Project Climate Analysis

Objective:

This project aims to select a dataset from Kaggle and apply the various data science concepts we have learned in class. Our journey will include data preprocessing, cleaning, utilizing MS Excel, navigating UNIX CLI, creating visualization charts, and building a machine-learning model. The outcome will be summarized in a 2–3 page report detailing the insights and methodologies applied. The project's Jupyter Notebook should be included in the appendix.

Step 1: Choosing the Right Dataset

Dataset selection is a critical step in this project, as it forms the bedrock of our analysis. Kaggle, with its vast array of datasets, was our starting point. I considered several options, including Popular Movies of IMDb, World Weather Repository, Reddit - Data is Beautiful and Ugly, Sleep Health and Lifestyle Dataset, and Climate Change Indicators.

To ensure alignment with the project goals, I conducted an initial exploratory data analysis (EDA) for each dataset in the Jupyter Notebook.

My review process involved loading the datasets into the notebook to assess their size, structure, and a snapshot of their contents. I then graded each dataset as either pass (P) or not pass (NP). This led to the exclusion of two datasets. The Reddit - Data is Beautiful and Ugly dataset was removed due to its heavy reliance on broken URL links. The Sleep Health and Lifestyle Dataset, despite its interesting content, was dismissed due to its small size, which limited its potential for in-depth analysis.

Next, I examined the remaining datasets for their data types, formats, and any missing values. I also conducted a statistical summary to evaluate the diversity of features available, which is crucial for our project's scope. I then discarded the IMDb dataset due to its heavy focus on categorical data, which did not match my project needs.

For the Climate Change Indicators, I analyzed the distribution of data from the first and last recorded years to assess changes over time. Similarly, I explored the World Weather Repository, focusing on the variety of climates and the distribution of temperature and humidity across countries. Both datasets proved exciting and relevant together mainly since they include country-specific data that could be useful for cross-dataset analysis.

Ultimately, despite the presence of outliers not being a deciding factor at this stage, I chose to proceed with both the Climate and Weather datasets. They offer rich opportunities for applying the data science skills we have learned, including visualization and machine learning.

For a detailed look at my analysis process and decision-making during Step 1, refer to the “Project_Step1” notebook in the appendix. The next step will involve a more thorough EDA to dig deeper into these datasets and start unraveling the stories they can tell.

Step 2: Data Cleaning Process

This phase unveiled that temperature changes are referenced against a baseline period from 1951 to 1980. Tackling the data cleaning journey, I turned to MS Excel, starting with the Climate dataset. I removed columns with redundant information, like ‘Indicator,’ ‘Unit,’ ‘Source,’ ‘CTS_Name,’ and ‘CTS_Full_Descriptor,’ as they contained uniform data across all rows. These details were assigned to the appendix as source references. I encountered peculiar formatting in the year columns, where each year was prefixed with a letter ‘F,’ which I promptly removed using the ‘Find and Replace’ feature.

To enrich the dataset, I aimed to include geographical regions and sub-regions, acknowledging that climate patterns transcend political borders and are influenced by geographical location. I found a complementary dataset (‘ISO country region code all.csv’) that provided this geographical classification. Initially, I attempted to merge this data in Excel, but Tableau Prep proved to be a more user-friendly tool for the task, especially with Power Query’s limitations in comparison. The merge was executed with an inner join, matching 224 countries but excluding the ‘World (WLD)’ average and

other entities not covered in our climate data. This process classified the countries into five regions and 17 sub-regions.

Regarding the Weather dataset, I manually added the ISO3 country codes to facilitate future analysis, navigating through 97 country name discrepancies between the datasets. This step, too, benefited from the intuitive interface of Tableau Prep, which also assisted in transforming the Climate dataset into a “melted” format needed for the visual analysis. Screenshots of “Project_Step2.1.png” and “Project_Step2.2.png” from MS Excel and Tableau Prep are provided in the appendix.

Step 3: Exploratory Visualizations

The visualization efforts are showcased in the accompanying Jupyter Notebook Project_Step3.ipynb attached to the appendix, illustrating the initial graphical exploration of our datasets. Figures.

Step 4: Preliminary Findings or Observations

Information on temperature changes in this dataset is based on a baseline climatology corresponding to the period 1951-1980. During this step, it became clear that Climate Change is not an easy subject. Even though it also became clear that temperatures were increasing over these five decades. I wanted to understand if the increase was true for all the countries and which ones had the most increase. I tried two types of pair plots to identify outliers and to get distribution insights. Using the pair plot with a sub-region was an excellent way to determine its relevance for the subject. The group of countries marking the most significant change in 1970 was totally different from the one marked in 2022, and this insight should be explored more with machine learning. I noted that I could explore more the group of the regions with more Spread, the ones that had more than 50% of marking above 1 or below -1, separated from the countries that had changed within 1ºC above or below. Another group that could help me explore was the ones with the most changes in 2022: Andorra, Austria, France, Germany, Lichtenstein, Monaco, Netherlands, Spain, and Switzerland, dropping the ones that have too many missing values. A line plot by region confirms that temperature

changes in Europe and Asia show a remarkable growth compared to other places. I plotted a cluster with 8 and 3 that confirmed this inference. Finally, I used a Z-Score to calculate anomalies in temperature change over the years. The scatter plot showed a very interesting view of the changes in climate but was too busy and confusing to read. For readability, I then printed a list of the anomalies, which confirmed the most significant growth in Europe and Asia in the later years. Here, I was able to observe that 2016 and 2020 were years with more anomalies than others.

Lastly, I printed a heat map of all countries and temperature changes and sorted them by the highest temperature changes in 2020, the most recent year with the most anomalies. I wanted to take notice of the countries in the top right and the lower left as two groups to be observed further.

I worked on an analysis for the weather dataset, hoping that a list of the top poor air quality countries could be the same and correlated to the list of highest climate change, but my findings were inconclusive. Many different calculations and plots were done; only one remains in the notebook Project_Step3.ipynb. I left this study as a suggestion for the next project and proceeded with this project with only the Climate dataset.

Step 5: Machine Learning

The machine learning analysis, documented in the dedicated Jupyter Notebook Project_Step5.ipynb in the appendix, explores the application of a predictive models to the climate data.

Step 6: Conclusion and Next

For this part of the project, I wanted to work with a ARIMA model which is good fit for climate predictions. I considered that this could more complex than I able to handle is my first semester learning ML models. My second choice was to work with a Random Forest Regression which could also be well suitable to this kind of dataset.

After running the model in the notebook Project_Step5, I got an acceptable mean squared error of 0.1050. Given the scale of climate change values varying from -1 to 3ºC, this is a good result.

Working with climate change is an extensive undertaking. For the following steps, I should explore the weather conditions to find relevant variables to train the models. Moreover, I should explore at least three more kinds of models, especially Neural Networks or Tensor Flows.

References

Dataset Climate: climate_change_indicators.csv
https://www.kaggle.com/datasets/tarunrm09/climate-change-indicators
Indicator: Temperature change with respect to a baseline climatology, corresponding to the period 1951-1980
Unit: Degree Celsius
Source: Food and Agriculture Organization of the United Nations (FAO). 2022. FAOSTAT Climate Change, Climate Indicators, Temperature change. License: CC BY-NC-SA 3.0 IGO. Extracted
CTS_Code: ECCS
CTS_Name: Surface Temperature Change
CTS_Full_Descriptor: Environment, Climate Change, Climate Indicators, Surface Temperature Change

Dataset ISO Region: ISO contry region code all.csv
https://github.com/lukes/ISO-3166-Countries-with-Regional-Codes/blob/master/README.md

Appendix

(Next page)

Project_Step1

0.0.1 1. Import Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
```

0.0.2 2. Load Datasets

[120]:
```
# Importing data frames
df_climate = pd.read_csv("climate_change_indicators.csv")
df_ugly = pd.read_csv("data_is_ugly.csv")
df_weather = pd.read_csv("GlobalWeatherRepository.csv")
df_sleep = pd.read_csv("Sleep_health_and_lifestyle_dataset.csv")
```

[125]:
```
df_imdb = pd.read_csv("IMDb_updated.csv")
```

0.0.3 3. Check Size and Structure

```
# Preview dataset Climate
print("Dataset Size: {} rows, {} columns".format(df_climate.shape[0],␣
```

[90]:
```
 ↪df_climate.shape[1]))
print(df_climate.head())
# Pass
```

```
Dataset Size: 225 rows, 72 columns
   ObjectId                        Country ISO2 ISO3  \
0         1  Afghanistan, Islamic Rep. of   AF  AFG
1         2                       Albania   AL  ALB
2         3                       Algeria   DZ  DZA
3         4                American Samoa   AS  ASM
4         5     Andorra, Principality of   AD  AND

                                           Indicator           Unit  \
0  Temperature change with respect to a baseline …  Degree Celsius
1  Temperature change with respect to a baseline …  Degree Celsius
2  Temperature change with respect to a baseline …  Degree Celsius
```

```
3  Temperature change with respect to a baseline …  Degree Celsius
4  Temperature change with respect to a baseline …  Degree Celsius

                                                 Source CTS_Code  \
0  Food and Agriculture Organization of the Unite…     ECCS
1  Food and Agriculture Organization of the Unite…     ECCS
2  Food and Agriculture Organization of the Unite…     ECCS
3  Food and Agriculture Organization of the Unite…     ECCS
4  Food and Agriculture Organization of the Unite…     ECCS

                    CTS_Name  \
0  Surface Temperature Change
1  Surface Temperature Change
2  Surface Temperature Change
3  Surface Temperature Change
4  Surface Temperature Change

                                CTS_Full_Descriptor  …  F2013  F2014  \
0  Environment, Climate Change, Climate Indicator…  …  1.281  0.456
1  Environment, Climate Change, Climate Indicator…  …  1.333  1.198
2  Environment, Climate Change, Climate Indicator…  …  1.192  1.690
3  Environment, Climate Change, Climate Indicator…  …  1.257  1.170
4  Environment, Climate Change, Climate Indicator…  …  0.831  1.946

   F2015  F2016  F2017  F2018  F2019  F2020  F2021  F2022
0  1.093  1.555  1.540  1.544  0.910  0.498  1.327  2.012
1  1.569  1.464  1.121  2.028  1.675  1.498  1.536  1.518
2  1.121  1.757  1.512  1.210  1.115  1.926  2.330  1.688
3  1.009  1.539  1.435  1.189  1.539  1.430  1.268  1.256
4  1.690  1.990  1.925  1.919  1.964  2.562  1.533  3.243

[5 rows x 72 columns]
```

[92]:
```
# Preview dataset Ugly
print("Dataset Size: {} rows, {} columns".format(df_ugly.shape[0], df_ugly.
  ↪shape[1]))
print(df_ugly.head())
# Not Pass, many links that are not working anymore
```

```
Dataset Size: 8596 rows, 10 columns
  post_id           created_at  \
0  xcmklj  2022-09-12 20:03:15
1  xckzxj  2022-09-12 18:54:50
2  xcd52w  2022-09-12 13:40:16
3  xcbjt6  2022-09-12 12:30:07
4  xcbjhv  2022-09-12 12:29:45

                                                 title link_flair_text  score  \
```

```
0  Infographic that came with my North Carolina v…              NaN      1
1  You cannot tell me those are different shades …              NaN      1
2                                            Neymar     Clusterfuck      1
3           Youngest WORLD Number 1 In the OPEN ERA              NaN      1
4                                why the tails omg              NaN      1

   num_comments   posted_by                              image_url  \
0             0  TheTraceur  https://i.redd.it/q5aqzkcqhhn91.jpg
1             0       Fnorv  https://i.redd.it/1r4jft2i2fn91.jpg
2             0  RedFlare07  https://i.redd.it/r6a3i6jllfn91.jpg
3             0  PhillyManc  https://i.redd.it/cq2akup8kbn91.png
4             0  biglezmate  https://i.redd.it/zv0nz7jmpcn91.png

                                          full_link   nsfw
0  https://www.reddit.com/r/dataisugly/comments/x…  False
1  https://www.reddit.com/r/dataisugly/comments/x…  False
2  https://www.reddit.com/r/dataisugly/comments/x…  False
3  https://www.reddit.com/r/dataisugly/comments/x…  False
4  https://www.reddit.com/r/dataisugly/comments/x…  False
```

[94]:
```
# Preview dataset Weather
print("Dataset Size: {} rows, {} columns".format(df_weather.shape[0],␣
 ↪df_weather.shape[1]))
print(df_weather.head())
# Pass
```

```
Dataset Size: 40346 rows, 41 columns
       country     location_name  latitude  longitude        timezone  \
0  Afghanistan             Kabul     34.52      69.18      Asia/Kabul
1      Albania            Tirana     41.33      19.82   Europe/Tirane
2      Algeria           Algiers     36.76       3.05  Africa/Algiers
3      Andorra  Andorra La Vella     42.50       1.52  Europe/Andorra
4       Angola            Luanda     -8.84      13.23   Africa/Luanda

   last_updated_epoch      last_updated  temperature_celsius  \
0          1693301400  2023-08-29 14:00                 28.8
1          1693301400  2023-08-29 11:30                 27.0
2          1693301400  2023-08-29 10:30                 28.0
3          1693301400  2023-08-29 11:30                 10.2
4          1693301400  2023-08-29 10:30                 25.0

   temperature_fahrenheit condition_text  …  air_quality_PM2.5  \
0                    83.8          Sunny  …                7.9
1                    80.6  Partly cloudy  …               28.2
2                    82.4  Partly cloudy  …                6.4
3                    50.4          Sunny  …                0.5
4                    77.0  Partly cloudy  …              139.6
```

```
   air_quality_PM10  air_quality_us-epa-index air_quality_gb-defra-index  \
0              11.1                         1                          1
1              29.6                         2                          3
2               7.9                         1                          1
3               0.8                         1                          1
4             203.3                         4                         10

    sunrise    sunset  moonrise   moonset      moon_phase  moon_illumination
0  05:24 AM  06:24 PM  05:39 PM  02:48 AM  Waxing Gibbous                 93
1  06:04 AM  07:19 PM  06:50 PM  03:25 AM  Waxing Gibbous                 93
2  06:16 AM  07:21 PM  06:46 PM  03:50 AM  Waxing Gibbous                 93
3  07:16 AM  08:34 PM  08:08 PM  04:38 AM  Waxing Gibbous                 93
4  06:11 AM  06:06 PM  04:43 PM  04:41 AM  Waxing Gibbous                 93

[5 rows x 41 columns]
```

[96]:
```
# Preview dataset Sleep
print("Dataset Size: {} rows, {} columns".format(df_sleep.shape[0], df_sleep.
 ↪shape[1]))
print(df_sleep.head())
# Not Pass, too small for this objective
```

```
Dataset Size: 374 rows, 13 columns
   Person ID Gender  Age            Occupation  Sleep Duration  \
0          1   Male   27     Software Engineer             6.1
1          2   Male   28                Doctor             6.2
2          3   Male   28                Doctor             6.2
3          4   Male   28  Sales Representative             5.9
4          5   Male   28  Sales Representative             5.9

   Quality of Sleep  Physical Activity Level  Stress Level BMI Category  \
0                 6                       42             6   Overweight
1                 6                       60             8       Normal
2                 6                       60             8       Normal
3                 4                       30             8        Obese
4                 4                       30             8        Obese

  Blood Pressure  Heart Rate  Daily Steps Sleep Disorder
0         126/83          77         4200            NaN
1         125/80          75        10000            NaN
2         125/80          75        10000            NaN
3         140/90          85         3000    Sleep Apnea
4         140/90          85         3000    Sleep Apnea
```

[98]:
```
# Preview dataset IMDB movies
print("Dataset Size: {} rows, {} columns".format(df_imdb.shape[0], df_imdb.
 ↪shape[1]))
```

```
print(df_imdb.head())
# Pass
```

```
Dataset Size: 10000 rows, 6 columns
   Unnamed: 0                          title  \
0           0                       Ad Astra
1           1                      Bloodshot
2           2              Bad Boys for Life
3           3                        Ant-Man
4           4  Percy Jackson: Sea of Monsters

                                            overview original_language  \
0  The near future, a time when both hope and har…                 en
1  After he and his wife are murdered, marine Ray…                 en
2  Marcus and Mike are forced to confront new thr…                 en
3  Armed with the astonishing ability to shrink i…                 en
4  In their quest to confront the ultimate evil, …                 en

   vote_count  vote_average
0        2853           5.9
1        1349           7.2
2        2530           7.1
3       13611           7.1
4        3542           5.9
```

0.0.4 4. Data Types, Formats and Missing Values

[101]:
```
# Data types climate
print(df_climate.dtypes)
print(df_climate.isnull().sum())
```

```
ObjectId       int64
Country       object
ISO2          object
ISO3          object
Indicator     object
              …
F2018        float64
F2019        float64
F2020        float64
F2021        float64
F2022        float64
Length: 72, dtype: object
ObjectId      0
Country       0
ISO2          2
ISO3          0
```

```
Indicator     0
              ..
F2018        12
F2019        12
F2020        13
F2021        12
F2022        12
Length: 72, dtype: int64
```

[103]:
```
# Data types Weather
print(df_weather.dtypes)
print(df_weather.isnull().sum())
```

```
country                          object
location_name                    object
latitude                        float64
longitude                       float64
timezone                         object
last_updated_epoch                int64
last_updated                     object
temperature_celsius             float64
temperature_fahrenheit          float64
condition_text                   object
wind_mph                        float64
wind_kph                        float64
wind_degree                       int64
wind_direction                   object
pressure_mb                     float64
pressure_in                     float64
precip_mm                       float64
precip_in                       float64
humidity                          int64
cloud                             int64
feels_like_celsius              float64
feels_like_fahrenheit           float64
visibility_km                   float64
visibility_miles                float64
uv_index                        float64
gust_mph                        float64
gust_kph                        float64
air_quality_Carbon_Monoxide     float64
air_quality_Ozone               float64
air_quality_Nitrogen_dioxide    float64
air_quality_Sulphur_dioxide     float64
air_quality_PM2.5               float64
air_quality_PM10                float64
air_quality_us-epa-index          int64
air_quality_gb-defra-index        int64
```

```
sunrise                          object
sunset                           object
moonrise                         object
moonset                          object
moon_phase                       object
moon_illumination                 int64
dtype: object
country                          0
location_name                    0
latitude                         0
longitude                        0
timezone                         0
last_updated_epoch               0
last_updated                     0
temperature_celsius              0
temperature_fahrenheit           0
condition_text                   0
wind_mph                         0
wind_kph                         0
wind_degree                      0
wind_direction                   0
pressure_mb                      0
pressure_in                      0
precip_mm                        0
precip_in                        0
humidity                         0
cloud                            0
feels_like_celsius               0
feels_like_fahrenheit            0
visibility_km                    0
visibility_miles                 0
uv_index                         0
gust_mph                         0
gust_kph                         0
air_quality_Carbon_Monoxide      0
air_quality_Ozone                0
air_quality_Nitrogen_dioxide     0
air_quality_Sulphur_dioxide      0
air_quality_PM2.5                0
air_quality_PM10                 0
air_quality_us-epa-index         0
air_quality_gb-defra-index       0
sunrise                          0
sunset                           0
moonrise                         0
moonset                          0
moon_phase                       0
moon_illumination                0
```

```
dtype: int64
```

[105]:
```
# Data types IMDB
print(df_imdb.dtypes)
print(df_imdb.isnull().sum())
```

```
Unnamed: 0             int64
title                 object
overview              object
original_language     object
vote_count             int64
vote_average         float64
dtype: object
Unnamed: 0            0
title                 0
overview             30
original_language     0
vote_count            0
vote_average          0
dtype: int64
```

0.0.5 5. Statistical Summary

[108]:
```
# Descriptive Statistics for Climate data
print(df_climate.describe())
# Pass
```

```
         ObjectId       F1961       F1962       F1963       F1964       F1965  \
count  225.000000  188.000000  189.000000  188.000000  188.000000  188.000000
mean   113.000000    0.163053   -0.013476   -0.006043   -0.070059   -0.247027
std     65.096083    0.405080    0.341812    0.387348    0.309305    0.270734
min      1.000000   -0.694000   -0.908000   -1.270000   -0.877000   -1.064000
25%     57.000000   -0.097000   -0.164000   -0.205500   -0.236500   -0.392500
50%    113.000000    0.064500   -0.056000   -0.003000   -0.056000   -0.230500
75%    169.000000    0.318500    0.114000    0.230500    0.132500   -0.091500
max    225.000000    1.892000    0.998000    1.202000    1.097000    0.857000

            F1966       F1967       F1968       F1969  …       F2013  \
count  192.000000  191.000000  191.000000  190.000000  …  216.000000
mean     0.105505   -0.110832   -0.199110    0.157942  …    0.931199
std      0.378423    0.339484    0.270131    0.308540  …    0.321595
min     -1.801000   -1.048000   -1.634000   -0.900000  …    0.118000
25%     -0.035750   -0.259500   -0.340000   -0.009000  …    0.743500
50%      0.098000   -0.146000   -0.187000    0.204000  …    0.897000
75%      0.277000    0.015000   -0.067000    0.349000  …    1.187500
max      1.151000    1.134000    0.476000    0.939000  …    1.643000

            F2014       F2015       F2016       F2017       F2018       F2019  \
```

```
count  216.000000  216.000000  213.000000  214.000000  213.000000  213.000000
mean     1.114815    1.269773    1.439521    1.280785    1.302113    1.443061
std      0.564903    0.462162    0.401091    0.393999    0.596786    0.467510
min     -0.092000   -0.430000    0.250000    0.017000    0.238000    0.050000
25%      0.744000    1.017750    1.147000    1.027500    0.865000    1.169000
50%      0.986500    1.215000    1.446000    1.282000    1.125000    1.412000
75%      1.335500    1.520500    1.714000    1.535000    1.834000    1.698000
max      2.704000    2.613000    2.459000    2.493000    2.772000    2.689000

            F2020       F2021       F2022
count  212.000000  213.000000  213.000000
mean     1.552038    1.343531    1.382113
std      0.621930    0.484692    0.669279
min      0.229000   -0.425000   -1.305000
25%      1.161750    1.019000    0.878000
50%      1.477000    1.327000    1.315000
75%      1.826250    1.629000    1.918000
max      3.691000    2.676000    3.243000

[8 rows x 63 columns]
```

[110]:
```
# Descriptive Statistics for Weather data
print(df_weather.describe())
# Pass
```

```
           latitude     longitude  last_updated_epoch  temperature_celsius  \
count  40346.000000  40346.000000        4.034600e+04         40346.000000
mean      19.298562     21.760141        1.702473e+09            19.179078
std       24.521626     65.682857        5.363903e+06            10.718576
min      -41.300000   -175.200000        1.693301e+09           -41.900000
25%        3.750000     -6.840000        1.697662e+09            12.000000
50%       17.250000     23.240000        1.702668e+09            22.000000
75%       41.320000     49.880000        1.707156e+09            27.000000
max       64.100000    179.220000        1.711556e+09            45.400000

       temperature_fahrenheit      wind_mph      wind_kph   wind_degree  \
count            40346.000000  40346.000000  40346.000000  40346.000000
mean                66.522233      7.471611     12.027016    162.518242
std                 19.293569      5.172169      8.325958    106.324676
min                -43.400000      2.200000      3.600000      1.000000
25%                 53.600000      3.800000      6.100000     70.000000
50%                 71.600000      5.800000      9.400000    150.000000
75%                 80.600000     10.500000     16.900000    250.000000
max                113.700000     91.900000    148.000000    360.000000

        pressure_mb   pressure_in  …      gust_kph  \
count  40346.000000  40346.000000  …  40346.000000
mean    1013.890943     29.939522  …     20.027718
```

```
std        7.437666      0.219491  …     11.906650
min      958.000000     28.290000  …      0.000000
25%     1010.000000     29.830000  …     11.100000
50%     1013.000000     29.910000  …     18.000000
75%     1018.000000     30.060000  …     26.300000
max     1074.000000     31.710000  …    155.200000

       air_quality_Carbon_Monoxide  air_quality_Ozone  \
count                 40346.000000       40346.000000
mean                    577.529794          43.759971
std                    1382.216080          32.658034
min                      96.800000           0.000000
25%                     237.000000          18.400000
50%                     290.400000          40.800000
75%                     447.300000          63.700000
max                   41870.102000         555.000000

       air_quality_Nitrogen_dioxide  air_quality_Sulphur_dioxide  \
count                  40346.000000                 40346.000000
mean                      14.511496                     8.242131
std                       27.044095                    22.470273
min                        0.000000                     0.000000
25%                        1.000000                     0.500000
50%                        4.500000                     1.800000
75%                       15.600000                     6.400000
max                      575.800000                   557.000000

       air_quality_PM2.5  air_quality_PM10  air_quality_us-epa-index  \
count       40346.000000       40346.00000              40346.000000
mean           25.276535          45.65170                  1.613791
std            63.684429         106.59568                  1.067879
min             0.500000           0.50000                  1.000000
25%             2.500000           4.70000                  1.000000
50%             7.500000          12.90000                  1.000000
75%            23.000000          39.50000                  2.000000
max          1558.800000        3566.40000                  6.000000

       air_quality_gb-defra-index  moon_illumination
count                40346.000000       40346.000000
mean                     2.456427          52.067268
std                      2.682182          35.314977
min                      1.000000           0.000000
25%                      1.000000          17.000000
50%                      1.000000          53.000000
75%                      2.000000          88.000000
max                     10.000000         100.000000

[8 rows x 30 columns]
```

[112]:
```
# Descriptive Statistics for IMDB data
print(df_imdb.describe())
# Not Pass
```

```
       Unnamed: 0    vote_count  vote_average
count  10000.00000  10000.000000  10000.000000
mean    4999.50000   1020.825100      6.306230
std     2886.89568   1992.305005      1.354259
min        0.00000      0.000000      0.000000
25%     2499.75000    143.000000      5.800000
50%     4999.50000    332.000000      6.500000
75%     7499.25000    926.250000      7.100000
max     9999.00000  25148.000000     10.000000
```

[114]:
```
# Descriptive Statistics for categorical IMDB data
for column in df_imdb.select_dtypes(include=['object']).columns:
    print(df_imdb[column].value_counts())
    print("\n")
# Not Pass
```

```
title
Dracula                                4
Beauty and the Beast                   4
Godzilla                               3
Fallen                                 3
Les Misérables                         3
                                      ..
The Imaginarium of Doctor Parnassus    1
Along Came Polly                       1
Grand Illusion                         1
Teenage Mutant Ninja Turtles III       1
Woochi: The Demon Slayer               1
Name: count, Length: 9689, dtype: int64


overview
Plot unknown.
5
The plot is unknown at this time.
2
Wilbur the pig is scared of the end of the season, because he knows that come
that time, he will end up on the dinner table. He hatches a plan with Charlotte,
a spider that lives in his pen, to ensure that this will never happen.
2
Bernie works at a Las Vegas casino, where he uses his innate ability to bring
about misfortune in those around him to jinx gamblers into losing. His imposing
boss, Shelly Kaplow, is happy with the arrangement. But Bernie finds unexpected
happiness when he begins dating attractive waitress Natalie Belisario.
```

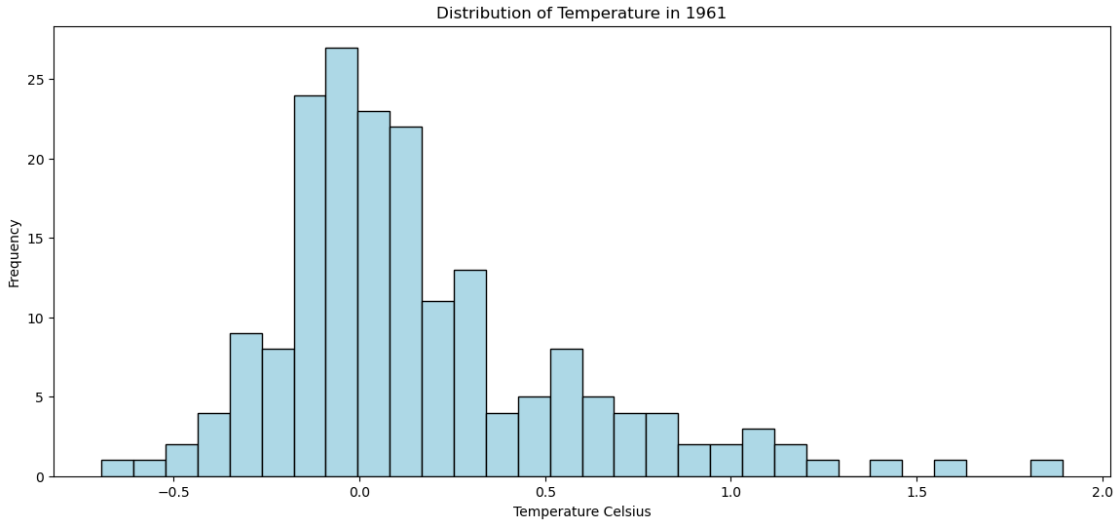
```
1
A fat Lawyer finds himself growing "Thinner" when an old gypsy man places a hex
on him. Now the lawyer must call upon his friends in organized crime to help him
persuade the gypsy to lift the curse. Time is running out for the desperate
lawyer as he draws closer to his own death, and grows ever thinner.
1
                          ..
Six short stories that explore the extremities of human behavior involving
people in distress.
1
King Randolph sends for his cousin Duchess Rowena to help turn his daughters,
Princess Genevieve and her 11 sisters, into better ladies. But the Duchess takes
away all the sisters fun, including the sisters favorite pastime:
dancing.Thinking all hope is lost they find a secret passageway to a magical
land were they can dance the night away.    1
An orphaned boy raised by underground creatures called Boxtrolls comes up from
the sewers and out of his box to save his family and the town from the evil
exterminator, Archibald Snatcher.
1
At an exclusive country club, an ambitious young caddy, Danny Noonan, eagerly
pursues a caddy scholarship in hopes of attending college and, in turn, avoiding
a job at the lumber yard. In order to succeed, he must first win the favour of
the elitist Judge Smails, and then the caddy golf tournament which Smails
sponsors.                          1
Spanning four centuries in Korea, this epic action-adventure concerns a powerful
pipe and a trio of wizards who will do anything to protect it.
1
Name: count, Length: 9964, dtype: int64


original_language
en    8326
fr     385
ja     269
it     152
es     145
de      96
ko      92
cn      72
zh      71
hi      70
ru      69
da      32
sv      30
pt      22
id      21
ta      18
nl      13
```

```
no      13
tr      12
pl      10
tl       9
th       9
ar       6
sr       6
te       5
fa       5
cs       4
ml       4
he       4
fi       3
hu       3
af       2
el       2
ro       2
ab       1
lv       1
mr       1
ka       1
ms       1
vi       1
et       1
eu       1
sh       1
bs       1
nb       1
pa       1
la       1
is       1
xx       1
mk       1
sq       1
uk       1
Name: count, dtype: int64
```

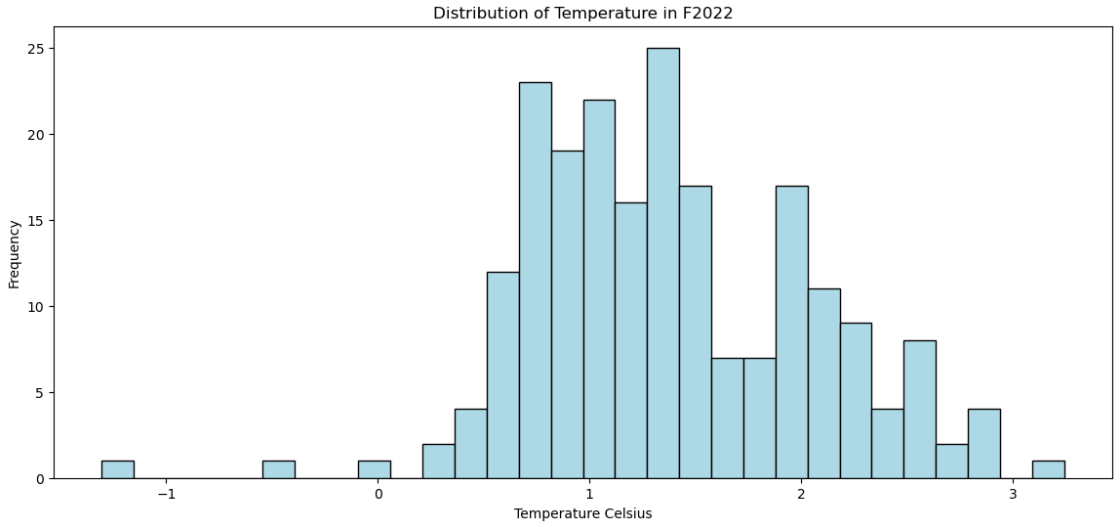
0.0.6 6. Distribution Analysis

[160]:
```
#df_climate
# Distribution of temperature in the first notation
plt.figure(figsize=(14, 6))
plt.hist(df_climate['F1961'], bins=30, color='lightblue', edgecolor='black')
plt.title('Distribution of Temperature in 1961')
plt.xlabel('Temperature Celsius')
```

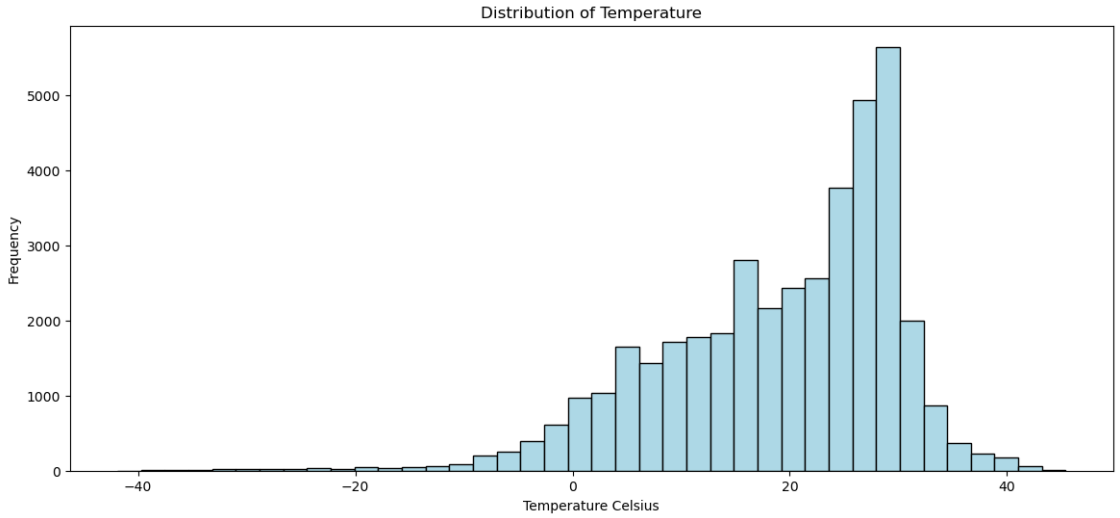
```
plt.ylabel('Frequency')
plt.show()
```



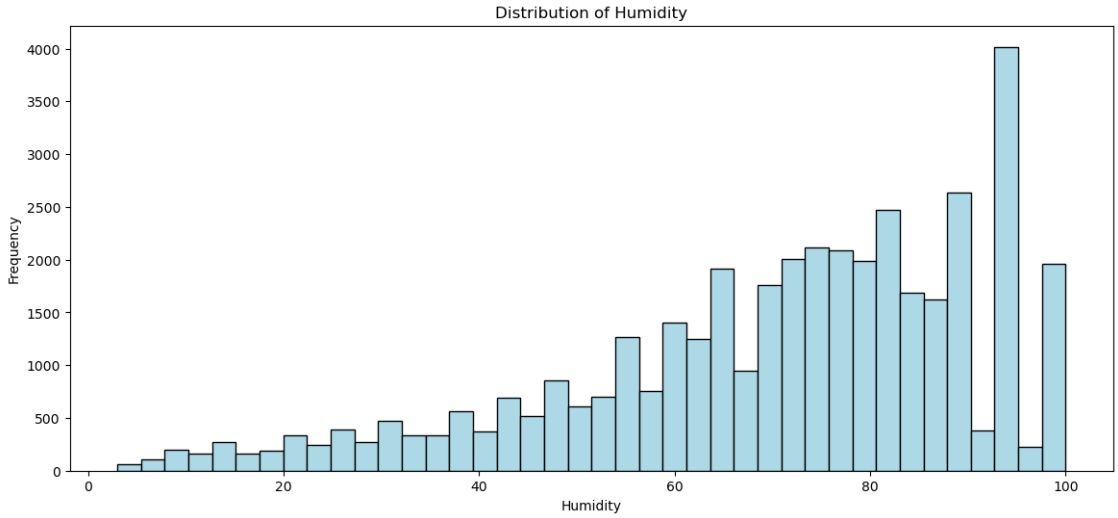
[162]:
```
#df_climate
# Distribution of temperature in the last notation
plt.figure(figsize=(14, 6))
plt.hist(df_climate['F2022'], bins=30, color='lightblue', edgecolor='black')
plt.title('Distribution of Temperature in F2022')
plt.xlabel('Temperature Celsius')
plt.ylabel('Frequency')
plt.show()
```



[164]:
```
#df_weather
# Distribution of temperature
plt.figure(figsize=(14, 6))
plt.hist(df_weather['temperature_celsius'], bins=40, color='lightblue',␣
 ↪edgecolor='black')
plt.title('Distribution of Temperature')
plt.xlabel('Temperature Celsius')
plt.ylabel('Frequency')
plt.show()
```

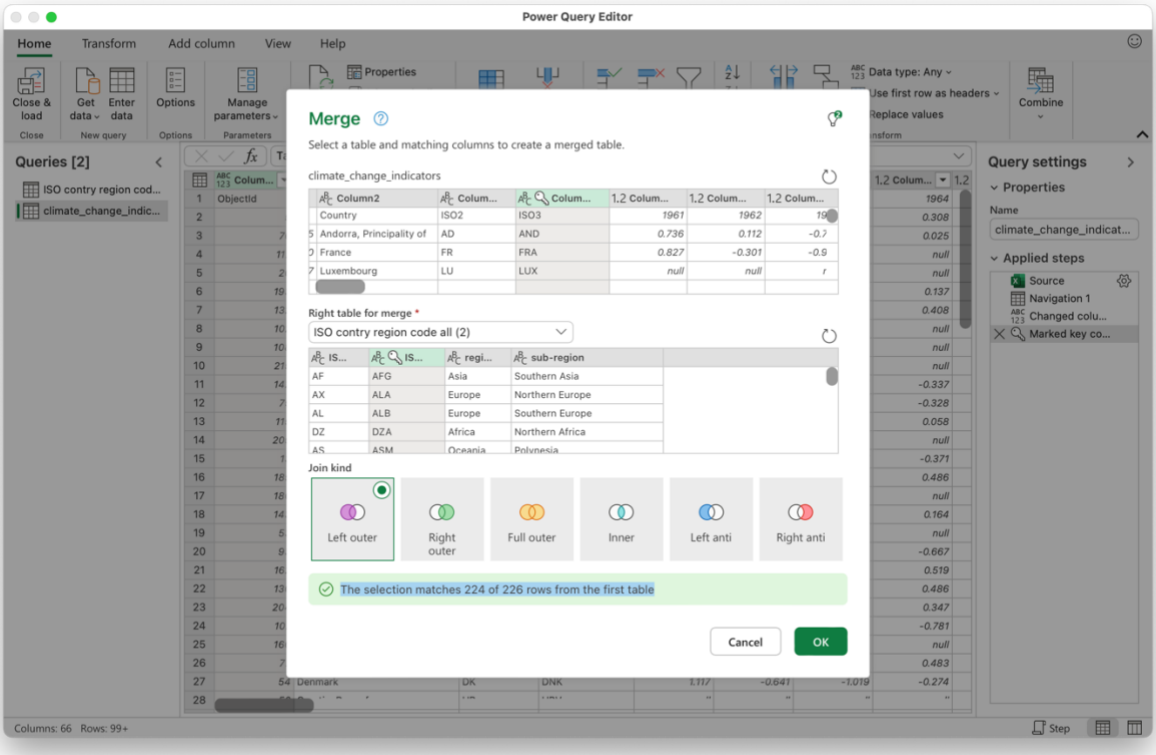


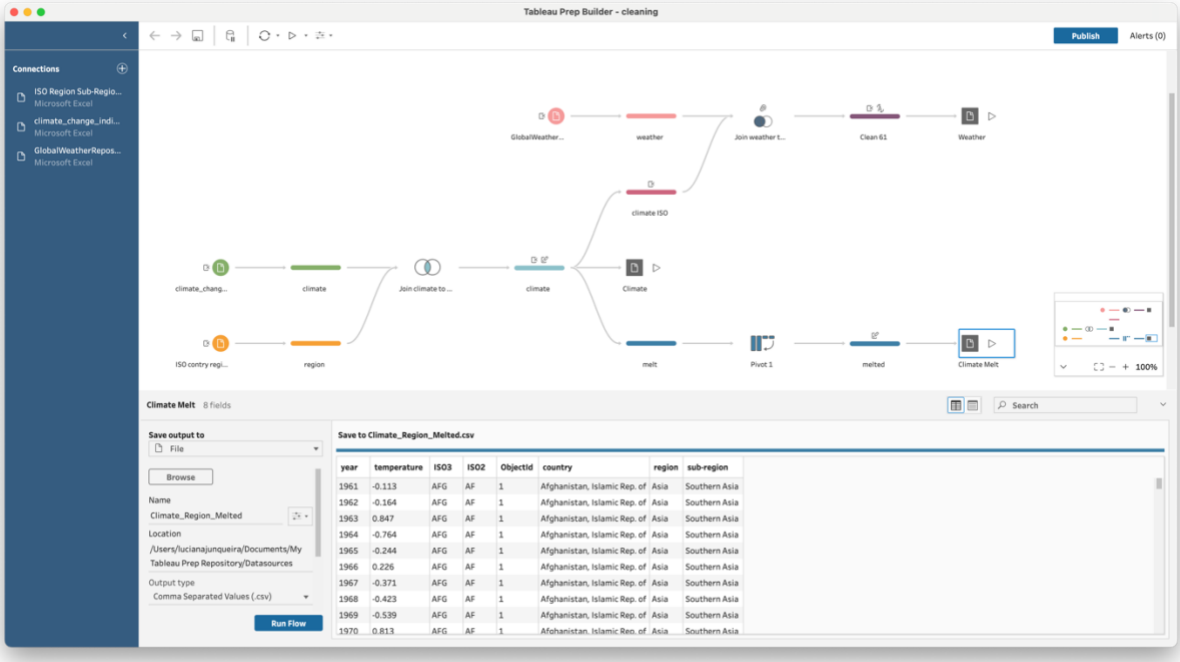
[166]:
```
#df_weather
# Distribution of Humidity
plt.figure(figsize=(14, 6))
plt.hist(df_weather['humidity'], bins=40, color='lightblue', edgecolor='black')
plt.title('Distribution of Humidity')
plt.xlabel('Humidity')
plt.ylabel('Frequency')
plt.show()
```


Distribution of Humidity
Frequency
Humidity
0
500
1000
1500
2000
2500
3000
3500
4000
0
20
40
60
80
100

[ ]:

Project_Step2





Project_Step3

0.0.1 1. Import Libraries

[99]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from scipy.stats import zscore
```

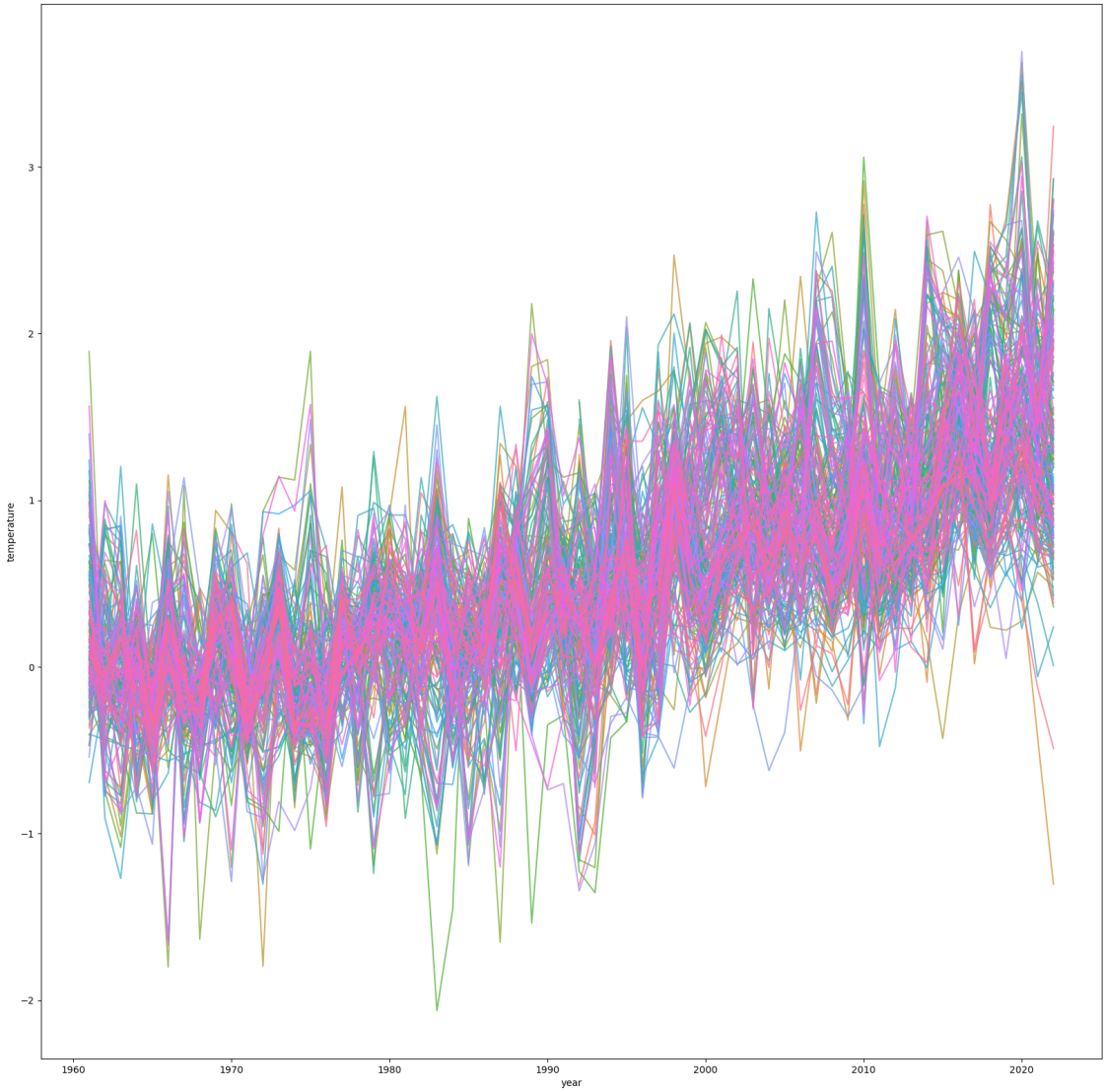
0.0.2 2. Load Datasets

```
# Importing data frames
df_climate = pd.read_csv("Climate_Region.csv")
df_climate_melt = pd.read_csv("Climate_Region_Melted.csv")
df_weather = pd.read_csv("Weather_ISO.csv")
```

[102]:
0.0.3 3. EDA —Country

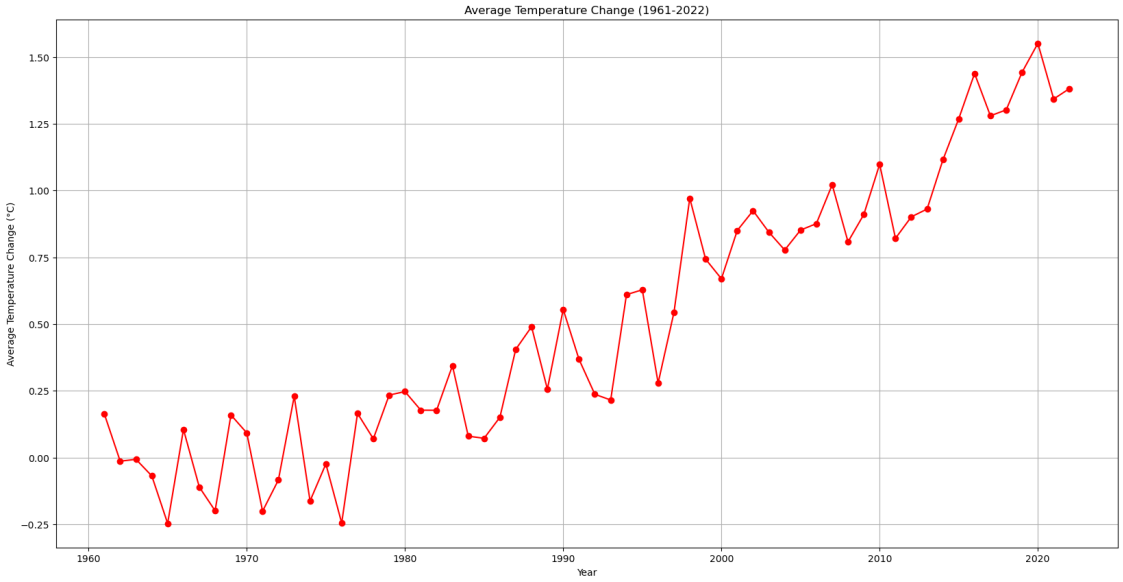
```
#Time Series Analysis of all data
plt.figure(figsize=(20, 20))
sns.lineplot(data=df_climate_melt, x='year', y='temperature', hue='country',␣
↪alpha=0.75,legend = False)
plt.show()
```

[105]:
```
#Overall temperature are increasing over the years
```

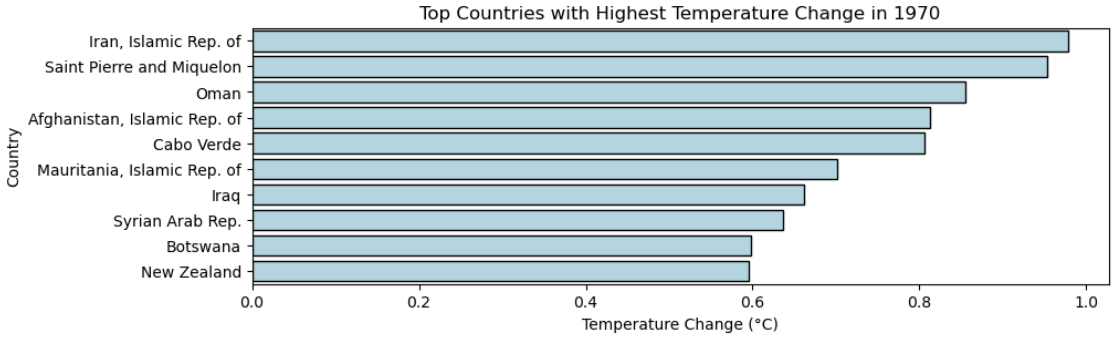


[107]:
```
#Time Series Analysis of all data - Average

# Calculate the mean temperature change for each year
yearly_averages = df_climate.loc[:, '1961':'2022'].mean()

# Reset index of the series
yearly_averages_df = yearly_averages.reset_index()
yearly_averages_df.columns = ['Year', 'Average Temperature Change']

# Convert 'Year' to numeric
yearly_averages_df['Year'] = pd.to_numeric(yearly_averages_df['Year'])

# Display Plot
```

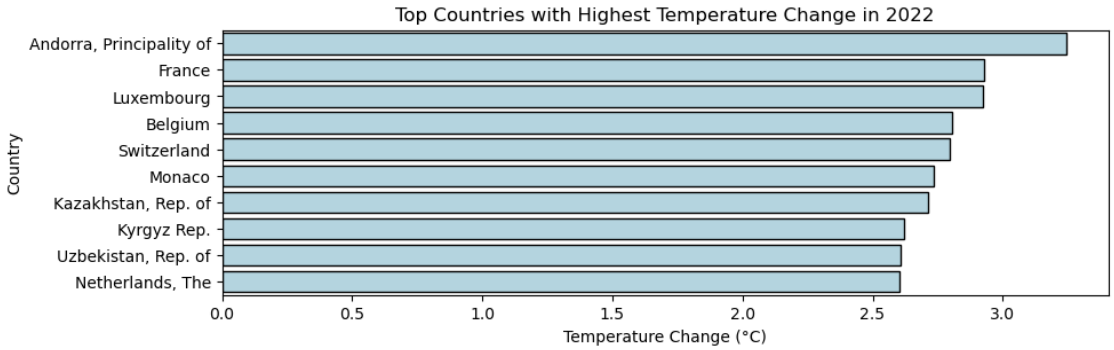
```
plt.figure(figsize=(20, 10))
plt.plot(yearly_averages_df['Year'], yearly_averages_df['Average Temperature␣
 ↪Change'], marker='o', color='red', linestyle='-')
plt.title('Average Temperature Change (1961-2022)')
plt.xlabel('Year')
plt.ylabel('Average Temperature Change (°C)')
plt.grid(True)
plt.show()
```



[109]:
```
# Country Comparison
# Highest Temperature in 1970

# Filter positive values
positive_1970 = df_climate[df_climate['1970'] > 0]

# Find Top 10 highest positive temperature changes 1970
top_10_1970 = positive_1970.sort_values(by='1970', ascending=False).head(10)

# Plot bar chart
plt.figure(figsize=(10, 3))
sns.barplot(x='1970', y='country', data=top_10_1970, color='lightblue',␣
 ↪edgecolor='black')
plt.title('Top Countries with Highest Temperature Change in 1970')
plt.xlabel('Temperature Change (°C)')
plt.ylabel('Country')
plt.show()
```



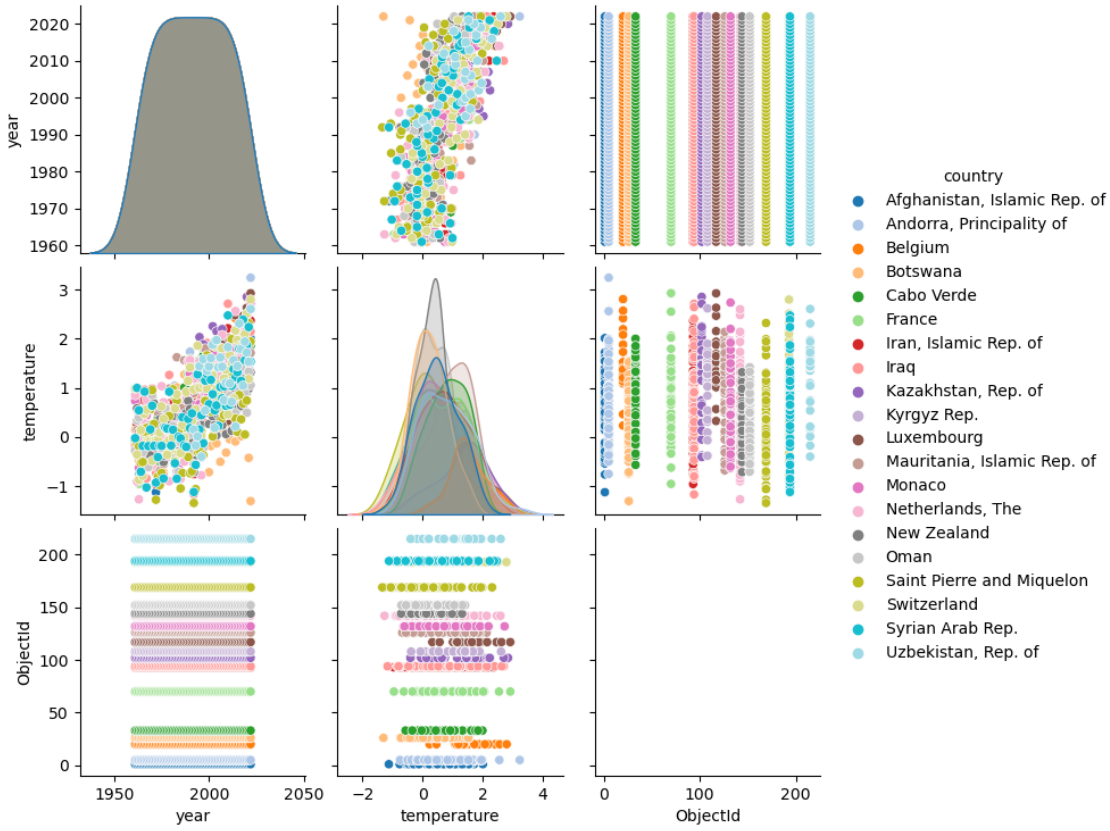
[111]:
```
# Country Comparison
# Highest Temperature in 2022

# Filter positive values
positive_2022 = df_climate[df_climate['2022'] > 0]

# Find Top 10 highest positive temperature changes 2022
top_10_2022 = positive_2022.sort_values(by='2022', ascending=False).head(10)

# Plot bar chart
plt.figure(figsize=(10, 3))
sns.barplot(x='2022', y='country', data=top_10_2022, color='lightblue',␣
 ↪edgecolor='black')
plt.title('Top Countries with Highest Temperature Change in 2022')
plt.xlabel('Temperature Change (°C)')
plt.ylabel('Country')
plt.show()

#Countries with Top Temperature change are different in 2022 from 1970, more␣
 ↪European countries
```



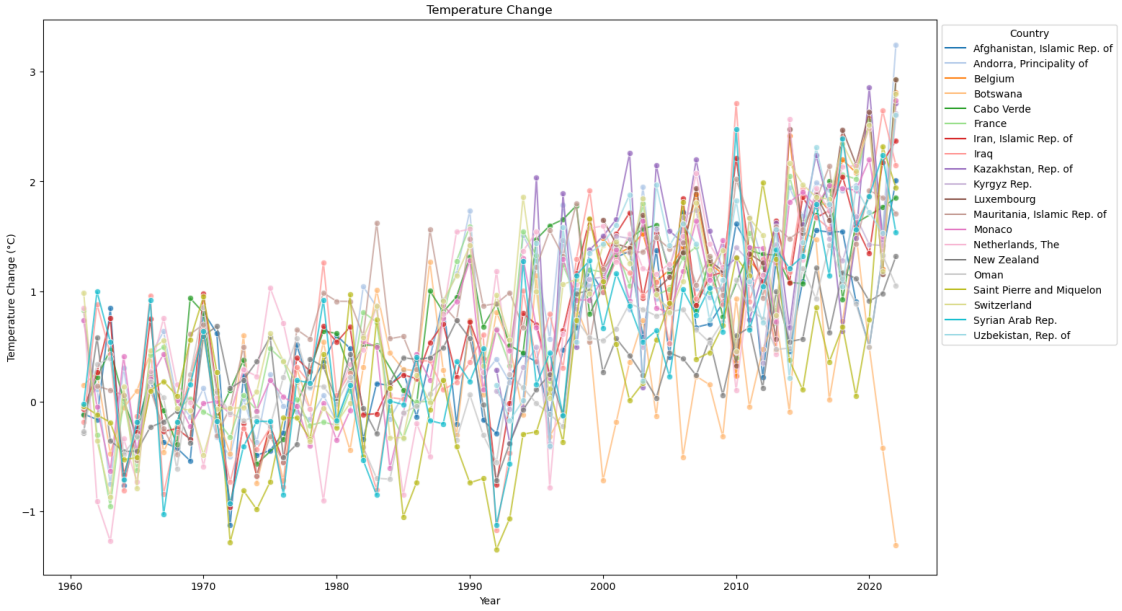
[113]:
```
#Top 10 1970
positive_1970 = df_climate[df_climate['1970'] > 0]
top_10_1970 = positive_1970.sort_values(by='1970', ascending=False).head(10)

#Top 10 2022
positive_2022 = df_climate[df_climate['2022'] > 0]
top_10_2022 = positive_2022.sort_values(by='2022', ascending=False).head(10)

# Combine the two lists of top countries, uniqueness
top_countries_combined = pd.concat([top_10_1970, top_10_2022]).
 ↪drop_duplicates(subset='country')
unique_top_countries = top_countries_combined['country'].unique()

# Filter melted dataset for the selected countries
df_filtered = df_climate_melt[df_climate_melt['country'].
 ↪isin(unique_top_countries)]

# Display plot
sns.pairplot(data=df_filtered, hue='country', palette='tab20')
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

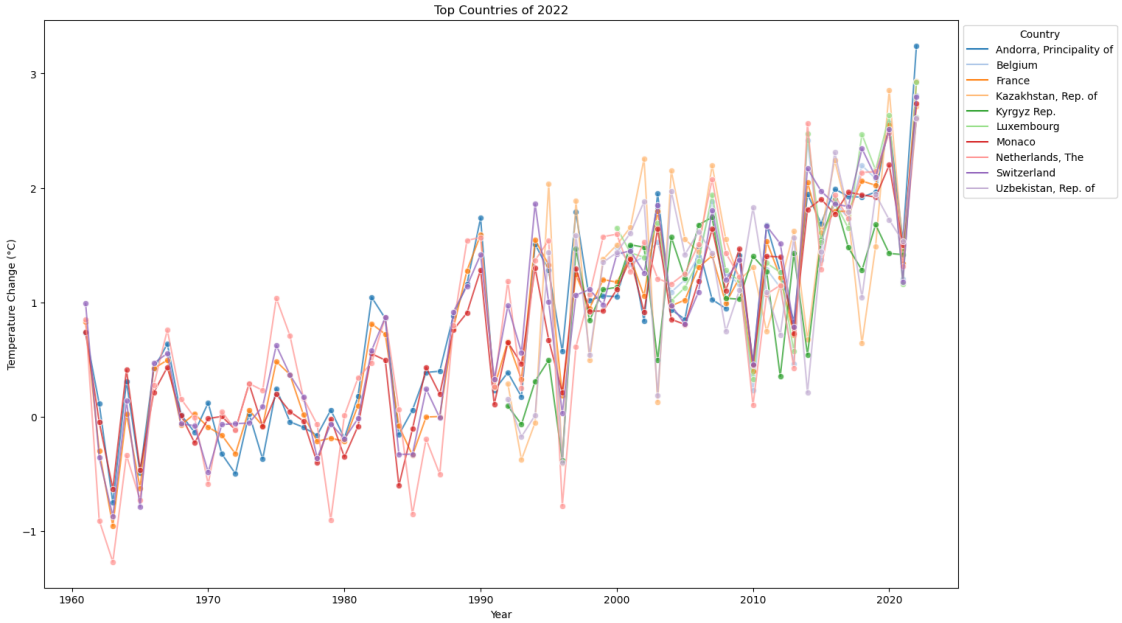
[113]:
```
<seaborn.axisgrid.PairGrid at 0x7f3f0b6f7b50>
```



[114]:
```
# Figure size
plt.figure(figsize=(16, 10))

# Create the line plot
sns.lineplot(data=df_filtered, x='year', y='temperature', hue='country',␣
 ↪marker='o', palette='tab20', alpha=0.75)

# Display the plot
plt.title('Temperature Change')
plt.xlabel('Year')
plt.ylabel('Temperature Change (°C)')
plt.legend(title='Country', bbox_to_anchor=(1,1), loc='upper left')
plt.show()
```

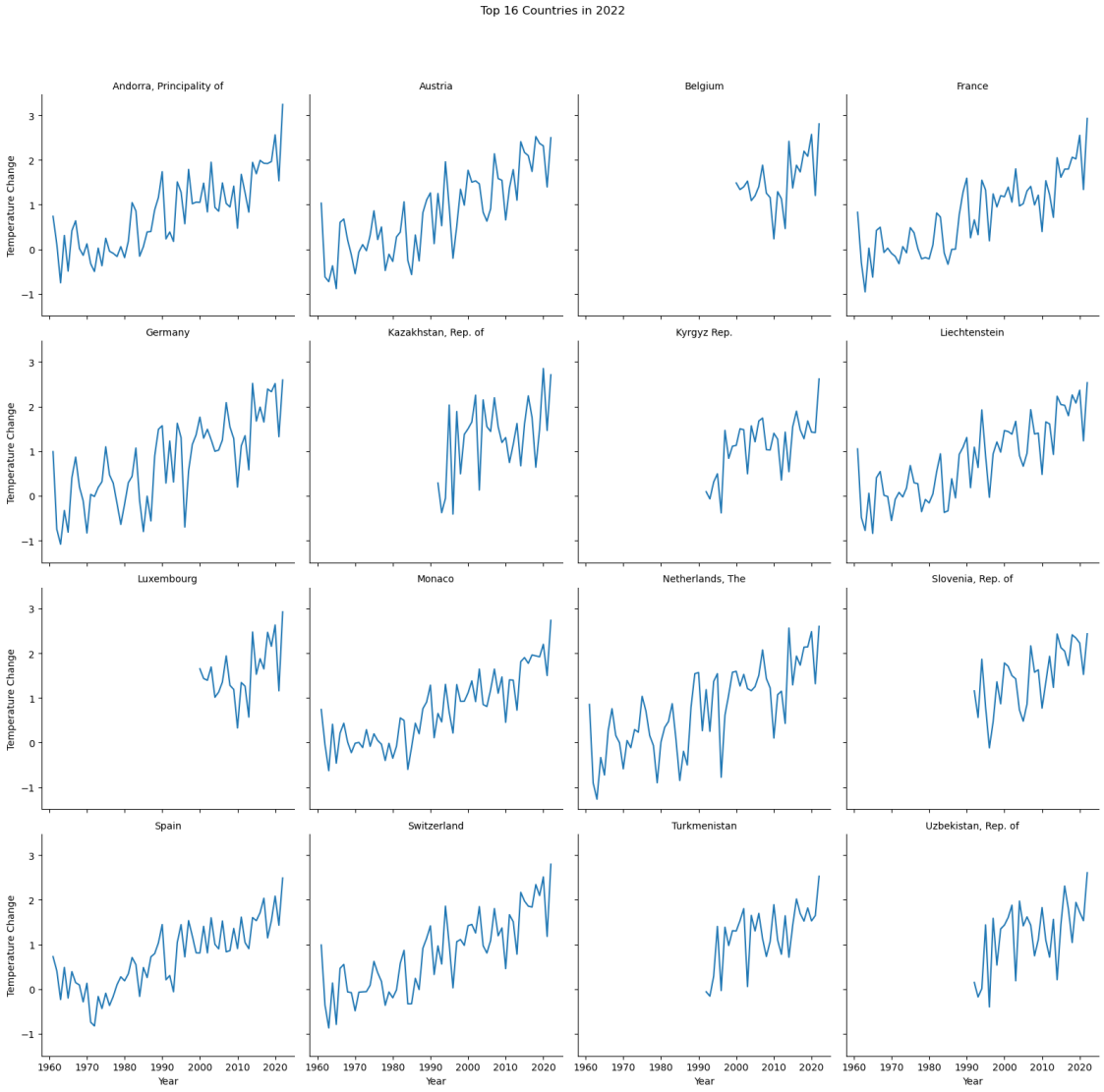


[115]:
```
# Figure size
plt.figure(figsize=(16, 10))

# Top 10 countries for 2022 with positive temperature change
positive_2022 = df_climate[df_climate['2022'] > 0]
top_10_2022 = positive_2022.sort_values(by='2022', ascending=False).head(10)
unique_top_countries_2022 = top_10_2022['country'].unique()

# Filtering the melted df for these top countries
df_filtered_2022 = df_climate_melt[df_climate_melt['country'].
 ↪isin(unique_top_countries_2022)]

# Display the plot
sns.lineplot(data=df_filtered_2022, x='year', y='temperature', hue='country',␣
 ↪marker='o', palette='tab20', alpha=0.75)
plt.title('Top Countries of 2022')
plt.xlabel('Year')
plt.ylabel('Temperature Change (°C)')
plt.legend(title='Country', bbox_to_anchor=(1,1), loc='upper left')
plt.show()
```



[116]:
```
# Set up Facet Grid for the top 16 countries
top_16_2022 = df_climate.sort_values(by='2022', ascending=False).
 ↪head(16)['country']
df_filtered = df_climate_melt[df_climate_melt['country'].isin(top_16_2022)]
g = sns.FacetGrid(df_filtered, col='country', col_wrap=4, sharex=True,␣
 ↪sharey=True, height=4)

# Display plot for each top country
g.map(sns.lineplot, 'year', 'temperature')
g.set_titles('{col_name}')
g.set_axis_labels('Year', 'Temperature Change')
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Top 16 Countries in 2022', fontsize=12)
plt.show()
# Get a group of countries to study further: Andora, Austria, France, Germany,
#Lichtenstein, Monaco, Netherlands, Spain and Switzerland.
#Dropping the ones that have too many missing values
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

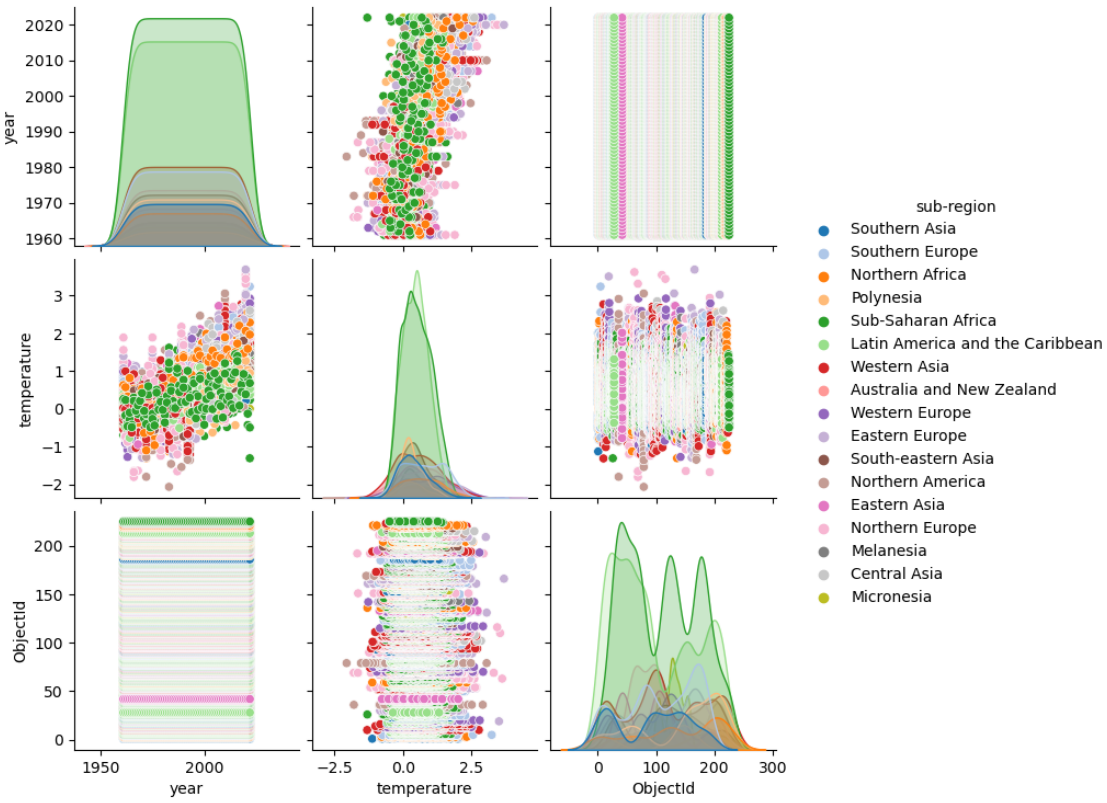


0.0.4 4. EDA — Sub Region

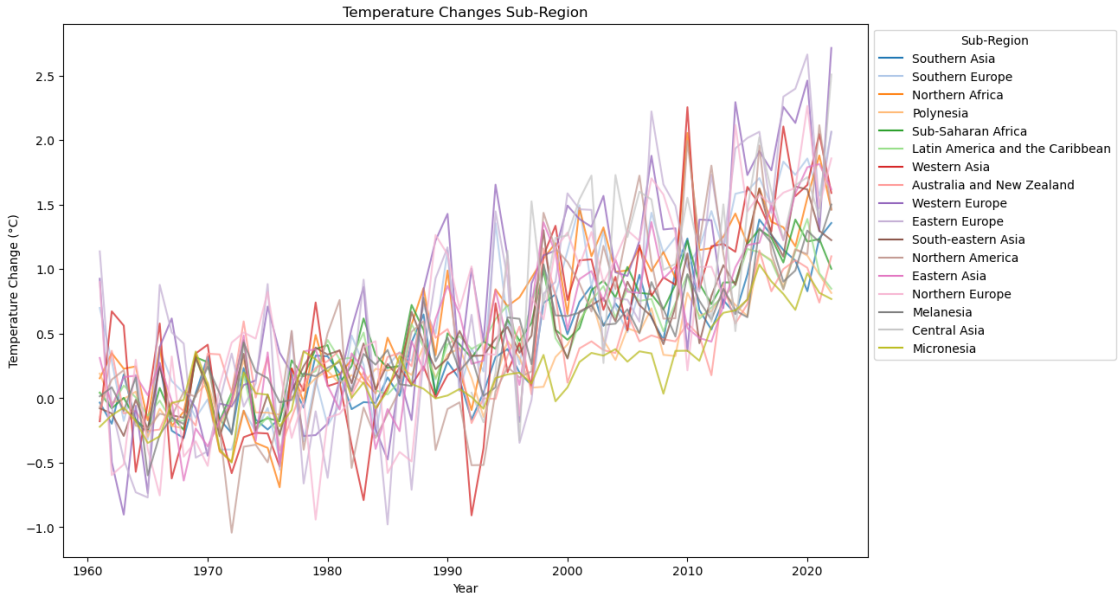
[121]:
```
# Pair plot of sub-region
sns.pairplot(data=df_climate_melt, hue='sub-region', palette= 'tab20')
# Sub-Region seems to be relevant for climate change
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

[121]:
```
<seaborn.axisgrid.PairGrid at 0x7f3f19e1fb50>
```

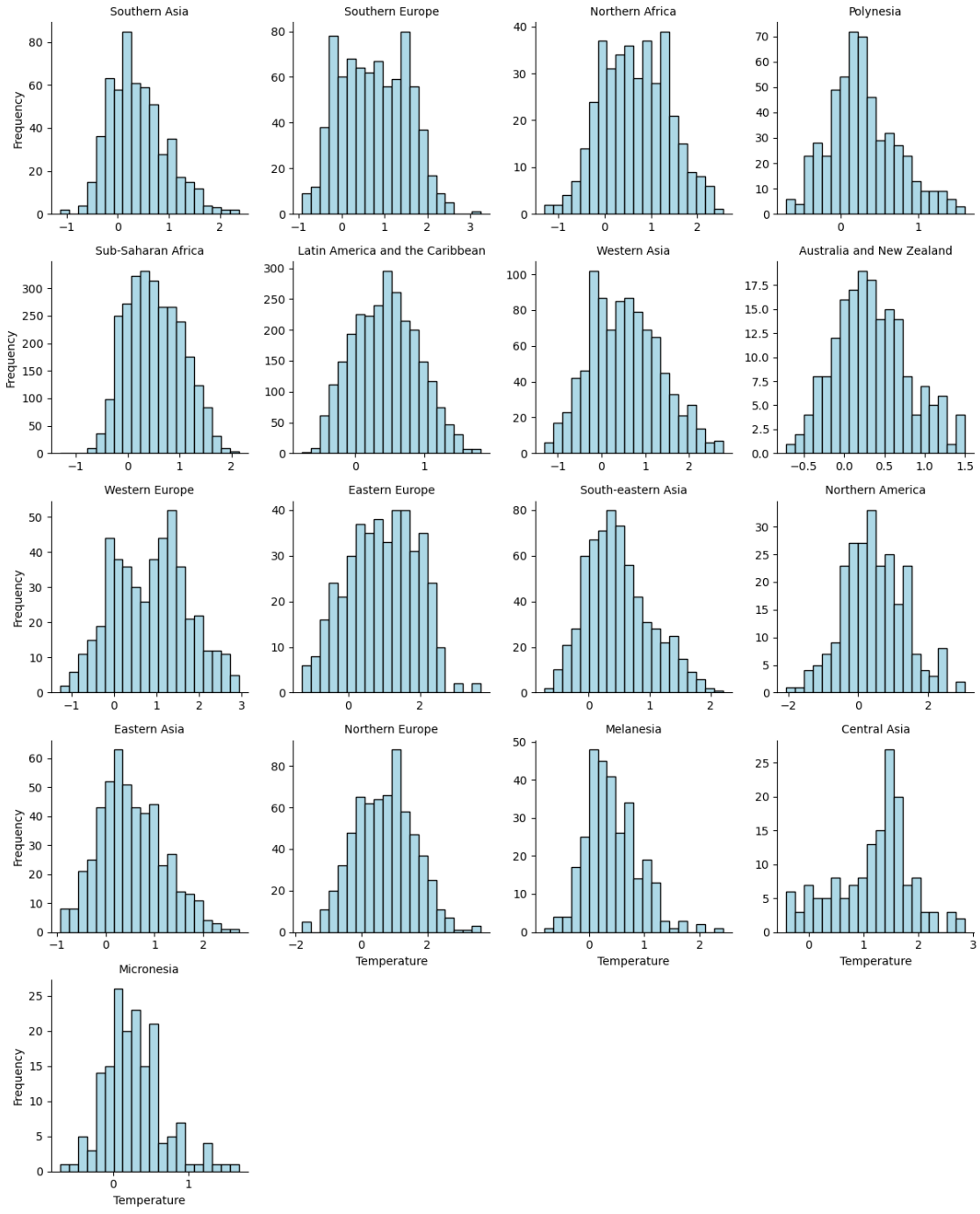


[122]:
```
# Convert year to numeric
df_climate_melt['year'] = pd.to_numeric(df_climate_melt['year'],␣
 ↪errors='coerce')

# Display plot
plt.figure(figsize=(12, 8))
sns.lineplot(data=df_climate_melt, x='year', y='temperature', hue='sub-region',␣
 ↪errorbar=None,palette='tab20', alpha=0.75)
plt.legend(title='Sub-Region', bbox_to_anchor=(1, 1), loc='upper left')
plt.title('Temperature Changes Sub-Region')
plt.xlabel('Year')
plt.ylabel('Temperature Change (°C)')
plt.show()
```



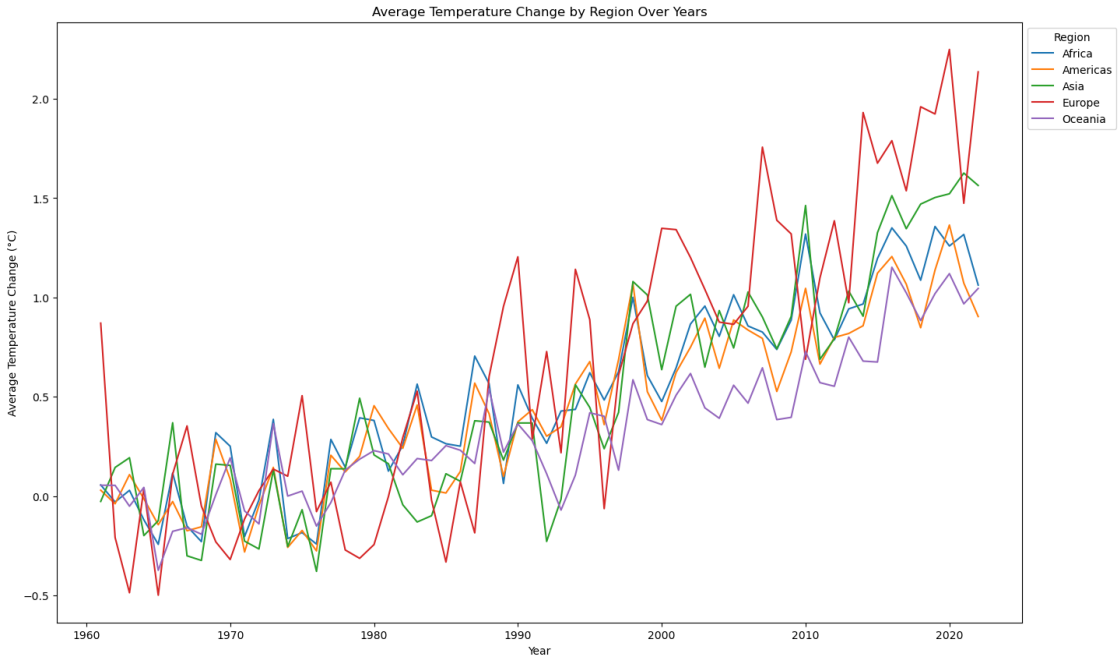
[123]:
```
# FacetGrid to plot multiple histograms, plot the histograms for each
g = sns.FacetGrid(df_climate_melt, col='sub-region', col_wrap=4, sharex=False,␣
↪sharey=False)
g.map(plt.hist, 'temperature', bins=20, color='lightblue', edgecolor='black')

# display plot
g.set_titles('{col_name}')
g.set_axis_labels('Temperature', 'Frequency')
plt.show()
# In the next step, make a group of the regions with more spread,
#the ones that had more than 50% of marking above 1 or below -1
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```



[124]:
```
# Aggregate it by 'year' and 'region' to get the average
region_year_avg = df_climate_melt.groupby(['year', 'region'])['temperature'].
  ↪mean().reset_index()

# Display line plot
plt.figure(figsize=(16, 10))
```

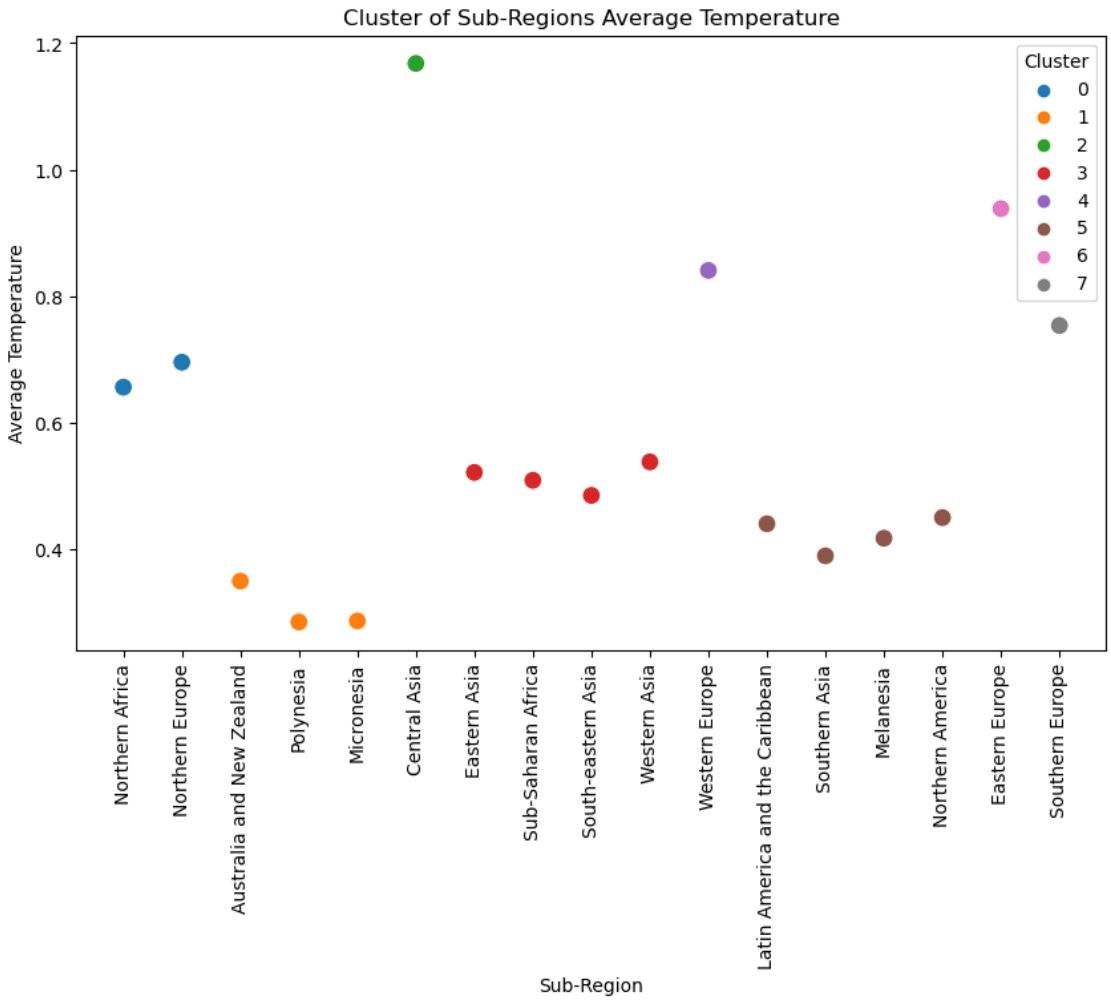
```
sns.lineplot(data=region_year_avg, x='year', y='temperature', hue='region',␣
 ↪color='tab10')
plt.title('Average Temperature Change by Region Over Years')
plt.ylabel('Average Temperature Change (°C)')
plt.xlabel('Year')
plt.legend(title='Region', bbox_to_anchor=(1, 1), loc='upper left')
plt.show()
#Temperature change in Europe and Asia show a remarkalbe growth compare to␣
 ↪others
```



[126]:
```
# n_clusters=8
#Confirming Temperature change in Europe and Asia

# Cluster based on the average temperature for each sub-region
data_for_clustering = df_climate_melt.groupby('sub-region')['temperature'].
 ↪mean().reset_index()

# Perform K-Means clustering, experiment with (n_clusters)
kmeans = KMeans(n_clusters=8, random_state=0).
 ↪fit(data_for_clustering[['temperature']])

# Assign the cluster labels
data_for_clustering['cluster'] = kmeans.labels_

# Sort the clusters
```

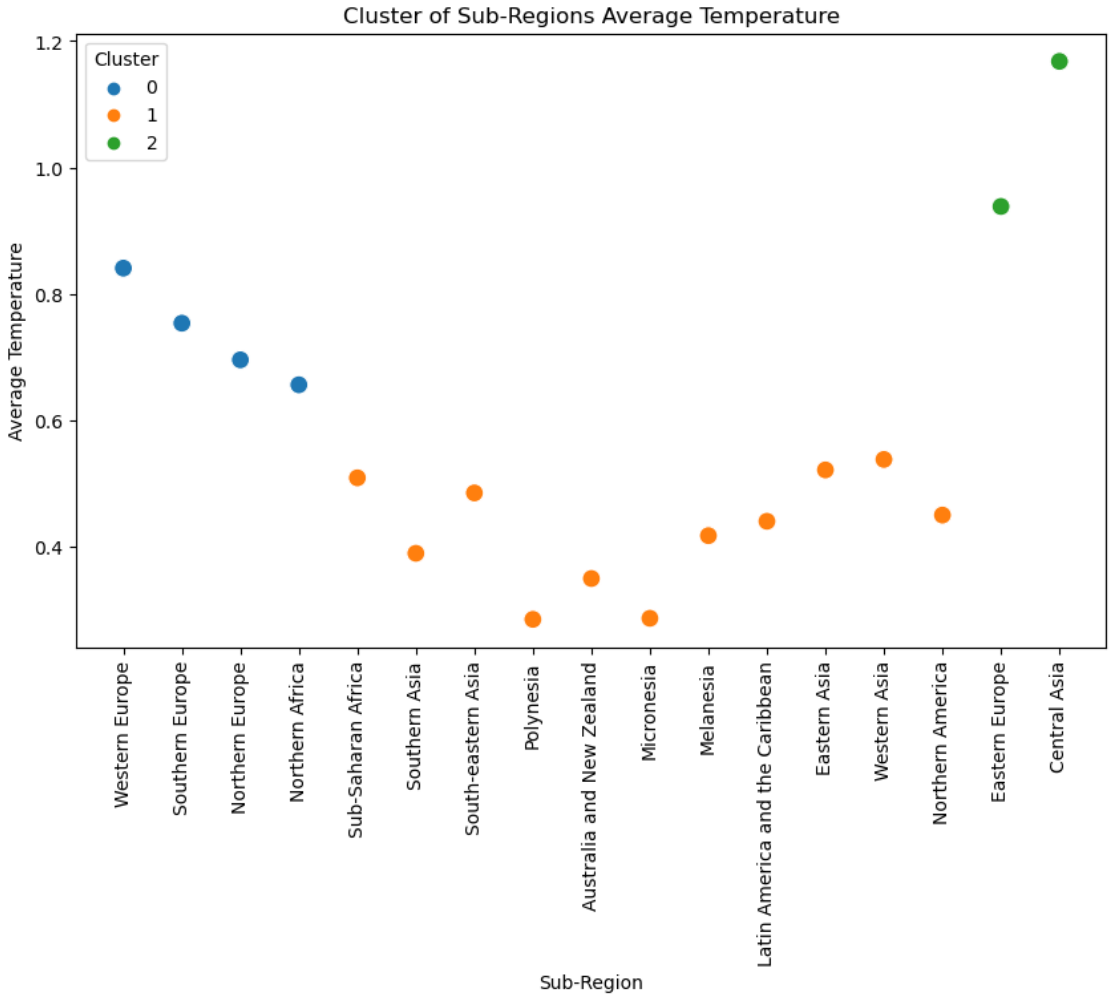
```
ordered_clusters = data_for_clustering.sort_values('cluster')

# Dis[lay plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=ordered_clusters, x='sub-region', y='temperature',␣
 ↪hue='cluster', palette='tab10', s=100)
plt.xticks(rotation=90)
plt.title('Cluster of Sub-Regions Average Temperature')
plt.xlabel('Sub-Region')
plt.ylabel('Average Temperature')
plt.legend(title='Cluster')
plt.show()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/cluster/_kmeans.py:1412: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



[127]:
```
# n_clusters=3
#Confirming Temperature change in Europe and Asia

# Cluster based on the average temperature for each sub-region
data_for_clustering = df_climate_melt.groupby('sub-region')['temperature'].
 ↪mean().reset_index()

# Perform K-Means clustering, experiment with (n_clusters)
kmeans = KMeans(n_clusters=3, random_state=0).
 ↪fit(data_for_clustering[['temperature']])

# Assign the cluster labels
data_for_clustering['cluster'] = kmeans.labels_

# Sort the clusters
ordered_clusters = data_for_clustering.sort_values('cluster')

# Dis[lay plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=ordered_clusters, x='sub-region', y='temperature',␣
 ↪hue='cluster', palette='tab10', s=100)
plt.xticks(rotation=90)
plt.title('Cluster of Sub-Regions Average Temperature')
plt.xlabel('Sub-Region')
plt.ylabel('Average Temperature')
plt.legend(title='Cluster')
plt.show()
```

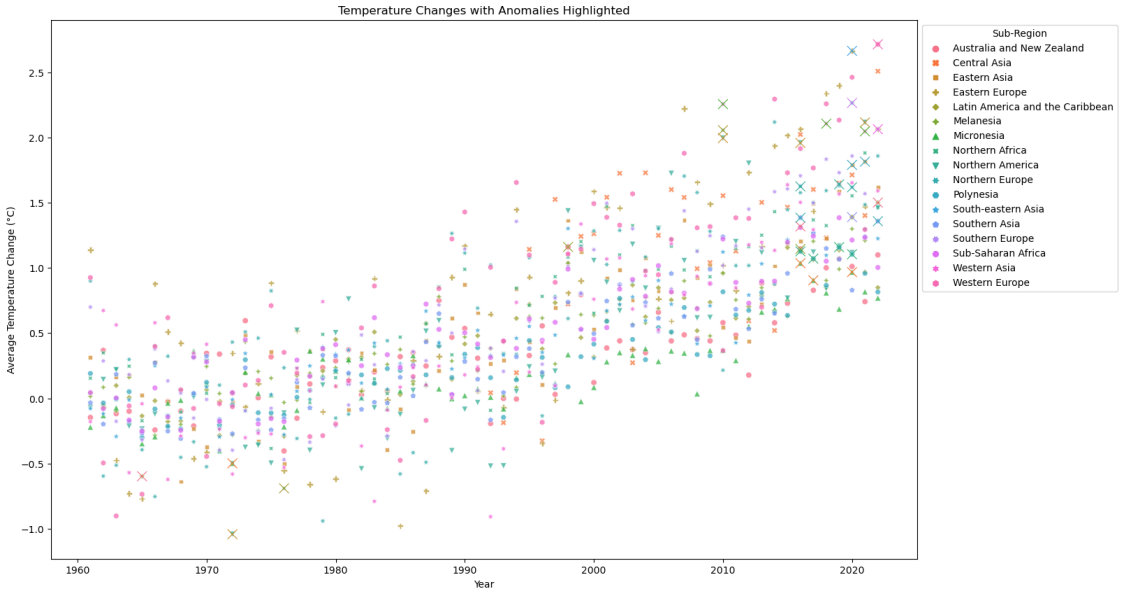
```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/cluster/_kmeans.py:1412: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



[128]:
```
# Group the data by 'year' and 'sub-region' and calculate the mean temperature␣
 ↪for each group
grouped_data = df_climate_melt.groupby(['year', 'sub-region'])['temperature'].
 ↪mean().reset_index()

# Calculate Z-score within each sub-region across different years
grouped_data['z_score'] = grouped_data.groupby('sub-region')['temperature'].
 ↪transform(lambda x: zscore(x, ddof=1))

# Filter to identify years with unusual temperature changes (absolute z_score >␣
 ↪2)
anomalies = grouped_data[grouped_data['z_score'].abs() > 2]

# Plot average temperature
plt.figure(figsize=(16, 10))
```

```
sns.scatterplot(data=grouped_data, x='year', y='temperature', hue='sub-region',␣
 ↪style='sub-region', alpha=0.75)

# Plot with Highlighted anomalies in the data
sns.scatterplot(data=anomalies, x='year', y='temperature', hue='sub-region',␣
 ↪s=100, legend=False, edgecolor='black', marker='x')

plt.title('Temperature Changes with Anomalies Highlighted')
plt.xlabel('Year')
plt.ylabel('Average Temperature Change (°C)')
plt.legend(title='Sub-Region', bbox_to_anchor=(1, 1), loc='upper left')
plt.show()
#Very interesting view of the changes in climate but too busy and confusing to␣
 ↪read
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/relational.py:573: UserWarning: You passed a
edgecolor/edgecolors ('black') for an unfilled marker ('x').  Matplotlib is
ignoring the edgecolor in favor of the facecolor.  This behavior may change in
the future.
  points = ax.scatter(x=x, y=y, **kws)
```



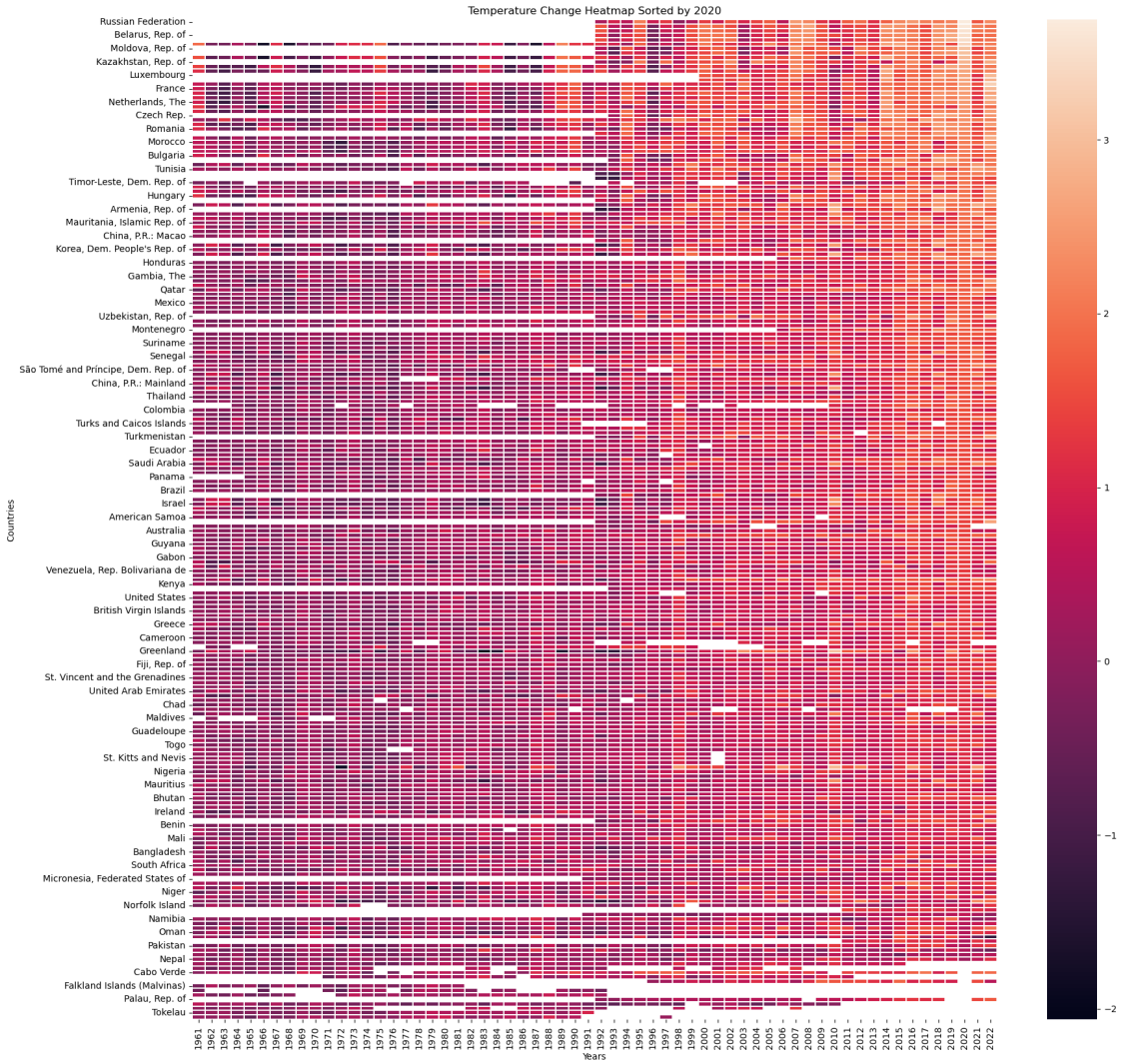
[129]:
```
# Display the anomalies Data Frame as a list
print(anomalies[['year', 'sub-region', 'temperature', 'z_score']].
 ↪to_string(index=False))
```

```
 year                    sub-region  temperature   z_score
 1965                     Melanesia    -0.597800 -2.347382
```

```
1972                     Micronesia    -0.498500 -2.086777
1972                Northern America    -1.041750 -2.030877
1976                 Northern Africa    -0.690000 -2.065177
1998       Australia and New Zealand     1.160667  2.119502
2010                 Northern Africa     2.055667  2.222212
2010                Northern America     1.994750  2.103725
2010                    Western Asia     2.255313  2.318433
2016       Australia and New Zealand     1.145333  2.079427
2016                       Melanesia     1.317750  2.038508
2016                      Micronesia     1.036000  2.421688
2016                Northern America     1.958000  2.053685
2016                      Polynesia     1.126857  2.176716
2016              South-eastern Asia     1.624545  2.303212
2016                   Southern Asia     1.384667  2.113459
2017                      Micronesia     0.905000  2.036801
2017                      Polynesia     1.071286  2.026469
2018                    Western Asia     2.105437  2.120036
2019                      Polynesia     1.158857  2.263233
2019              South-eastern Asia     1.640455  2.335232
2020                    Eastern Asia     1.789625  2.144988
2020                  Eastern Europe     2.663500  2.167614
2020 Latin America and the Caribbean     1.388927  2.279632
2020                      Micronesia     0.967667  2.220920
2020                  Northern Europe     2.264500  2.094737
2020                      Polynesia     1.104571  2.116463
2020              South-eastern Asia     1.617818  2.289673
2021                    Eastern Asia     1.814500  2.187053
2021                Northern America     2.115250  2.267802
2021                    Western Asia     2.046562  2.042100
2022                      Melanesia     1.501000  2.458082
2022                   Southern Asia     1.357778  2.056622
2022                  Southern Europe     2.063188  2.062366
2022                   Western Europe     2.714000  2.270726
```

[133]:
```
# Sort the countries based on their temperature change in 2020,
# the year with more anomalies
sorted_countries = df_climate.set_index('country').sort_values(by='2020',␣
 ↪ascending=False)

# Drop other columns
heatmap_data = sorted_countries.drop(['ISO3', 'ISO2', 'ObjectId', 'region',␣
 ↪'sub-region'], axis=1)

# Display plot heatmap
plt.figure(figsize=(20, 20))
sns.color_palette("flare", as_cmap=True)
sns.heatmap(heatmap_data, linecolor='white', linewidths=0.1)
```

```
plt.title('Temperature Change Heatmap Sorted by 2020')
plt.xlabel('Years')
plt.ylabel('Countries')
plt.show()
```

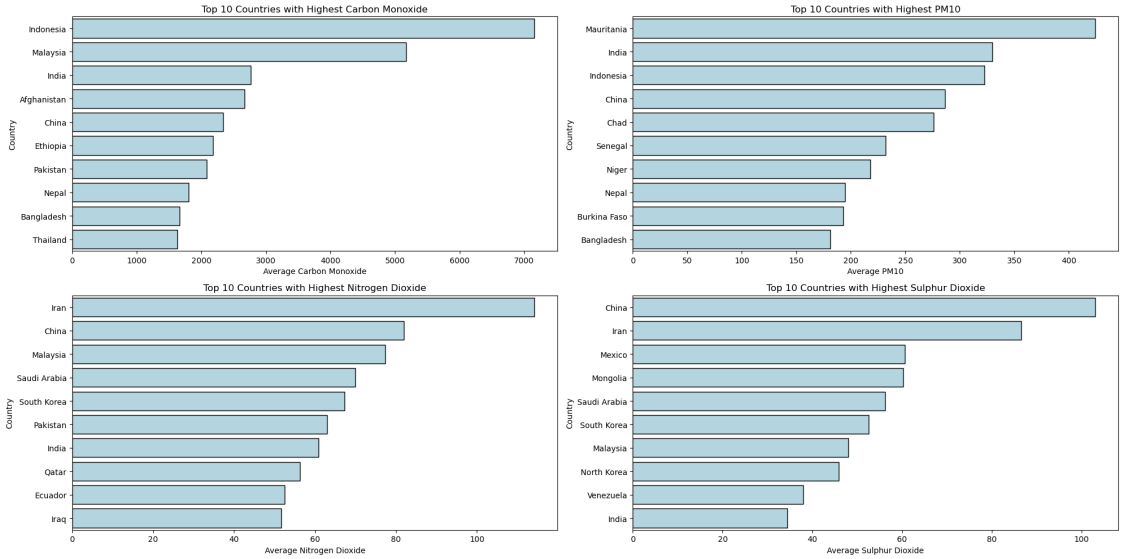


0.0.5 4. EDA — Weather

[138]:
```
# Based on finding of google search I will use 'air_quality_Carbon_Monoxide' ,␣
 ↪'air_quality_PM10' ,
# 'air_quality_Nitrogen_dioxide', and 'air_quality_Sulphur_dioxide to plot␣
 ↪average of countries
# to understand if the worst values for this air quality match with the highest␣
 ↪temprature change
```

```
# Aggregate data for each country
avg_air_quality = df_weather.groupby('country')[[
    'air_quality_Carbon_Monoxide',
    'air_quality_Nitrogen_dioxide',
    'air_quality_Sulphur_dioxide',
    'air_quality_PM10'
]].mean().reset_index()

# Sort the countries based on each air quality measurement
sorted_co = avg_air_quality.sort_values(by='air_quality_Carbon_Monoxide',␣
 ↪ascending=False).head(10)
sorted_pm10 = avg_air_quality.sort_values(by='air_quality_PM10',␣
 ↪ascending=False).head(10)
sorted_no2 = avg_air_quality.sort_values(by='air_quality_Nitrogen_dioxide',␣
 ↪ascending=False).head(10)
sorted_so2 = avg_air_quality.sort_values(by='air_quality_Sulphur_dioxide',␣
 ↪ascending=False).head(10)


fig, axes = plt.subplots(2, 2, figsize=(20, 10))

# Plot for 'air_quality_Carbon_Monoxide'
sns.barplot(x='air_quality_Carbon_Monoxide', y='country', data=sorted_co,␣
 ↪ax=axes[0, 0], color='lightblue', edgecolor='black')
axes[0, 0].set_title('Top 10 Countries with Highest Carbon Monoxide')
axes[0, 0].set_xlabel('Average Carbon Monoxide')
axes[0, 0].set_ylabel('Country')

# Plot for 'air_quality_PM10'
sns.barplot(x='air_quality_PM10', y='country', data=sorted_pm10, ax=axes[0, 1],␣
 ↪color='lightblue', edgecolor='black')
axes[0, 1].set_title('Top 10 Countries with Highest PM10')
axes[0, 1].set_xlabel('Average PM10')
axes[0, 1].set_ylabel('Country')

# Plot for 'air_quality_Nitrogen_dioxide'
sns.barplot(x='air_quality_Nitrogen_dioxide', y='country', data=sorted_no2,␣
 ↪ax=axes[1, 0], color='lightblue', edgecolor='black')
axes[1, 0].set_title('Top 10 Countries with Highest Nitrogen Dioxide')
axes[1, 0].set_xlabel('Average Nitrogen Dioxide')
axes[1, 0].set_ylabel('Country')

# Plot for 'air_quality_Sulphur_dioxide'
sns.barplot(x='air_quality_Sulphur_dioxide', y='country', data=sorted_so2,␣
 ↪ax=axes[1, 1], color='lightblue', edgecolor='black')
axes[1, 1].set_title('Top 10 Countries with Highest Sulphur Dioxide')
```

```
axes[1, 1].set_xlabel('Average Sulphur Dioxide')
axes[1, 1].set_ylabel('Country')

plt.tight_layout()
plt.show()

# Weather dataset is inconclusive, much more explorations are needed,
# this could be done in another project folowing this one.
```



[ ]:

Project_Step5

0.0.1 1. Import Libraries

[74]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
print('I am done')
```

```
I am done
```

0.0.2 2. Load Datasets

[76]:
```
# Importing data frames
df_climate = pd.read_csv("Climate_Region.csv")
df_climate_melt = pd.read_csv("Climate_Region_Melted.csv")
```

0.0.3 3. Random Forest

[84]:
```
#Random Forest

# Drop 'temperature' = NaN
df_climate_melt = df_climate_melt.dropna(subset=['temperature'])

# Define x and y
X = df_climate_melt.drop(['temperature', 'ISO3', 'ISO2', 'ObjectId',␣
 ↪'country'], axis=1)
y = df_climate_melt['temperature']
```

```
# Categorical and numerical features
categorical_features = ['region', 'sub-region']
numeric_features = ['year']

# Transform features
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

# Create a pipeline
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('regressor',␣
 ↪RandomForestRegressor(n_estimators=100, random_state=42))])

# Split the data into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train the Random Forest Regressor
pipeline.fit(X_train, y_train)

# Predict on the test data
y_pred = pipeline.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print('Mean squared error =', mse)
```

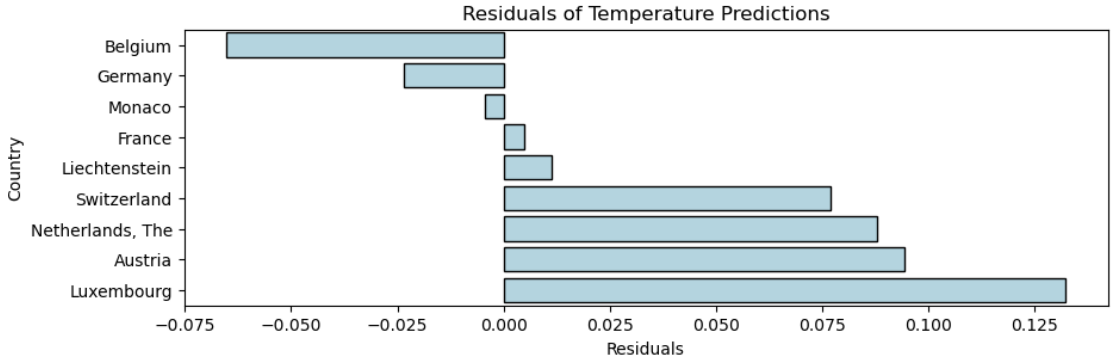
```
Mean squared error = 0.10505764046690876
```

Given the scale of climate change values varying from -1 to 3ºC the mean squared error suggests that the model has a acceptable predictive capability for this project.

0.0.4 4. Prediction

[131]:
```
# Generate predictions - Western Europe

predicted_temperatures = pipeline.predict(X_test)
df_climate_melt.loc[X_test.index, 'predicted_temperature'] =␣
 ↪predicted_temperatures

# Filter European countries
european_countries = df_climate_melt[df_climate_melt['sub-region'] == 'Western␣
 ↪Europe']
```

```
# Calculate residuals only for rows where 'predicted_temperature' is not NaN
european_countries = european_countries.dropna(subset=['predicted_temperature'])
european_countries['residuals'] = european_countries['temperature'] -␣
 ↪european_countries['predicted_temperature']

# Aggregate residuals by country and calculate the mean residual
residuals_by_country = european_countries.groupby('country')['residuals'].
 ↪mean().reset_index()

# Plotting residuals
plt.figure(figsize=(10, 3))
sns.barplot(x='residuals', y='country', data=residuals_by_country.
 ↪sort_values(by='residuals'), color='lightblue', edgecolor='black')
plt.xlabel('Residuals')
plt.ylabel('Country')
plt.title('Residuals of Temperature Predictions')
plt.show()
```



[133]:
```
# Generate predictions - Eastern Europe

predicted_temperatures = pipeline.predict(X_test)
df_climate_melt.loc[X_test.index, 'predicted_temperature'] =␣
 ↪predicted_temperatures

# Filter European countries
european_countries = df_climate_melt[df_climate_melt['sub-region'] == 'Eastern␣
 ↪Europe']

# Calculate residuals only for rows where 'predicted_temperature' is not NaN
european_countries = european_countries.dropna(subset=['predicted_temperature'])
european_countries['residuals'] = european_countries['temperature'] -␣
 ↪european_countries['predicted_temperature']
```

```
# Aggregate residuals by country and calculate the mean residual
residuals_by_country = european_countries.groupby('country')['residuals'].
 ↪mean().reset_index()

# Plotting residuals
plt.figure(figsize=(10, 3))
sns.barplot(x='residuals', y='country', data=residuals_by_country.
 ↪sort_values(by='residuals'), color='lightblue', edgecolor='black')
plt.xlabel('Residuals')
plt.ylabel('Country')
plt.title('Residuals of Temperature Predictions')
plt.show()
```