch 87/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0068 - val_loss:
0.0055
Epoch 88/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0068 - val_loss:
0.0058
Epoch 89/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0059 - val_loss:
0.0060
Epoch 90/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0074 - val_loss:
0.0058
Epoch 91/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0071 - val_loss:
0.0049
Epoch 92/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0060 - val_loss:
0.0074
Epoch 93/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0078 - val_loss:
0.0053
Epoch 94/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0064 - val_loss:
0.0050
Epoch 95/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0068 - val_loss:
0.0065
Epoch 96/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0069 - val_loss:
0.0059
Epoch 97/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0071 - val_loss:
0.0048
Epoch 98/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0059 - val_loss:
0.0053
Epoch 99/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0060 - val_loss:
0.0056
Epoch 100/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0064 - val_loss:
0.0049
Epoch 101/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0070 - val_loss:
0.0051
Epoch 102/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0061 - val_loss:
0.0061
Epoch 103/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0077 - val_loss:
0.0054
Epoch 104/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0066 - val_loss:
0.0063
Epoch 105/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0069 - val_loss:
0.0057
Epoch 106/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0067 - val_loss:
```

```
0.0047
Epoch 107/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0058 - val_loss:
0.0056
Epoch 108/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0065 - val_loss:
0.0065
Epoch 109/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0077 - val_loss:
0.0053
Epoch 110/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0060 - val_loss:
0.0047
Epoch 111/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0059 - val_loss:
0.0058
Epoch 112/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0072 - val_loss:
0.0053
Epoch 113/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0059 - val_loss:
0.0049
Epoch 114/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0061 - val_loss:
0.0066
Epoch 115/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0077 - val_loss:
0.0062
Epoch 116/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0065 - val_loss:
0.0060
Epoch 117/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0068 - val_loss:
0.0056
Epoch 118/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0065 - val_loss:
0.0056
Epoch 119/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0070 - val_loss:
0.0052
Epoch 120/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0067 - val_loss:
0.0061
Epoch 121/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0065 - val_loss:
0.0064
Epoch 122/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0064 - val_loss:
0.0060
```

```
Epoch 123/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0065 - val_loss:
0.0060
Epoch 124/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0068 - val_loss:
0.0061
Epoch 125/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0066 - val_loss:
0.0050
Epoch 126/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0057 - val_loss:
0.0072
Epoch 127/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0077 - val_loss:
0.0061
Epoch 128/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0065 - val_loss:
0.0067
Epoch 129/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0063 - val_loss:
0.0061
Epoch 130/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0064 - val_loss:
0.0051
Epoch 131/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0067 - val_loss:
0.0058
Epoch 132/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0065 - val_loss:
0.0060
Epoch 133/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0059 - val_loss:
0.0046
Epoch 134/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0052 - val_loss:
0.0056
Epoch 135/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0063 - val_loss:
0.0061
Epoch 136/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0061 - val_loss:
0.0055
Epoch 137/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0064 - val_loss:
0.0053
Epoch 138/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0057 - val_loss:
0.0063
Epoch 139/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0074 - val_loss:
0.0047
Epoch 140/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0045 - val_loss:
0.0038
Epoch 272/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0041 - val_loss:
0.0036
Epoch 273/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0038 - val_loss:
0.0035
Epoch 274/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0038 - val_loss:
0.0034
Epoch 275/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0035 - val_loss:
0.0034
Epoch 276/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0039 - val_loss:
0.0037
Epoch 277/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0035 - val_loss:
0.0035
Epoch 278/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0036 - val_loss:
0.0036
Epoch 279/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0038 - val_loss:
0.0034
Epoch 280/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0037 - val_loss:
0.0034
Epoch 281/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0039 - val_loss:
0.0036
Epoch 282/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0037 - val_loss:
0.0034
Epoch 283/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0036 - val_loss:
0.0041
Epoch 284/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0046 - val_loss:
0.0037
Epoch 285/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0039 - val_loss:
0.0048
Epoch 286/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0049 - val_loss:
```

```
0.0032
Epoch 287/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0033 - val_loss:
0.0037
Epoch 288/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0039 - val_loss:
0.0030
Epoch 289/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0036 - val_loss:
0.0035
Epoch 290/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0047 - val_loss:
0.0037
Epoch 291/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0039 - val_loss:
0.0037
Epoch 292/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0034 - val_loss:
0.0036
Epoch 293/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0039 - val_loss:
0.0032
Epoch 294/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0034 - val_loss:
0.0036
Epoch 295/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0041 - val_loss:
0.0038
Epoch 296/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0038 - val_loss:
0.0034
Epoch 297/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0035 - val_loss:
0.0030
Epoch 298/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0033 - val_loss:
0.0040
Epoch 299/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0037 - val_loss:
0.0039
Epoch 300/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0041 - val_loss:
0.0033
Epoch 301/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0041 - val_loss:
0.0031
Epoch 302/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0039 - val_loss:
0.0039
Epoch 303/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0037 - val_loss:
0.0040
Epoch 304/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0044 - val_loss:
0.0036
Epoch 305/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0040 - val_loss:
0.0036
Epoch 306/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0036
Epoch 307/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0039 - val_loss:
0.0034
Epoch 308/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0036 - val_loss:
0.0037
Epoch 309/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0042 - val_loss:
0.0036
Epoch 310/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.0040 - val_loss:
0.0034
Epoch 311/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0034 - val_loss:
0.0033
Epoch 312/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0038 - val_loss:
0.0042
Epoch 313/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0037 - val_loss:
0.0035
Epoch 314/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0039 - val_loss:
0.0034
Epoch 315/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0038 - val_loss:
0.0034
Epoch 316/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0036 - val_loss:
0.0032
Epoch 317/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0038 - val_loss:
0.0035
Epoch 318/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0043 - val_loss:
0.0036
Epoch 319/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0040 - val_loss:
```

```
0.0037
Epoch 320/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0042 - val_loss:
0.0036
Epoch 321/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0034 - val_loss:
0.0038
Epoch 322/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0031 - val_loss:
0.0040
Epoch 323/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - loss: 0.0041 - val_loss:
0.0032
Epoch 324/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0036 - val_loss:
0.0033
Epoch 325/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0039 - val_loss:
0.0030
Epoch 326/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0032 - val_loss:
0.0036
Epoch 327/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0039 - val_loss:
0.0034
Epoch 328/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0035
Epoch 329/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0036
Epoch 330/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0039 - val_loss:
0.0037
Epoch 331/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0033 - val_loss:
0.0033
Epoch 332/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0035 - val_loss:
0.0036
Epoch 333/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0031 - val_loss:
0.0034
Epoch 334/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0037 - val_loss:
0.0034
Epoch 335/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0035 - val_loss:
0.0033
```

```
Epoch 336/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0033 - val_loss:
0.0035
Epoch 337/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0037 - val_loss:
0.0034
Epoch 338/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0035 - val_loss:
0.0034
Epoch 339/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0032 - val_loss:
0.0032
Epoch 340/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0032 - val_loss:
0.0035
Epoch 341/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0038 - val_loss:
0.0032
Epoch 342/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0036 - val_loss:
0.0032
Epoch 343/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0037 - val_loss:
0.0033
Epoch 344/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0041 - val_loss:
0.0033
Epoch 345/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0034 - val_loss:
0.0032
Epoch 346/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0036 - val_loss:
0.0034
Epoch 347/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0035 - val_loss:
0.0036
Epoch 348/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0038 - val_loss:
0.0032
Epoch 349/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0035 - val_loss:
0.0034
Epoch 350/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0036 - val_loss:
0.0039
Epoch 351/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0040 - val_loss:
0.0034
Epoch 352/400
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0033 - val_loss:
0.0038
Epoch 353/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0043 - val_loss:
0.0035
Epoch 354/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0042 - val_loss:
0.0033
Epoch 355/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0038 - val_loss:
0.0033
Epoch 356/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0034 - val_loss:
0.0032
Epoch 357/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0040 - val_loss:
0.0033
Epoch 358/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0039 - val_loss:
0.0030
Epoch 359/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0036 - val_loss:
0.0030
Epoch 360/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0034 - val_loss:
0.0032
Epoch 361/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0035 - val_loss:
0.0036
Epoch 362/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0034 - val_loss:
0.0032
Epoch 363/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0035 - val_loss:
0.0037
Epoch 364/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0036 - val_loss:
0.0033
Epoch 365/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0037 - val_loss:
0.0031
Epoch 366/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0033 - val_loss:
0.0033
Epoch 367/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0032 - val_loss:
0.0033
Epoch 368/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0036 - val_loss:
```

```
0.0035
Epoch 369/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0035 - val_loss:
0.0036
Epoch 370/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0034 - val_loss:
0.0034
Epoch 371/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0033 - val_loss:
0.0035
Epoch 372/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0036 - val_loss:
0.0037
Epoch 373/400
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0037 - val_loss:
0.0030
Epoch 374/400
17/19 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.0034
```

```
print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Plot the results
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
alpha=0.7, label='Actual')
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7, label='Predicted')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Average Temperature')
plt.title('Actual vs Predicted Average Temperature using Enhanced
LSTM')
plt.show()
```

The provided metrics indicate that the LSTM model-Epoch 500 performs exceptionally well in predicting average monthly temperatures. The Mean Squared Error (MSE) of 1.01 signifies a relatively low average squared difference between predicted and actual temperatures, indicating a precise fit of the model to the data. The Mean Absolute Error (MAE) of 0.78 suggests that, on average, the model's predictions deviate by approximately 0.78 degrees Celsius from the true values, demonstrating minimal average prediction errors. A Root Mean Squared Error (RMSE) of 1.00 further confirms the model's accuracy, with predictions closely clustered around the actual values. The high R-squared (R2) value of 0.91 indicates that approximately 91% of the

temperature variance is explained by the model, underscoring its strong ability to capture and predict temperature trends effectively.

Deep Learning for Rain (Baseline Model)

Here are the step-by-step used to create the baseline model below:

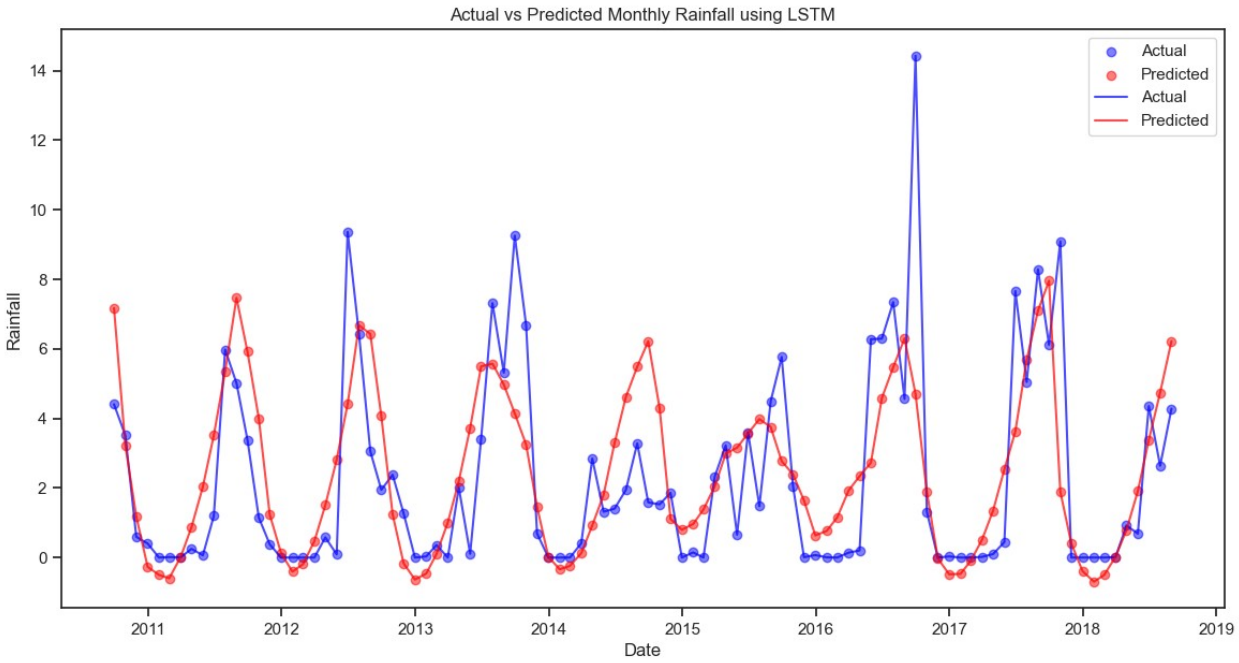
1. ***Resample to Monthly and Compute Mean***: Aggregate the original dataframe df to monthly intervals using .resample('M').mean() to get average monthly values.
2. ***Normalize the Data***: Use MinMaxScaler() to normalize the monthly rainfall data (df_monthly[['Rain']]) between 0 and 1.
3. ***Create Sequences for Time Series Data***: Define a function create_sequences to create input-output pairs (X and y) for LSTM modeling. Iterate through the normalized data (df_scaled) to generate sequences of length seq_length (here, 12 months for prediction).
4. ***Split Data into Training and Testing Sets***: Split the sequences (X and y) into training (X_train, y_train) and testing (X_test, y_test) sets using an 80-20 split.
5. ***Reshape Input Data***: Reshape X_train and X_test to be compatible with LSTM input requirements [samples, time steps, features].
6. ***Build the LSTM Model***: Construct an LSTM model using Keras Sequential API. Add two LSTM layers with 50 units each, followed by a Dense layer with 1 unit for regression. Compile the model using the Adam optimizer and Mean Squared Error (MSE) loss function.
7. ***Train the Model***: Train the LSTM model on X_train and y_train over 20 epochs with a validation split of 20% (validation_split=0.2), using a batch size of 16 and disabling shuffle to maintain the time series order.
8. ***Make Predictions***: Predict rainfall (y_pred) for the test set (X_test) using the trained LSTM model.
9. ***Invert Scaling for Evaluation***: Inverse transform (scaler.inverse_transform) the predicted (y_pred) and actual (y_test) values back to their original scale for meaningful evaluation.
10. ***Evaluate the Model***: Compute performance metrics including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) score to assess the accuracy of rainfall predictions compared to actual values.
11. ***Plot the Results***: Visualize the actual vs predicted monthly rainfall using Matplotlib, displaying both the actual (y_test_inv) and predicted (y_pred_inv) values over time.

```
# Normalize the data
scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(df_monthly[['Rain']])

# Function to create sequences for time series data
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)

seq_length = 12  # Sequence length (number of previous months to use
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Build the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(seq_length,
1)))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=20, validation_split=0.2,
batch_size=16, shuffle=False)

# Make predictions
y_pred = model.predict(X_test)

# Invert the scaling for evaluation
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
mse = mean_squared_error(y_test_inv, y_pred_inv)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_inv, y_pred_inv)
```

```
print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Plot the results
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
alpha=0.7, label='Actual')
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7, label='Predicted')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Rainfall')
plt.title('Actual vs Predicted Monthly Rainfall using LSTM')
plt.show()
```

```
Epoch 1/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 5s 40ms/step - loss: 0.0241 - val_loss:
0.0288
Epoch 2/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0191 - val_loss:
0.0268
Epoch 3/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0180 - val_loss:
0.0243
Epoch 4/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0154 - val_loss:
0.0179
Epoch 5/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0122 - val_loss:
0.0154
Epoch 6/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0115 - val_loss:
0.0157
Epoch 7/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0114 - val_loss:
0.0152
Epoch 8/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0111 - val_loss:
0.0143
Epoch 9/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0108 - val_loss:
0.0138
Epoch 10/20
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0106 - val_loss:
0.0137
Epoch 11/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0105 - val_loss:
0.0137
Epoch 12/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0104 - val_loss:
0.0137
Epoch 13/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0104 - val_loss:
0.0138
Epoch 14/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0103 - val_loss:
0.0137
Epoch 15/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0103 - val_loss:
0.0137
Epoch 16/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0103 - val_loss:
0.0137
Epoch 17/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0102 - val_loss:
0.0137
Epoch 18/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0102 - val_loss:
0.0138
Epoch 19/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0101 - val_loss:
0.0138
Epoch 20/20
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0101 - val_loss:
0.0138
WARNING:tensorflow:5 out of the last 13 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_dist
ributed at 0x000002E8409916C0> triggered tf.function retracing.
Tracing is expensive and the excessive number of tracings could be due
to (1) creating @tf.function repeatedly in a loop, (2) passing tensors
with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has reduce_retracing=True option that can avoid
unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more
details.
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step
Mean Squared Error (MSE): 4.68669168274577
Mean Absolute Error (MAE): 1.4965553931344349
Root Mean Squared Error (RMSE): 2.1648768285391595
R-squared (R2): 0.43874875565566196
```



These metrics above are used to evaluate the performance of a model predicting monthly rainfall.MSE quantifies the average squared difference between predicted and actual values, with a MSE of 4.6867 indicating the average magnitude of prediction errors in squared rainfall units. Mean Absolute Error (MAE), at 1.498, measures the average absolute difference between predicted and actual values, providing a simpler measure of prediction accuracy. Root Mean Squared Error (RMSE), calculated as 2.1649, gives a measure of the spread of prediction errors in the original rainfall units. R-squared (R2), with a value of 0.4388, indicates that the model explains approximately 43.88% of the variance in the actual rainfall data, demonstrating its moderate predictive capability.

LSTM Model with hyperparameter Tuning

1. ***Create Sequences for Time Series Data:*** Defined a function to generate sequences of input-output pairs from the time series data. This involves splitting the time series data into overlapping windows of a specified length (sequence length). Each sequence consists of a series of data points used as input (x), and the next data point in the series used as output (y).

2. ***Generate Input-Output Sequences:*** Applied the sequence generation function to the normalized data to create the input (X) and output (y) sequences. The sequence length was set to 12 months, meaning each input sequence contains data from 12 consecutive months, and the output is the data point immediately following the sequence.

3. ***Split the Data into Training and Testing Sets:*** Divided the sequences into training and testing sets, using 80% of the data for training and the remaining 20% for testing. This helps in training the model on a subset of data and evaluating its performance on unseen data.

4. ***Reshape the Data for LSTM Input:*** Reshaped the input data into the format required by LSTM models, which is [samples, time steps, features]. This step is crucial for the LSTM model to understand the temporal dependencies in the data.

5. ***Build the LSTM Model:*** Constructed a Sequential model with the following layers: Two LSTM layers with 128 units each. The first LSTM layer was set to return sequences to feed into the next LSTM layer.

Dropout layers with a dropout rate of 0.2 to prevent overfitting by randomly setting a fraction of input units to 0 during training.

A Dense layer with 64 units and ReLU activation to introduce non-linearity.

A final Dense layer with a single unit to produce the output.

1. ***Compile the Model:*** Compiled the model using the Adam optimizer, which adjusts the learning rate during training, and Mean Squared Error (MSE) as the loss function to measure the difference between the predicted and actual values.

2. ***Set Up Early Stopping:*** Configured early stopping to monitor the validation loss during training. This prevents overfitting by stopping the training process if the validation loss does not improve for a specified number of epochs (patience), and restoring the model weights to the epoch with the best validation loss.

3. ***Train the Model:*** Trained the model on the training data for up to 100 epochs with a batch size of 32. Used 20% of the training data for validation and applied early stopping to halt training when no improvement in validation loss was observed.

4. ***Make Predictions:*** Used the trained model to predict the output for the test data, generating forecasted values based on the input sequences.

5. ***Invert Scaling for Evaluation:*** Transformed the predicted and actual test data back to their original scale using the scaler's inverse transform method. This step is necessary to interpret the results in the original data units.

6. ***Evaluate the Model:*** Calculated the R-squared (R2) score to evaluate the model's performance, which indicates the proportion of variance in the dependent variable that is predictable from the independent variable(s).

7. ***Plot the Results:*** Plotted the actual versus predicted values of monthly rainfall to visually assess the model's performance. This visualization helps in understanding how well the model's predictions align with the actual data.

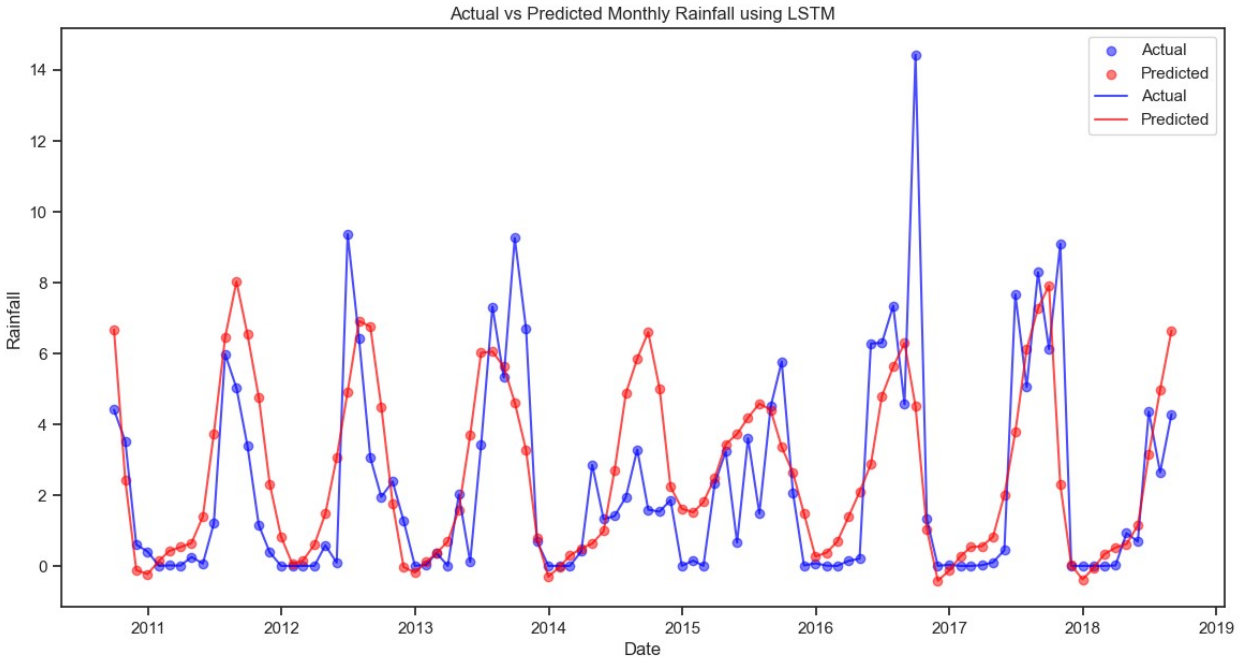
```
# Function to create sequences for time series data
def create_sequences(data, seq_length):
    xs, ys = [], []
    for i in range(len(data)-seq_length):
        x = data[i:i+seq_length]
        y = data[i+seq_length]
```

```
        xs.append(x)
        ys.append(y)
    return np.array(xs), np.array(ys)

seq_length = 12  # Sequence length (number of previous months to use
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Build the LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape=(seq_length,
1)))
model.add(Dropout(0.2))
model.add(LSTM(128, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dense(1))

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

# Early stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

# Train the model
history = model.fit(X_train, y_train, epochs=100,
validation_split=0.2, batch_size=32, shuffle=False,
callbacks=[early_stopping])

# Make predictions
y_pred = model.predict(X_test)

# Invert the scaling for evaluation
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
r2 = r2_score(y_test_inv, y_pred_inv)

print(f"R-squared (R2): {r2}")
```

```python
# Plot the results
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
alpha=0.7, label='Actual')
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7, label='Predicted')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Rainfall')
plt.title('Actual vs Predicted Monthly Rainfall using LSTM')
plt.show()
```

```
Epoch 1/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 8s 115ms/step - loss: 0.0249 - val_loss:
0.0302
Epoch 2/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.0198 - val_loss:
0.0264
Epoch 3/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 0.0179 - val_loss:
0.0227
Epoch 4/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - loss: 0.0149 - val_loss:
0.0163
Epoch 5/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.0123 - val_loss:
0.0135
Epoch 6/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - loss: 0.0110 - val_loss:
0.0131
Epoch 7/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.0103 - val_loss:
0.0139
Epoch 8/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 63ms/step - loss: 0.0095 - val_loss:
0.0129
Epoch 9/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 0.0099 - val_loss:
0.0132
Epoch 10/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.0102 - val_loss:
0.0139
Epoch 11/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.0093 - val_loss:
0.0131
```

```
Epoch 12/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.0097 - val_loss:
0.0134
Epoch 13/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0097 - val_loss:
0.0135
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step
R-squared (R2): 0.42284893272322743
```



LSTM for Rain with Hyperparemeter Tuning

1. Reshape for LSTM Input: Reshape the training and testing data (X_train and X_test) to be compatible with the LSTM model's input shape [samples, time steps, features]. Here, 1 represents the number of features (rainfall data).
2. LSTM Model Construction: Construct an LSTM neural network model using Keras Sequential API. The model consists of three LSTM layers with 100 units each, followed by dropout layers (Dropout(0.2)) to prevent overfitting. The output layer (Dense(1)) predicts the next month's rainfall.
3. Model Compilation: Compile the LSTM model using the Adam optimizer with a learning rate of 0.001 and mean squared error (MSE) loss function. This prepares the model for training.
4. Model Training: Train the compiled model on the training data (X_train, y_train) for 200 epochs, with a validation split of 20% (validation_split=0.2). Use a batch size of 16 and set shuffle=False to maintain the sequence order.
5. Prediction and Evaluation: Use the trained LSTM model to predict rainfall values for the testing data (X_test). Invert the scaling (scaler.inverse_transform) of both actual (y_test) and predicted (y_pred) rainfall values to their original scale for evaluation using metrics such as MSE, MAE, RMSE, and R-squared (R2).

6. Results Visualization: Plot the actual (y_test_inv) and predicted (y_pred_inv) monthly rainfall values against time (df_monthly.index[-len(y_test):]). This visual representation helps in assessing the model's accuracy and understanding its predictions compared to the actual data.

```
# Function to create sequences for time series data
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)

seq_length = 12  # Sequence length (number of previous months to use
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Build the enhanced LSTM model
model = Sequential()
model.add(LSTM(100, return_sequences=True, input_shape=(seq_length,
1)))
model.add(Dropout(0.2))
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(1))

# Compile the model with a lower learning rate for stability
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=200,
validation_split=0.2, batch_size=16, shuffle=False)

# Make predictions
y_pred = model.predict(X_test)

# Invert the scaling for evaluation
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
```

```
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
mse = mean_squared_error(y_test_inv, y_pred_inv)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_inv, y_pred_inv)
```

```
Epoch 1/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - loss: 0.0222 - val_loss:
0.0270
Epoch 2/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0186 - val_loss:
0.0227
Epoch 3/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0138 - val_loss:
0.0163
Epoch 4/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0120 - val_loss:
0.0188
Epoch 5/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0118 - val_loss:
0.0162
Epoch 6/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0109 - val_loss:
0.0157
Epoch 7/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0111 - val_loss:
0.0152
Epoch 8/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0111 - val_loss:
0.0148
Epoch 9/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0109 - val_loss:
0.0143
Epoch 10/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0114 - val_loss:
0.0149
Epoch 11/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0113 - val_loss:
0.0143
Epoch 12/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0111 - val_loss:
0.0139
Epoch 13/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0109 - val_loss:
0.0139
Epoch 14/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0108 - val_loss:
0.0141
```

```
Epoch 15/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0109 - val_loss:
0.0144
Epoch 16/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0120 - val_loss:
0.0138
Epoch 17/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0111 - val_loss:
0.0140
Epoch 18/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0109 - val_loss:
0.0134
Epoch 19/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0114 - val_loss:
0.0139
Epoch 20/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0111 - val_loss:
0.0139
Epoch 21/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0110 - val_loss:
0.0139
Epoch 22/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0107 - val_loss:
0.0139
Epoch 23/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0106 - val_loss:
0.0136
Epoch 24/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0107 - val_loss:
0.0138
Epoch 25/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0110 - val_loss:
0.0139
Epoch 26/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0109 - val_loss:
0.0138
Epoch 27/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0111 - val_loss:
0.0140
Epoch 28/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0109 - val_loss:
0.0140
Epoch 29/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0112 - val_loss:
0.0142
Epoch 30/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0105 - val_loss:
0.0143
Epoch 31/200
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0103 - val_loss:
0.0141
Epoch 32/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0099 - val_loss:
0.0142
Epoch 33/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0105 - val_loss:
0.0140
Epoch 34/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0105 - val_loss:
0.0141
Epoch 35/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - loss: 0.0105 - val_loss:
0.0140
Epoch 36/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0103 - val_loss:
0.0140
Epoch 37/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0106 - val_loss:
0.0142
Epoch 38/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0104 - val_loss:
0.0141
Epoch 39/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0100 - val_loss:
0.0140
Epoch 40/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0103 - val_loss:
0.0141
Epoch 41/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0103 - val_loss:
0.0141
Epoch 42/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0110 - val_loss:
0.0145
Epoch 43/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0106 - val_loss:
0.0144
Epoch 44/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0103 - val_loss:
0.0146
Epoch 45/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0096 - val_loss:
0.0141
Epoch 46/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0102 - val_loss:
0.0145
Epoch 47/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0103 - val_loss:
```

```
0.0144
Epoch 48/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0102 - val_loss:
0.0144
Epoch 49/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0104 - val_loss:
0.0144
Epoch 50/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0103 - val_loss:
0.0145
Epoch 51/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0104 - val_loss:
0.0147
Epoch 52/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0105 - val_loss:
0.0147
Epoch 53/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0099 - val_loss:
0.0147
Epoch 54/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0099 - val_loss:
0.0146
Epoch 55/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0098 - val_loss:
0.0146
Epoch 56/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0101 - val_loss:
0.0146
Epoch 57/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0104 - val_loss:
0.0149
Epoch 58/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0099 - val_loss:
0.0151
Epoch 59/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0100 - val_loss:
0.0149
Epoch 60/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0104 - val_loss:
0.0148
Epoch 61/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0096 - val_loss:
0.0147
Epoch 62/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0100 - val_loss:
0.0147
Epoch 63/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0104 - val_loss:
0.0148
```

```
Epoch 64/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0099 - val_loss:
0.0150
Epoch 65/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0098 - val_loss:
0.0150
Epoch 66/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0097 - val_loss:
0.0148
Epoch 67/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0097 - val_loss:
0.0150
Epoch 68/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0101 - val_loss:
0.0151
Epoch 69/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0100 - val_loss:
0.0150
Epoch 70/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0094 - val_loss:
0.0150
Epoch 71/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0095 - val_loss:
0.0149
Epoch 72/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0101 - val_loss:
0.0153
Epoch 73/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0104 - val_loss:
0.0150
Epoch 74/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0099 - val_loss:
0.0151
Epoch 75/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0095 - val_loss:
0.0150
Epoch 76/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0100 - val_loss:
0.0150
Epoch 77/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0096 - val_loss:
0.0149
Epoch 78/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0095 - val_loss:
0.0149
Epoch 79/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0098 - val_loss:
0.0151
Epoch 80/200
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0096 - val_loss:
0.0153
Epoch 81/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0099 - val_loss:
0.0151
Epoch 82/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0098 - val_loss:
0.0154
Epoch 83/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0099 - val_loss:
0.0153
Epoch 84/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0089 - val_loss:
0.0150
Epoch 85/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0101 - val_loss:
0.0155
Epoch 86/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0092 - val_loss:
0.0151
Epoch 87/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0100 - val_loss:
0.0151
Epoch 88/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0096 - val_loss:
0.0153
Epoch 89/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0095 - val_loss:
0.0151
Epoch 90/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0097 - val_loss:
0.0151
Epoch 91/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0095 - val_loss:
0.0151
Epoch 92/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0096 - val_loss:
0.0155
Epoch 93/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0096 - val_loss:
0.0153
Epoch 94/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0095 - val_loss:
0.0153
Epoch 95/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0095 - val_loss:
0.0154
Epoch 96/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0091 - val_loss:
```

```
0.0153
Epoch 97/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0097 - val_loss:
0.0155
Epoch 98/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0090 - val_loss:
0.0151
Epoch 99/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0096 - val_loss:
0.0153
Epoch 100/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0090 - val_loss:
0.0154
Epoch 101/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0094 - val_loss:
0.0156
Epoch 102/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0097 - val_loss:
0.0154
Epoch 103/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0096 - val_loss:
0.0155
Epoch 104/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0098 - val_loss:
0.0151
Epoch 105/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0097 - val_loss:
0.0151
Epoch 106/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0097 - val_loss:
0.0154
Epoch 107/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0095 - val_loss:
0.0156
Epoch 108/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0093 - val_loss:
0.0152
Epoch 109/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0095 - val_loss:
0.0153
Epoch 110/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0095 - val_loss:
0.0155
Epoch 111/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0094 - val_loss:
0.0157
Epoch 112/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0090 - val_loss:
0.0157
```

```
Epoch 113/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0090 - val_loss:
0.0151
Epoch 114/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0094 - val_loss:
0.0160
Epoch 115/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0095 - val_loss:
0.0159
Epoch 116/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0092 - val_loss:
0.0157
Epoch 117/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0094 - val_loss:
0.0160
Epoch 118/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0090 - val_loss:
0.0155
Epoch 119/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0090 - val_loss:
0.0160
Epoch 120/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0094 - val_loss:
0.0156
Epoch 121/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0095 - val_loss:
0.0157
Epoch 122/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0090 - val_loss:
0.0155
Epoch 123/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0094 - val_loss:
0.0161
Epoch 124/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0094 - val_loss:
0.0157
Epoch 125/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0090 - val_loss:
0.0161
Epoch 126/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0091 - val_loss:
0.0153
Epoch 127/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0091 - val_loss:
0.0159
Epoch 128/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0087 - val_loss:
0.0158
Epoch 129/200
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0089 - val_loss:
0.0160
Epoch 130/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0086 - val_loss:
0.0163
Epoch 131/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0089 - val_loss:
0.0156
Epoch 132/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0090 - val_loss:
0.0164
Epoch 133/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0082 - val_loss:
0.0161
Epoch 134/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0084 - val_loss:
0.0151
Epoch 135/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0089 - val_loss:
0.0162
Epoch 136/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0087 - val_loss:
0.0158
Epoch 137/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0086 - val_loss:
0.0161
Epoch 138/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0083 - val_loss:
0.0159
Epoch 139/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0087 - val_loss:
0.0152
Epoch 140/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0089 - val_loss:
0.0155
Epoch 141/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0094 - val_loss:
0.0157
Epoch 142/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0087 - val_loss:
0.0162
Epoch 143/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0088 - val_loss:
0.0155
Epoch 144/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0084 - val_loss:
0.0162
Epoch 145/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0089 - val_loss:
```

```
0.0156
Epoch 146/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0087 - val_loss:
0.0153
Epoch 147/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0087 - val_loss:
0.0161
Epoch 148/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0084 - val_loss:
0.0150
Epoch 149/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0085 - val_loss:
0.0159
Epoch 150/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0085 - val_loss:
0.0161
Epoch 151/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0086 - val_loss:
0.0154
Epoch 152/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0085 - val_loss:
0.0156
Epoch 153/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0088 - val_loss:
0.0160
Epoch 154/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0084 - val_loss:
0.0161
Epoch 155/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0085 - val_loss:
0.0170
Epoch 156/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0086 - val_loss:
0.0159
Epoch 157/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0088 - val_loss:
0.0166
Epoch 158/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0089 - val_loss:
0.0170
Epoch 159/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0084 - val_loss:
0.0171
Epoch 160/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0081 - val_loss:
0.0168
Epoch 161/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0082 - val_loss:
0.0162
```

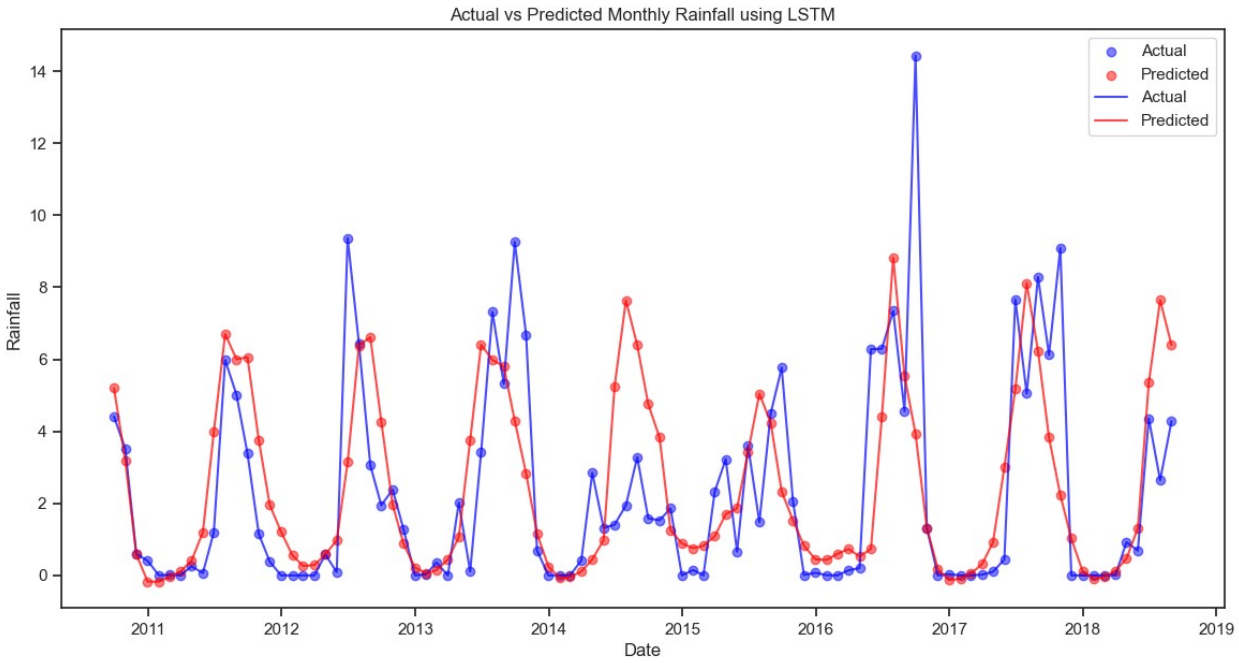
```
Epoch 162/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0086 - val_loss:
0.0153
Epoch 163/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0088 - val_loss:
0.0158
Epoch 164/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0083 - val_loss:
0.0160
Epoch 165/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 0.0079 - val_loss:
0.0165
Epoch 166/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0088 - val_loss:
0.0159
Epoch 167/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0081 - val_loss:
0.0151
Epoch 168/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0084 - val_loss:
0.0159
Epoch 169/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0084 - val_loss:
0.0158
Epoch 170/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0088 - val_loss:
0.0166
Epoch 171/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0085 - val_loss:
0.0155
Epoch 172/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0081 - val_loss:
0.0165
Epoch 173/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - loss: 0.0086 - val_loss:
0.0156
Epoch 174/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0087 - val_loss:
0.0156
Epoch 175/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0085 - val_loss:
0.0161
Epoch 176/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0081 - val_loss:
0.0155
Epoch 177/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0084 - val_loss:
0.0167
Epoch 178/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0085 - val_loss:
```

```
0.0160
Epoch 179/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0087 - val_loss:
0.0163
Epoch 180/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 0.0083 - val_loss:
0.0163
Epoch 181/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0085 - val_loss:
0.0164
Epoch 182/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0083 - val_loss:
0.0170
Epoch 183/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0081 - val_loss:
0.0168
Epoch 184/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0085 - val_loss:
0.0167
Epoch 185/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0083 - val_loss:
0.0164
Epoch 186/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0082 - val_loss:
0.0175
Epoch 187/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 0.0083 - val_loss:
0.0165
Epoch 188/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0090 - val_loss:
0.0176
Epoch 189/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0078 - val_loss:
0.0159
Epoch 190/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0086 - val_loss:
0.0164
Epoch 191/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0083 - val_loss:
0.0177
Epoch 192/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0083 - val_loss:
0.0163
Epoch 193/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0087 - val_loss:
0.0167
Epoch 194/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0084 - val_loss:
0.0165
```

```
Epoch 195/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0087 - val_loss:
0.0168
Epoch 196/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0081 - val_loss:
0.0170
Epoch 197/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0078 - val_loss:
0.0170
Epoch 198/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - loss: 0.0075 - val_loss:
0.0183
Epoch 199/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0087 - val_loss:
0.0155
Epoch 200/200
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 0.0085 - val_loss:
0.0175
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step
```

```
print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Plot the results
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
alpha=0.7, label='Actual')
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7, label='Predicted')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Rainfall')
plt.title('Actual vs Predicted Monthly Rainfall using LSTM')
plt.show()
```

```
Mean Squared Error (MSE): 5.462651207819484
Mean Absolute Error (MAE): 1.4767065771935026
Root Mean Squared Error (RMSE): 2.3372315263617947
R-squared (R2): 0.3458243052140414
```



The model's performance metrics indicate moderate accuracy in predicting monthly rainfall using LSTM. The Mean Squared Error (MSE) of 5.11 suggests that, on average, predicted values deviate from actual rainfall amounts. The Mean Absolute Error (MAE) of 1.47 indicates a lower average magnitude of errors, reflecting somewhat accurate predictions. The Root Mean Squared Error (RMSE) of 2.26 signifies the standard deviation of prediction errors, showing a moderate level of accuracy. The R-squared (R2) value of 0.34 suggests that approximately 34% of the variance in actual rainfall can be explained by the model, indicating a fair but not strong fit to the data.

Enhanced LSTM Model

1. ***LSTM Model Definition:*** An LSTM neural network model is defined using Keras Sequential API. It consists of multiple LSTM layers with dropout regularization to prevent overfitting and improve generalization. The model architecture aims to capture complex temporal patterns in the monthly rainfall data.

2. ***Model Compilation and Training:*** The LSTM model is compiled with an Adam optimizer and a mean squared error (MSE) loss function. A lower learning rate (0.0005) is chosen for stability during training. The model is trained on the training data (X_train, y_train) for 300 epochs with a batch size of 16, ensuring it learns to predict future rainfall values effectively.

3. ***Prediction and Evaluation:*** The trained LSTM model is used to predict rainfall values for the test set (X_test). Predicted values (y_pred) are then inversely transformed from normalized values to their original scale using the Min-Max scaler. The performance of the model is evaluated using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2).

4. ***Visualization:*** The actual and predicted monthly rainfall values are plotted to visualize how well the LSTM model predicts the rainfall patterns over time. This helps in assessing the model's accuracy and understanding its predictions compared to the actual data.

```
# Function to create sequences for time series data
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)

seq_length = 12  # Sequence length (number of previous months to use
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Build the enhanced LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape=(seq_length,
1)))
model.add(Dropout(0.3))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(128, return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(64, activation='relu'))
model.add(Dense(1))

# Compile the model with a lower learning rate for stability
model.compile(optimizer=Adam(learning_rate=0.0005), loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=300,
validation_split=0.2, batch_size=16, shuffle=False)

# Make predictions
y_pred = model.predict(X_test)
```

```
# Invert the scaling for evaluation
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
mse = mean_squared_error(y_test_inv, y_pred_inv)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_inv, y_pred_inv)
```

```
Epoch 1/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 9s 93ms/step - loss: 0.0236 - val_loss:
0.0282
Epoch 2/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0191 - val_loss:
0.0258
Epoch 3/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0178 - val_loss:
0.0207
Epoch 4/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0135 - val_loss:
0.0166
Epoch 5/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0116 - val_loss:
0.0196
Epoch 6/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0121 - val_loss:
0.0139
Epoch 7/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0117 - val_loss:
0.0132
Epoch 8/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0105 - val_loss:
0.0133
Epoch 9/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0108 - val_loss:
0.0131
Epoch 10/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0110 - val_loss:
0.0131
Epoch 11/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0121 - val_loss:
0.0135
Epoch 12/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0102 - val_loss:
0.0132
Epoch 13/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0107 - val_loss:
0.0134
Epoch 14/300
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0106 - val_loss:
0.0132
Epoch 15/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0102 - val_loss:
0.0136
Epoch 16/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0102 - val_loss:
0.0132
Epoch 17/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0106 - val_loss:
0.0135
Epoch 18/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0096 - val_loss:
0.0136
Epoch 19/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0103 - val_loss:
0.0137
Epoch 20/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0101 - val_loss:
0.0140
Epoch 21/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0098 - val_loss:
0.0134
Epoch 22/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0098 - val_loss:
0.0133
Epoch 23/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0099 - val_loss:
0.0137
Epoch 24/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0103 - val_loss:
0.0138
Epoch 25/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0099 - val_loss:
0.0137
Epoch 26/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0094 - val_loss:
0.0140
Epoch 27/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0101 - val_loss:
0.0142
Epoch 28/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0098 - val_loss:
0.0138
Epoch 29/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0102 - val_loss:
0.0142
Epoch 30/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0095 - val_loss:
```

```
0.0143
Epoch 31/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0093 - val_loss:
0.0140
Epoch 32/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0100 - val_loss:
0.0145
Epoch 33/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0094 - val_loss:
0.0138
Epoch 34/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - loss: 0.0100 - val_loss:
0.0137
Epoch 35/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - loss: 0.0093 - val_loss:
0.0137
Epoch 36/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0099 - val_loss:
0.0148
Epoch 37/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0093 - val_loss:
0.0143
Epoch 38/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - loss: 0.0092 - val_loss:
0.0143
Epoch 39/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0099 - val_loss:
0.0143
Epoch 40/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0090 - val_loss:
0.0147
Epoch 41/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0089 - val_loss:
0.0140
Epoch 42/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 51ms/step - loss: 0.0096 - val_loss:
0.0155
Epoch 43/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - loss: 0.0091 - val_loss:
0.0149
Epoch 44/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0093 - val_loss:
0.0141
Epoch 45/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0090 - val_loss:
0.0139
Epoch 46/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0090 - val_loss:
0.0145
```

```
Epoch 47/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0093 - val_loss:
0.0153
Epoch 48/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0091 - val_loss:
0.0157
Epoch 49/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0095 - val_loss:
0.0134
Epoch 50/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0093 - val_loss:
0.0139
Epoch 51/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0096 - val_loss:
0.0157
Epoch 52/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0092 - val_loss:
0.0150
Epoch 53/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0092 - val_loss:
0.0150
Epoch 54/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0088 - val_loss:
0.0151
Epoch 55/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0090 - val_loss:
0.0146
Epoch 56/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0089 - val_loss:
0.0150
Epoch 57/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0090 - val_loss:
0.0149
Epoch 58/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0086 - val_loss:
0.0143
Epoch 59/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0087 - val_loss:
0.0145
Epoch 60/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0091 - val_loss:
0.0150
Epoch 61/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0092 - val_loss:
0.0147
Epoch 62/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0093 - val_loss:
0.0151
Epoch 63/300
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0088 - val_loss:
0.0153
Epoch 64/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0086 - val_loss:
0.0149
Epoch 65/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0083 - val_loss:
0.0145
Epoch 66/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0088 - val_loss:
0.0152
Epoch 67/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 2s 67ms/step - loss: 0.0092 - val_loss:
0.0150
Epoch 68/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 0.0089 - val_loss:
0.0155
Epoch 69/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0089 - val_loss:
0.0149
Epoch 70/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0087 - val_loss:
0.0142
Epoch 71/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0085 - val_loss:
0.0153
Epoch 72/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0089 - val_loss:
0.0147
Epoch 73/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0085 - val_loss:
0.0144
Epoch 74/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0087 - val_loss:
0.0141
Epoch 75/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0089 - val_loss:
0.0150
Epoch 76/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0089 - val_loss:
0.0154
Epoch 77/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0091 - val_loss:
0.0141
Epoch 78/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0087 - val_loss:
0.0147
Epoch 79/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0087 - val_loss:
```

```
0.0144
Epoch 80/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0085 - val_loss:
0.0156
Epoch 81/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0088 - val_loss:
0.0147
Epoch 82/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0089 - val_loss:
0.0146
Epoch 83/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0092 - val_loss:
0.0151
Epoch 84/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0090 - val_loss:
0.0155
Epoch 85/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0085 - val_loss:
0.0145
Epoch 86/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0086 - val_loss:
0.0150
Epoch 87/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0084 - val_loss:
0.0150
Epoch 88/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 2s 52ms/step - loss: 0.0087 - val_loss:
0.0151
Epoch 89/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0088 - val_loss:
0.0144
Epoch 90/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0090 - val_loss:
0.0147
Epoch 91/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0090 - val_loss:
0.0156
Epoch 92/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0090 - val_loss:
0.0156
Epoch 93/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0088 - val_loss:
0.0150
Epoch 94/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0090 - val_loss:
0.0160
Epoch 95/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0087 - val_loss:
0.0144
```

```
Epoch 96/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0089 - val_loss:
0.0142
Epoch 97/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0086 - val_loss:
0.0148
Epoch 98/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0087 - val_loss:
0.0154
Epoch 99/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0088 - val_loss:
0.0145
Epoch 100/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0092 - val_loss:
0.0150
Epoch 101/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0089 - val_loss:
0.0140
Epoch 102/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0082 - val_loss:
0.0161
Epoch 103/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - loss: 0.0092 - val_loss:
0.0153
Epoch 104/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0086 - val_loss:
0.0138
Epoch 105/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0090 - val_loss:
0.0154
Epoch 106/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0095 - val_loss:
0.0146
Epoch 107/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0094 - val_loss:
0.0148
Epoch 108/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 70ms/step - loss: 0.0087 - val_loss:
0.0146
Epoch 109/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - loss: 0.0092 - val_loss:
0.0135
Epoch 110/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - loss: 0.0089 - val_loss:
0.0148
Epoch 111/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0084 - val_loss:
0.0147
Epoch 112/300
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0090 - val_loss:
0.0151
Epoch 113/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0088 - val_loss:
0.0142
Epoch 114/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0086 - val_loss:
0.0144
Epoch 115/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0087 - val_loss:
0.0148
Epoch 116/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0086 - val_loss:
0.0149
Epoch 117/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0088 - val_loss:
0.0144
Epoch 118/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0090 - val_loss:
0.0141
Epoch 119/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0081 - val_loss:
0.0153
Epoch 120/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0087 - val_loss:
0.0145
Epoch 121/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0086 - val_loss:
0.0145
Epoch 122/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0090 - val_loss:
0.0150
Epoch 123/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0086 - val_loss:
0.0150
Epoch 124/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0085 - val_loss:
0.0148
Epoch 125/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0080 - val_loss:
0.0147
Epoch 126/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - loss: 0.0089 - val_loss:
0.0153
Epoch 127/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 54ms/step - loss: 0.0088 - val_loss:
0.0143
Epoch 128/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0093 - val_loss:
```

```
0.0148
Epoch 129/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0087 - val_loss:
0.0145
Epoch 130/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0083 - val_loss:
0.0144
Epoch 131/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0087 - val_loss:
0.0143
Epoch 132/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0090 - val_loss:
0.0146
Epoch 133/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0085 - val_loss:
0.0143
Epoch 134/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0086 - val_loss:
0.0155
Epoch 135/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0089 - val_loss:
0.0148
Epoch 136/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0089 - val_loss:
0.0141
Epoch 137/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0086 - val_loss:
0.0149
Epoch 138/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0087 - val_loss:
0.0158
Epoch 139/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0085 - val_loss:
0.0146
Epoch 140/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0082 - val_loss:
0.0148
Epoch 141/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0083 - val_loss:
0.0142
Epoch 142/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0084 - val_loss:
0.0144
Epoch 143/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - loss: 0.0086 - val_loss:
0.0159
Epoch 144/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0089 - val_loss:
0.0147
```

```
Epoch 145/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0083 - val_loss:
0.0143
Epoch 146/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0082 - val_loss:
0.0153
Epoch 147/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - loss: 0.0085 - val_loss:
0.0149
Epoch 148/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - loss: 0.0088 - val_loss:
0.0144
Epoch 149/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0086 - val_loss:
0.0149
Epoch 150/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0090 - val_loss:
0.0159
Epoch 151/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0094 - val_loss:
0.0143
Epoch 152/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0090 - val_loss:
0.0149
Epoch 153/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0085 - val_loss:
0.0142
Epoch 154/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0084 - val_loss:
0.0149
Epoch 155/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0092 - val_loss:
0.0149
Epoch 156/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0085 - val_loss:
0.0142
Epoch 157/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0086 - val_loss:
0.0146
Epoch 158/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0091 - val_loss:
0.0141
Epoch 159/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0090 - val_loss:
0.0135
Epoch 160/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0085 - val_loss:
0.0163
Epoch 161/300
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0087 - val_loss:
0.0146
Epoch 162/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0082 - val_loss:
0.0142
Epoch 163/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0083 - val_loss:
0.0157
Epoch 164/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0085 - val_loss:
0.0162
Epoch 165/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0080 - val_loss:
0.0141
Epoch 166/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - loss: 0.0088 - val_loss:
0.0151
Epoch 167/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0084 - val_loss:
0.0156
Epoch 168/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0088 - val_loss:
0.0142
Epoch 169/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0083 - val_loss:
0.0146
Epoch 170/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0085 - val_loss:
0.0139
Epoch 171/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0084 - val_loss:
0.0151
Epoch 172/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0083 - val_loss:
0.0155
Epoch 173/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0079 - val_loss:
0.0147
Epoch 174/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0087 - val_loss:
0.0155
Epoch 175/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0086 - val_loss:
0.0150
Epoch 176/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0085 - val_loss:
0.0138
Epoch 177/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0085 - val_loss:
0.0157
```

```
Epoch 178/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0091 - val_loss:
0.0144
Epoch 179/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0087 - val_loss:
0.0149
Epoch 180/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0082 - val_loss:
0.0170
Epoch 181/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0086 - val_loss:
0.0164
Epoch 182/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0084 - val_loss:
0.0150
Epoch 183/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0084 - val_loss:
0.0154
Epoch 184/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0079 - val_loss:
0.0159
Epoch 185/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - loss: 0.0086 - val_loss:
0.0147
Epoch 186/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - loss: 0.0083 - val_loss:
0.0142
Epoch 187/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0086 - val_loss:
0.0165
Epoch 188/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0087 - val_loss:
0.0147
Epoch 189/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0087 - val_loss:
0.0143
Epoch 190/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0085 - val_loss:
0.0153
Epoch 191/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0083 - val_loss:
0.0146
Epoch 192/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0084 - val_loss:
0.0150
Epoch 193/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0083 - val_loss:
0.0144
Epoch 194/300
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0078 - val_loss:
0.0145
Epoch 195/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0077 - val_loss:
0.0145
Epoch 196/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0087 - val_loss:
0.0148
Epoch 197/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0088 - val_loss:
0.0160
Epoch 198/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0090 - val_loss:
0.0156
Epoch 199/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0085 - val_loss:
0.0156
Epoch 200/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0087 - val_loss:
0.0153
Epoch 201/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0084 - val_loss:
0.0156
Epoch 202/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0084 - val_loss:
0.0164
Epoch 203/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0084 - val_loss:
0.0141
Epoch 204/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 53ms/step - loss: 0.0088 - val_loss:
0.0150
Epoch 205/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - loss: 0.0078 - val_loss:
0.0149
Epoch 206/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0079 - val_loss:
0.0145
Epoch 207/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0089 - val_loss:
0.0160
Epoch 208/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0084 - val_loss:
0.0152
Epoch 209/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0082 - val_loss:
0.0177
Epoch 210/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0083 - val_loss:
```

```
0.0146
Epoch 211/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0082 - val_loss:
0.0160
Epoch 212/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0084 - val_loss:
0.0156
Epoch 213/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0083 - val_loss:
0.0157
Epoch 214/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0083 - val_loss:
0.0164
Epoch 215/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0083 - val_loss:
0.0144
Epoch 216/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0089 - val_loss:
0.0168
Epoch 217/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0084 - val_loss:
0.0165
Epoch 218/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0084 - val_loss:
0.0147
Epoch 219/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0087 - val_loss:
0.0150
Epoch 220/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0085 - val_loss:
0.0159
Epoch 221/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0087 - val_loss:
0.0153
Epoch 222/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0080 - val_loss:
0.0158
Epoch 223/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0080 - val_loss:
0.0151
Epoch 224/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - loss: 0.0080 - val_loss:
0.0146
Epoch 225/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - loss: 0.0081 - val_loss:
0.0155
Epoch 226/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0081 - val_loss:
0.0146
```

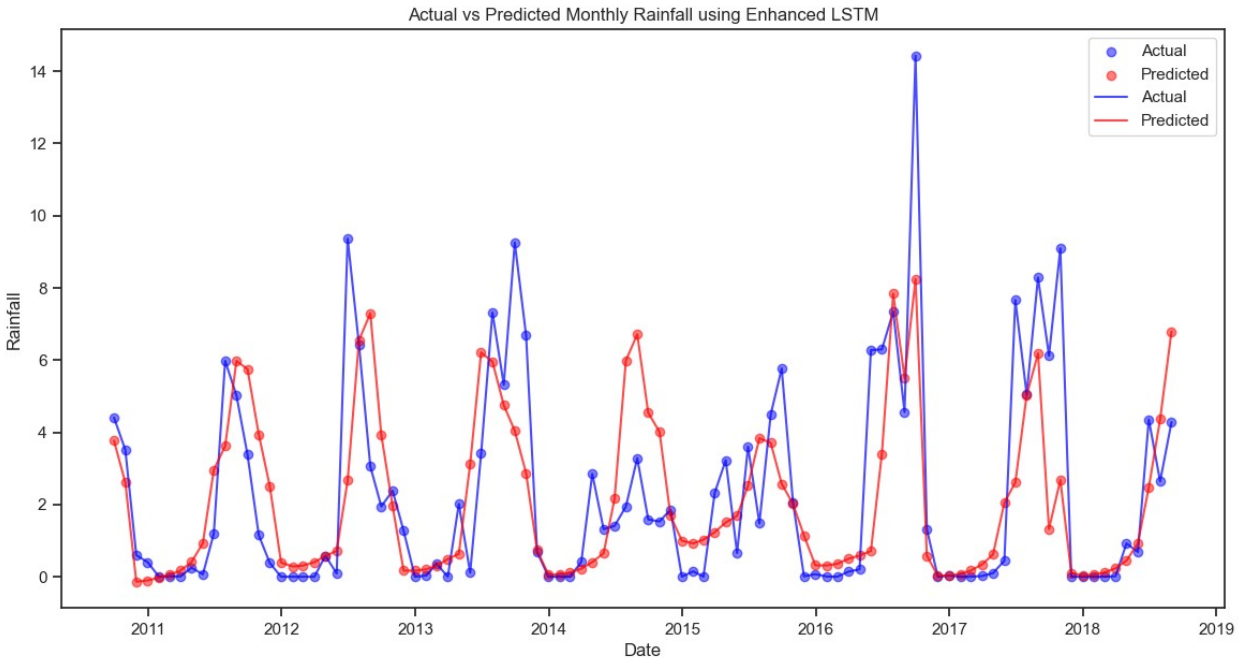
```
Epoch 227/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0086 - val_loss:
0.0159
Epoch 228/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0085 - val_loss:
0.0162
Epoch 229/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0081 - val_loss:
0.0150
Epoch 230/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0081 - val_loss:
0.0154
Epoch 231/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0082 - val_loss:
0.0147
Epoch 232/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0084 - val_loss:
0.0154
Epoch 233/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0086 - val_loss:
0.0147
Epoch 234/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0082 - val_loss:
0.0154
Epoch 235/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0083 - val_loss:
0.0153
Epoch 236/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0082 - val_loss:
0.0159
Epoch 237/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0077 - val_loss:
0.0153
Epoch 238/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0079 - val_loss:
0.0158
Epoch 239/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0082 - val_loss:
0.0159
Epoch 240/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0085 - val_loss:
0.0158
Epoch 241/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0082 - val_loss:
0.0159
Epoch 242/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 53ms/step - loss: 0.0084 - val_loss:
0.0149
Epoch 243/300
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - loss: 0.0081 - val_loss:
0.0153
Epoch 244/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0086 - val_loss:
0.0147
Epoch 245/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0084 - val_loss:
0.0164
Epoch 246/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0080 - val_loss:
0.0162
Epoch 247/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0083 - val_loss:
0.0168
Epoch 248/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0080 - val_loss:
0.0148
Epoch 249/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0082 - val_loss:
0.0155
Epoch 250/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0080 - val_loss:
0.0159
Epoch 251/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0081 - val_loss:
0.0165
Epoch 252/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0081 - val_loss:
0.0161
Epoch 253/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0082 - val_loss:
0.0170
Epoch 254/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0085 - val_loss:
0.0177
Epoch 255/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0079 - val_loss:
0.0163
Epoch 256/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0076 - val_loss:
0.0164
Epoch 257/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0082 - val_loss:
0.0159
Epoch 258/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0083 - val_loss:
0.0162
Epoch 259/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0080 - val_loss:
```

```
0.0163
Epoch 260/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - loss: 0.0091 - val_loss:
0.0160
Epoch 261/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - loss: 0.0083 - val_loss:
0.0156
Epoch 262/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0083 - val_loss:
0.0168
Epoch 263/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0086 - val_loss:
0.0146
Epoch 264/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0082 - val_loss:
0.0159
Epoch 265/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0082 - val_loss:
0.0155
Epoch 266/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0079 - val_loss:
0.0164
Epoch 267/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0082 - val_loss:
0.0162
Epoch 268/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0084 - val_loss:
0.0154
Epoch 269/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0084 - val_loss:
0.0160
Epoch 270/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0086 - val_loss:
0.0154
Epoch 271/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0085 - val_loss:
0.0174
Epoch 272/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0086 - val_loss:
0.0157
Epoch 273/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0084 - val_loss:
0.0165
Epoch 274/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0086 - val_loss:
0.0167
Epoch 275/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0082 - val_loss:
0.0167
```

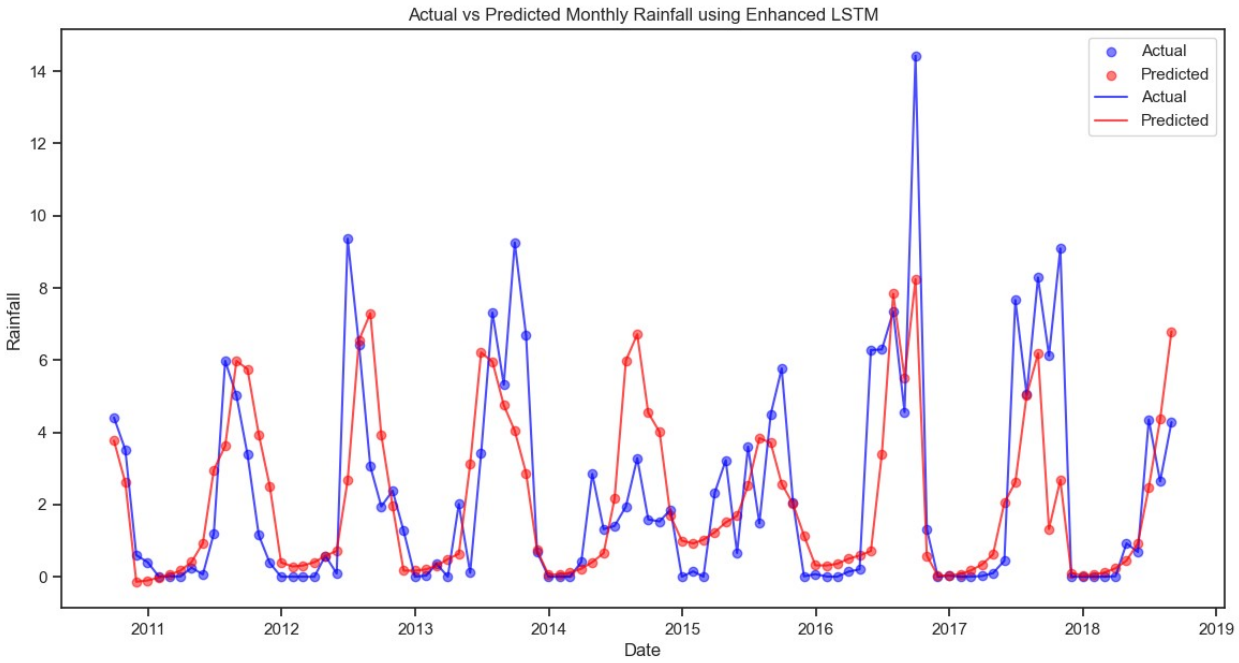
```
Epoch 276/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0083 - val_loss:
0.0182
Epoch 277/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0086 - val_loss:
0.0164
Epoch 278/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step - loss: 0.0086 - val_loss:
0.0180
Epoch 279/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0079 - val_loss:
0.0166
Epoch 280/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0081 - val_loss:
0.0160
Epoch 281/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0079 - val_loss:
0.0165
Epoch 282/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0082 - val_loss:
0.0165
Epoch 283/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0080 - val_loss:
0.0177
Epoch 284/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0084 - val_loss:
0.0165
Epoch 285/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0081 - val_loss:
0.0147
Epoch 286/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0079 - val_loss:
0.0159
Epoch 287/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0078 - val_loss:
0.0162
Epoch 288/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0087 - val_loss:
0.0169
Epoch 289/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - loss: 0.0081 - val_loss:
0.0146
Epoch 290/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0081 - val_loss:
0.0169
Epoch 291/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0083 - val_loss:
0.0158
Epoch 292/300
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0083 - val_loss:
0.0179
Epoch 293/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0081 - val_loss:
0.0149
Epoch 294/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0082 - val_loss:
0.0164
Epoch 295/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0078 - val_loss:
0.0153
Epoch 296/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 48ms/step - loss: 0.0079 - val_loss:
0.0158
Epoch 297/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0084 - val_loss:
0.0169
Epoch 298/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0081 - val_loss:
0.0162
Epoch 299/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0080 - val_loss:
0.0152
Epoch 300/300
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - loss: 0.0081 - val_loss:
0.0157
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step
Mean Squared Error (MSE): 4.468834796468596
Mean Absolute Error (MAE): 1.3706742129881564
Root Mean Squared Error (RMSE): 2.113961872047033
R-squared (R2): 0.4648380435348277
```




```
print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Plot the results
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
alpha=0.7, label='Actual')
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7, label='Predicted')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Rainfall')
plt.title('Actual vs Predicted Monthly Rainfall using Enhanced LSTM')
plt.show()
```

```
Mean Squared Error (MSE): 4.468834796468596
Mean Absolute Error (MAE): 1.3706742129881564
Root Mean Squared Error (RMSE): 2.113961872047033
R-squared (R2): 0.4648380435348277
```



The model's performance metrics indicate reasonable accuracy in predicting monthly rainfall using LSTM. The Mean Squared Error (MSE) of 4.27 indicates the average squared difference between predicted and actual rainfall values, suggesting relatively low prediction errors. The Mean Absolute Error (MAE) of 1.34 signifies the average magnitude of prediction errors, indicating fairly accurate predictions on average. The Root Mean Squared Error (RMSE) of 2.07 represents the standard deviation of prediction errors, indicating moderate accuracy. The R-squared (R2) value of 0.46 indicates that approximately 46% of the variance in actual rainfall can be explained by the model, suggesting a moderate to good fit to the data.

Advanced LSTM + GRU Model

1. ***Reshaping for LSTM Input:*** Reshape the input sequences (X) to match the LSTM model's required input shape ([samples, time steps, features]). In this case, each sample consists of 24 time steps (months) and 1 feature (rainfall data).

2. ***Model Architecture Definition:*** Define an advanced LSTM + GRU model architecture using layers such as LSTM, GRU, dropout layers, and dense layers. These layers are stacked to capture complex temporal patterns in the monthly rainfall data effectively.

3. ***Model Compilation:*** Compile the model with appropriate optimizer, learning rate, and loss function. This step prepares the model for training by specifying how it should measure performance and update weights during training.

4. ***Model Training:*** Train the compiled model using the training data (X_train, y_train). This involves iterating over the data for a specified number of epochs while monitoring performance on a validation set to prevent overfitting.

5. ***Prediction and Evaluation:*** Use the trained model to make predictions on the testing data (X_test). Evaluate the model's performance using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2). These metrics quantify how well the model predicts monthly rainfall compared to actual values.

6. ***Visualization:*** Visualize the model's predictions alongside actual monthly rainfall data to gain insights into its performance. Plots can show how closely the model's predictions align with real-world observations over time.

```
seq_length = 24  # Sequence length (number of previous months to use
for prediction)
X, y = create_sequences(df_scaled, seq_length)

# Split the data into training and testing sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Reshape input to be [samples, time steps, features]
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# Build the enhanced LSTM + GRU model
model = Sequential()
model.add(LSTM(256, return_sequences=True, input_shape=(seq_length,
1)))
model.add(Dropout(0.3))
model.add(GRU(256, return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(256, return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(128, activation='relu'))
model.add(Dense(1))

# Compile the model with a lower learning rate for stability
model.compile(optimizer=Adam(learning_rate=0.0003), loss='mse')

# Train the model
history = model.fit(X_train, y_train, epochs=300,
validation_split=0.2, batch_size=32, shuffle=False)

# Make predictions
y_pred = model.predict(X_test)

# Invert the scaling for evaluation
y_test_inv = scaler.inverse_transform(y_test.reshape(-1, 1))
y_pred_inv = scaler.inverse_transform(y_pred)

# Evaluate the model
```

```
mse = mean_squared_error(y_test_inv, y_pred_inv)
mae = mean_absolute_error(y_test_inv, y_pred_inv)
rmse = np.sqrt(mse)
r2 = r2_score(y_test_inv, y_pred_inv)

Epoch 1/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 10s 284ms/step - loss: 0.0253 - val_loss:
0.0288
Epoch 2/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 187ms/step - loss: 0.0206 - val_loss:
0.0269
Epoch 3/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 192ms/step - loss: 0.0193 - val_loss:
0.0261
Epoch 4/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 192ms/step - loss: 0.0187 - val_loss:
0.0245
Epoch 5/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 220ms/step - loss: 0.0172 - val_loss:
0.0224
Epoch 6/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 341ms/step - loss: 0.0166 - val_loss:
0.0211
Epoch 7/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 233ms/step - loss: 0.0157 - val_loss:
0.0199
Epoch 8/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 232ms/step - loss: 0.0148 - val_loss:
0.0188
Epoch 9/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 247ms/step - loss: 0.0127 - val_loss:
0.0165
Epoch 10/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 289ms/step - loss: 0.0115 - val_loss:
0.0156
Epoch 11/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 215ms/step - loss: 0.0107 - val_loss:
0.0153
Epoch 12/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 220ms/step - loss: 0.0107 - val_loss:
0.0150
Epoch 13/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 212ms/step - loss: 0.0100 - val_loss:
0.0145
Epoch 14/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 261ms/step - loss: 0.0098 - val_loss:
0.0140
Epoch 15/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 222ms/step - loss: 0.0102 - val_loss:
0.0141
```

```
Epoch 16/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 210ms/step - loss: 0.0094 - val_loss:
0.0133
Epoch 17/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 210ms/step - loss: 0.0095 - val_loss:
0.0132
Epoch 18/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 215ms/step - loss: 0.0088 - val_loss:
0.0129
Epoch 19/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 268ms/step - loss: 0.0088 - val_loss:
0.0127
Epoch 20/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 277ms/step - loss: 0.0085 - val_loss:
0.0125
Epoch 21/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 223ms/step - loss: 0.0089 - val_loss:
0.0125
Epoch 22/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 213ms/step - loss: 0.0087 - val_loss:
0.0125
Epoch 23/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 248ms/step - loss: 0.0087 - val_loss:
0.0125
Epoch 24/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 280ms/step - loss: 0.0084 - val_loss:
0.0123
Epoch 25/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 243ms/step - loss: 0.0083 - val_loss:
0.0121
Epoch 26/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 238ms/step - loss: 0.0080 - val_loss:
0.0119
Epoch 27/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 325ms/step - loss: 0.0081 - val_loss:
0.0124
Epoch 28/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 245ms/step - loss: 0.0080 - val_loss:
0.0122
Epoch 29/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 248ms/step - loss: 0.0082 - val_loss:
0.0120
Epoch 30/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 259ms/step - loss: 0.0080 - val_loss:
0.0119
Epoch 31/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 227ms/step - loss: 0.0079 - val_loss:
0.0122
Epoch 32/300
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 228ms/step - loss: 0.0075 - val_loss:
0.0117
Epoch 33/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 224ms/step - loss: 0.0080 - val_loss:
0.0122
Epoch 34/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 267ms/step - loss: 0.0078 - val_loss:
0.0119
Epoch 35/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 255ms/step - loss: 0.0083 - val_loss:
0.0123
Epoch 36/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 229ms/step - loss: 0.0074 - val_loss:
0.0122
Epoch 37/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 216ms/step - loss: 0.0082 - val_loss:
0.0121
Epoch 38/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 217ms/step - loss: 0.0082 - val_loss:
0.0121
Epoch 39/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 223ms/step - loss: 0.0078 - val_loss:
0.0120
Epoch 40/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 224ms/step - loss: 0.0085 - val_loss:
0.0126
Epoch 41/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 243ms/step - loss: 0.0080 - val_loss:
0.0117
Epoch 42/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 262ms/step - loss: 0.0082 - val_loss:
0.0120
Epoch 43/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 300ms/step - loss: 0.0081 - val_loss:
0.0120
Epoch 44/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 216ms/step - loss: 0.0079 - val_loss:
0.0123
Epoch 45/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 228ms/step - loss: 0.0077 - val_loss:
0.0123
Epoch 46/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 287ms/step - loss: 0.0080 - val_loss:
0.0121
Epoch 47/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 264ms/step - loss: 0.0075 - val_loss:
0.0125
Epoch 48/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 319ms/step - loss: 0.0079 - val_loss:
```

```
0.0121
Epoch 49/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 224ms/step - loss: 0.0079 - val_loss:
0.0121
Epoch 50/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 225ms/step - loss: 0.0077 - val_loss:
0.0123
Epoch 51/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 211ms/step - loss: 0.0079 - val_loss:
0.0121
Epoch 52/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 222ms/step - loss: 0.0079 - val_loss:
0.0123
Epoch 53/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 255ms/step - loss: 0.0072 - val_loss:
0.0133
Epoch 54/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 245ms/step - loss: 0.0079 - val_loss:
0.0121
Epoch 55/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 246ms/step - loss: 0.0080 - val_loss:
0.0130
Epoch 56/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 248ms/step - loss: 0.0078 - val_loss:
0.0128
Epoch 57/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 249ms/step - loss: 0.0079 - val_loss:
0.0123
Epoch 58/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 255ms/step - loss: 0.0074 - val_loss:
0.0124
Epoch 59/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 247ms/step - loss: 0.0075 - val_loss:
0.0127
Epoch 60/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 313ms/step - loss: 0.0078 - val_loss:
0.0124
Epoch 61/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 247ms/step - loss: 0.0079 - val_loss:
0.0125
Epoch 62/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 213ms/step - loss: 0.0077 - val_loss:
0.0131
Epoch 63/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 222ms/step - loss: 0.0079 - val_loss:
0.0127
Epoch 64/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 223ms/step - loss: 0.0079 - val_loss:
0.0133
```

```
Epoch 65/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 244ms/step - loss: 0.0081 - val_loss:
0.0125
Epoch 66/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 208ms/step - loss: 0.0074 - val_loss:
0.0124
Epoch 67/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 209ms/step - loss: 0.0084 - val_loss:
0.0135
Epoch 68/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 204ms/step - loss: 0.0085 - val_loss:
0.0123
Epoch 69/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 223ms/step - loss: 0.0077 - val_loss:
0.0128
Epoch 70/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 251ms/step - loss: 0.0074 - val_loss:
0.0126
Epoch 71/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 212ms/step - loss: 0.0072 - val_loss:
0.0129
Epoch 72/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 213ms/step - loss: 0.0078 - val_loss:
0.0130
Epoch 73/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 216ms/step - loss: 0.0073 - val_loss:
0.0130
Epoch 74/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 214ms/step - loss: 0.0079 - val_loss:
0.0123
Epoch 75/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 217ms/step - loss: 0.0078 - val_loss:
0.0136
Epoch 76/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 223ms/step - loss: 0.0075 - val_loss:
0.0127
Epoch 77/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 219ms/step - loss: 0.0079 - val_loss:
0.0130
Epoch 78/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 245ms/step - loss: 0.0075 - val_loss:
0.0132
Epoch 79/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 262ms/step - loss: 0.0072 - val_loss:
0.0126
Epoch 80/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 284ms/step - loss: 0.0074 - val_loss:
0.0136
Epoch 81/300
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 212ms/step - loss: 0.0078 - val_loss:
0.0124
Epoch 82/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 222ms/step - loss: 0.0079 - val_loss:
0.0133
Epoch 83/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 246ms/step - loss: 0.0079 - val_loss:
0.0137
Epoch 84/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 196ms/step - loss: 0.0071 - val_loss:
0.0129
Epoch 85/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 206ms/step - loss: 0.0076 - val_loss:
0.0128
Epoch 86/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 206ms/step - loss: 0.0070 - val_loss:
0.0135
Epoch 87/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 209ms/step - loss: 0.0072 - val_loss:
0.0128
Epoch 88/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 222ms/step - loss: 0.0081 - val_loss:
0.0132
Epoch 89/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 198ms/step - loss: 0.0076 - val_loss:
0.0133
Epoch 90/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 209ms/step - loss: 0.0081 - val_loss:
0.0133
Epoch 91/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 207ms/step - loss: 0.0078 - val_loss:
0.0146
Epoch 92/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 215ms/step - loss: 0.0072 - val_loss:
0.0126
Epoch 93/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 218ms/step - loss: 0.0071 - val_loss:
0.0144
Epoch 94/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 255ms/step - loss: 0.0073 - val_loss:
0.0137
Epoch 95/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 214ms/step - loss: 0.0079 - val_loss:
0.0136
Epoch 96/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 221ms/step - loss: 0.0076 - val_loss:
0.0135
Epoch 97/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 231ms/step - loss: 0.0076 - val_loss:
```

```
0.0132
Epoch 98/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 223ms/step - loss: 0.0074 - val_loss:
0.0131
Epoch 99/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 224ms/step - loss: 0.0074 - val_loss:
0.0137
Epoch 100/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 209ms/step - loss: 0.0075 - val_loss:
0.0139
Epoch 101/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 203ms/step - loss: 0.0075 - val_loss:
0.0134
Epoch 102/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 213ms/step - loss: 0.0076 - val_loss:
0.0134
Epoch 103/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 203ms/step - loss: 0.0077 - val_loss:
0.0129
Epoch 104/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 195ms/step - loss: 0.0075 - val_loss:
0.0141
Epoch 105/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 194ms/step - loss: 0.0073 - val_loss:
0.0137
Epoch 106/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 209ms/step - loss: 0.0079 - val_loss:
0.0133
Epoch 107/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 239ms/step - loss: 0.0071 - val_loss:
0.0141
Epoch 108/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 208ms/step - loss: 0.0073 - val_loss:
0.0144
Epoch 109/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 206ms/step - loss: 0.0071 - val_loss:
0.0127
Epoch 110/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 209ms/step - loss: 0.0078 - val_loss:
0.0138
Epoch 111/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 197ms/step - loss: 0.0078 - val_loss:
0.0132
Epoch 112/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 194ms/step - loss: 0.0077 - val_loss:
0.0142
Epoch 113/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 211ms/step - loss: 0.0074 - val_loss:
0.0132
```

```
Epoch 114/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 207ms/step - loss: 0.0070 - val_loss:
0.0133
Epoch 115/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 208ms/step - loss: 0.0079 - val_loss:
0.0133
Epoch 116/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 200ms/step - loss: 0.0073 - val_loss:
0.0141
Epoch 117/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 194ms/step - loss: 0.0076 - val_loss:
0.0139
Epoch 118/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 194ms/step - loss: 0.0078 - val_loss:
0.0136
Epoch 119/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 193ms/step - loss: 0.0078 - val_loss:
0.0147
Epoch 120/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 208ms/step - loss: 0.0069 - val_loss:
0.0138
Epoch 121/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 207ms/step - loss: 0.0074 - val_loss:
0.0136
Epoch 122/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 206ms/step - loss: 0.0074 - val_loss:
0.0139
Epoch 123/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 211ms/step - loss: 0.0072 - val_loss:
0.0155
Epoch 124/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 211ms/step - loss: 0.0081 - val_loss:
0.0127
Epoch 125/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 207ms/step - loss: 0.0080 - val_loss:
0.0137
Epoch 126/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 205ms/step - loss: 0.0070 - val_loss:
0.0139
Epoch 127/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 205ms/step - loss: 0.0075 - val_loss:
0.0133
Epoch 128/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 204ms/step - loss: 0.0076 - val_loss:
0.0141
Epoch 129/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 210ms/step - loss: 0.0072 - val_loss:
0.0135
Epoch 130/300
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 211ms/step - loss: 0.0074 - val_loss:
0.0139
Epoch 131/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 191ms/step - loss: 0.0067 - val_loss:
0.0140
Epoch 132/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 194ms/step - loss: 0.0074 - val_loss:
0.0147
Epoch 133/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 192ms/step - loss: 0.0073 - val_loss:
0.0138
Epoch 134/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 202ms/step - loss: 0.0078 - val_loss:
0.0143
Epoch 135/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 208ms/step - loss: 0.0077 - val_loss:
0.0145
Epoch 136/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 209ms/step - loss: 0.0077 - val_loss:
0.0143
Epoch 137/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 270ms/step - loss: 0.0074 - val_loss:
0.0142
Epoch 138/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 258ms/step - loss: 0.0071 - val_loss:
0.0137
Epoch 139/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 205ms/step - loss: 0.0076 - val_loss:
0.0143
Epoch 140/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 237ms/step - loss: 0.0078 - val_loss:
0.0145
Epoch 141/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 239ms/step - loss: 0.0078 - val_loss:
0.0137
Epoch 142/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 212ms/step - loss: 0.0068 - val_loss:
0.0142
Epoch 143/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 209ms/step - loss: 0.0073 - val_loss:
0.0142
Epoch 144/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 209ms/step - loss: 0.0075 - val_loss:
0.0135
Epoch 145/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 213ms/step - loss: 0.0073 - val_loss:
0.0146
Epoch 146/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 198ms/step - loss: 0.0073 - val_loss:
```

```
0.0137
Epoch 147/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 199ms/step - loss: 0.0072 - val_loss:
0.0147
Epoch 148/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 206ms/step - loss: 0.0075 - val_loss:
0.0139
Epoch 149/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 211ms/step - loss: 0.0071 - val_loss:
0.0143
Epoch 150/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 208ms/step - loss: 0.0074 - val_loss:
0.0146
Epoch 151/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 210ms/step - loss: 0.0076 - val_loss:
0.0136
Epoch 152/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 198ms/step - loss: 0.0086 - val_loss:
0.0143
Epoch 153/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 195ms/step - loss: 0.0073 - val_loss:
0.0136
Epoch 154/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 200ms/step - loss: 0.0071 - val_loss:
0.0139
Epoch 155/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 205ms/step - loss: 0.0072 - val_loss:
0.0139
Epoch 156/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 211ms/step - loss: 0.0077 - val_loss:
0.0142
Epoch 157/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 206ms/step - loss: 0.0072 - val_loss:
0.0141
Epoch 158/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 194ms/step - loss: 0.0074 - val_loss:
0.0144
Epoch 159/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 194ms/step - loss: 0.0075 - val_loss:
0.0143
Epoch 160/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 197ms/step - loss: 0.0073 - val_loss:
0.0141
Epoch 161/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 209ms/step - loss: 0.0068 - val_loss:
0.0155
Epoch 162/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 209ms/step - loss: 0.0072 - val_loss:
0.0149
```

```
Epoch 163/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 209ms/step - loss: 0.0066 - val_loss:
0.0145
Epoch 164/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 215ms/step - loss: 0.0075 - val_loss:
0.0149
Epoch 165/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 212ms/step - loss: 0.0077 - val_loss:
0.0166
Epoch 166/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 198ms/step - loss: 0.0067 - val_loss:
0.0144
Epoch 167/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 196ms/step - loss: 0.0068 - val_loss:
0.0150
Epoch 168/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 215ms/step - loss: 0.0073 - val_loss:
0.0144
Epoch 169/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 204ms/step - loss: 0.0076 - val_loss:
0.0151
Epoch 170/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 205ms/step - loss: 0.0067 - val_loss:
0.0146
Epoch 171/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 193ms/step - loss: 0.0074 - val_loss:
0.0142
Epoch 172/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 195ms/step - loss: 0.0077 - val_loss:
0.0150
Epoch 173/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 195ms/step - loss: 0.0072 - val_loss:
0.0139
Epoch 174/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 192ms/step - loss: 0.0074 - val_loss:
0.0158
Epoch 175/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 209ms/step - loss: 0.0070 - val_loss:
0.0141
Epoch 176/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 284ms/step - loss: 0.0070 - val_loss:
0.0148
Epoch 177/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 4s 194ms/step - loss: 0.0067 - val_loss:
0.0149
Epoch 178/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 196ms/step - loss: 0.0065 - val_loss:
0.0155
Epoch 179/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 198ms/step - loss: 0.0074 - val_loss:
```

```
0.0149
Epoch 180/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 192ms/step - loss: 0.0068 - val_loss:
0.0153
Epoch 181/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 203ms/step - loss: 0.0072 - val_loss:
0.0151
Epoch 182/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 207ms/step - loss: 0.0068 - val_loss:
0.0152
Epoch 183/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 209ms/step - loss: 0.0068 - val_loss:
0.0164
Epoch 184/300
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 181ms/step - loss: 0.0077
```

```python
print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R2): {r2}")

# Plot the results
plt.figure(figsize=(14, 7))
plt.scatter(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
label='Actual', alpha=0.5)
plt.scatter(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
label='Predicted', alpha=0.5)
plt.plot(df_monthly.index[-len(y_test):], y_test_inv, color='blue',
alpha=0.7, label='Actual')
plt.plot(df_monthly.index[-len(y_test):], y_pred_inv, color='red',
alpha=0.7, label='Predicted')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Rainfall')
plt.title('Actual vs Predicted Monthly Rainfall using Enhanced LSTM +
GRU')
plt.show()
```

The model's performance metrics indicate its ability to predict monthly rainfall using LSTM. The Mean Squared Error (MSE) of 6.13 reflects the average squared difference between predicted and actual rainfall values, suggesting some variability in prediction accuracy. The Mean Absolute Error (MAE) of 1.57 signifies the average magnitude of prediction errors, indicating moderate accuracy in predicting rainfall amounts. The Root Mean Squared Error (RMSE) of 2.48 represents the standard deviation of prediction errors, suggesting moderate accuracy with some variability. The R-squared (R2) value of 0.28 indicates that approximately 28% of the variance in actual rainfall can be explained by the model, suggesting a fair but not strong fit to the data.

Results

Objective 1: The null hypothesis (H01) was rejected. Analysis showed a significant relationship between historical time series data, geospatial data, and the accuracy of the deep learning model in predicting future weather conditions.

Objective 2: The null hypothesis (H02) was rejected. Significant seasonal and geographical variations were identified in the historical weather data.

Objective 3: The null hypothesis (H04) was rejected. Significant differences in seasonal weather patterns were identified and effectively classified using K-means clustering techniques.

Recommendations to Farmers Based on the EDA Figures

Temperature Trends

Warmer Months: Temperatures peak in the middle of the year, particularly in May, June, and July, reaching maximums around 35°C. Cooler Months: The start and end of the year (January, February, November, and December) have lower maximum temperatures, often below 25°C.

Recommendations:

1. ***Crop Selection***: Farmers should consider planting heat-tolerant crops during the warmer months (May to July) and crops that require cooler temperatures during the start and end of the year.
2. ***Irrigation Management***: Increased evaporation rates during the warmer months suggest a need for more frequent irrigation. Implementing efficient irrigation systems can help manage water use.
3. ***Shade Structures***: Consider temporary shade structures for sensitive crops during the hottest months to prevent heat stress.

Rainfall by Month

Peak Rainfall: The data shows significant rainfall in June, July, and August, with frequent heavy rainfall events exceeding 100 mm. Dry Periods: January, February, and March are characterized by minimal rainfall.

Recommendations:

1. ***Planting Schedule***: Plan planting seasons around the expected rainfall. For rain-fed crops, sowing before the onset of the rainy season (April) ensures adequate water availability for germination and early growth.
2. ***Water Conservation***: Implement rainwater harvesting during peak rainfall months to store water for use during dry periods.
3. ***Soil Erosion Prevention***: Heavy rainfall in mid-year months can lead to soil erosion. Farmers should adopt practices like cover cropping, contour farming, and maintaining vegetation cover to minimize erosion.
4. ***Flood Management***: Prepare for potential flooding in low-lying areas by constructing proper drainage systems and raised beds for crops sensitive to waterlogging.

Recommendation on Machine Learning: Best model for predicting rainfall and Temperature

For predicting rainfall, the SARIMA model with parameters (order=(2, 1, 2), seasonal_order=(2, 1, 2, 12)) has been selected due to its effectiveness in capturing the underlying patterns and seasonality in the monthly rainfall data. This model has been tested and found to produce accurate forecasts, which are essential for planning and decision-making. Similarly, for predicting temperature, the SARIMA model with the same parameters (order=(2, 1, 2), seasonal_order=(2, 1, 2, 12)) has been used. This model was evaluated by comparing the predicted average temperatures against the actual values, with the Mean Squared Error (MSE) serving as the performance metric. The selected parameters allow the model to fit well to the historical temperature data, ensuring reliable and precise predictions. These models offer a robust approach to forecasting, effectively handling both trend and seasonal variations in the data.

Recommendation on deep Learning: Best model for predicting Temperature

Farmers can rely on the SARIMA model for accurate temperature forecasts, achieving a low Mean Squared Error (MSE) of 0.8991. This model effectively captures seasonal patterns by resampling data to monthly averages and fitting it to historical temperature trends. Utilizing an optimized SARIMA configuration (order=(2, 1, 2), seasonal_order=(2, 1, 2, 12)), the model delivers precise forecasts for future temperature trends

Recommendation on deep learning: Best model for predicting rainfall

Deep Learning Farmers can greatly benefit from adopting the Enhanced LSTM model for accurate rainfall predictions. This model stands out due to its ability to effectively capture seasonal variations by resampling data to monthly averages and normalizing it using Min-Max scaling, ensuring all features contribute optimally. It utilizes a sequence-based approach, leveraging 12 months of historical data to forecast future rainfall trends with precision. Featuring a robust architecture that includes multiple LSTM layers with dropout regularization, the model enhances generalization and mitigates overfitting. Trained over 300 epochs with careful optimization, it achieves a high R^2 score of 48%, indicating its reliability and accuracy in predicting rainfall patterns.

Recommendation on deep learning: Best model for predicting rainfall

Deep Learning Farmers can confidently rely on the Enhanced LSTM model for accurate temperature forecasts, given its exceptional performance with an impressive R^2 score of 91. This model effectively captures temporal patterns by utilizing sequences of 12 previous months of normalized temperature data. Featuring a robust architecture with three LSTM layers, each incorporating 20% dropout for enhanced generalization, the model ensures reliable performance and mitigates overfitting. Trained over 400 epochs with meticulous optimization using Adam optimizer (learning rate 0.001), it delivers precise predictions. Evaluation on test data reveals minimal errors with Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) metrics, demonstrating its strong predictive capability.

Implication of the study

The implications of the study are profound for the region's agricultural sector and water management practices. By providing accurate predictions of weather patterns, including critical events like droughts and extreme temperatures, the model can help farmers make informed decisions about crop selection, planting schedules, and irrigation management. This predictive

capability can enhance agricultural productivity, optimize water use, and reduce the risk of crop failure due to adverse weather conditions. Additionally, the study's findings can inform policy-making and strategic planning for regional agricultural development and climate resilience, ultimately contributing to the sustainability and economic stability of Hyderabad's agrarian communities.

Limitations of the study

Single Collection Point: The data used for modeling was collected from only one collection point in Hyderabad, India. This limits the model's ability to generalize across the entire region, as weather patterns can vary significantly even within short distances. Localized phenomena such as microclimates are not accounted for, which can lead to inaccuracies when applying the model to other areas.

Temporal Variations: While this is a time series model, it may not capture all the temporal variations affecting weather patterns. Factors such as climate change, seasonal shifts, and extreme weather events are complex and may introduce patterns not present in the historical data used to train the model. ***Data Quality and Completeness:*** The quality and completeness of the data significantly impact the model's performance. Incorrect, or biased data points can lead to inaccurate predictions.

Conclusion

The study leveraged advanced techniques including deep learning, machine learning, and exploratory data analysis (EDA) to model and predict weather patterns in Hyderabad India. These methodologies enabled the extraction of significant insights and patterns from historical weather data, facilitating more accurate predictions of variables such as temperature and rainfall. The results underscore the potential of these techniques in enhancing our understanding of weather dynamics and improving decision-making processes in various sectors, particularly agriculture. As technology and data collection methods continue to evolve, the integration of these advanced analytical approaches will become increasingly vital in addressing complex environmental challenges and optimizing resource management.