DANIEL KARUME

SCT211-0072/2022

DATA VISUALISATION USING PANDAS AND MATPLOTLIB

LAB-07

Transition to a Low-Carbon Economy - NGFS Transition Pathways

Dataset Link

Overview: This dataset, titled "Transition to a Low-Carbon Economy - NGFS Transition Pathways," presents a comprehensive compilation and visualization of key indicators derived from climate scenarios developed by the Network for Greening the Financial System (NGFS). The dataset is curated in collaboration with the International Monetary Fund (IMF), reflecting a joint effort to provide a detailed and insightful representation of crucial metrics and data points.

Purpose: The primary objective of this dataset is to offer a comprehensive understanding of the intricate dynamics and implications associated with climate-related scenarios. The selected indicators aim to illuminate the intersection between environmental factors and financial considerations within the overarching framework of NGFS climate scenarios. By focusing on these key indicators, stakeholders can gain valuable insights into the potential impacts of climate scenarios on both the environment and the financial system.

Dataset Details:

- **Title:** Transition to a Low-Carbon Economy - NGFS Transition Pathways
- **Source:** Network for Greening the Financial System (NGFS), in collaboration with the International Monetary Fund (IMF)
- **Scope:** Climate scenarios covering the period 2020-2050
- **Release Date:** November 07, 2023
- **Number of Indicators:** 184

This dataset serves as a valuable resource for researchers, policymakers, and analysts aiming to explore and understand the transition pathways towards a low-carbon economy as envisioned by the NGFS. The collaboration between the NGFS and IMF enhances the dataset's credibility and ensures a rigorous representation of key factors shaping the future landscape of sustainable finance and environmental considerations.

In [7]:
```
from tabulate import tabulate
import pandas as pd
import matplotlib.pyplot as plt
```

The code above imports three Python libraries: `tabulate`, `pandas`, and `matplotlib.pyplot`.

Here's a brief note on what each part of the code does:

1. `from tabulate import tabulate` : This line imports the `tabulate` module, which is used for formatting and displaying tabular data in a visually appealing way. It provides functions to convert data structures like pandas DataFrames into formatted tables.

2. `import pandas as pd` : This line imports the `pandas` library and assigns it the alias `pd` . `pandas` is a powerful data manipulation and analysis library in Python. The alias `pd` is commonly used for brevity when referring to pandas functions.

3. `import matplotlib.pyplot as plt` : This line imports the `pyplot` module from the `matplotlib` library and assigns it the alias `plt` . `matplotlib.pyplot` provides a collection of functions for creating various types of plots and visualizations.

In [4]:
```
df = pd.read_csv("./data/NGFS_Key_Indicators.csv")
```

The code `df = pd.read_csv("./data/NGFS_Key_Indicators.csv")` reads a CSV (Comma-Separated Values) file named "NGFS_Key_Indicators.csv" and loads its contents into a pandas DataFrame. Here's a breakdown of each part:

- **`pd` alias for pandas**: The `import pandas as pd` statement was used to import the pandas library and give it the alias `pd` . This is a common convention to make the code more concise.

- **`pd.read_csv("./data/NGFS_Key_Indicators.csv")`** : The `pd.read_csv()` function is a pandas function that reads data from a CSV file. In this case, it reads the contents of the file located at "./data/NGFS_Key_Indicators.csv". The result is a pandas DataFrame (`df`), which is a two-dimensional, tabular data structure.

- **DataFrame (`df`)**: The DataFrame is a fundamental pandas object that organizes data into rows and columns, similar to a table in a relational database or a spreadsheet. It provides various methods and functions for data manipulation, exploration, and analysis.

DATA EXPLORATION

In this section, we delve into exploring the dataset to gain a comprehensive understanding. Various techniques are employed to uncover insights about the data.

Preview of the Dataset

Head

We begin by inspecting the initial rows of the dataset using the `head()` function. This provides a snapshot of the first few records, offering a quick overview.

```
print(df.head())
```

Tail

Similarly, we explore the end of the dataset with the `tail()` function. This reveals the concluding records, ensuring a complete view of the dataset.

```
print(df.tail())
```

Statistical Summary

To gain a statistical overview of the dataset, we utilize the `describe()` function. This method furnishes essential statistical measures such as mean, standard deviation, and quartiles for numerical columns.

```
print(df.describe())
```

Below is Python code with an indepth definition of the code.

In [17]:
```
def explore_dataframe(dataframe):
    """
    Print the head, tail, and summary statistics of a DataFrame.

    Parameters:
    - dataframe (pd.DataFrame): The DataFrame to explore.

    Returns:
    None
    """

    print("\nDataFrame Head:")
    print(tabulate(dataframe.head(), headers='keys', tablefmt='pretty', showindex=False)

    print("\nDataFrame Tail:")
    print(tabulate(dataframe.tail(), headers='keys', tablefmt='pretty', showindex=False)

    print("\nDataFrame Summary Statistics:")
    print(tabulate(dataframe.describe(), headers='keys', tablefmt='pretty', showindex=Fa


explore_dataframe(df)
```

```
DataFrame Head:
+----------+------------------------------+------+------+---------------------+---------
--------+----------------+-------+--------------------+-------------------------------
-----------------------------------------------------------+----------+----------+-------
--------------+----------+----------+----------+----------+----------+----------+-------
---+
| ObjectId |           Country            | ISO2 | ISO3 |      Indicator      |    Varia
ble     |      Unit      | Model |     Scenarios      |
         Source                                            | CTS_Code | CTS_Name | CTS_Fu
ll_Descriptor |  F2020   |  F2025   |  F2030   |  F2035   |  F2040   |  F2045   |  F2050
   |
+----------+------------------------------+------+------+---------------------+---------
--------+----------------+-------+--------------------+-------------------------------
-----------------------------------------------------------+----------+----------+-------
--------------+----------+----------+----------+----------+----------+----------+-------
---+
|    1     | Afghanistan, Islamic Rep. of |  AF  | AFG  | Surface temperature | 50th per
centile | Degree Celsius | GCAM  |     Below 2°C      | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 13.90978 | 14.20393 | 14.42557 | 14.63334 | 14.76049 | 14.81978 | 14.870
96 |
|    2     | Afghanistan, Islamic Rep. of |  AF  | AFG  | Surface temperature | 50th per
centile | Degree Celsius | GCAM  |  Current Policies  | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 13.93854 | 14.20575 | 14.40551 | 14.62693 | 14.78472 | 14.9447  | 15.155
75 |
|    3     | Afghanistan, Islamic Rep. of |  AF  | AFG  | Surface temperature | 50th per
centile | Degree Celsius | GCAM  | Delayed transition | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 13.90871 | 14.18438 | 14.39542 | 14.65662 | 14.83033 | 14.91812 | 14.964
96 |
|    4     | Afghanistan, Islamic Rep. of |  AF  | AFG  | Surface temperature | 50th per
centile | Degree Celsius | GCAM  |  Fragmented World  | Sources: NGFS (2023), Scenarios
```

```
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 13.90964 | 14.15194 | 14.33653 | 14.52832 | 14.73503 | 14.91799 | 15.096
18 |
|    5     | Afghanistan, Islamic Rep. of |  AF  | AFG  | Surface temperature | 50th per
centile | Degree Celsius | GCAM  |     Low Demand     | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 13.90845 | 14.17709 | 14.39153 | 14.58657 | 14.69253 | 14.72458 | 14.683
49 |
+----------+------------------------------+------+------+---------------------+---------
--------+----------------+-------+--------------------+--------------------------------
--------------------------------------------------------+----------+----------+-------
--------------+----------+----------+----------+----------+----------+----------+-------
---+

DataFrame Tail:
+----------+------------------------------+------+------+-------------+----------+-----
--+-------+------------------------------------------------+--------------------------------
--------------------------------------------------------+----------+----------+------
---------------+--------+--------+--------+--------+--------+--------+--------+
| ObjectId |           Country            | ISO2 | ISO3 |  Indicator   | Variable | Unit
  | Model |                 Scenarios                  |
         Source                                          | CTS_Code | CTS_Name | CTS_F
ull_Descriptor | F2020  | F2025  | F2030  | F2035  | F2040  | F2045  | F2050  |
+----------+------------------------------+------+------+-------------+----------+-----
--+-------+------------------------------------------------+--------------------------------
--------------------------------------------------------+----------+----------+------
---------------+--------+--------+--------+--------+--------+--------+--------+
|   495    | Afghanistan, Islamic Rep. of |  AF  | AFG  | Final energy |  Solids  | EJ/y
r | GCAM  |              Current Policies              | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 0.187  | 0.1882 | 0.2006 | 0.2095 | 0.2178 | 0.2345 | 0.2449 |
|   496    | Afghanistan, Islamic Rep. of |  AF  | AFG  | Final energy |  Solids  | EJ/y
r | GCAM  |             Delayed transition             | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 0.1871 | 0.1882 | 0.2008 | 0.2083 | 0.1933 |  0.17  | 0.1307 |
|   497    | Afghanistan, Islamic Rep. of |  AF  | AFG  | Final energy |  Solids  | EJ/y
r | GCAM  |              Fragmented World              | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 0.1845 | 0.1909 | 0.207  | 0.2156 | 0.2196 | 0.2317 | 0.2349 |
|   498    | Afghanistan, Islamic Rep. of |  AF  | AFG  | Final energy |  Solids  | EJ/y
r | GCAM  |                 Low Demand                 | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 0.1861 | 0.1912 | 0.2046 | 0.1993 | 0.1703 | 0.1236 | 0.0883 |
|   499    | Afghanistan, Islamic Rep. of |  AF  | AFG  | Final energy |  Solids  | EJ/y
r | GCAM  | Nationally Determined Contributions (NDCs) | Sources: NGFS (2023), Scenarios
Portal; and IIASA (2023), NGFS Phase 4 Scenario Explorer. |   nan    |   nan    |
  nan         | 0.1852 | 0.1941 | 0.2077 | 0.2171 | 0.2213 | 0.2325 | 0.2319 |
+----------+------------------------------+------+------+-------------+----------+-----
--+-------+------------------------------------------------+--------------------------------
--------------------------------------------------------+----------+----------+------
---------------+--------+--------+--------+--------+--------+--------+--------+

DataFrame Summary Statistics:
+--------------------+----------+----------+--------------------+--------------------+-
--------------------+--------------------+----------------------+--------------------+----
----------------+--------------------+
|      ObjectId      | CTS_Code | CTS_Name | CTS_Full_Descriptor |       F2020        |
      F2025        |       F2030        |         F2035        |       F2040        |
   F2045        |       F2050        |
+--------------------+----------+----------+--------------------+--------------------+-
--------------------+--------------------+----------------------+--------------------+----
----------------+--------------------+
|       499.0        |   0.0    |   0.0    |         0.0        |       489.0        |
      497.0        |       497.0        |         499.0        |       499.0        |
   499.0        |       499.0        |
|       250.0        |   nan    |   nan    |         nan        | 2.766486274662577  |
```

```
3.3266730396981887 | 5.962311828672033 |   7.609071070881763   | 9.495128483927855 | 12.
464856263807615 | 15.141707295971944 |
| 144.19315748906627 |   nan    |   nan    |         nan         | 4.9958874682018255 |
7.351311837661896  | 34.35868187788973 |  38.777621609316924   | 49.81839042350675 | 71.
23962677038003  | 88.65917346373874  |
|        1.0         |   nan    |   nan    |         nan         |      -0.1776       |
     -1.8188       |     -24.6789      |        -4.4926        |      -5.5027      |
  -5.9483       |      -6.4265       |
|       125.5        |   nan    |   nan    |         nan         |        0.0         |
      0.0002       |      0.0007       | 0.0020499999999999997 |      0.0038       |
   0.0073       |       0.0131       |
|       250.0        |   nan    |   nan    |         nan         |       0.014        |
      0.0202       |      0.0348       |        0.0511         |       0.083       |
   0.1252       |       0.1725       |
|       374.5        |   nan    |   nan    |         nan         |       2.6053       |
      3.6745       |      4.2757       |        5.1169         |      5.57625      |
   6.3291       |       6.7751       |
|       499.0        |   nan    |   nan    |         nan         |      15.0224       |
     99.5255       |     689.8968      |       691.1356        |     808.4434      |
1046.3668      |     1275.5484      |
+--------------------+----------+----------+---------------------+--------------------+-
-------------------+-------------------+-----------------------+-------------------+----
---------------+--------------------+
```

In [9]:

```python
def print_categorical_value_counts(dataframe):
    """
    Print value counts for categorical columns in a nicely formatted table.

    Parameters:
    - dataframe (pd.DataFrame): The DataFrame containing the data.

    Returns:
    None
    """
    categorical_columns = ["Model", "Scenarios"]
    for column in categorical_columns:
        table = tabulate(
            dataframe[column].value_counts().reset_index(),
            headers=["Value", "Count"],
            tablefmt="pretty",
        )
        print(f"\nValue Counts for {column}:\n{table}")


print_categorical_value_counts(df)
```

```
Value Counts for Model:
+---+-------------------+-------+
|   |       Value       | Count |
+---+-------------------+-------+
| 0 |   REMIND-MAgPIE   |  171  |
| 1 | MESSAGEix-GLOBIOM |  168  |
| 2 |       GCAM        |  160  |
+---+-------------------+-------+

Value Counts for Scenarios:
+---+-----------------------------------------------+-------+
|   |                     Value                     | Count |
+---+-----------------------------------------------+-------+
| 0 |                  Below 2°C                    |  72   |
| 1 |             Delayed transition                |  72   |
| 2 |                 Low Demand                    |  72   |
| 3 | Nationally Determined Contributions (NDCs)    |  72   |
| 4 |               Net Zero 2050                   |  71   |
| 5 |              Current Policies                 |  70   |
```

```
| 6 |              Fragmented World              |  70   |
+---+--------------------------------------------+-------+
```

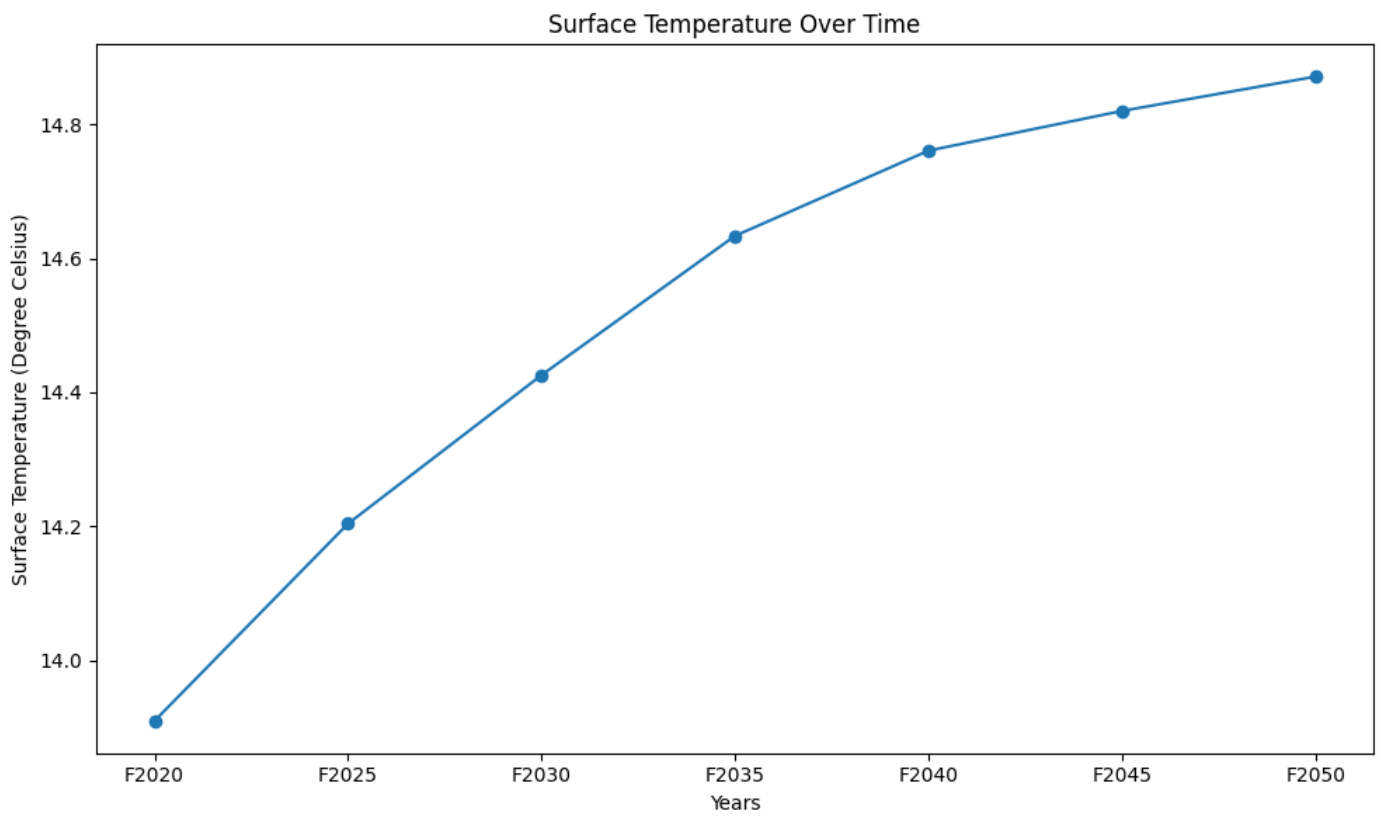
DATA VISUALIZATION

In this section, we embark on a visual exploration of the dataset, aiming to gain insights into the patterns, trends, and relationships within the data. Leveraging the power of data visualization through popular Python libraries like pandas and matplotlib, we employ various plotting techniques to graphically represent the dataset.

Line Plot Over Time

To understand the temporal evolution of surface temperatures, we commence with a line plot spanning the years from 2020 to 2050. The selected columns, such as "F2020," "F2025," and so forth, are plotted against the corresponding years, providing a visual narrative of the temperature changes over time.

In [10]:
```python
def plot_surface_temperature_over_time(dataframe):
    """
    Plot a line graph showing surface temperature over time.

    Parameters:
    - dataframe: Pandas DataFrame containing the dataset.

    Returns:
    - None
    """
    plt.figure(figsize=(10, 6))
    plt.plot(
        ["F2020", "F2025", "F2030", "F2035", "F2040", "F2045", "F2050"],
        dataframe.loc[0, "F2020":"F2050"],
        marker="o",
    )
    plt.title("Surface Temperature Over Time")
    plt.xlabel("Years")
    plt.ylabel("Surface Temperature (Degree Celsius)")
    plt.tight_layout()
    plt.show()


plot_surface_temperature_over_time(df)
```

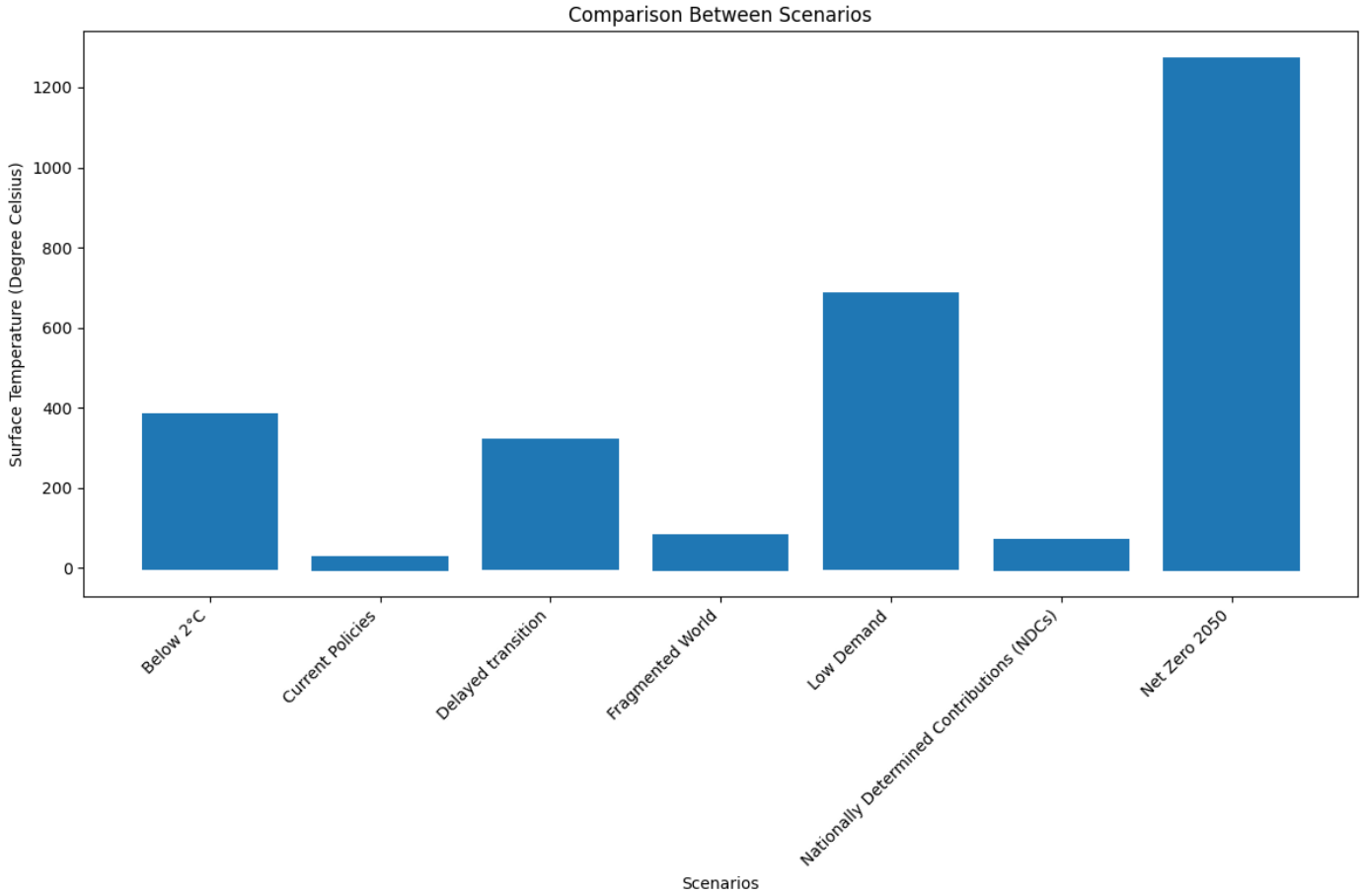


Comparison Between Scenarios

Next, we delve into scenario comparisons by plotting a bar chart, where different scenarios are juxtaposed based on the temperature projections for the year 2050. This visual representation aids in discerning the potential impacts of distinct scenarios on surface temperatures.

In [11]:

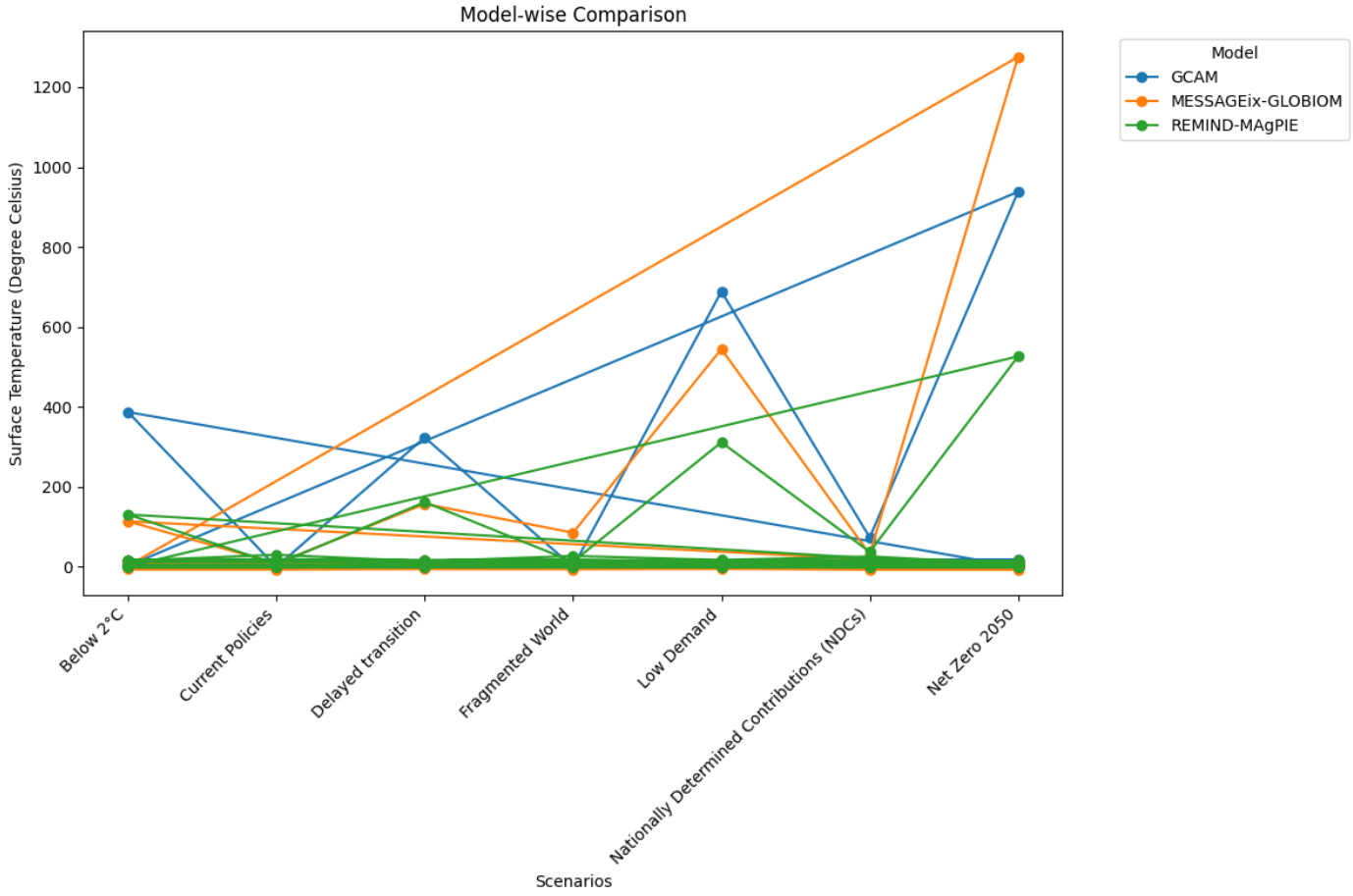
```
def plot_scenario_comparison(dataframe):
    """
    Plot a bar chart comparing surface temperatures between scenarios.

    Parameters:
    - dataframe: Pandas DataFrame containing the dataset.

    Returns:
    - None
    """
    plt.figure(figsize=(12, 8))
    plt.bar(dataframe["Scenarios"], dataframe["F2050"])
    plt.title("Comparison Between Scenarios")
    plt.xlabel("Scenarios")
    plt.ylabel("Surface Temperature (Degree Celsius)")
    plt.xticks(rotation=45, ha="right")
    plt.tight_layout()
    plt.show()


plot_scenario_comparison(df)
```



Model-wise Comparison

For a nuanced perspective, we conduct a model-wise comparison, plotting surface temperatures for different scenarios using distinct models. This line plot facilitates the identification of variations between models and their projections.

In [12]:
```
def plot_model_comparison(dataframe):
    """
    Plot a line graph comparing surface temperatures between models.

    Parameters:
    - dataframe: Pandas DataFrame containing the dataset.

    Returns:
    - None
    """
    plt.figure(figsize=(12, 8))
    for model in dataframe["Model"].unique():
        model_data = dataframe[dataframe["Model"] == model]
        plt.plot(model_data["Scenarios"], model_data["F2050"], label=model, marker="o")
    plt.title("Model-wise Comparison")
    plt.xlabel("Scenarios")
    plt.ylabel("Surface Temperature (Degree Celsius)")
    plt.xticks(rotation=45, ha="right")
    plt.legend(title="Model", bbox_to_anchor=(1.05, 1), loc="upper left")
    plt.tight_layout()
    plt.show()


plot_model_comparison(df)
```

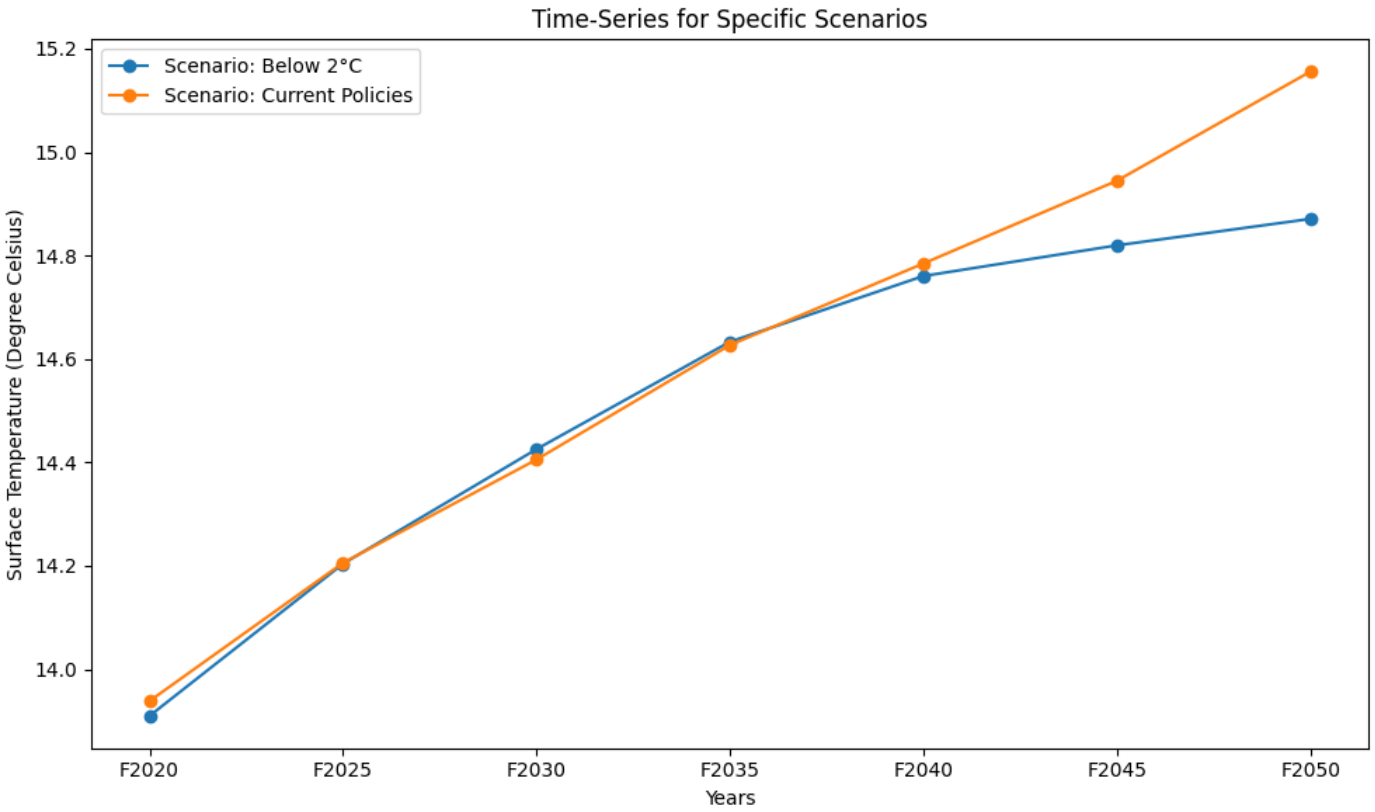


Regional Temperature Variations

To explore temperature variations across different regions, we employ a bar chart where countries or regions are plotted against their corresponding surface temperatures in 2050.

In [13]:
```python
def plot_time_series_for_scenarios(dataframe):
    """
    Plot time-series line graphs for specific scenarios.

    Parameters:
    - dataframe: Pandas DataFrame containing the dataset.

    Returns:
    - None
    """
    plt.figure(figsize=(10, 6))
    plt.plot(
        ["F2020", "F2025", "F2030", "F2035", "F2040", "F2045", "F2050"],
        dataframe.loc[0, "F2020":"F2050"],
        marker="o",
        label="Scenario: Below 2°C",
    )
    plt.plot(
        ["F2020", "F2025", "F2030", "F2035", "F2040", "F2045", "F2050"],
        dataframe.loc[1, "F2020":"F2050"],
        marker="o",
        label="Scenario: Current Policies",
    )
    plt.title("Time-Series for Specific Scenarios")
    plt.xlabel("Years")
    plt.ylabel("Surface Temperature (Degree Celsius)")
    plt.legend()
    plt.tight_layout()
    plt.show()
```

```
plot_time_series_for_scenarios(df)
```

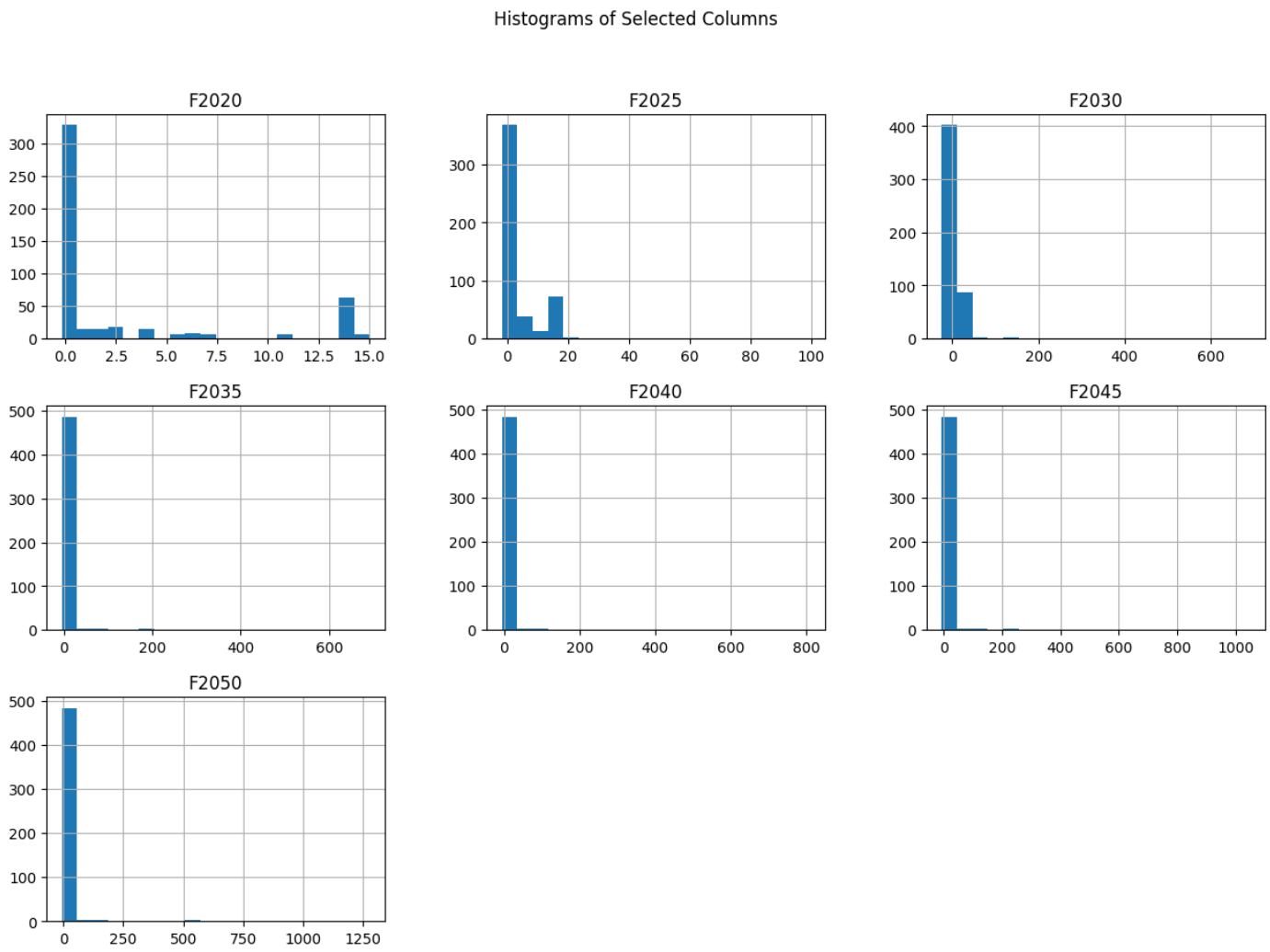


Histograms for Selected Columns

In this analysis, we utilize histograms to visually represent the distribution of data across selected columns: "F2020," "F2025," "F2030," "F2035," "F2040," "F2045," and "F2050." Each histogram provides insights into the frequency and spread of surface temperature projections for specific years. The use of bins facilitates a granular view of the data distribution.

These histograms collectively contribute to a deeper exploration of the distributional characteristics of the selected columns, aiding stakeholders in identifying patterns and potential trends within the dataset. The visualization offers a concise summary of the surface temperature projections, allowing for a quick and intuitive grasp of the underlying data distribution.

In [15]:
```
def plot_selected_histograms(df):
    """
    Plot histograms for a predefined set of columns in the DataFrame.

    Parameters:
    - df (pd.DataFrame): The DataFrame containing the data.

    Returns:
    None
    """
    selected_columns = ["F2020", "F2025", "F2030", "F2035", "F2040", "F2045", "F2050"]
    df[selected_columns].hist(bins=20, figsize=(15, 10))
    plt.suptitle("Histograms of Selected Columns")
    plt.show()


plot_selected_histograms(df)
```



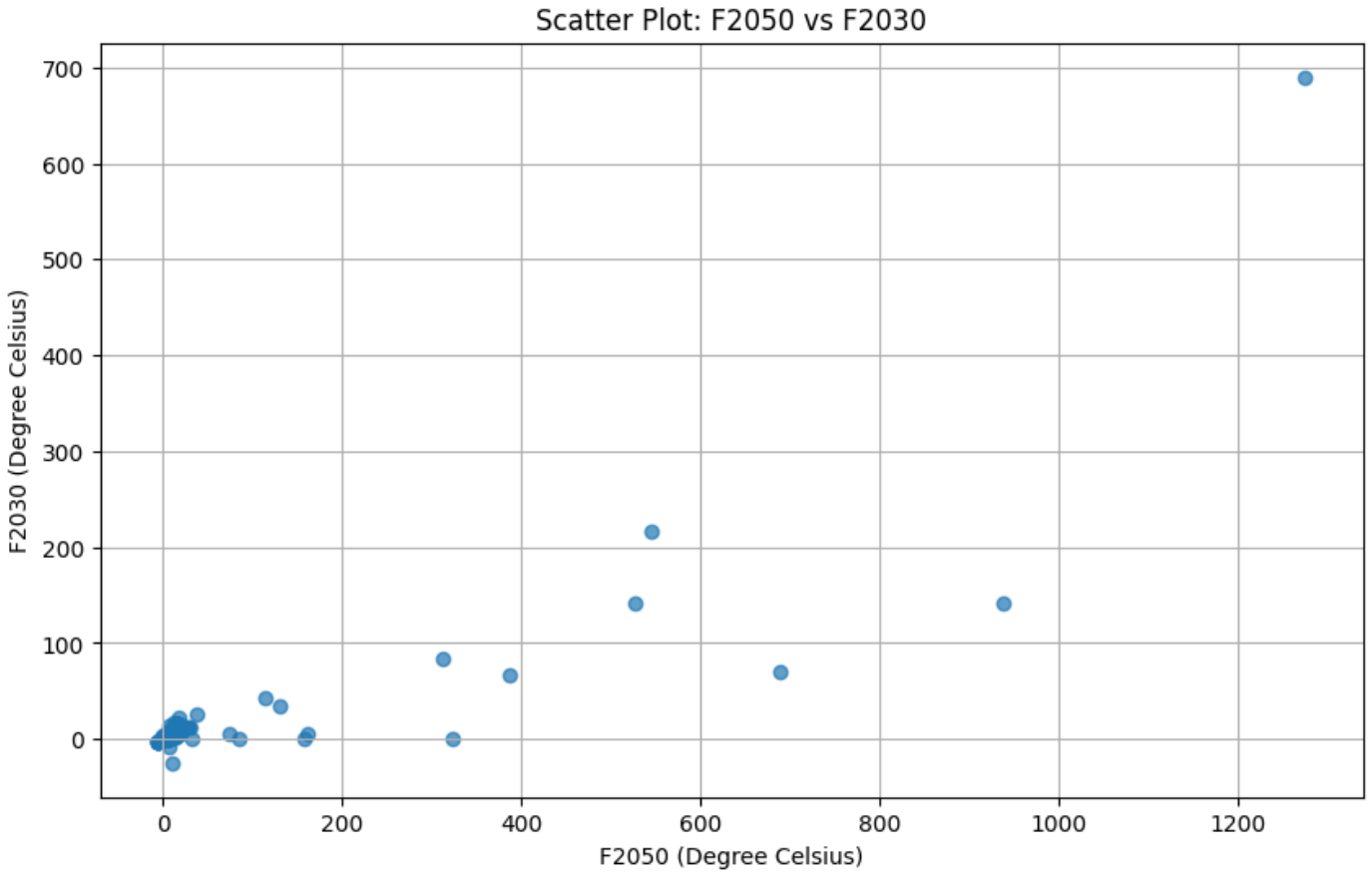
Scatter Plot: F2050 vs F2030

For a comprehensive exploration of the relationship between surface temperature projections in the years F2050 and F2030, we present a scatter plot. The x-axis represents the temperature projection for the year F2050, while the y-axis corresponds to the projection for F2030. Each data point on the plot signifies a specific scenario, providing insights into the potential correlation between these two time points.

The scatter plot is a valuable tool for identifying patterns, trends, and potential outliers in the dataset. The alpha parameter is set to 0.7 to enhance visibility of overlapping data points. The resulting visualization offers a clear depiction of the dispersion of temperature projections and aids in discerning any discernible relationships between the selected years.

In [16]:
```python
def scatter_plot(dataframe):
    """
    Generate a scatter plot and save it to a specified path.

    Parameters:
    - dataframe: DataFrame containing the data
    """

    x_values = dataframe["F2050"]
    y_values = dataframe["F2030"]

    plt.figure(figsize=(10, 6))
    plt.scatter(x_values, y_values, marker="o", alpha=0.7)
    plt.title("Scatter Plot: F2050 vs F2030")
    plt.xlabel("F2050 (Degree Celsius)")
    plt.ylabel("F2030 (Degree Celsius)")
```

```
    plt.grid(True)
    plt.show()


scatter_plot(df)
```



CONCLUSION

In summary, the "Transition to a Low-Carbon Economy - NGFS Transition Pathways" dataset, developed in collaboration with the International Monetary Fund (IMF) and curated by the Network for Greening the Financial System (NGFS), offers a comprehensive insight into climate scenarios. Covering the years 2020 to 2050, the dataset comprises 184 key indicators meticulously chosen to illuminate the pathways toward a low-carbon economy. These indicators provide a nuanced understanding of the intricate interplay between environmental factors and financial considerations, making the dataset a valuable resource for researchers, policymakers, and analysts.

The collaborative effort between NGFS and IMF enhances the dataset's credibility, ensuring a robust representation of data essential for shaping strategies and policies aligned with sustainable finance. As the world addresses the challenges of climate change, this dataset stands as a cornerstone for informed decision-making, offering detailed insights into the potential implications of climate-related scenarios on environmental sustainability and financial systems.