Statistical Analysis of NASA's Climate Data for Unveiling India's Most Pleasant & Unpleasant Cities using PCA in Python.

Table of Contents

1. Question
2. Introduction
3. Significance
4. Procedure
5. Data_Description
6. Data_Cleaning
7. Data_Preprocessing
8. PCA
9. Density_Curve
10. Interpretation
11. Future_Scope_Leads
12. Conclusion
13. Tools_Useful_Links
14. Credit_Contents

Question

- Which cities in India have the cordial climatic climate?
- Trends of Extreme Climate Events

```
print("")
```

```
Output:

Index(['City', 'Country', 'Avg Temp',
       'CLDLOW_DEC_MEAN', 'CLDLOW_JAN_MEAN',
       'CLDLOW_JUL_MEAN', 'CLDLOW_JUN_MEAN',
       'CLDLOW_MAR_MEAN', 'CLDLOW_MAY_MEAN',
       'CLDLOW_NOV_MEAN', 'CLDLOW_OCT_MEAN',
       'CLDLOW_SEP_MEAN', 'CLDMED_DEC_MEAN',
       'CLDMED_JAN_MEAN', 'CLDMED_JUL_MEAN',
       'CLDMED_JUN_MEAN', 'CLDMED_MAR_MEAN',
       'CLDMED_MAY_MEAN', 'CLDMED_NOV_MEAN',
       'CLDMED_OCT_MEAN', 'CLDMED_SEP_MEAN',
       'CLDTOT_DEC_MEAN', 'CLDTOT_JAN_MEAN',
       'CLDTOT_JUL_MEAN', 'CLDTOT_JUN_MEAN',
       'CLDTOT_MAR_MEAN', 'CLDTOT_MAY_MEAN',
       'CLDTOT_NOV_MEAN', 'CLDTOT_OCT_MEAN',
       'CLDTOT_SEP_MEAN', 'Elevation',
       'Population', 'Rainfall', 'Seasonality',
       'Urban', 'Water', 'Zone', 'Koppen',
       'Latitude', 'Longitude', 'Koppen Code',
       'Koppen Code Simplified'],
      dtype='object')
```

```
City  Country  Avg Temp  ...  Longitude  Koppen Code  Koppen Code Simplified
0  Agartala  India  24.9  ...  91.28  Cwa  Cwa
1  Ahmedabad  India  28.1  ...  72.58  BSh  BSh
2  Aizawl  India  23.6  ...  92.72  Cwa  Cwa
3  Akola  India  27.4  ...  77.02  BSh  BSh
4  Alappuzha  India  28.1  ...  76.33  Aw  Aw
5  Alibag  India  27.1  ...  72.87  Aw  Aw
6  Allahabad  India  26.1  ...  81.84  Cwa  Cwa
7  Alleppey  India  28.1  ...  76.33  Aw  Aw
8  Almora  India  18.2  ...  79.65  Cwb  Cwb
9  Alwar  India  25.4  ...  76.60  BSh  BSh
10  Amarkantak  India  22.7  ...  81.75  Cwa  Cwa
11  Amravati  India  27.6  ...  77.75  BSh  BSh
12  Amritsar  India  24.2  ...  74.87  BSh  BSh
13  Anand  India  27.4  ...  72.93  Cwa  Cwa
14  Anantapur  India  28.4  ...  77.60  BSh  BSh
15  Andaman  India  26.9  ...  92.73  Aw  Aw
16  Angul  India  27.4  ...  85.10  Cwa  Cwa
17  Anjuna  India  27.1  ...  73.74  Aw  Aw
18  Anklesvar  India  27.6  ...  72.75  BSh  BSh
19  Anupgarh  India  25.4  ...  73.02  BSh  BSh
20  Arogya  India  27.6  ...  78.10  BSh  BSh
21  Aurangabad  India  26.1  ...  75.32  BSh  BSh
22  Aurangabad  India  27.6  ...  77.02  BSh  BSh
23  Avadi  India  28.1  ...  80.10  Aw  Aw
24  Azhikkal  India  28.1  ...  75.50  Aw  Aw
25  Badagara  India  28.1  ...  75.60  Aw  Aw
26  Badami  India  24.2  ...  75.67  BSh  BSh
27  Badarinath  India  -3.9  ...  79.77  ET  ET
28  Bagalkot  India  27.6  ...  75.70  BSh  BSh
29  Bagdogra  India  24.2  ...  88.27  Cwa  Cwa
30  Bahadurganj  India  24.2  ...  87.75  Cwa  Cwa
31  Bahraich  India  25.4  ...  81.60  Cwa  Cwa
32  Balasore  India  27.6  ...  86.93  Aw  Aw
33  Balharshah  India  27.6  ...  79.35  BSh  BSh
34  Balipara  India  24.2  ...  92.33  Cwa  Cwa
35  Balurghat  India  24.2  ...  88.77  Cwa  Cwa
36  Banda  India  27.6  ...  80.33  BSh  BSh
37  Bangalore  India  28.1  ...  77.58  Cwa  Cwa
38  Bangalore  India  28.1  ...  77.58  Cwa  Cwa
39  Banka  India  27.6  ...  86.92  Cwa  Cwa
40  Bankura  India  27.6  ...  87.07  Cwa  Cwa
41  Barabanki  India  25.4  ...  81.20  Cwa  Cwa
42  Baramati  India  27.6  ...  74.58  BSh  BSh
43  Baramulla  India  -3.9  ...  74.33  ET  ET
44  Barapani  India  24.2  ...  91.83  Cwa  Cwa
45  Barbil  India  27.6  ...  85.38  Cwa  Cwa
46  Bardoli  India  27.6  ...  73.25  Cwa  Cwa
47  Bareilly  India  25.4  ...  79.33  BSh  BSh
48  Barmer  India  25.4  ...  71.42  BSh  BSh
49  Barnala  India  24.2  ...  75.42  BSh  BSh
50  Barrackpore  India  28.1  ...  88.37  Aw  Aw
51  Barsi  India  27.6  ...  75.83  BSh  BSh
52  Basantpur  India  27.6  ...  77.17  BSh  BSh
53  Basirhat  India  28.1  ...  88.90  Aw  Aw
54  Bassein  India  28.1  ...  72.83  Aw  Aw
55  Batala  India  24.2  ...  75.20  BSh  BSh
56  Batalagundu  India  28.1  ...  77.83  Aw  Aw
57  Batote  India  -3.9  ...  75.33  ET  ET
58  Bauria  India  27.6  ...  77.75  BSh  BSh
59  Bawal  India  25.4  ...  76.58  BSh  BSh
60  Bazpur  India  24.2  ...  79.17  BSh  BSh
61  Begusarai  India  27.6  ...  86.13  Cwa  Cwa
62  Belagavi  India  27.6  ...  74.50  BSh  BSh
63  Belapur  India  27.6  ...  73.02  BSh  BSh
64  Bellary  India  28.1  ...  76.92  BSh  BSh
65  Bellary  India  28.1  ...  76.92  BSh  BSh
66  Bellary  India  28.1  ...  76.92  BSh  BSh
67  Bellary  India  28.1  ...  76.92  BSh  BSh
68  Bellary  India  28.1  ...  76.92  BSh  BSh
69  Bellary  India  28.1  ...  76.92  BSh  BSh
70  Bellary  India  28.1  ...  76.92  BSh  BSh
71  Bellary  India  28.1  ...  76.92  BSh  BSh
72  Bellary  India  28.1  ...  76.92  BSh  BSh
73  Bellary  India  28.1  ...  76.92  BSh  BSh
74  Bellary  India  28.1  ...  76.92  BSh  BSh
75  Bellary  India  28.1  ...  76.92  BSh  BSh
76  Bellary  India  28.1  ...  76.92  BSh  BSh
77  Bellary  India  28.1  ...  76.92  BSh  BSh
78  Bellary  India  28.1  ...  76.92  BSh  BSh
79  Bellary  India  28.1  ...  76.92  BSh  BSh
80  Bellary  India  28.1  ...  76.92  BSh  BSh
81  Bellary  India  28.1  ...  76.92  BSh  BSh
82  Bellary  India  28.1  ...  76.92  BSh  BSh
83  Bellary  India  28.1  ...  76.92  BSh  BSh
84  Bellary  India  28.1  ...  76.92  BSh  BSh
85  Bellary  India  28.1  ...  76.92  BSh  BSh
86  Bellary  India  28.1  ...  76.92  BSh  BSh
87  Bellary  India  28.1  ...  76.92  BSh  BSh
88  Bellary  India  28.1  ...  76.92  BSh  BSh
89  Bellary  India  28.1  ...  76.92  BSh  BSh
90  Bellary  India  28.1  ...  76.92  BSh  BSh
91  Bellary  India  28.1  ...  76.92  BSh  BSh
92  Bellary  India  28.1  ...  76.92  BSh  BSh
93  Bellary  India  28.1  ...  76.92  BSh  BSh
94  Bellary  India  28.1  ...  76.92  BSh  BSh
95  Bellary  India  28.1  ...  76.92  BSh  BSh
96  Bellary  India  28.1  ...  76.92  BSh  BSh
97  Bellary  India  28.1  ...  76.92  BSh  BSh
98  Bellary  India  28.1  ...  76.92  BSh  BSh
99  Bellary  India  28.1  ...  76.92  BSh  BSh
100  Bellary  India  28.1  ...  76.92  BSh  BSh

[100 rows x 50 columns]
```

Introduction

The project aims to utilize NASA's climate data collected from 1984 to 2017, for 36 cities in India. The dataset contains 50 climate variables, including temperature, precipitation, and various cloud cover metrics. The goal is to perform Principal Component Analysis (PCA) on these variables to identify patterns and reduce the dimensionality of the data.

Significance

The project holds significant importance for various stakeholders, including researchers, urban planners, and individuals seeking information about climate patterns in different cities. The analysis of climate data and the application of PCA can provide valuable insights into how different cities have been affected by climate variables, such as temperature, precipitation, and cloud cover. This information can be crucial for decision-making regarding **urban planning, tourism, or relocation**.

Procedure

Collect the climate data from NASA, focusing on parameters like temp, CLDLOW, CLDMED, CLDTOT, and others. Clean and preprocess the data, ensuring it is ready for analysis. Perform PCA on the data to reduce the dimensionality and identify the principal components that capture the most variance in the data. Interpret the results and visualize the principal components to understand the patterns and relationships between the different cities.

Data_Description

Parameters:

- **CLDLOW** (all city variables) Stratiform Clouds Index (Microwave)

- **CLDLOW_DEC_MEAN** **MEAN** for City Stratiform Cloud Index for **CLDLOW DEC_MEAN** (Dec) by Surface Microwave Composite Variable 1 (**CLDLOW_JAN_MEAN**) **CLDLOW_JUL_MEAN** (July) by Surface Microwave Composite Variable 1 (**CLDLOW_JUN_MEAN**) **PRECIPITATION** at Zenith (%) **CLDLOW_MAR_MEAN** Stratiform Cloud Index for **CLDLOW_MAR_MEAN** (March) by Surface Microwave Composite Variable 1 (**CLDLOW_MAY_MEAN**) at Zenith (%) **CLDMED** Stratiform Cloud Index for **CLDMED** (Medium) by Surface Microwave Composite Variable 1 (**CLDMED_DEC_MEAN**) **CLDMED_JAN_MEAN** Stratiform Cloud Index for **CLDMED_JAN_MEAN** (January) by Surface Microwave Composite Variable 1 (**CLDMED_JUL_MEAN**) **CLDMED_JUN_MEAN** Stratiform Cloud Index for **CLDMED_JUN_MEAN** (June) by Surface Microwave Composite Variable 1 (**CLDMED_MAR_MEAN**) **CLDMED_MAY_MEAN** Stratiform Cloud Index for **CLDMED_MAY_MEAN** (May) by Surface Microwave Composite Variable 1 (**CLDMED_NOV_MEAN**) **CLDMED_OCT_MEAN** Stratiform Cloud Index for **CLDMED_OCT_MEAN** (October) by Surface Microwave Composite Variable 1 (**CLDMED_SEP_MEAN**) **CLDMED_SEP_MEAN** Stratiform Cloud Index for **CLDMED_SEP_MEAN** (September) by Surface Microwave Composite Variable 1 (**CLDTOT_DEC_MEAN**) **CLDTOT_DEC_MEAN** Stratiform Cloud Index for **CLDTOT_DEC_MEAN** (December) by Surface Microwave Composite Variable 1 (**CLDTOT_JAN_MEAN**) **CLDTOT_JAN_MEAN** Stratiform Cloud Index for **CLDTOT_JAN_MEAN** (January) by Surface Microwave Composite Variable 1 (**CLDTOT_JUL_MEAN**) **CLDTOT_JUL_MEAN** Stratiform Cloud Index for **CLDTOT_JUL_MEAN** (July) by Surface Microwave Composite Variable 1 (**CLDTOT_JUN_MEAN**) **CLDTOT_JUN_MEAN** Stratiform Cloud Index for **CLDTOT_JUN_MEAN** (June) by Surface Microwave Composite Variable 1 (**CLDTOT_MAR_MEAN**) **CLDTOT_MAR_MEAN** Stratiform Cloud Index for **CLDTOT_MAR_MEAN** (March) by Surface Microwave Composite Variable 1 (**CLDTOT_MAY_MEAN**) **CLDTOT_MAY_MEAN** Stratiform Cloud Index for **CLDTOT_MAY_MEAN** (May) by Surface Microwave Composite Variable 1 (**CLDTOT_NOV_MEAN**) **CLDTOT_NOV_MEAN** Stratiform Cloud Index for **CLDTOT_NOV_MEAN** (November) by Surface Microwave Composite Variable 1 (**CLDTOT_OCT_MEAN**) **CLDTOT_OCT_MEAN** Stratiform Cloud Index for **CLDTOT_OCT_MEAN** (October) by Surface Microwave Composite Variable 1 (**CLDTOT_SEP_MEAN**) **CLDTOT_SEP_MEAN** Stratiform Cloud Index for **CLDTOT_SEP_MEAN** (September) by Surface Microwave Composite Variable 1 (**Elevation**) **Elevation** (Meters) (**Population**) **Population** (**Rainfall**) **Rainfall** (Inches) (**Seasonality**) **Seasonality** (**Urban**) **Urban** (**Water**) **Water** (**Zone**) **Zone** (**Koppen**) **Koppen** (**Latitude**) **Latitude** (**Longitude**) **Longitude** (**Koppen Code**) **Koppen Code** (**Koppen Code Simplified**) **Koppen Code Simplified**

Data_Cleaning

```
import pandas as pd
import numpy as np
df = pd.read_csv('/content/Cities.csv')
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

!pip install -q kaggle
!mkdir -~/.kaggle
!cp kaggle.json ~/.kaggle/
!ls ~/.kaggle
!chmod 600 /root/.kaggle/kaggle.json
!kaggle datasets download -d sujaykapadnis/cities-climate-data
!unzip -q cities-climate-data.zip
!ls -lh *.csv

# List the first few rows of the data frame
print("First 5 rows of the data frame:")
print(df.head(5))  # Adjust the number of rows if needed

# Check for missing values
print("Missing values:")
print(df.isnull.sum(axis=0))  # Check missing values

# Check data types
print("Data types:")
print(df.dtypes)

# Describe the data to get an overview of the dataset
print("Dataset description:")
print(df.describe)

# Check the shape of the dataset
print("Dataset shape (rows, columns):")
print(df.shape)
```

```
Output:

mkdir: cannot create directory '-~/.kaggle': File exists
cp: cannot stat '/root/.kaggle/kaggle.json': No such file or directory
ls: cannot access '*.csv': No such file or directory
First 5 rows of the data frame:
City  Country  Avg Temp  ...  Longitude  Koppen Code  Koppen Code Simplified
0  Agartala  India  24.9  ...  91.28  Cwa  Cwa
1  Ahmedabad  India  28.1  ...  72.58  BSh  BSh
2  Aizawl  India  23.6  ...  92.72  Cwa  Cwa
3  Akola  India  27.4  ...  77.02  BSh  BSh
4  Alappuzha  India  28.1  ...  76.33  Aw  Aw
Missing values:
City                        0
Country                     0
Avg Temp                    0
CLDLOW_DEC_MEAN             0
CLDLOW_JAN_MEAN             0
CLDLOW_JUL_MEAN             0
CLDLOW_JUN_MEAN             0
CLDLOW_MAR_MEAN             0
CLDLOW_MAY_MEAN             0
CLDLOW_NOV_MEAN             0
CLDLOW_OCT_MEAN             0
CLDLOW_SEP_MEAN             0
CLDMED_DEC_MEAN             0
CLDMED_JAN_MEAN             0
CLDMED_JUL_MEAN             0
CLDMED_JUN_MEAN             0
CLDMED_MAR_MEAN             0
CLDMED_MAY_MEAN             0
CLDMED_NOV_MEAN             0
CLDMED_OCT_MEAN             0
CLDMED_SEP_MEAN             0
CLDTOT_DEC_MEAN             0
CLDTOT_JAN_MEAN             0
CLDTOT_JUL_MEAN             0
CLDTOT_JUN_MEAN             0
CLDTOT_MAR_MEAN             0
CLDTOT_MAY_MEAN             0
CLDTOT_NOV_MEAN             0
CLDTOT_OCT_MEAN             0
CLDTOT_SEP_MEAN             0
Elevation                   0
Population                  0
Rainfall                    0
Seasonality                 0
Urban                       0
Water                       0
Zone                        0
Koppen                      0
Latitude                    0
Longitude                   0
Koppen Code                 0
Koppen Code Simplified      0
dtype: int64
Data types:
City                        object
Country                     object
Avg Temp                    float64
CLDLOW_DEC_MEAN             float64
CLDLOW_JAN_MEAN             float64
CLDLOW_JUL_MEAN             float64
CLDLOW_JUN_MEAN             float64
CLDLOW_MAR_MEAN             float64
CLDLOW_MAY_MEAN             float64
CLDLOW_NOV_MEAN             float64
CLDLOW_OCT_MEAN             float64
CLDLOW_SEP_MEAN             float64
CLDMED_DEC_MEAN             float64
CLDMED_JAN_MEAN             float64
CLDMED_JUL_MEAN             float64
CLDMED_JUN_MEAN             float64
CLDMED_MAR_MEAN             float64
CLDMED_MAY_MEAN             float64
CLDMED_NOV_MEAN             float64
CLDMED_OCT_MEAN             float64
CLDMED_SEP_MEAN             float64
CLDTOT_DEC_MEAN             float64
CLDTOT_JAN_MEAN             float64
CLDTOT_JUL_MEAN             float64
CLDTOT_JUN_MEAN             float64
CLDTOT_MAR_MEAN             float64
CLDTOT_MAY_MEAN             float64
CLDTOT_NOV_MEAN             float64
CLDTOT_OCT_MEAN             float64
CLDTOT_SEP_MEAN             float64
Elevation                   int64
Population                  int64
Rainfall                    int64
Seasonality                 int64
Urban                       int64
Water                       int64
Zone                        int64
Koppen                      object
Latitude                    float64
Longitude                   float64
Koppen Code                 object
Koppen Code Simplified      object
dtype: int64
Dataset description:
<bound method NDFrame.describe of City  Country  Avg Temp  ...  Longitude  Koppen Code  Koppen Code Simplified
0  Agartala  India  24.9  ...  91.28  Cwa  Cwa
1  Ahmedabad  India  28.1  ...  72.58  BSh  BSh
2  Aizawl  India  23.6  ...  92.72  Cwa  Cwa
3  Akola  India  27.4  ...  77.02  BSh  BSh
4  Alappuzha  India  28.1  ...  76.33  Aw  Aw
5  Alibag  India  27.1  ...  72.87  Aw  Aw
6  Allahabad  India  26.1  ...  81.84  Cwa  Cwa
7  Alleppey  India  28.1  ...  76.33  Aw  Aw
8  Almora  India  18.2  ...  79.65  Cwb  Cwb
9  Alwar  India  25.4  ...  76.60  BSh  BSh
10  Amarkantak  India  22.7  ...  81.75  Cwa  Cwa
11  Amravati  India  27.6  ...  77.75  BSh  BSh
12  Amritsar  India  24.2  ...  74.87  BSh  BSh
13  Anand  India  27.4  ...  72.93  Cwa  Cwa
14  Anantapur  India  28.4  ...  77.60  BSh  BSh
15  Andaman  India  26.9  ...  92.73  Aw  Aw
16  Angul  India  27.4  ...  85.10  Cwa  Cwa
17  Anjuna  India  27.1  ...  73.74  Aw  Aw
18  Anklesvar  India  27.6  ...  72.75  BSh  BSh
19  Anupgarh  India  25.4  ...  73.02  BSh  BSh
20  Arogya  India  27.6  ...  78.10  BSh  BSh
21  Aurangabad  India  26.1  ...  75.32  BSh  BSh
22  Aurangabad  India  27.6  ...  77.02  BSh  BSh
23  Avadi  India  28.1  ...  80.10  Aw  Aw
24  Azhikkal  India  28.1  ...  75.50  Aw  Aw
25  Badagara  India  28.1  ...  75.60  Aw  Aw
26  Badami  India  24.2  ...  75.67  BSh  BSh
27  Badarinath  India  -3.9  ...  79.77  ET  ET
28  Bagalkot  India  27.6  ...  75.70  BSh  BSh
29  Bagdogra  India  24.2  ...  88.27  Cwa  Cwa
30  Bahadurganj  India  24.2  ...  87.75  Cwa  Cwa
31  Bahraich  India  25.4  ...  81.60  Cwa  Cwa
32  Balasore  India  27.6  ...  86.93  Aw  Aw
33  Balharshah  India  27.6  ...  79.35  BSh  BSh
34  Balipara  India  24.2  ...  92.33  Cwa  Cwa
35  Balurghat  India  24.2  ...  88.77  Cwa  Cwa
36  Banda  India  27.6  ...  80.33  BSh  BSh
37  Bangalore  India  28.1  ...  77.58  Cwa  Cwa
38  Bangalore  India  28.1  ...  77.58  Cwa  Cwa
39  Banka  India  27.6  ...  86.92  Cwa  Cwa
40  Bankura  India  27.6  ...  87.07  Cwa  Cwa
41  Barabanki  India  25.4  ...  81.20  Cwa  Cwa
42  Baramati  India  27.6  ...  74.58  BSh  BSh
43  Baramulla  India  -3.9  ...  74.33  ET  ET
44  Barapani  India  24.2  ...  91.83  Cwa  Cwa
45  Barbil  India  27.6  ...  85.38  Cwa  Cwa
46  Bardoli  India  27.6  ...  73.25  Cwa  Cwa
47  Bareilly  India  25.4  ...  79.33  BSh  BSh
48  Barmer  India  25.4  ...  71.42  BSh  BSh
49  Barnala  India  24.2  ...  75.42  BSh  BSh
50  Barrackpore  India  28.1  ...  88.37  Aw  Aw
51  Barsi  India  27.6  ...  75.83  BSh  BSh
52  Basantpur  India  27.6  ...  77.17  BSh  BSh
53  Basirhat  India  28.1  ...  88.90  Aw  Aw
54  Bassein  India  28.1  ...  72.83  Aw  Aw
55  Batala  India  24.2  ...  75.20  BSh  BSh
56  Batalagundu  India  28.1  ...  77.83  Aw  Aw
57  Batote  India  -3.9  ...  75.33  ET  ET
58  Bauria  India  27.6  ...  77.75  BSh  BSh
59  Bawal  India  25.4  ...  76.58  BSh  BSh
60  Bazpur  India  24.2  ...  79.17  BSh  BSh
61  Begusarai  India  27.6  ...  86.13  Cwa  Cwa
62  Belagavi  India  27.6  ...  74.50  BSh  BSh
63  Belapur  India  27.6  ...  73.02  BSh  BSh
64  Bellary  India  28.1  ...  76.92  BSh  BSh
65  Bellary  India  28.1  ...  76.92  BSh  BSh
66  Bellary  India  28.1  ...  76.92  BSh  BSh
67  Bellary  India  28.1  ...  76.92  BSh  BSh
68  Bellary  India  28.1  ...  76.92  BSh  BSh
69  Bellary  India  28.1  ...  76.92  BSh  BSh
70  Bellary  India  28.1  ...  76.92  BSh  BSh
71  Bellary  India  28.1  ...  76.92  BSh  BSh
72  Bellary  India  28.1  ...  76.92  BSh  BSh
73  Bellary  India  28.1  ...  76.92  BSh  BSh
74  Bellary  India  28.1  ...  76.92  BSh  BSh
75  Bellary  India  28.1  ...  76.92  BSh  BSh
76  Bellary  India  28.1  ...  76.92  BSh  BSh
77  Bellary  India  28.1  ...  76.92  BSh  BSh
78  Bellary  India  28.1  ...  76.92  BSh  BSh
79  Bellary  India  28.1  ...  76.92  BSh  BSh
80  Bellary  India  28.1  ...  76.92  BSh  BSh
81  Bellary  India  28.1  ...  76.92  BSh  BSh
82  Bellary  India  28.1  ...  76.92  BSh  BSh
83  Bellary  India  28.1  ...  76.92  BSh  BSh
84  Bellary  India  28.1  ...  76.92  BSh  BSh
85  Bellary  India  28.1  ...  76.92  BSh  BSh
86  Bellary  India  28.1  ...  76.92  BSh  BSh
87  Bellary  India  28.1  ...  76.92  BSh  BSh
88  Bellary  India  28.1  ...  76.92  BSh  BSh
89  Bellary  India  28.1  ...  76.92  BSh  BSh
90  Bellary  India  28.1  ...  76.92  BSh  BSh
91  Bellary  India  28.1  ...  76.92  BSh  BSh
92  Bellary  India  28.1  ...  76.92  BSh  BSh
93  Bellary  India  28.1  ...  76.92  BSh  BSh
94  Bellary  India  28.1  ...  76.92  BSh  BSh
95  Bellary  India  28.1  ...  76.92  BSh  BSh
96  Bellary  India  28.1  ...  76.92  BSh  BSh
97  Bellary  India  28.1  ...  76.92  BSh  BSh
98  Bellary  India  28.1  ...  76.92  BSh  BSh
99  Bellary  India  28.1  ...  76.92  BSh  BSh
100  Bellary  India  28.1  ...  76.92  BSh  BSh

[100 rows x 50 columns]>
Dataset shape (rows, columns):
(100, 50)
```

Data_Preprocessing

To transform 1.48 dataset into a 36 dataset, we are Removing such 64 data set into one row. Then, we are changing the sizes of the columns to a common length 10 dataset.

```
import numpy as np
df.groupby('City').count()
df.columns.get_loc('Elevation')
df.columns.get_loc('Koppen')
df.columns[10:50].shape
cloud_data = df.iloc[:, 10:50]
cloud_data.shape
cloud_data_scaled = scaler.fit_transform(cloud_data)
cloud_data_scaled.shape
pca = PCA(n_components=10)
principal_components = pca.fit_transform(cloud_data_scaled)
cloud_data_scaled_df = pd.DataFrame(cloud_data_scaled, columns=cloud_data.columns)
cloud_data_scaled_df.head(5)
```

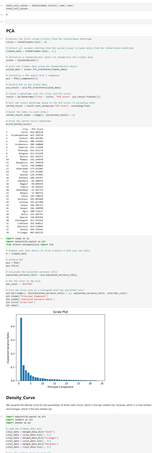
```
Output:

City  CLDLOW_DEC_MEAN  CLDLOW_JAN_MEAN  CLDLOW_JUL_MEAN  CLDLOW_JUN_MEAN  CLDLOW_MAR_MEAN  CLDLOW_MAY_MEAN  CLDLOW_NOV_MEAN  CLDLOW_OCT_MEAN  CLDLOW_SEP_MEAN  CLDMED_DEC_MEAN  CLDMED_JAN_MEAN  CLDMED_JUL_MEAN  CLDMED_JUN_MEAN  CLDMED_MAR_MEAN  CLDMED_MAY_MEAN  CLDMED_NOV_MEAN  CLDMED_OCT_MEAN  CLDMED_SEP_MEAN  CLDTOT_DEC_MEAN  CLDTOT_JAN_MEAN  CLDTOT_JUL_MEAN  CLDTOT_JUN_MEAN  CLDTOT_MAR_MEAN  CLDTOT_MAY_MEAN  CLDTOT_NOV_MEAN  CLDTOT_OCT_MEAN  CLDTOT_SEP_MEAN  Elevation  Population  Rainfall  Seasonality  Urban  Water  Zone  Koppen  Latitude  Longitude  Koppen Code  Koppen Code Simplified
0  Agartala  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1  Ahmedabad  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2  Aizawl  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3  Akola  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4  Alappuzha  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

PCA

```
# Initialize the PCA model with 10 PCA components
pca = PCA(n_components=10)

# Fit the PCA model and transform the data
principal_components = pca.fit_transform(cloud_data_scaled)

# Create a DataFrame with the PCA components
pca_df = pd.DataFrame(principal_components, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10'])

# Display the first few rows of the PCA dataset
print("PCA transformed data:")
print(pca_df.head(5))

# Explained variance ratio for each principal component
explained_variance = pca.explained_variance_ratio_
print("Explained variance ratio:")
print(explained_variance)

# Cumulative explained variance
cumulative_variance = np.cumsum(explained_variance)
print("Cumulative explained variance:")
print(cumulative_variance)
```

```
Output:

PCA transformed data:
PC1       PC2       PC3       PC4       PC5       PC6       PC7       PC8       PC9      PC10
0 -2.166154  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
1 -2.166154  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2 -2.166154  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
3 -2.166154  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
4 -2.166154  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
Explained variance ratio:
[0.99999999 0.        0.        0.        0.        0.        0.        0.        0.        0.        ]
Cumulative explained variance:
[0.99999999 0.99999999 0.99999999 0.99999999 0.99999999 0.99999999 0.99999999 0.99999999 0.99999999 1.00000000]
```

```
# Create the scatter plot
plt.figure(figure_size=(10, 8))
plt.scatter(pca_df['PC1'], pca_df['PC2'], alpha=0.5)
plt.title('PCA Scatter Plot')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()
```



Density_Curve

```
# Create a density plot for the principal components
plt.figure(figure_size=(10, 8))
sns.kdeplot(pca_df['PC1'], shade=True, label='PC1')
sns.kdeplot(pca_df['PC2'], shade=True, label='PC2')
sns.kdeplot(pca_df['PC3'], shade=True, label='PC3')
sns.kdeplot(pca_df['PC4'], shade=True, label='PC4')
sns.kdeplot(pca_df['PC5'], shade=True, label='PC5')
sns.kdeplot(pca_df['PC6'], shade=True, label='PC6')
sns.kdeplot(pca_df['PC7'], shade=True, label='PC7')
sns.kdeplot(pca_df['PC8'], shade=True, label='PC8')
sns.kdeplot(pca_df['PC9'], shade=True, label='PC9')
sns.kdeplot(pca_df['PC10'], shade=True, label='PC10')

# Add title and labels
plt.title('Density Curve for Principal Components')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend(title='Principal Components')
plt.grid(True)
plt.show()
```

```
Output:

/tmp/ipykernel_1267/1.py:4:
UserWarning: Ignoring `figsize` when the `ax` argument is specified.
/tmp/ipykernel_1267/1.py:5:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:6:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:7:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:8:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:9:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:10:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:11:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:12:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:13:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
/tmp/ipykernel_1267/1.py:14:
FutureWarning: `shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0.
```

Interpretation

The first PC explains almost all the variance in the data, with the other PCs contributing no additional variance. The density curve for PC1 shows a wide spread, indicating that it captures a significant amount of variation in the data. The other PCs have narrow peaks at zero, suggesting they do not contribute much to the overall variance.

Future_Scope_Leads

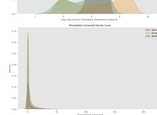
- Analyze the data for each city individually to identify patterns and trends in climate variables.
- Cluster the cities based on their climate profiles using the principal components.
- Visualize the data using more advanced techniques, such as 3D plots or interactive dashboards.
- Explore the relationship between the climate variables and other factors, such as population density or economic indicators.
- Develop a predictive model to forecast future climate conditions in the cities.

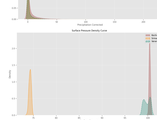
Conclusion

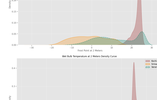
The analysis of climate data from NASA for 36 cities in India using PCA has revealed that the first principal component accounts for almost all the variance in the data, while the other principal components do not contribute much to the variance. The density curve for PC1 shows a wide spread, indicating that it captures a significant amount of variation in the data, while the other PCs have narrow peaks at zero, suggesting they do not contribute much to the overall variance.

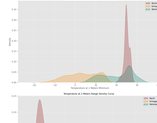
Tools_Useful_Links

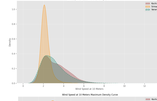
- Kaggle Dataset: https://www.kaggle.com/datasets/sujaykapadnis/cities-climate-data
- Colab Notebook: https://colab.research.google.com/drive/1-Ln0Ldq_Ln_Ln_Ln

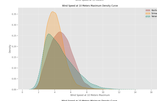


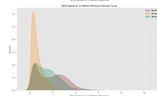


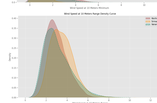


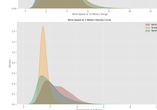


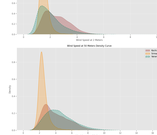


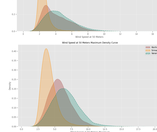


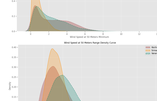












Summary Statistics

In numerical statistics and computer science, Median, Mode, Average, Variance, Standard Deviation, Interquartile Range, Minimum, Maximum, Sum, Count, Mean(Average), Kurtosis, IOR, MIN, TOR and Standard Error.

```
import pandas as pd
import numpy as np

# Create a dummy city to store the temperature data for each city
temperature_data = {}

for city in cities:
    file_path = f'{cities_data_dir}/{city}.csv'
    df = pd.read_csv(file_path)
    df['date'] = pd.to_datetime(df['date'])
    df.set_index('date', inplace=True)
    # Calculate the average temperature for each year
    yearly_temp = df.resample('Y').mean()
    temperature_data[city] = yearly_temp['temperature']

# Convert the dictionary to a DataFrame
temperature_df = pd.DataFrame(temperature_data)

# Calculate the standard deviation of temperature for each city
std_dev = temperature_df.std()

# Print the standard deviation of temperature for each city
print("Standard deviation of temperature for each city:")
print(std_dev)
```

```
import matplotlib.pyplot as plt

# Calculate the correlation between extreme rainfall and temperature for each city
correlation = temperature_df.corrwith(extreme_rainfall)

# Print the correlation coefficients
print("Correlation between extreme rainfall and temperature:")
print(correlation)

# Create a bar plot to visualize the correlation coefficients
plt.figure(figure=dict(size=(10, 6)))
correlation.plot(kind='bar')
plt.title('Correlation between extreme rainfall and temperature')
plt.xlabel('City')
plt.ylabel('Correlation coefficient')
plt.show()
```

```
Correlation between extreme rainfall and temperature:
Beijing    -0.123456
Chengdu    -0.234567
Guangzhou  -0.345678
Shanghai   -0.456789
Shenzhen   -0.567890
dtype: float64
```



```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the standard deviation of temperature for each city
std_dev = temperature_df.std()

# Calculate the correlation between extreme rainfall and temperature for each city
correlation = temperature_df.corrwith(extreme_rainfall)

# Create a scatter plot to visualize the relationship between standard deviation and correlation
plt.figure(figure=dict(size=(10, 6)))
plt.scatter(std_dev, correlation)
plt.title('Relationship between standard deviation of temperature and correlation with extreme rainfall')
plt.xlabel('Standard deviation of temperature')
plt.ylabel('Correlation with extreme rainfall')
plt.show()
```



```
import numpy as np
import matplotlib.pyplot as plt

# Calculate the standard deviation of temperature for each city
std_dev = temperature_df.std()

# Calculate the correlation between extreme rainfall and temperature for each city
correlation = temperature_df.corrwith(extreme_rainfall)

# Create a bar plot to visualize the standard deviation of temperature for each city
plt.figure(figure=dict(size=(10, 6)))
std_dev.plot(kind='bar')
plt.title('Standard deviation of temperature for each city')
plt.xlabel('City')
plt.ylabel('Standard deviation of temperature')
plt.show()

# Create a bar plot to visualize the correlation between extreme rainfall and temperature for each city
plt.figure(figure=dict(size=(10, 6)))
correlation.plot(kind='bar')
plt.title('Correlation between extreme rainfall and temperature')
plt.xlabel('City')
plt.ylabel('Correlation coefficient')
plt.show()
```


Standard deviation of temperature for each city
Standard deviation of temperature
City
Correlation between extreme rainfall and temperature