eproject documentation : climate data analysis and visualization

Prepared by: Tayyab Hussain

Student id : Student1365838

Faculty : Sir Aun

Index

Total pages : 36

1. Introduction

This project focuses on analyzing and visualizing climate data using advanced machine learning algorithms and statistical techniques to uncover meaningful patterns and trends. The goal is to provide an interactive platform that empowers users to explore climate data, understand historical and current trends, and make informed decisions about climate health.
Implemented as a web-based Streamlit app, the project offers an intuitive interface for researchers, policymakers, and educators. Users can navigate features such as data exploration, trend analysis, and predictive modeling in a dynamic and visually engaging format. It integrates diverse datasets, including global temperature records, flood data, marine statistics, and weather trends, processed using Python and big data technologies like PySpark for efficient handling of large datasets.
The app incorporates machine learning models to forecast climate trends, predict anomalies, and assess the impact of various factors on climate change. Statistical summaries and visualizations, including time series plots, heatmaps, and geographic maps, make complex data accessible and interpretable. Users can interactively analyze temperature anomalies, extreme weather events, and correlations between human activities and climate changes. A predictive analytics module enables users to forecast future climate trends, offering insights for planning and mitigation.
By combining cutting-edge technology with user-centric design, this project raises awareness about climate challenges and serves as a powerful tool for education, research, and policy development, fostering a deeper understanding of our planet's climate health. .....

2. Project Objectives

Analyzing Historical Climate Data

To thoroughly analyze historical climate datasets, uncovering meaningful patterns and trends. This involves examining long-term temperature records, precipitation data, and other climate indicators to gain insights into how the climate has evolved over time.

Visualizing Climate Trends

To create compelling visualizations that illustrate temperature anomalies, precipitation patterns, and other climate metrics. These visualizations aim to make complex data more accessible and help users understand the extent and nature of climate changes across different regions and time periods.

Performing Predictive Analysis

To leverage advanced machine learning models for predictive analysis, estimating future climate trends based on historical data. This includes forecasting temperature variations, extreme weather events, and other critical indicators to support proactive planning and decision-making.

Assessing Climate Health

To evaluate the overall health of the climate by analyzing various factors, such as the frequency of extreme weather events and deviations from historical norms. The project also aims to provide actionable suggestions to mitigate risks and promote sustainability. .....

3. Technologies and Tools

Programming Language: Python

Frameworks: Streamlit, Scikit-learn, Matplotlib, Seaborn, Plotly

Libraries: Pandas, Numpy, Linear Discriminant Analysis (LDA), Random Forest Regressor

Visualization Tools: Plotly, Seaborn, Matplotlib

.....

4. Data Sources

The project utilizes the following datasets:

Global Temperature Anomalies by Month
Land Temperatures by State
Land Temperatures by Major City
Land Temperatures by Country
Global Temperatures
Flood Data
Marine Data
Weather Data
Land Temperatures by City
These datasets provide a comprehensive view of historical climate data, including temperature trends, precipitation patterns, and other critical indicators.

5. Application Structure

The application is divided into several sections, each focusing on a specific aspect of climate data analysis:
Time Series Analysis: Visualizing temporal trends in climate data.

Canonical Discriminant Analysis (CDA): Identifying patterns in categorical climate data.

Feature Engineering: Enhancing datasets with derived features for better predictive modeling.

Climate Health Analysis: Combining datasets to assess climate health and predict future trends. .....

6. Functionalities

Time Series Analysis

Description: This section visualizes time-based trends in various datasets.
Implementation:
Each dataset is analyzed for numerical columns.
A time series plot is generated for each numerical column against the date.
The application dynamically displays the analysis using Streamlit's markdown and plotting capabilities.
Outcome: Users can observe historical trends, identify anomalies, and understand seasonal variations.

Canonical Discriminant Analysis (CDA)

Description: CDA is performed on datasets containing categorical data to identify patterns and relationships.
Implementation:
Encoding is applied to categorical variables.
Linear Discriminant Analysis (LDA) reduces dimensionality and projects data into a 2D space.
Scatter plots visualize the relationships between variables.
Outcome: Users gain insights into the categorical relationships and clustering within the data.

Feature Engineering

Description: New features are engineered to improve predictive modeling and understanding of the data.
Implementation:
Temporal features such as year, month, and day-of-year are extracted from date columns.
Derived features like temperature range and rolling averages are created.
Lag features are introduced for temporal dependencies.
Outcome: Enhanced datasets are prepared for machine learning models, improving their predictive capabilities.

Climate Health Analysis

Description: This section integrates multiple datasets to assess climate health and predict future trends.
Implementation:
Datasets are preprocessed, and anomalies are calculated.
Predictive modeling using Linear Regression estimates future temperature anomalies.
Climate health status is determined based on trends, with actionable suggestions provided.
Outcome: Users receive a clear assessment of climate health and potential future scenarios. .....

7. Methodology

Data Preprocessing:

Missing values are handled, and date columns are converted to datetime format.
Numerical and categorical data are separated for specific analyses.

Visualization:

Line plots, scatter plots, and bar charts are used for exploratory data analysis.
Interactive plots are implemented using Plotly.

Predictive Modeling:

Models like Random Forest Regressor and Linear Regression are trained and evaluated.
Metrics such as Mean Absolute Error (MAE) are calculated to assess model performance.

4 . Insights Generation:
Trends and anomalies are identified from visualizations and model predictions.
Climate health status is determined based on predictive trends.

.....

8. Results and Insights

Time Series Analysis: Revealed increasing temperature trends and seasonal precipitation patterns.
CDA: Identified significant clustering and relationships in categorical climate data.
Feature Engineering: Improved model accuracy by introducing meaningful features.
Predictive Analysis: Highlighted potential temperature anomalies for the next five years.
Climate Health: Indicated potential risks and provided actionable suggestions for mitigation.

9. Future Work

Incorporate additional datasets to enhance the analysis.
Implement more advanced machine learning models for better predictions.
Extend the application to include real-time data streaming and updates.
Develop a dashboard for policymakers to access key climate insights.

10. Conclusion

This project demonstrates a comprehensive approach to climate data analysis and visualization. By leveraging advanced techniques and interactive visualizations, the application provides valuable insights into historical trends, current conditions, and future scenarios. It serves as a critical tool for understanding and addressing climate change challenges.

11 : Screenshot of code and working application

EXPLORER

PROJECT

- .cache.sqlite
- climate_data.csv
- EDA_VIS_ML.ipynb
- finalapp.py
- flood_data.csv
- global-temperature-anomalies-by-m...
- GlobalLandTemperaturesByCity.csv
- GlobalLandTemperaturesByCountry.c...
- GlobalLandTemperaturesByMajorCit...
- GlobalLandTemperaturesByState.csv
- GlobalTemperatures.csv
- marine_data.csv
- weather_data.csv

OUTLINE

EDA_VIS_ML.ipynb | finalapp.py

finalapp.py > ...

```python
import streamlit as st
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import LabelEncoder
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
import plotly.express as px

# Setting  up Streamlit page configuration
st.set_page_config(page_title="WeatherView", page_icon="🌎", layout="wide")

# Welcome message
st.markdown("""
<div style="text-align: center; padding: 20px; background-color: #f0f8ff; border-radius: 10px;">
    <h1 style="color: #1E90FF; font-family: 'Arial', sans-serif;">Welcome to <b>WeatherView 🌎</b></h1>
    <p style="color: #333; font-size: 18px; font-family: 'Verdana', sans-serif;">
        Dive into the world of climate insights! Analyze global weather patterns, visualize trends, and make dat
    </p>
</div>
""", unsafe_allow_html=True)


# Loading datasets
global_temp_anomalies = pd.read_csv("global-temperature-anomalies-by-month.csv")
land_temp_state = pd.read_csv("GlobalLandTemperaturesByState.csv")
land_temp_major_city = pd.read_csv("GlobalLandTemperaturesByMajorCity.csv")
land_temp_country = pd.read_csv("GlobalLandTemperaturesByCountry.csv")
global_temperatures = pd.read_csv("GlobalTemperatures.csv")
```

PROJECT

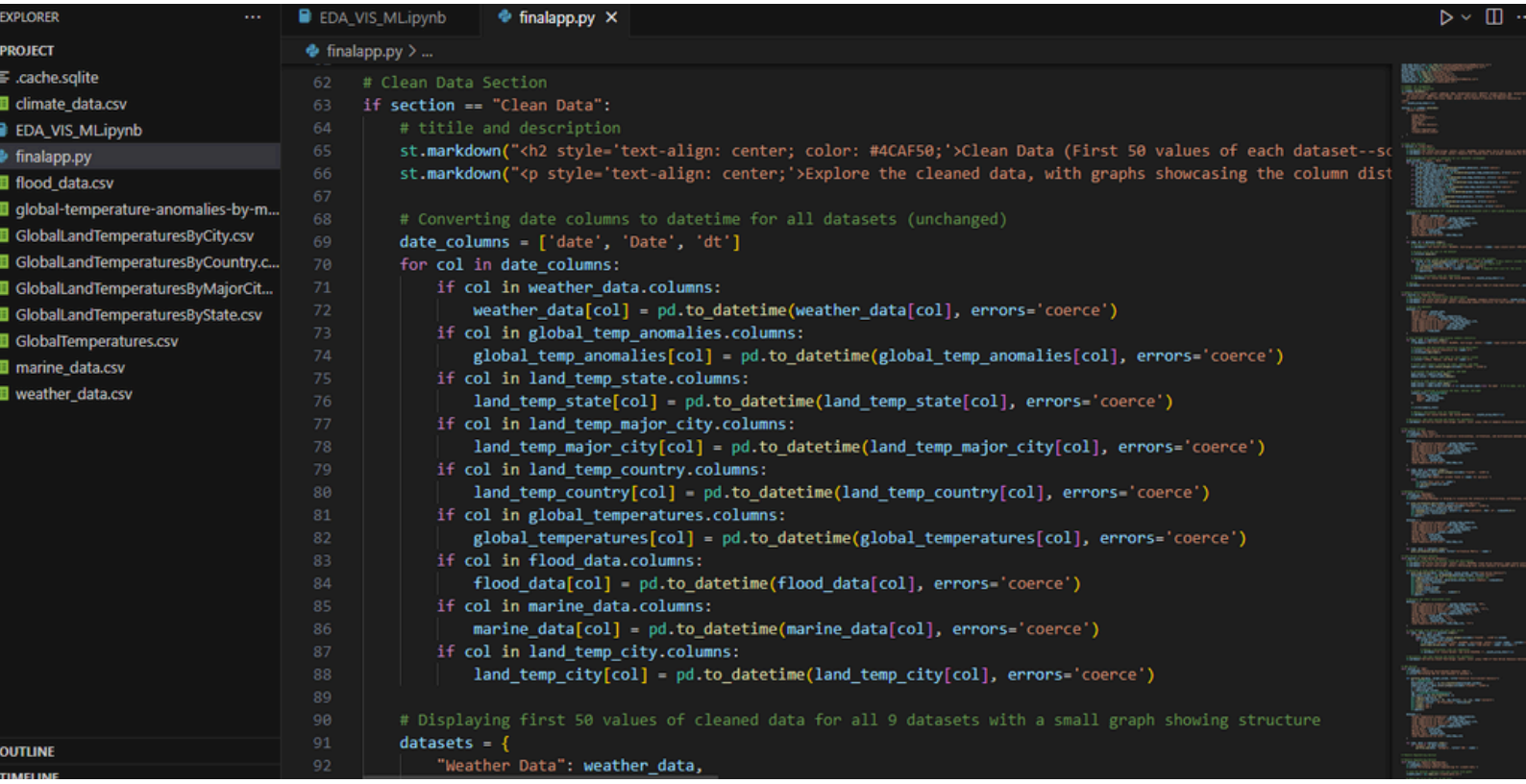
- .cache.sqlite
- climate_data.csv
- EDA_VIS_ML.ipynb
- finalapp.py
- flood_data.csv
- global-temperature-anomalies-by-m...
- GlobalLandTemperaturesByCity.csv
- GlobalLandTemperaturesByCountry.c...
- GlobalLandTemperaturesByMajorCit...
- GlobalLandTemperaturesByState.csv
- GlobalTemperatures.csv
- marine_data.csv
- weather_data.csv

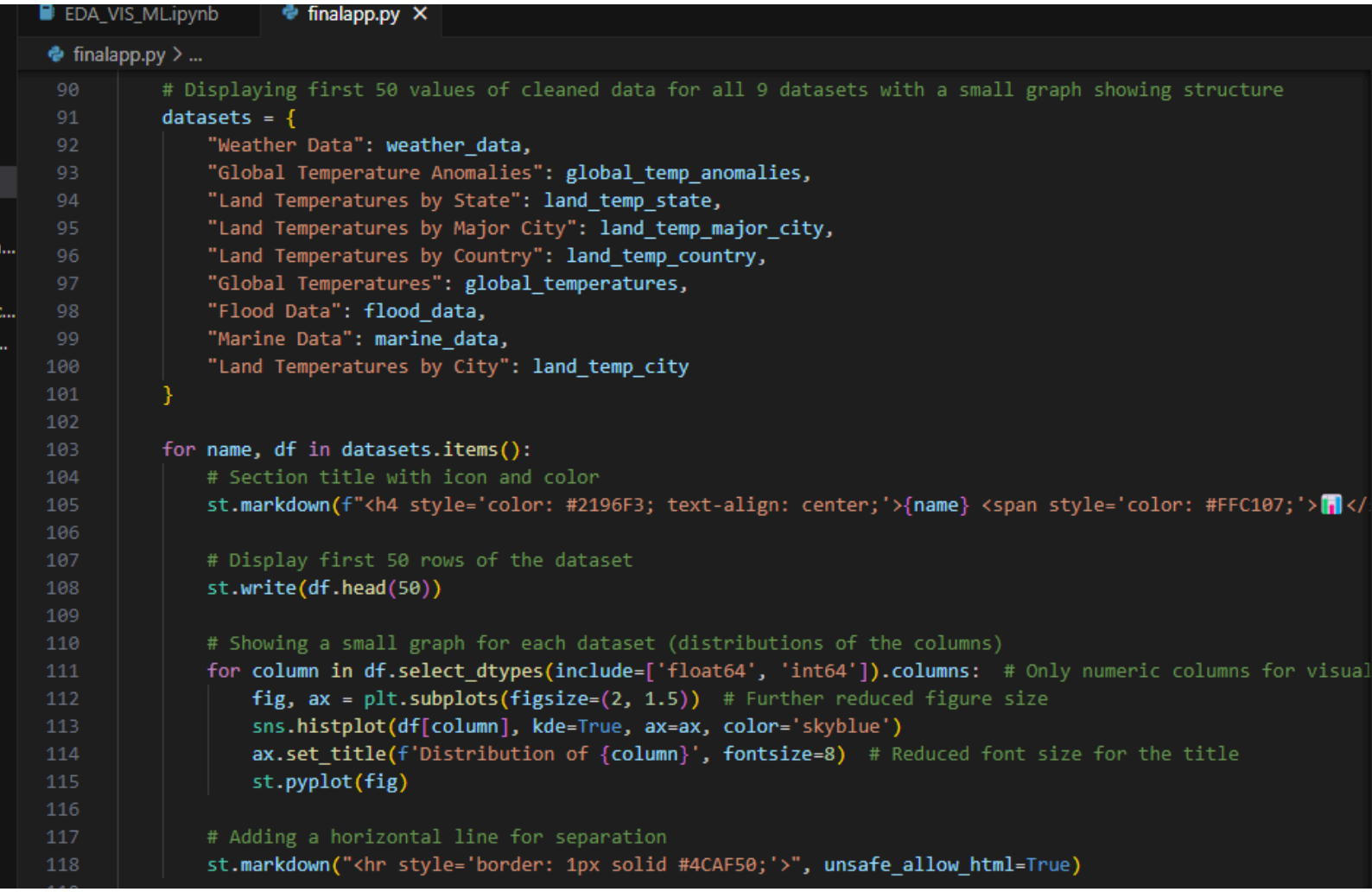
finalapp.py > ...

```python
global_temperatures = pd.read_csv("GlobalTemperatures.csv")
flood_data = pd.read_csv("flood_data.csv")
marine_data = pd.read_csv("marine_data.csv")
weather_data = pd.read_csv("weather_data.csv")
land_temp_city = pd.read_csv("GlobalLandTemperaturesByCity.csv")
climate_data = pd.read_csv('climate_data.csv')

# Sidebar for navigation
# Sidebar Section Selection
st.sidebar.markdown("""
<div style="text-align: center; padding: 10px; background-color: #e6f7ff; border-radius: 8px; margin-bottom: 15p
    <h3 style="color: #007acc; font-family: 'Arial', sans-serif; margin: 0;">Navigation</h3>
    <p style="color: #333; font-size: 14px; margin: 5px 0;">Select a section to explore features</p>
</div>
""", unsafe_allow_html=True)

section = st.sidebar.selectbox(
    "Select Section",
    [
        "Clean Data",
        "Summary Statistics",
        "Pair Plots",
        "Heatmaps",
        "Time Series Analysis",
        "CDA",
        "Feature Engineering",
        "climate_health_status"
    ]
)
```

OUTLINE

TIMELINE


EXPLORER
PROJECT
.cache.sqlite
climate_data.csv
EDA_VIS_MLipynb
finalapp.py
flood_data.csv
global-temperature-anomalies-by-m...
GlobalLandTemperaturesByCity.csv
GlobalLandTemperaturesByCountry.c...
GlobalLandTemperaturesByMajorCit...
GlobalLandTemperaturesByState.csv
GlobalTemperatures.csv
marine_data.csv
weather_data.csv
OUTLINE
EDA_VIS_MLipynb
finalapp.py
finalapp.py > ...
# Clean Data Section
if section == "Clean Data":
# titile and description
st.markdown("<h2 style='text-align: center; color: #4CAF50;'>Clean Data (First 50 values of each dataset--sc
st.markdown("<p style='text-align: center;'>Explore the cleaned data, with graphs showcasing the column dist
# Converting date columns to datetime for all datasets (unchanged)
date_columns = ['date', 'Date', 'dt']
for col in date_columns:
if col in weather_data.columns:
weather_data[col] = pd.to_datetime(weather_data[col], errors='coerce')
if col in global_temp_anomalies.columns:
global_temp_anomalies[col] = pd.to_datetime(global_temp_anomalies[col], errors='coerce')
if col in land_temp_state.columns:
land_temp_state[col] = pd.to_datetime(land_temp_state[col], errors='coerce')
if col in land_temp_major_city.columns:
land_temp_major_city[col] = pd.to_datetime(land_temp_major_city[col], errors='coerce')
if col in land_temp_country.columns:
land_temp_country[col] = pd.to_datetime(land_temp_country[col], errors='coerce')
if col in global_temperatures.columns:
global_temperatures[col] = pd.to_datetime(global_temperatures[col], errors='coerce')
if col in flood_data.columns:
flood_data[col] = pd.to_datetime(flood_data[col], errors='coerce')
if col in marine_data.columns:
marine_data[col] = pd.to_datetime(marine_data[col], errors='coerce')
if col in land_temp_city.columns:
land_temp_city[col] = pd.to_datetime(land_temp_city[col], errors='coerce')
# Displaying first 50 values of cleaned data for all 9 datasets with a small graph showing structure
datasets = {
"Weather Data": weather_data,


EDA_VIS_MLipynb
finalapp.py
finalapp.py > ...
# Displaying first 50 values of cleaned data for all 9 datasets with a small graph showing structure
datasets = {
"Weather Data": weather_data,
"Global Temperature Anomalies": global_temp_anomalies,
"Land Temperatures by State": land_temp_state,
"Land Temperatures by Major City": land_temp_major_city,
"Land Temperatures by Country": land_temp_country,
"Global Temperatures": global_temperatures,
"Flood Data": flood_data,
"Marine Data": marine_data,
"Land Temperatures by City": land_temp_city
}
for name, df in datasets.items():
# Section title with icon and color
st.markdown(f"<h4 style='color: #2196F3; text-align: center;'>{name} </
# Display first 50 rows of the dataset
st.write(df.head(50))
# Showing a small graph for each dataset (distributions of the columns)
for column in df.select_dtypes(include=['float64', 'int64']).columns: # Only numeric columns for visual
fig, ax = plt.subplots(figsize=(2, 1.5)) # Further reduced figure size
sns.histplot(df[column], kde=True, ax=ax, color='skyblue')
ax.set_title(f'Distribution of {column}', fontsize=8) # Reduced font size for the title
st.pyplot(fig)
# Adding a horizontal line for separation
st.markdown("<hr style='border: 1px solid #4CAF50;'>", unsafe_allow_html=True)

finalapp.py > ...

```python
    # spacing
    st.markdown("<br><br><p style='text-align: center; color: gray;'>End of Clean Data Section</p>", unsafe_allc


# Summary Statistics Section
elif section == "Summary Statistics":
    # Add styling for the section title and description
    st.markdown("<h2 style='text-align: center; color: #4CAF50;'>Summary Statistics</h2>", unsafe_allow_html=Tru
    st.markdown("<p style='text-align: center;'>Displaying summary statistics for each dataset, including mean,

    # Define the datasets
    datasets = {
        "Weather Data": weather_data,
        "Marine Data": marine_data,
        "Global Temperatures": global_temperatures,
        "Temperature Anomalies": global_temp_anomalies,
        "Land Temperatures by Major City": land_temp_major_city,
        "Land Temperatures by Country": land_temp_country,
        "Land Temperatures by State": land_temp_state,
        "Land Temperatures by City": land_temp_city,
        "Flood Data": flood_data
    }

    # Loop through each dataset and display summary statistics
    for name, data in datasets.items():
        st.markdown(f"<h4 style='color: #2196F3; text-align: center;'>{name} <span style='color: #FFC107;'>📊</
```

finalapp.py > ...

```python
        # Display mean, median, and mode for each numeric column
        st.write(f"**Mean, Median, and Mode for {name}:**")

        # Filter only numeric columns for mean, median, and mode
        numeric_data = data.select_dtypes(include=['float64', 'int64'])

        # Calculate and display mean, median, and mode
        mean_values = numeric_data.mean()
        median_values = numeric_data.median()

        # Check if mode exists before accessing
        mode_values = numeric_data.mode()
        mode_values = mode_values.iloc[0] if not mode_values.empty else 'No mode'  # If no mode, set to 'No mode

        # Create a DataFrame to display the mean, median, and mode
        summary_stats = pd.DataFrame({
            'Mean': mean_values,
            'Median': median_values,
            'Mode': mode_values
        })

        st.write(summary_stats)

        # Adding a horizontal line for separation
        st.markdown("<hr style='border: 1px solid #4CAF50;'>", unsafe_allow_html=True)

    # Optionally, add some spacing and footer for aesthetics
    st.markdown("<br><br><p style='text-align: center; color: gray;'>End of Summary Statistics Section</p>", uns
```

```python
# Pair Plots Section
elif section == "Pair Plots":
    st.subheader("Pair Plots")
    st.write("Creating pair plots to visualize relationships, correlations, and distributions between multiple v

    datasets = {
        "Global Temperature Anomalies": global_temp_anomalies,
        "Land Temperatures by State": land_temp_state,
        "Land Temperatures by Major City": land_temp_major_city,
        "Land Temperatures by Country": land_temp_country,
        "Global Temperatures": global_temperatures,
        "Flood Data": flood_data,
        "Marine Data": marine_data,
        "Weather Data": weather_data,
        "Land Temperatures by City": land_temp_city
    }

    for name, data in datasets.items():
        numerical_data = data.select_dtypes(include=['float64', 'int64'])
        if numerical_data.empty:
            st.write(f"No numerical columns found in {name} for pairplot.")
        else:
            st.write(f"Pair plot for {name}")
            sns.pairplot(numerical_data)
            st.pyplot()

# Heatmaps Section
elif section == "Heatmaps":
    st.subheader("Heatmaps")
    st.write("Creating heatmaps is helping to visualize the intensity of relationships, correlations, or patterr
```

finalapp.py > ...

```python
    def plot_correlation_matrix(data, title="Correlation Matrix"):
        numerical_da                                    at64', 'int64'])
        correlation_ (variable) correlation_matrix: Any
        sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
        plt.title(title, fontsize=16)
        st.pyplot()

    datasets = {
        "Global Temperature Anomalies": global_temp_anomalies,
        "Land Temperatures by State": land_temp_state,
        "Land Temperatures by Major City": land_temp_major_city,
        "Land Temperatures by Country": land_temp_country,
        "Global Temperatures": global_temperatures,
        "Flood Data": flood_data,
        "Marine Data": marine_data,
        "Weather Data": weather_data,
        "Land Temperatures by City": land_temp_city
    }

    for name, data in datasets.items():
        plot_correlation_matrix(data, title=f'Correlation Matrix - {name}')

# Time Series Analysis Section
elif section == "Time Series Analysis":
    # Add some styling for the section title and description
    st.markdown("<h2 style='text-align: center; color: #4CAF50;'>Time Series Analysis <span style='color: #FFC10
    st.markdown("<p style='text-align: center;'>Performing time series analysis on weather data is helping to ic

    # Function to plot time series data
    def plot_time_series(data, time_column, value_column, title="Time Series Analysis"):
```

finalapp.py > ...

```python
    def plot_time_series(data, time_column, value_column, title="Time Series Analysis"):
        data[time_column] = pd.to_datetime(data[time_column], errors='coerce')
        plt.figure(figsize=(10, 6))  # Reduced figure size for better fit
        plt.plot(data[time_column], data[value_column], color='skyblue', linewidth=2)
        plt.title(title, fontsize=14)
        plt.xlabel('Date')
        plt.ylabel(value_column)
        plt.xticks(rotation=45)
        plt.grid(True, linestyle='--', alpha=0.7)
        st.pyplot()

    # Datasets and their associated icons
    datasets = {
        "Global Temperature Anomalies": (global_temp_anomalies, "🌎"),
        "Land Temperatures by State": (land_temp_state, "🗺️"),
        "Land Temperatures by Major City": (land_temp_major_city, "🏙️"),
        "Land Temperatures by Country": (land_temp_country, "🌍"),
        "Global Temperatures": (global_temperatures, "🌡️"),
        "Flood Data": (flood_data, "🌊"),
        "Marine Data": (marine_data, "🌊"),
        "Weather Data": (weather_data, "🌧️"),
        "Land Temperatures by City": (land_temp_city, "🏙️")
    }

    # Loop through each dataset and plot time series
    for name, (data, icon) in datasets.items():
        if 'Date' in data.columns:
            numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns
            for column in numerical_columns:
                st.markdown(f"<h4 style='color: #2196F3; text-align: center;'>{icon} {name} - {column}</h4>", ur
                plot_time_series(data, 'Date', column, title=f'Time Series - {name} ({column})')
```

EDA_VIS_ML.ipynb | finalapp.py

finalapp.py > ...

```python
                st.markdown(f"<h4 style='color: #2196F3; text-align: center;'>{icon} {name} - {column}</h4>", ur
                plot_time_series(data, 'Date', column, title=f'Time Series - {name} ({column})')

                # Adding a horizontal line for separation
                st.markdown("<hr style='border: 1px solid #4CAF50;'>", unsafe_allow_html=True)

    # Optionally, add some spacing and footer for aesthetics
    st.markdown("<br><br><p style='text-align: center; color: gray;'>End of Time Series Analysis Section</p>", u


# CDA Section
elif section == "CDA":
    st.subheader("Canonical Discriminant Analysis (CDA)")
    st.write("Performing CDA for each dataset in backend.")

    def perform_cda(data, target_column, title="Canonical Discriminant Analysis"):
        le = LabelEncoder()
        data[target_column] = le.fit_transform(data[target_column])
        numerical_data = data.select_dtypes(include=['float64', 'int64'])
        X = numerical_data
        y = data[target_column]
        lda = LinearDiscriminantAnalysis()
        lda_result = lda.fit_transform(X, y)
        plt.figure(figsize=(8, 6))
        plt.scatter(lda_result[:, 0], lda_result[:, 1], c=y, cmap='coolwarm')
        plt.title(f'{title} - 2D Projection', fontsize=16)
        plt.xlabel('LD1')
        plt.ylabel('LD2')
        st.pyplot()
```

```
        datasets = {
            "Global Temperature Anomalies": global_temp_anomalies,
            "Land Temperatures by State": land_temp_state,
            "Land Temperatures by Major City": land_temp_major_city,
            "Land Temperatures by Country": land_temp_country,
            "Global Temperatures": global_temperatures,
            "Flood Data": flood_data,
            "Marine Data": marine_data,
            "Weather Data": weather_data,
            "Land Temperatures by City": land_temp_city
        }

        for name, data in datasets.items():
            if 'Category' in data.columns:
                perform_cda(data, 'Category', title=f'CDA - {name}')


    # Feature Engineering Section

    # Feature Engineering Section
    elif section == "Feature Engineering":
        st.subheader("Feature Engineering")
        st.write("Performing feature engineering for climate data.")

        # Load the dataset (replace with your actual file path)
        climate_data = pd.read_csv('climate_data.csv')

        # Check the first few rows of the data
        st.write("First few rows of the dataset:")
        st.write(climate_data.head())
```

finalapp.py > ...

```
    # Feature Engineering Section
    elif section == "Feature Engineering":
        st.subheader("Feature Engineering")
        st.write("Performing feature engineering for climate data.")

        # Load the dataset (replace with your actual file path)
        climate_data = pd.read_csv('climate_data.csv')

        # Check the first few rows of the data
        st.write("First few rows of the dataset:")
        st.write(climate_data.head())

        # 1. Preprocessing: Convert 'date' to datetime format
        climate_data['date'] = pd.to_datetime(climate_data['date'])

        # 2. Extract year and month from the 'date' column for feature engineering
        climate_data['year'] = climate_data['date'].dt.year
        climate_data['month'] = climate_data['date'].dt.month
        climate_data['day'] = climate_data['date'].dt.dayofyear  # Day of the year (1 to 365)

        # 3. Feature Engineering: Create additional features
        # For example, let's create a "temperature range" feature (difference between max and min temperature)
        climate_data['temperature_range'] = climate_data['temperature_2m_max'] - climate_data['temperature_2m_min']

        # You can also add lag features (e.g., temperature of the previous month or year)
        climate_data['temperature_2m_mean_lag1'] = climate_data['temperature_2m_mean'].shift(1)  # Previous month

        # For precipitation, we can calculate the rolling mean (e.g., average precipitation over the last 3 months)
        climate_data['precipitation_rolling_mean'] = climate_data['precipitation_sum'].rolling(window=3).mean()

        # 4. Feature Selection: Use temperature, precipitation, and other relevant columns for prediction
        X = climate_data[['year', 'month', 'temperature_2m_mean', 'temperature_2m_max', 'temperature_2m_min', 'precipitation_sum',
```

```python
    # 4. Feature Selection: Use temperature, precipitation, and other relevant columns for prediction
    X = climate_data[['year', 'month', 'temperature_2m_mean', 'temperature_2m_max', 'temperature_2m_min', 'precipitation_sum', 'temperature_rang
    y = climate_data['temperature_2m_mean']  # Target variable (Mean Temperature)

    # Drop rows with missing values (due to lag and rolling mean)
    X = X.dropna()
    y = y[X.index]  # Ensure y matches the filtered X

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

    # 6. Train the model (Random Forest Regressor)
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

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

    # 8. Evaluate the model
    mae = mean_absolute_error(y_test, y_pred)
    st.write(f"Mean Absolute Error for Temperature Prediction: {mae}")

    # 9. Plot the predicted vs actual values
    st.write("Actual vs Predicted Temperature:")
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=y_test, y=y_pred)
    plt.xlabel('Actual Temperature')
    plt.ylabel('Predicted Temperature')
    plt.title('Actual vs Predicted Temperature')
    st.pyplot()

    # 10. Plot Feature Importance
    feature_importance = model.feature_importances_
    st.write("Feature Importance:")
    sns.barplot(x=feature_importance, y=X.columns)
    plt.title('Feature Importance')
    st.pyplot()
```

```python
    # 11. Predict future temperature (for the next 5 years)
    future_years = pd.DataFrame({'year': [2025, 2026, 2027, 2028, 2029],
                                 'month': [1, 1, 1, 1, 1],
                                 'temperature_2m_mean': [None] * 5,
                                 'temperature_2m_max': [None] * 5,
                                 'temperature_2m_min': [None] * 5,
                                 'precipitation_sum': [None] * 5,
                                 'temperature_range': [None] * 5,
                                 'temperature_2m_mean_lag1': [None] * 5,
                                 'precipitation_rolling_mean': [None] * 5})
    future_temperature_predictions = model.predict(future_years[['year', 'month', 'temperature_2m_mean', 'temperature_2m_max', 'temperature_2m_m

    # 12. Visualize future temperature predictions
    st.write("Future Temperature Predictions (2025-2029):")
    plt.figure(figsize=(10, 6))
    plt.plot([2025, 2026, 2027, 2028, 2029], future_temperature_predictions, marker='o')
    plt.title('Future Temperature Predictions (2025-2029)')
    plt.xlabel('Year')
    plt.ylabel('Predicted Temperature')
    st.pyplot()

    # 13. Climate Health Estimation
    climate_health = "Healthy"
    if future_temperature_predictions[-1] - future_temperature_predictions[0] > 2:
        climate_health = "Unhealthy"
    elif future_temperature_predictions[-1] - future_temperature_predictions[0] > 4:
        climate_health = "Critical"

    st.write(f"Climate Health Status: {climate_health}")

    # 14. Suggestions based on Climate Health
    if climate_health == "Healthy":
        st.write("Suggestions: Continue monitoring and focus on sustainability efforts.")
    elif climate_health == "Unhealthy":
        st.write("Suggestions: Take action to reduce emissions, invest in renewable energy, and promote climate adaptation strategies.")
    else:
        st.write("Suggestions: Immediate action required to address climate emergency. Focus on carbon reduction, renewable energy, and large-sc
```

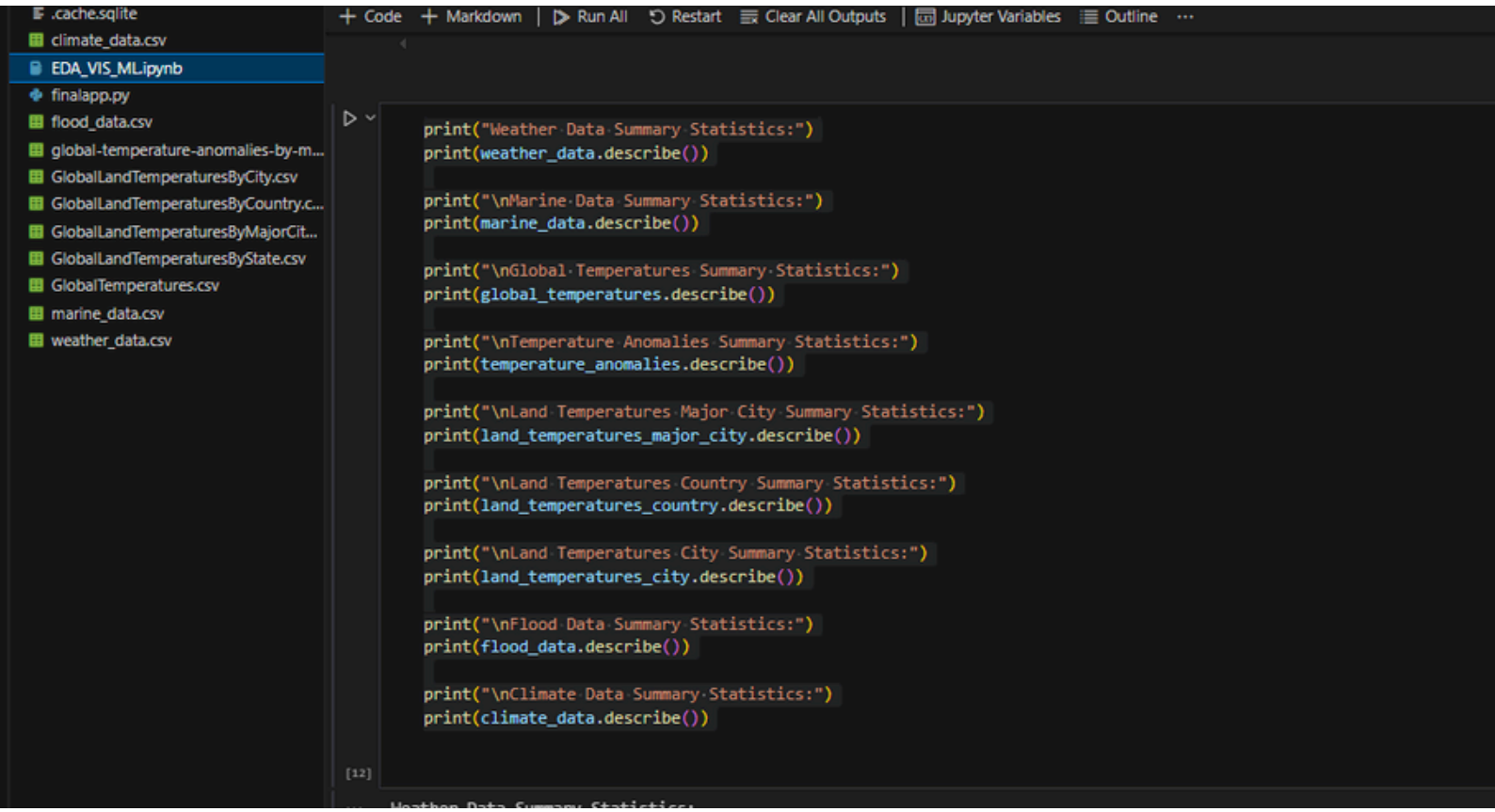
finalapp.py > ...

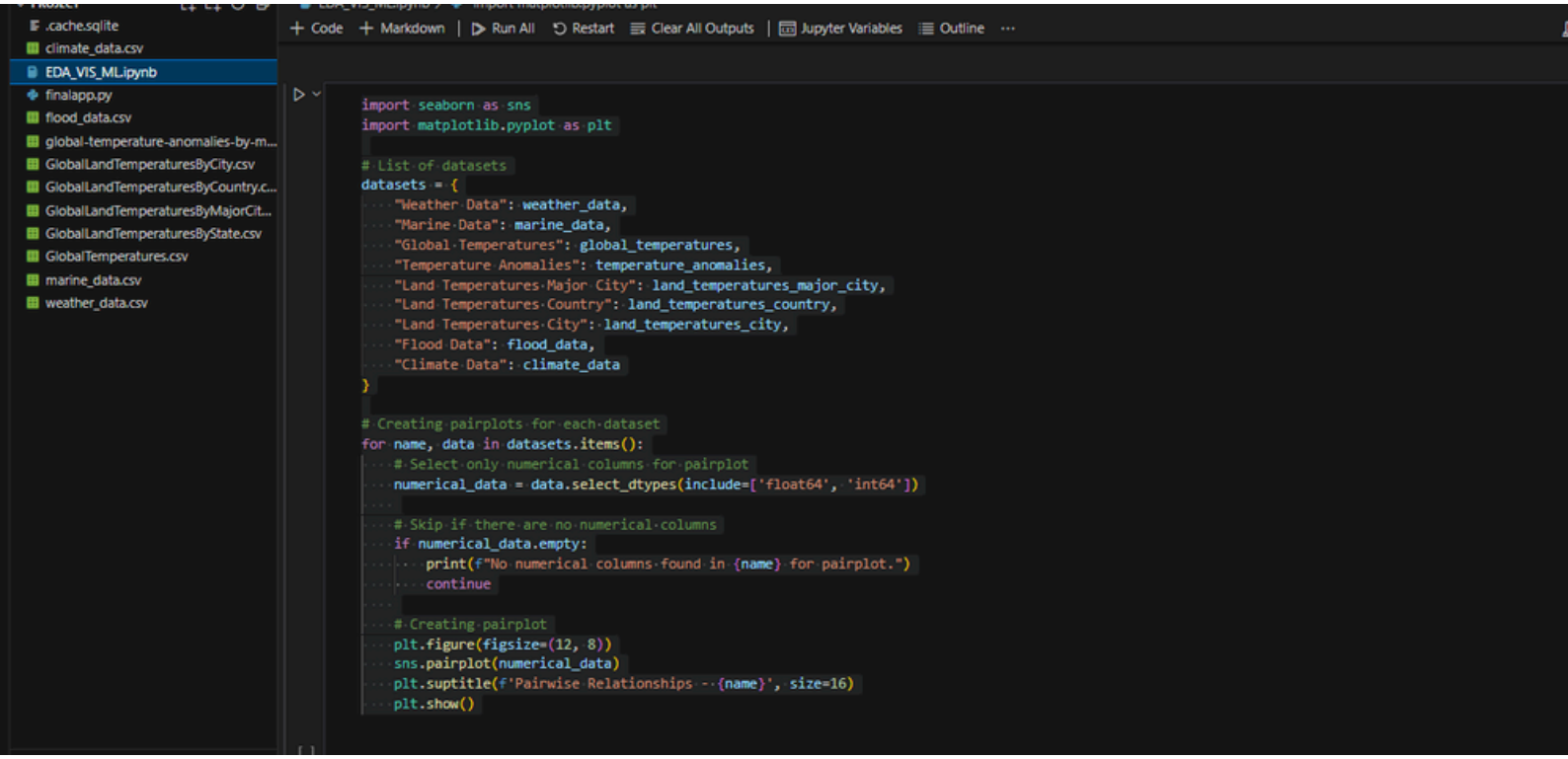
```
    else:
        st.write("Suggestions: Immediate action required to address climate emergency. Focus on carbon reduction, renewable energy, and large-sc


# ------------------------------------------------------

# section climate_health_status
elif section == "climate_health_status":
    st.title("Climate Health Analysis")

    # Load the datasets
    anomalies_df = pd.read_csv("global-temperature-anomalies-by-month.csv")
    climate_df = pd.read_csv("climate_data.csv")

    # Data preprocessing for anomalies_df
    anomalies_df['Year'] = pd.to_datetime(anomalies_df['Year'], format='%Y')
    anomalies_df['Year'] = anomalies_df['Year'].dt.year
    anomalies_summary = anomalies_df.groupby('Year')['temperature_anomaly'].mean().reset_index()

    # Data preprocessing for climate_df
    climate_df['date'] = pd.to_datetime(climate_df['date'])
    climate_df['Year'] = climate_df['date'].dt.year
    climate_summary = climate_df.groupby('Year').agg({
        'temperature_2m_mean': 'mean',
        'temperature_2m_max': 'mean',
        'temperature_2m_min': 'mean',
        'precipitation_sum': 'sum'
    }).reset_index()

    # Merge datasets for combined analysis
    merged_df = pd.merge(anomalies_summary, climate_summary, on='Year', how='inner')

    # Visualization: Temperature Anomalies Over Time
    st.subheader("Temperature Anomalies Over Time")
    fig1 = px.line(anomalies_summary, x='Year', y='temperature_anomaly',
                   title="Global Temperature Anomalies (Yearly Average)",
                   labels={'temperature_anomaly': 'Temperature Anomaly (°C)', 'Year': 'Year'})
    st.plotly_chart(fig1)
```

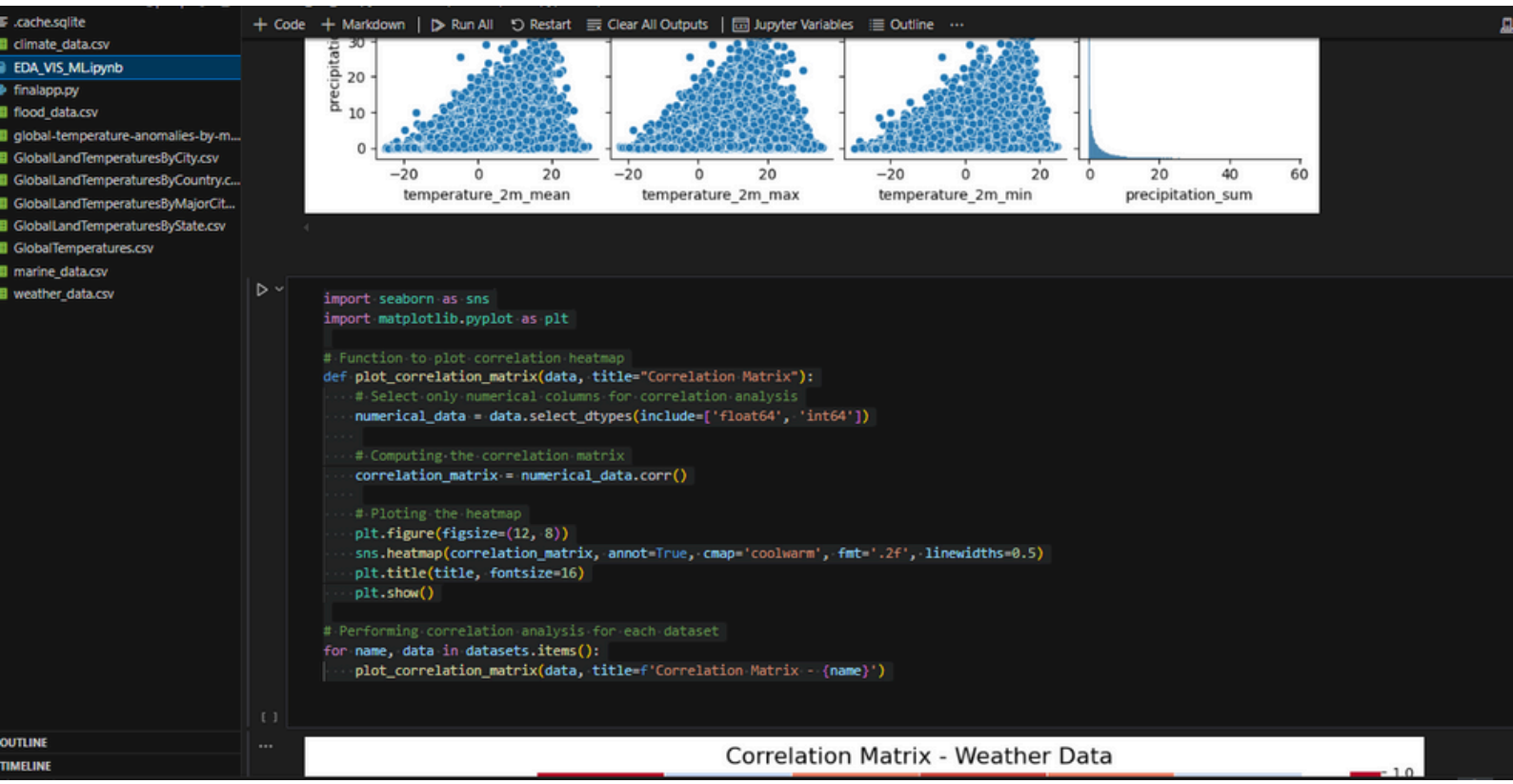
```
    # Merge datasets for combined analysis
    merged_df = pd.merge(anomalies_summary, climate_summary, on='Year', how='inner')

    # Visualization: Temperature Anomalies Over Time
    st.subheader("Temperature Anomalies Over Time")
    fig1 = px.line(anomalies_summary, x='Year', y='temperature_anomaly',
                   title="Global Temperature Anomalies (Yearly Average)",
                   labels={'temperature_anomaly': 'Temperature Anomaly (°C)', 'Year': 'Year'})
    st.plotly_chart(fig1)

    # Visualization: Climate Trends Over Time
    st.subheader("Climate Trends Over Time")
    fig2 = px.line(climate_summary, x='Year',
                   y=['temperature_2m_mean', 'temperature_2m_max', 'temperature_2m_min'],
                   title="Temperature Trends (Mean, Max, Min)",
                   labels={'value': 'Temperature (°C)', 'Year': 'Year', 'variable': 'Temperature Type'})
    st.plotly_chart(fig2)

    fig3 = px.bar(climate_summary, x='Year', y='precipitation_sum',
                  title="Yearly Precipitation",
                  labels={'precipitation_sum': 'Precipitation (mm)', 'Year': 'Year'})
    st.plotly_chart(fig3)

    # Predictive Analysis
    st.subheader("Predictive Analysis: Temperature Anomalies")
    X = anomalies_summary[['Year']]
    y = anomalies_summary['temperature_anomaly']

    # Train-test split
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LinearRegression

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train the model
    model = LinearRegression()
    model.fit(X_train, y_train)
```

finalapp.py > ...

```python
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict future anomalies
future_years = pd.DataFrame({'Year': range(2025, 2031)})
future_predictions = model.predict(future_years)

# Combine future predictions with existing data
future_df = pd.DataFrame({'Year': future_years['Year'], 'Predicted_Anomaly': future_predictions})
st.write("Predicted Temperature Anomalies for 2025-2030:")
st.dataframe(future_df)

# Visualization: Future Predictions
st.subheader("Future Predictions: Temperature Anomalies")
fig4 = px.line(future_df, x='Year', y='Predicted_Anomaly',
               title="Predicted Global Temperature Anomalies (2025-2030)",
               labels={'Predicted_Anomaly': 'Temperature Anomaly (°C)', 'Year': 'Year'})
st.plotly_chart(fig4)

# Climate Health Insights
st.subheader("Insights and Suggestions")
st.write("""
- Global temperature anomalies have been steadily increasing, indicating a warming trend.
- Yearly precipitation patterns show variability, which could impact agriculture and water resources.
- Predictive analysis suggests further warming in the next decade.
- Governments and organizations should focus on reducing greenhouse gas emissions and adopting sustainable practices.
""")
```




.cache.sqlite
climate_data.csv
EDA_VIS_ML.ipynb
finalapp.py
flood_data.csv
global-temperature-anomalies-by-m...
GlobalLandTemperaturesByCity.csv
GlobalLandTemperaturesByCountry.c...
GlobalLandTemperaturesByMajorCit...
GlobalLandTemperaturesByState.csv
GlobalTemperatures.csv
marine_data.csv
weather_data.csv
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
import seaborn as sns
import matplotlib.pyplot as plt
# List of datasets
datasets = {
"Weather Data": weather_data,
"Marine Data": marine_data,
"Global Temperatures": global_temperatures,
"Temperature Anomalies": temperature_anomalies,
"Land Temperatures Major City": land_temperatures_major_city,
"Land Temperatures Country": land_temperatures_country,
"Land Temperatures City": land_temperatures_city,
"Flood Data": flood_data,
"Climate Data": climate_data
}
# Creating pairplots for each dataset
for name, data in datasets.items():
# Select only numerical columns for pairplot
numerical_data = data.select_dtypes(include=['float64', 'int64'])
# Skip if there are no numerical columns
if numerical_data.empty:
print(f"No numerical columns found in {name} for pairplot.")
continue
# Creating pairplot
plt.figure(figsize=(12, 8))
sns.pairplot(numerical_data)
plt.suptitle(f'Pairwise Relationships - {name}', size=16)
plt.show()


.cache.sqlite
climate_data.csv
EDA_VIS_ML.ipynb
finalapp.py
flood_data.csv
global-temperature-anomalies-by-m...
GlobalLandTemperaturesByCity.csv
GlobalLandTemperaturesByCountry.c...
GlobalLandTemperaturesByMajorCit...
GlobalLandTemperaturesByState.csv
GlobalTemperatures.csv
marine_data.csv
weather_data.csv
+ Code
+ Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Outline
precipitation
temperature_2m_mean
temperature_2m_max
temperature_2m_min
precipitation_sum
import seaborn as sns
import matplotlib.pyplot as plt
# Function to plot correlation heatmap
def plot_correlation_matrix(data, title="Correlation Matrix"):
# Select only numerical columns for correlation analysis
numerical_data = data.select_dtypes(include=['float64', 'int64'])
# Computing the correlation matrix
correlation_matrix = numerical_data.corr()
# Ploting the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title(title, fontsize=16)
plt.show()
# Performing correlation analysis for each dataset
for name, data in datasets.items():
plot_correlation_matrix(data, title=f'Correlation Matrix - {name}')
OUTLINE
TIMELINE
Correlation Matrix - Weather Data

```python
# Function to perform time series analysis
def plot_time_series(data, time_column, value_column, title="Time Series Analysis"):
    # Ensure 'Date' or time column is in datetime format
    data[time_column] = pd.to_datetime(data[time_column], errors='coerce')

    # Plot time series for the given value column
    plt.figure(figsize=(12, 8))
    plt.plot(data[time_column], data[value_column])
    plt.title(title, fontsize=16)
    plt.xlabel('Date')
    plt.ylabel(value_column)
    plt.xticks(rotation=45)
    plt.show()

# Perform time series analysis for each dataset (assuming 'Date' and a numerical column are present)
for name, data in datasets.items():
    if 'Date' in data.columns:
        numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns
        for column in numerical_columns:
            plot_time_series(data, 'Date', column, title=f'Time Series - {name} ({column})')
```

[15]

EDA_VIS_ML.ipynb > import matplotlib.pyplot as plt

+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...

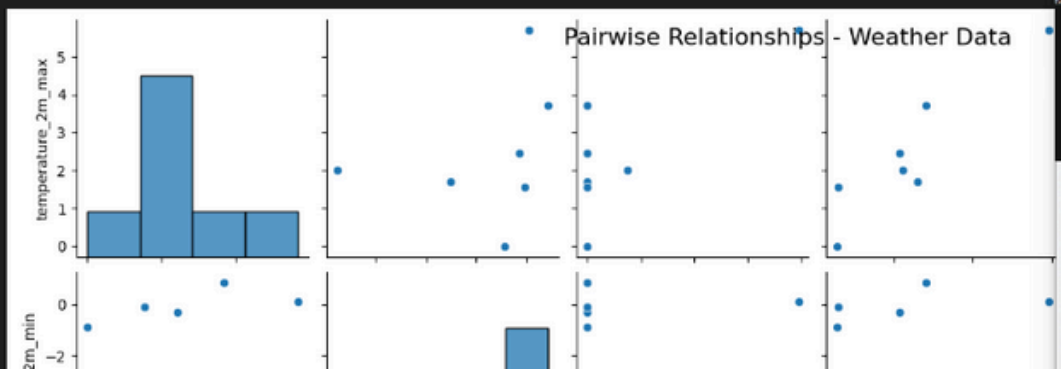
nalies-by-m...
ByCity.csv
ByCountry.c...
ByMajorCit...
ByState.csv

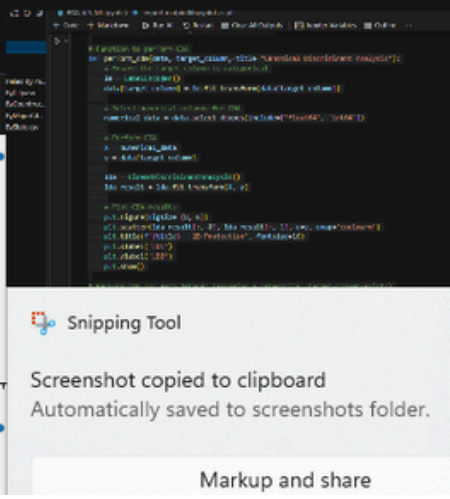
```python
# Function to perform CDA
def perform_cda(data, target_column, title="Canonical Discriminant Analysis"):
    # Ensure the target column is categorical
    le = LabelEncoder()
    data[target_column] = le.fit_transform(data[target_column])

    # Select numerical columns for CDA
    numerical_data = data.select_dtypes(include=['float64', 'int64'])

    # Perform CDA
    X = numerical_data
    y = data[target_column]

    lda = LinearDiscriminantAnalysis()
    lda_result = lda.fit_transform(X, y)

    # Plot CDA results
    plt.figure(figsize=(8, 6))
    plt.scatter(lda_result[:, 0], lda_result[:, 1], c=y, cmap='coolwarm')
    plt.title(f'{title} - 2D Projection', fontsize=16)
    plt.xlabel('LD1')
    plt.ylabel('LD2')
    plt.show()

# Perform CDA for each dataset (assuming a categorical target column exists)
for name, data in datasets.items():
    # Assume 'Category' is the target column (replace with actual target column name)
    if 'Category' in data.columns:
        perform_cda(data, 'Category', title=f'CDA - {name}')
```

[16]

```
[16]
# Pairwise relationships (already covered)
for name, data in datasets.items():
    numerical_data = data.select_dtypes(include=['float64', 'int64'])

    if numerical_data.empty:
        print(f"No numerical columns found in {name} for pairplot.")
        continue

    plt.figure(figsize=(12, 8))
    sns.pairplot(numerical_data)
    plt.suptitle(f'Pairwise Relationships - {name}', size=16)
    plt.show()
```

```
[17]
<Figure size 1200x800 with 0 Axes>
```





Snipping Tool

Screenshot copied to clipboard
Automatically saved to screenshots folder.

Markup and share

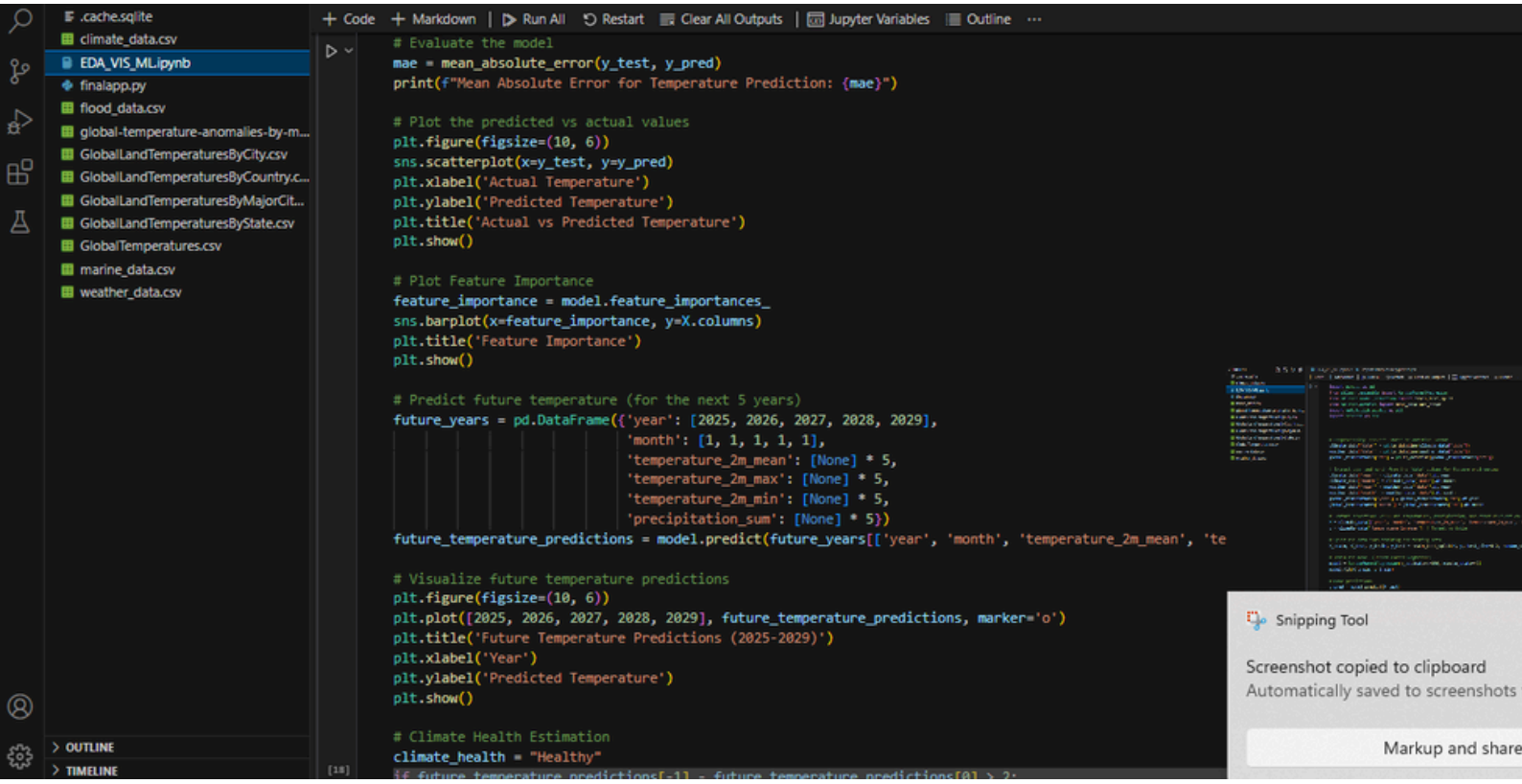
PROJECT

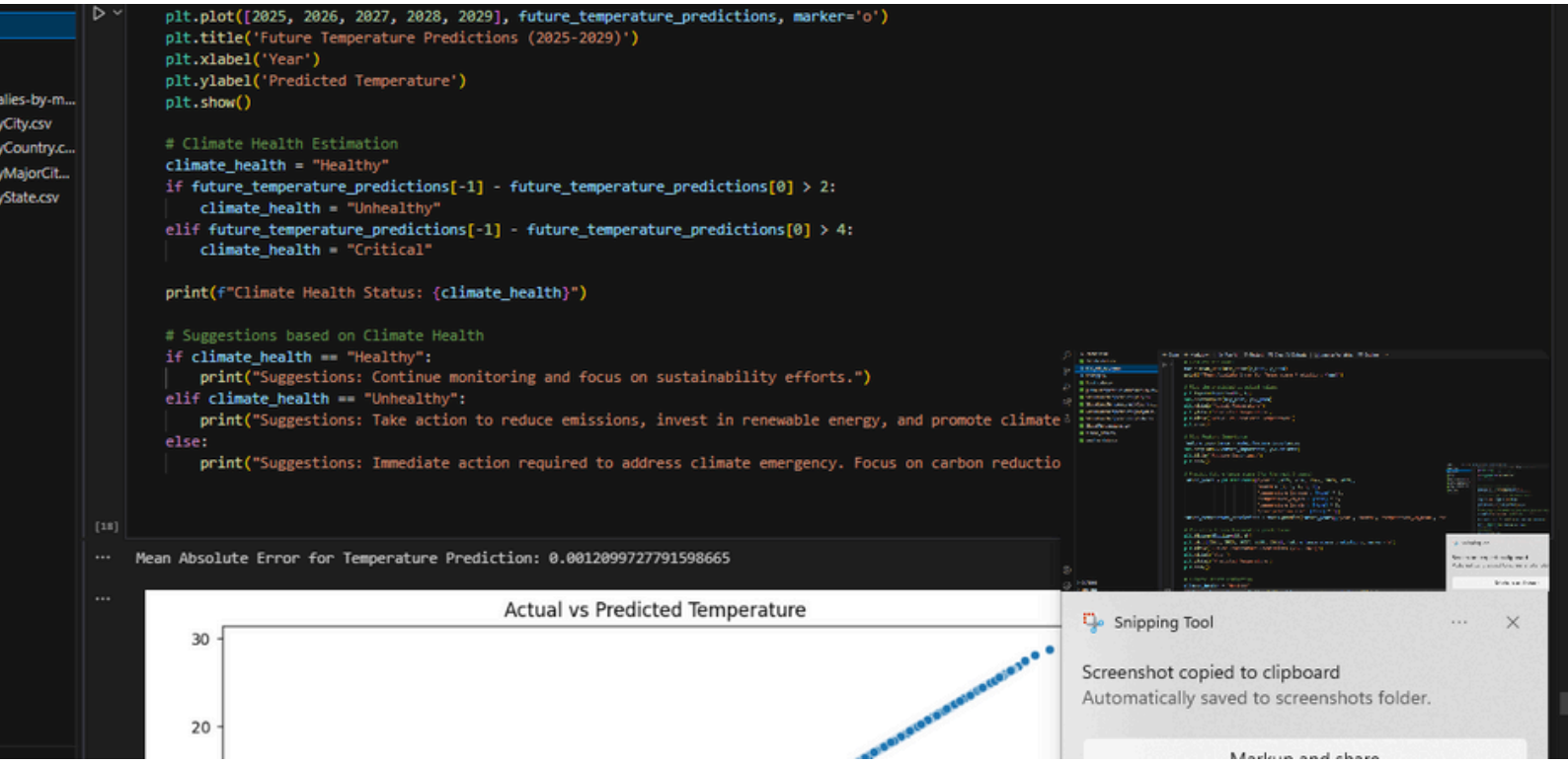
- .cache.sqlite
- climate_data.csv
- EDA_VIS_ML.ipynb
- finalapp.py
- flood_data.csv
- global-temperature-anomalies-by-m...
- GlobalLandTemperaturesByCity.csv
- GlobalLandTemperaturesByCountry.c...
- GlobalLandTemperaturesByMajorCit...
- GlobalLandTemperaturesByState.csv
- GlobalTemperatures.csv
- marine_data.csv
- weather_data.csv

EDA_VIS_ML.ipynb > import matplotlib.pyplot as plt

+ Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline ...

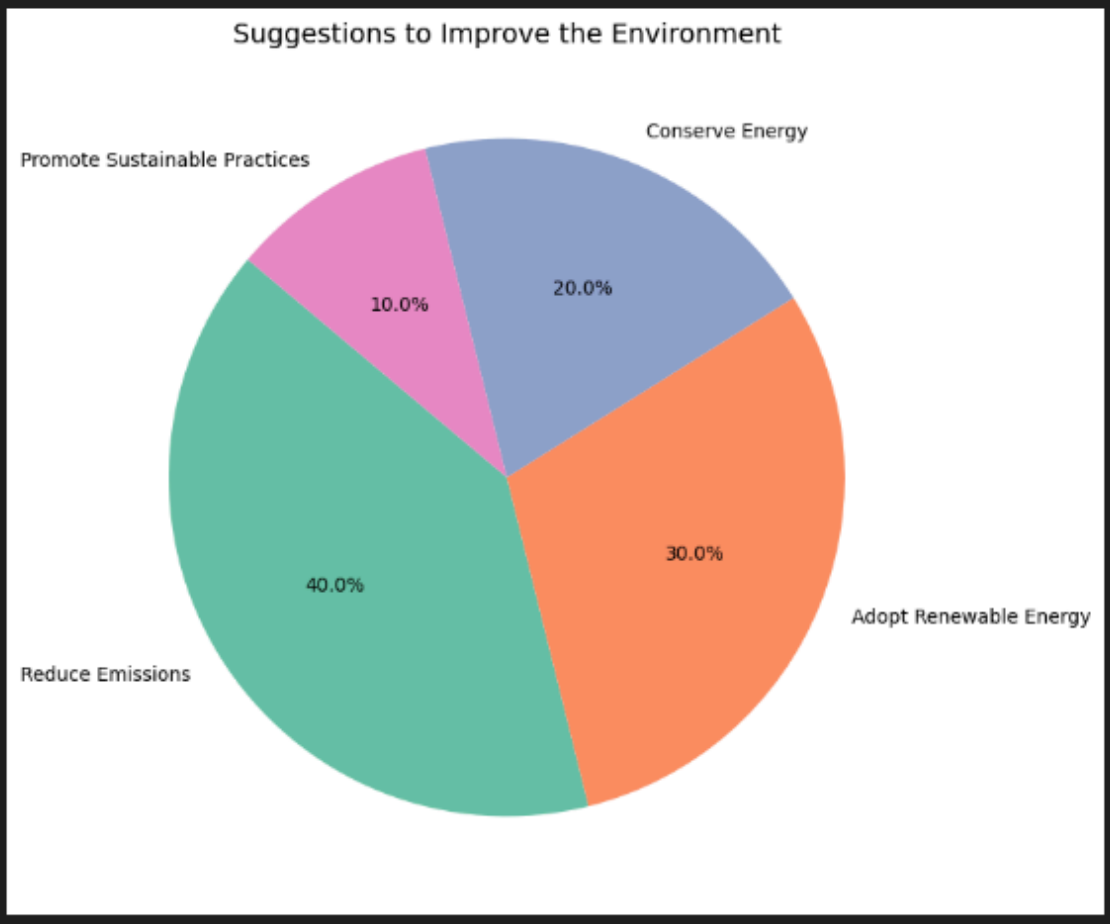
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sns



# Preprocessing: Convert 'date' to datetime format
climate_data['date'] = pd.to_datetime(climate_data['date'])
weather_data['date'] = pd.to_datetime(weather_data['date'])
global_temperatures['dt'] = pd.to_datetime(global_temperatures['dt'])

# Extract year and month from the 'date' column for feature engineering
climate_data['year'] = climate_data['date'].dt.year
climate_data['month'] = climate_data['date'].dt.month
weather_data['year'] = weather_data['date'].dt.year
weather_data['month'] = weather_data['date'].dt.month
global_temperatures['year'] = global_temperatures['dt'].dt.year
global_temperatures['month'] = global_temperatures['dt'].dt.month

# Feature selection: We'll use temperature, precipitation, and other relevant columns for prediction
X = climate_data[['year', 'month', 'temperature_2m_mean', 'temperature_2m_max', 'temperature_2m_min', 'precipitation_sum']]
y = climate_data['temperature_2m_mean']  # Target variable

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

# Train the model (Random Forest Regressor)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

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

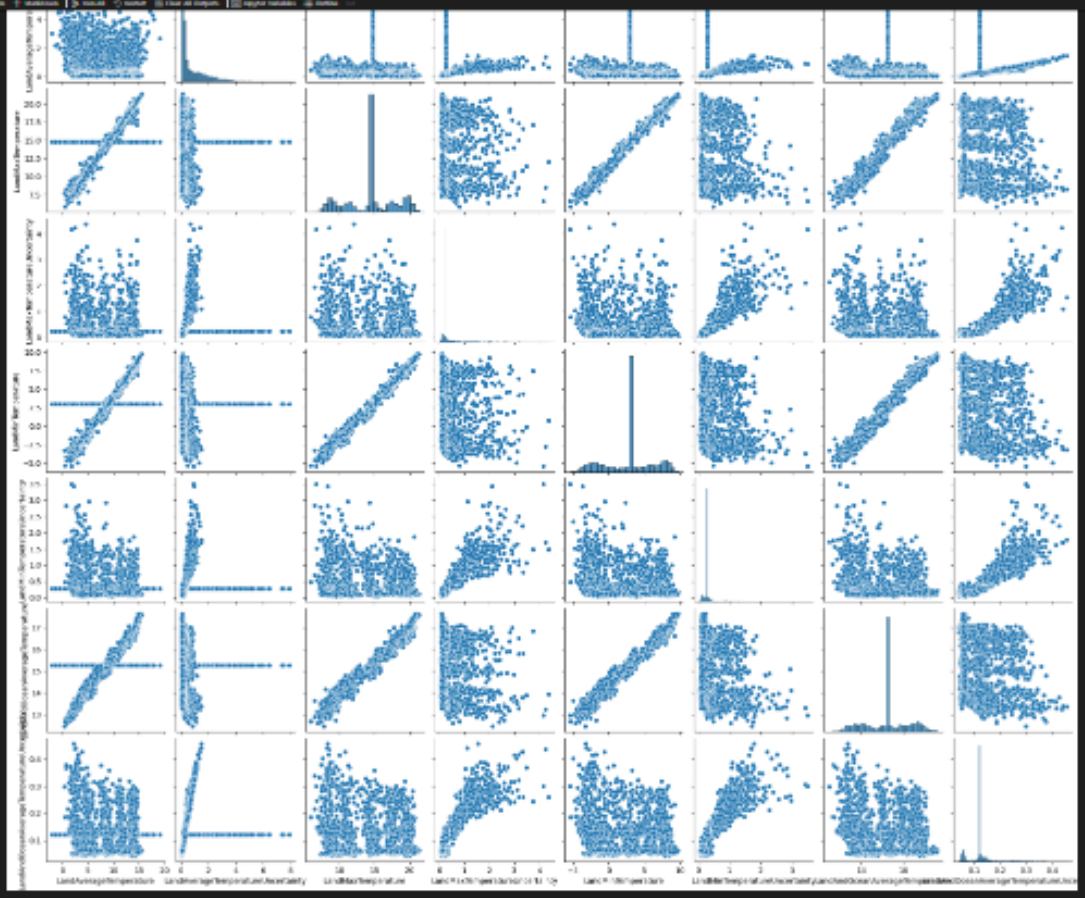

.cache.sqlite
climate_data.csv
EDA_VIS_ML.ipynb
finalapp.py
flood_data.csv
global-temperature-anomalies-by-m...
GlobalLandTemperaturesByCity.csv
GlobalLandTemperaturesByCountry.c...
GlobalLandTemperaturesByMajorCit...
GlobalLandTemperaturesByState.csv
GlobalTemperatures.csv
marine_data.csv
weather_data.csv
OUTLINE
TIMELINE
+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...
# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error for Temperature Prediction: {mae}")
# Plot the predicted vs actual values
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel('Actual Temperature')
plt.ylabel('Predicted Temperature')
plt.title('Actual vs Predicted Temperature')
plt.show()
# Plot Feature Importance
feature_importance = model.feature_importances_
sns.barplot(x=feature_importance, y=X.columns)
plt.title('Feature Importance')
plt.show()
# Predict future temperature (for the next 5 years)
future_years = pd.DataFrame({'year': [2025, 2026, 2027, 2028, 2029],
'month': [1, 1, 1, 1, 1],
'temperature_2m_mean': [None] * 5,
'temperature_2m_max': [None] * 5,
'temperature_2m_min': [None] * 5,
'precipitation_sum': [None] * 5})
future_temperature_predictions = model.predict(future_years[['year', 'month', 'temperature_2m_mean', 'te
# Visualize future temperature predictions
plt.figure(figsize=(10, 6))
plt.plot([2025, 2026, 2027, 2028, 2029], future_temperature_predictions, marker='o')
plt.title('Future Temperature Predictions (2025-2029)')
plt.xlabel('Year')
plt.ylabel('Predicted Temperature')
plt.show()
# Climate Health Estimation
climate_health = "Healthy"
Snipping Tool
Screenshot copied to clipboard
Automatically saved to screenshots
Markup and share

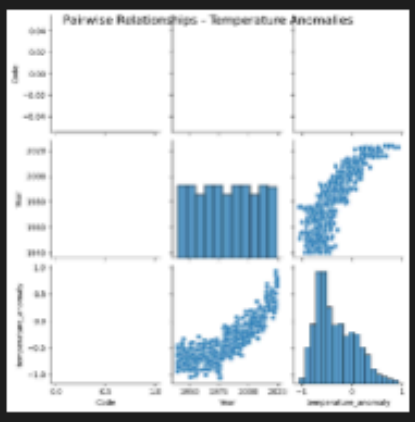

plt.plot([2025, 2026, 2027, 2028, 2029], future_temperature_predictions, marker='o')
plt.title('Future Temperature Predictions (2025-2029)')
plt.xlabel('Year')
plt.ylabel('Predicted Temperature')
plt.show()
# Climate Health Estimation
climate_health = "Healthy"
if future_temperature_predictions[-1] - future_temperature_predictions[0] > 2:
climate_health = "Unhealthy"
elif future_temperature_predictions[-1] - future_temperature_predictions[0] > 4:
climate_health = "Critical"
print(f"Climate Health Status: {climate_health}")
# Suggestions based on Climate Health
if climate_health == "Healthy":
print("Suggestions: Continue monitoring and focus on sustainability efforts.")
elif climate_health == "Unhealthy":
print("Suggestions: Take action to reduce emissions, invest in renewable energy, and promote climate
else:
print("Suggestions: Immediate action required to address climate emergency. Focus on carbon reductio
Mean Absolute Error for Temperature Prediction: 0.0012099727791598665
Actual vs Predicted Temperature
30
20
Snipping Tool
Screenshot copied to clipboard
Automatically saved to screenshots folder.

```
+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ···  Python
```

```python
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Climate Health Visualization
# Create a bar plot to show the temperature rise and corresponding climate health
temperature_rise = future_temperature_predictions[-1] - future_temperature_predictions[0]
climate_health = "Healthy"
if temperature_rise > 2:
    climate_health = "Unhealthy"
if temperature_rise > 4:
    climate_health = "Critical"

# Plotting the climate health based on temperature rise
plt.figure(figsize=(8, 6))
sns.barplot(x=["Temperature Rise"], y=[temperature_rise], palette="coolwarm")
plt.title(f"Climate Health Based on Temperature Rise: {climate_health}", fontsize=14)
plt.ylabel("Temperature Rise (°C)")
plt.show()

# 2. Suggestions for Action
# Pie chart showing possible actions to improve the environment
actions = ['Reduce Emissions', 'Adopt Renewable Energy', 'Conserve Energy', 'Promote Sustainable Practices']
action_values = [40, 30, 20, 10]

plt.figure(figsize=(8, 8))
plt.pie(action_values, labels=actions, autopct='%1.1f%%', startangle=140, colors=sns.color_palette("Set2"))
plt.title("Suggestions to Improve the Environment", fontsize=14)
plt.show()

# 3. Impact of Inaction (Temperature rise without action)
# Simulating a scenario where no action is taken and temperature continues to rise
future_temperature_no_action = future_temperature_predictions[-1] + (temperature_rise * 0.5)  # Exaggerated rise if no action

plt.figure(figsize=(10, 6))
plt.plot([2025, 2026, 2027, 2028, 2029], future_temperature_predictions, label="With Action", marker='o', color='green')
plt.plot([2025, 2026, 2027, 2028, 2029], [future_temperature_predictions[0]] * 5, linestyle="--", color="gray", label="No Action (Baseline)")
plt.title("Impact of Inaction on Future Temperature", fontsize=14)
plt.xlabel("Year")
plt.ylabel("Predicted Temperature (°C)")
plt.legend()
plt.show()
```






Pairwise Relationships - Temperature Anomalies

RUNNING... Stop Deploy

Navigation

Select a section to explore features

Select Section

Clean Data

- Clean Data
- Summary Statistics
- Pair Plots
- Heatmaps
- Time Series Analysis
- CDA
- Feature Engineering

Explore the cleaned data, with graphs showcasing the column distributions for better insights.

Weather Data

	date	temperature_2m_max	temperature_2m_min	precipitation_sum	windspeed_10m_max	windgusts_
0	2024-12-20 23:00:00+00:00	5.7	0.1	7.9	29.52	
1	2024-12-20 23:00:00+00:00	3.7	0.85	0	14.04	
2	2024-12-20 23:00:00+00:00	1.7	-3	0	12.96	
3	2024-12-20 23:00:00+00:00	2	-7.5	1.5	11.0901	
4	2024-12-20 23:00:00+00:00	2.45	-0.3	0	10.7036	
5	2024-12-20 23:00:00+00:00	1.55	-0.1	0	2.9686	
6	2024-12-20 23:00:00+00:00	0	-0.9	0	2.9024	

Distribution of temperature_2m_max

RUNNING... Stop Deploy

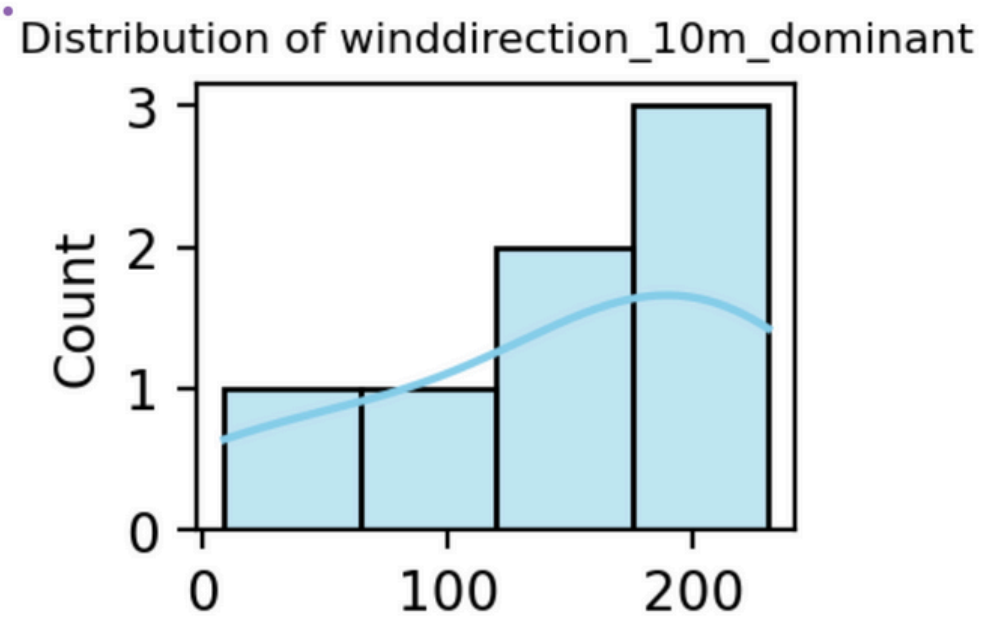
Navigation

Select a section to explore features

Select Section

Clean Data

- Clean Data
- Summary Statistics
- Pair Plots
- Heatmaps
- Time Series Analysis
- CDA
- Feature Engineering



RUNNING... Stop Deploy

Navigation

Select a section to explore features

Select Section

Clean Data

- Clean Data
- Summary Statistics
- Pair Plots
- Heatmaps
- Time Series Analysis
- CDA
- Feature Engineering

Global Temperature Anomalies

	Entity	Code	Year	temperature_anomaly
0	April	None	1,940	-0.6048
1	April	None	1,941	-0.8068
2	April	None	1,942	-0.8034
3	April	None	1,943	-0.7722
4	April	None	1,944	-0.6304
5	April	None	1,945	-0.6644
6	April	None	1,946	-0.6237
7	April	None	1,947	-0.5139
8	April	None	1,948	-0.7421
9	April	None	1,949	-0.7331

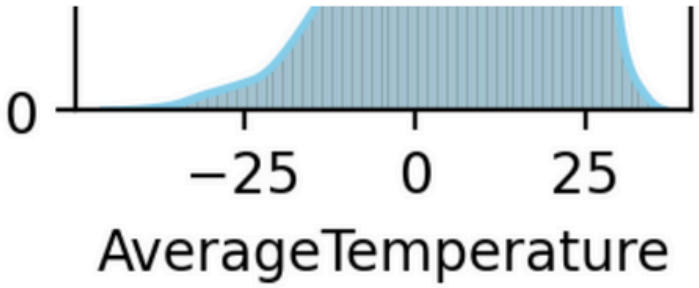
Navigation

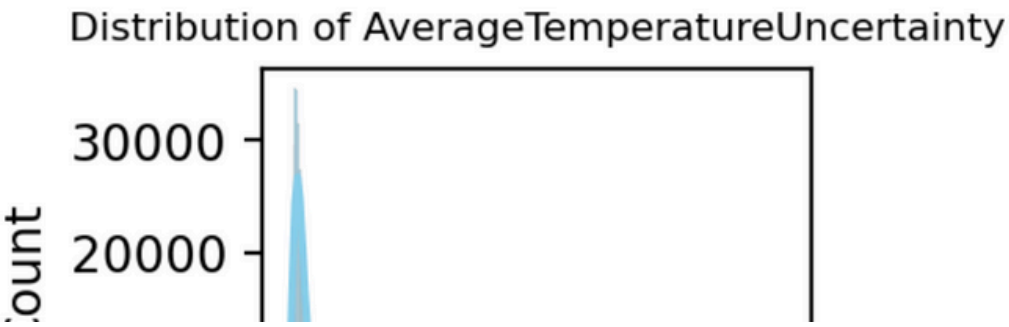
Select a section to explore features

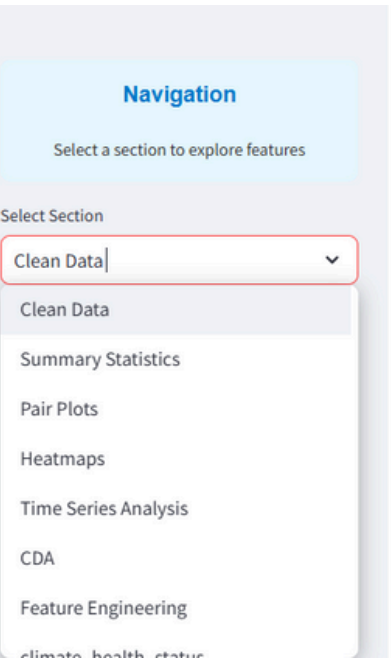
lect Section

Clean Data

- Clean Data
- Summary Statistics
- Pair Plots
- Heatmaps
- Time Series Analysis
- CDA
- Feature Engineering

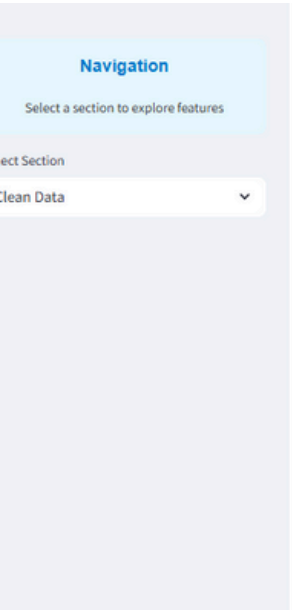






	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude
27	1851-04-01 00:00:00	26.604	1.654	Abidjan	Côte D'Ivoire	5.63N	3.23W
28	1851-05-01 00:00:00	26.212	1.195	Abidjan	Côte D'Ivoire	5.63N	3.23W
29	1851-06-01 00:00:00	24.898	1.329	Abidjan	Côte D'Ivoire	5.63N	3.23W
30	1851-07-01 00:00:00	24.318	1.463	Abidjan	Côte D'Ivoire	5.63N	3.23W
31	1851-08-01 00:00:00	23.752	1.394	Abidjan	Côte D'Ivoire	5.63N	3.23W
32	1851-09-01 00:00:00	24.083	1.121	Abidjan	Côte D'Ivoire	5.63N	3.23W
33	1851-10-01 00:00:00	25.138	1.505	Abidjan	Côte D'Ivoire	5.63N	3.23W
34	1851-11-01 00:00:00	25.527	1.409	Abidjan	Côte D'Ivoire	5.63N	3.23W
35	1851-12-01 00:00:00	26.203	2.358	Abidjan	Côte D'Ivoire	5.63N	3.23W
36	1852-01-01 00:00:00	None	None	Abidjan	Côte D'Ivoire	5.63N	3.23W

Distribution of AverageTemperature



Global Temperatures

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature	LandMaxTemperatureUncertainty	LandMinTemperature	LandMinTemperat
15	1751-04-01 00:00:00	7.67	2.368	None	None	None	
16	1751-05-01 00:00:00	None	None	None	None	None	
17	1751-06-01 00:00:00	13.827	1.801	None	None	None	
18	1751-07-01 00:00:00	None	None	None	None	None	
19	1751-08-01 00:00:00	14.405	2.296	None	None	None	
20	1751-09-01 00:00:00	10.673	2.656	None	None	None	
21	1751-10-01 00:00:00	None	None	None	None	None	
22	1751-11-01 00:00:00	None	None	None	None	None	
23	1751-12-01 00:00:00	None	None	None	None	None	
24	1752-01-01 00:00:00	0.348	3.789	None	None	None	

Distribution of LandAverageTemperature

Navigation

Select a section to explore features

Select Section

Summary Statistics

Weather Data

Descriptive Statistics for Weather Data:

	temperature_2m_max	temperature_2m_min	precipitation_sum	windspeed_10m_max	windgusts_10m_max	winddirection_10m_dominant
count	7	7	7	7	7	7
mean	2.4429	-1.55	1.3429	12.0264	24.7886	148.5447
std	1.8121	2.8889	2.945	8.9337	12.6452	82.8415
min	0	-7.5	0	2.9024	11.52	9.0664
25%	1.625	-1.95	0	6.8361	16.92	108.0362
50%	2	-0.3	0	11.0901	21.6	154.9686
75%	3.075	0	0.75	13.5	29.34	214.5375
max	5.7	0.85	7.9	29.52	47.88	230.6307

Mean, Median, and Mode for Weather Data:

	Mean	Median	Mode
temperature_2m_max	2.4429	2	0
temperature_2m_min	-1.55	-0.3	-7.5
precipitation_sum	1.3429	0	0
windspeed_10m_max	12.0264	11.0901	2.9024
windgusts_10m_max	24.7886	21.6	11.52
winddirection_10m_dominant	148.5447	154.9686	9.0664

Navigation

Select a section to explore features

Select Section

Summary Statistics

Clean Data

Summary Statistics

Pair Plots

Heatmaps

Time Series Analysis

CDA

Feature Engineering

Deploy

Global Temperatures

Descriptive Statistics for Global Temperatures:

	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature	LandMaxTemperatureUncertainty	LandMinTemperature	LandMinTemperatureUncertainty	LandAndOceanAverageTemperature	LandAndOceanAverageTemperatureUncertainty
count	3,180	3,180	1,992	1,992	1,992	1,992	1,992	1,992
mean	8.3747	0.9385	14.3506	0.4798	2.7436	0.4318	15.2126	0.1285
std	4.3813	1.0964	4.3096	0.5632	4.1558	0.4458	1.2741	0.0736
min	2.08	0.034	3.9	0.044	5.407	0.045	12.475	0.042
25%	4.312	0.1868	10.217	0.142	-1.3345	0.155	14.047	0.063
50%	8.6105	0.392	14.76	0.252	2.9495	0.279	15.251	0.122
75%	12.5483	1.4193	18.4515	0.539	6.7788	0.4583	16.3963	0.153
max	19.021	7.88	21.32	4.373	9.715	3.498	17.611	0.457

Mean, Median, and Mode for Global Temperatures:

	Mean	Median	Mode
LandAv	8.3747	8.6105	13.293
LandAv	0.9385	0.392	0.087
LandMa	14.3506	14.76	8.555
LandMa	0.4798	0.252	0.093
LandMi	2.7436	2.9495	-1.339
LandMi	0.4318	0.279	0.237
LandAn	15.2126	15.251	15.005
LandAn	0.1285	0.122	0.061

Temperature Anomalies

Descriptive Statistics for Temperature Anomalies:

	Code	Year	temperature_anomaly
count	0	1,018	1,018
mean	None	1,961.9175	0.3286

White Blue Modern Project Outlin | WeatherView

localhost:8501

eatsheet-a5.pdf | HTML Examples | New folder | New tab | W3Schools Online... | Dell | Imported From IE | Imported From IE (1) | Gmail

Navigation

lect a section to explore features

tion

ary Statistics

Descriptive Statistics for Land Temperatures by

	AverageTemperature	AverageTemperatureUncertainty
count	228,175	228,175
mean	18.126	0.9693
std	10.0248	0.9706
min	26.772	0.04
25%	12.71	0.34
50%	20.428	0.592
75%	25.918	1.32
max	38.283	14.037

Mean, Median, and Mode for Land Temperatures by Major City:

	Mean	Median	Mode
AverageTemperature	18.126	20.428	26.612
AverageTemperatureUncertainty	0.9693	0.592	0.256

Land Temperatures by Country

Descriptive Statistics for Land Temperatures by Country:

	AverageTemperature	AverageTemperatureUncertainty
count	544,811	545,550
mean	17.1934	1.0191
std	10.954	1.2019
min	37.658	0.052
25%	10.025	0.323
50%	20.901	0.571
75%	25.814	1.206
max	38.842	15.003

Mean, Median, and Mode for Land Temperatures by Country:

Navigation

Select a section to explore features

lect Section

Summary Statistics

Flood Data

Descriptive Statistics for Flood Data:

	river_discharge	river_discharge_mean	river_discharge_median	river_discharge_max	river_discharge_min	river_discharge_p25	river_discharge_p75
count	92	92	92	92	92	92	92
mean	5.5638	4.5234	3.8543	12.2772	3.0859	3.4948	4.8495
std	2.4152	0.7001	0.9339	4.1298	0.7696	0.8311	0.8584
min	3.1879	3.5963	3.0133	5.0534	2.1117	2.7154	3.6885
25%	4.233	4.0549	3.2243	9.4894	2.5975	2.9253	4.2811
50%	4.8681	4.2654	3.4515	11.4173	2.7111	3.1376	4.6172
75%	5.8951	4.8091	4.2152	14.1009	3.4952	3.7727	5.1564
max	18.5885	6.891	7.0538	24.5859	5.8391	5.8391	7.9228

Mean, Median, and Mode for Flood Data:

	Mean	Median	Mode
river_d	5.5638	4.8681	4.4828
river_d	4.5234	4.2654	3.5963
river_d	3.8543	3.4515	3.2025
river_d	12.2772	11.4173	9.2565
river_d	3.0859	2.7111	2.6898
river_d	3.4948	3.1376	2.8107
river_d	4.8495	4.6172	3.6885

White Blue Modern Project Outlin | WeatherView

localhost:8501

cheatsheet-a5.pdf | HTML Examples | New folder | New tab | W3Schools Online... | Dell | Imported From IE | Imported From IE (1) | Gmail | YouTube | Maps | Arif Malik: One of to...

RUNNING... Stop Deploy

Navigation

Select a section to explore features

Select Section

Pair Plots

Welcome to WeatherView

Dive into the world of climate insights! Analyze global weather patterns, visualize trends, and make data-driven predictions.

Pair Plots

Creating pair plots to visualize relationships, correlations, and distributions between multiple variables in a dataset for data exploration and analysis.

Pair plot for Global Temperature Anomalies

Calling `st.pyplot()` without providing a figure argument has been deprecated and will be removed in a later version as it requires the use of Matplotlib's global figure object, which is not thread-safe.

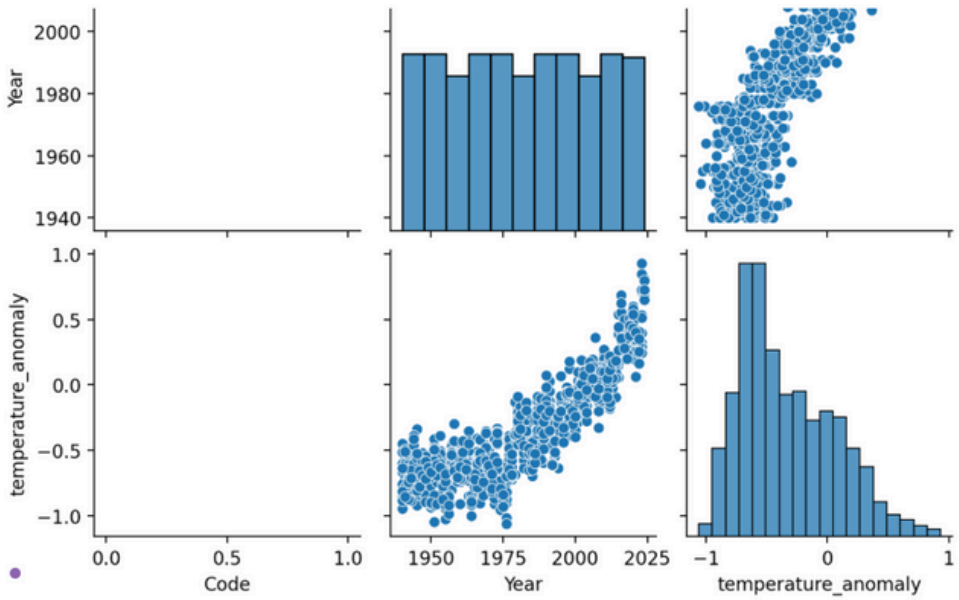
RUNNING... Stop Deploy

Navigation

Select a section to explore features

Select Section

Pair Plots



Pair plot for Land Temperatures by State

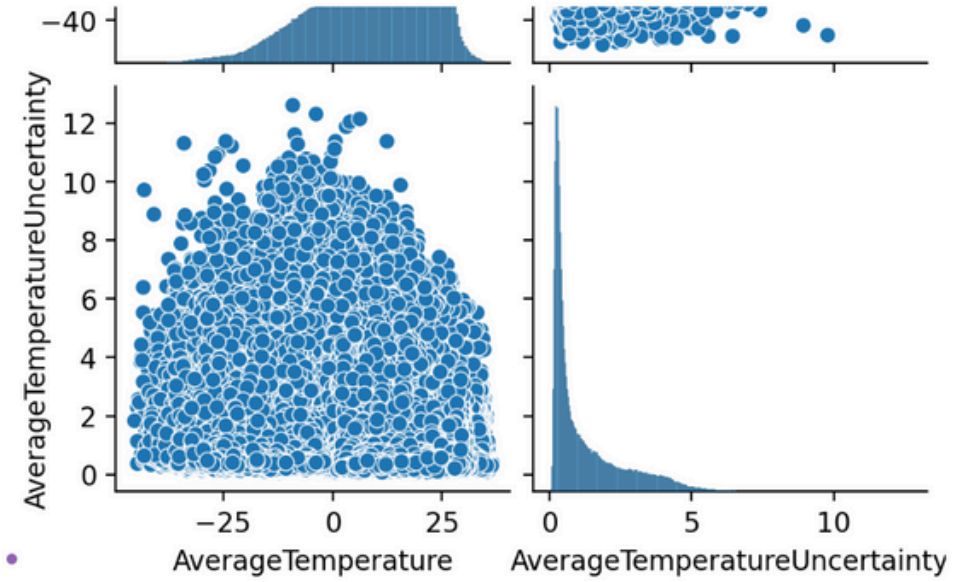
Navigation

Select a section to explore features

Select Section

Pair Plots

RUNNING... Stop Deploy



Pair plot for Land Temperatures by Major City

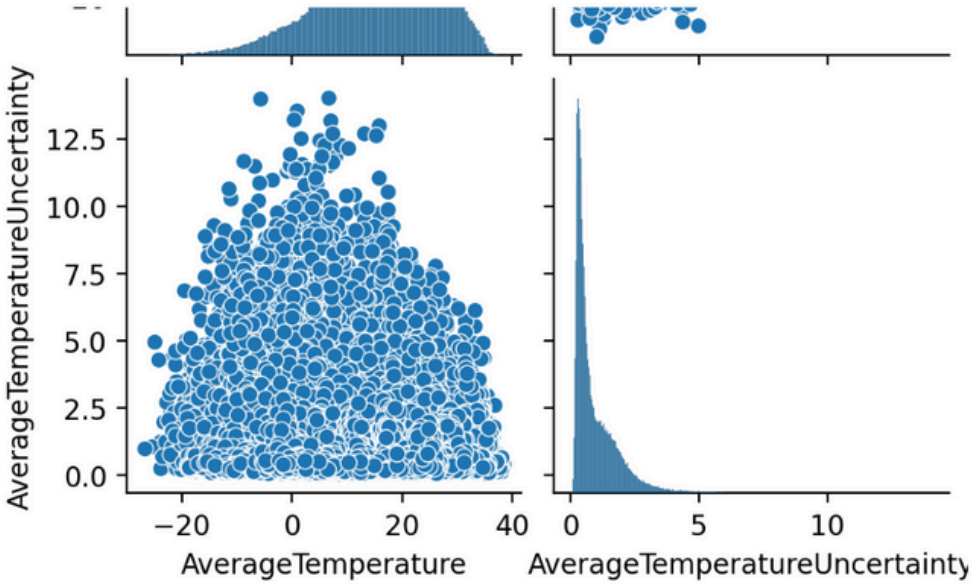
Navigation

Select a section to explore features

Select Section

Pair Plots

RUNNING... Stop Deploy



Pair plot for Land Temperatures by Country

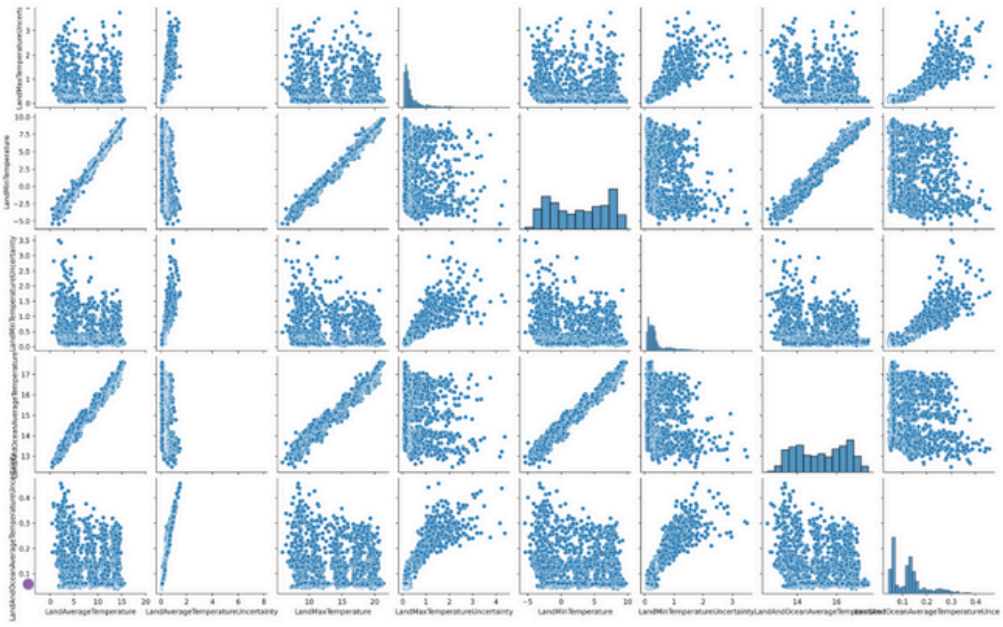
RUNNING... Stop Deploy

Navigation

Select a section to explore features

elect Section

Pair Plots



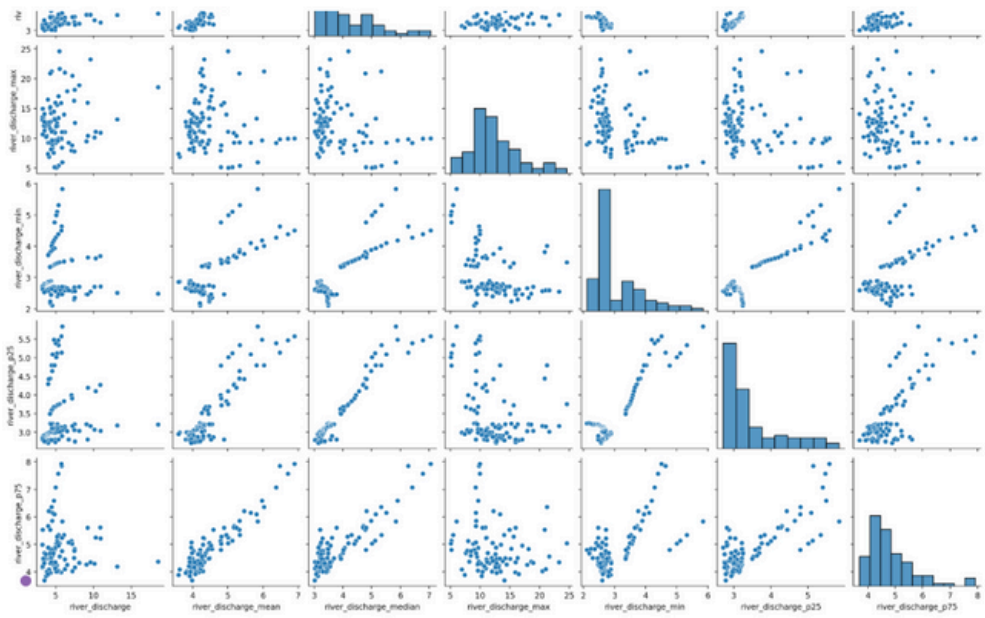
Pair plot for Flood Data

Navigation

Select a section to explore features

lect Section

Pair Plots



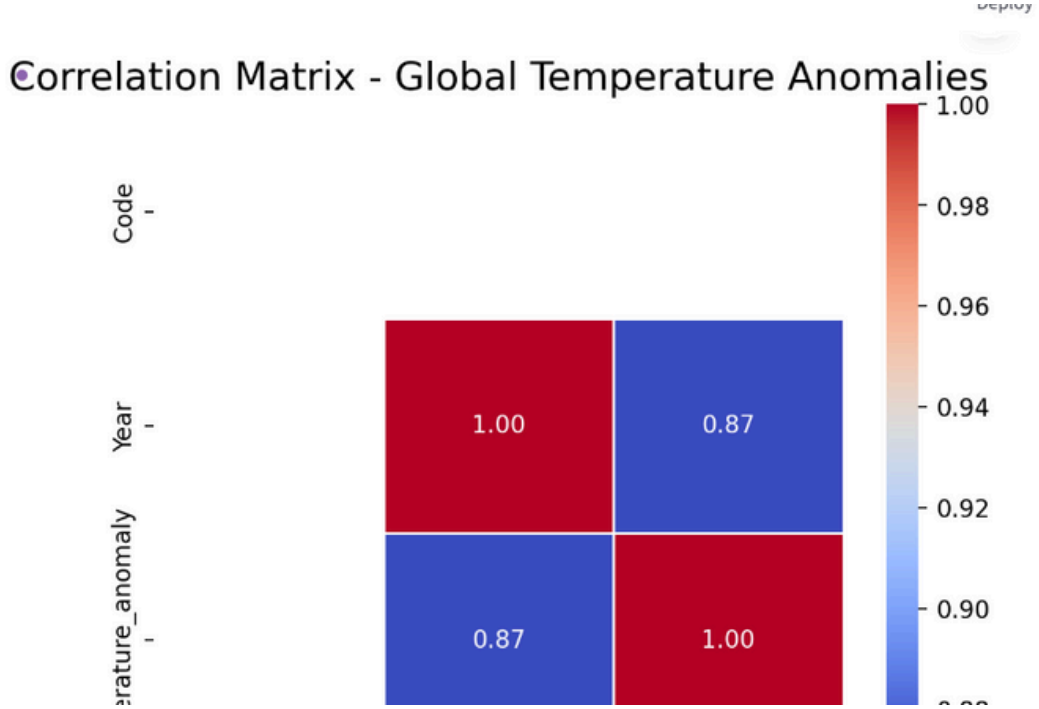
Pair plot for Marine Data

Navigation

Select a section to explore features

elect Section

Heatmaps

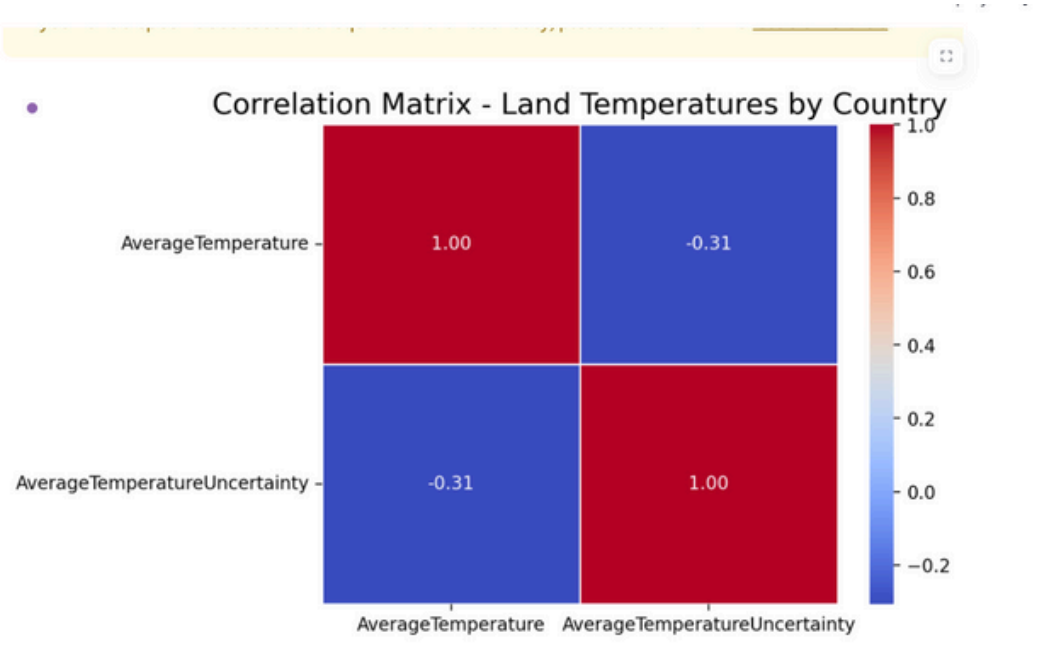


Navigation

Select a section to explore features

Select Section

Heatmaps



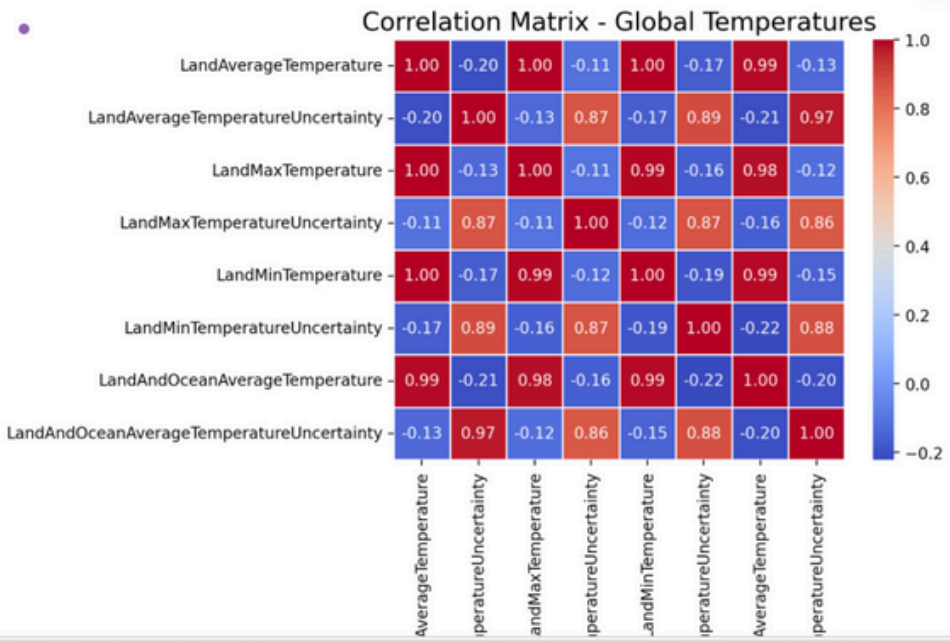
Deploy

Navigation

Select a section to explore features

Select Section

Heatmaps



58°F Smoke

Search

11:43 PM 1/11/2025

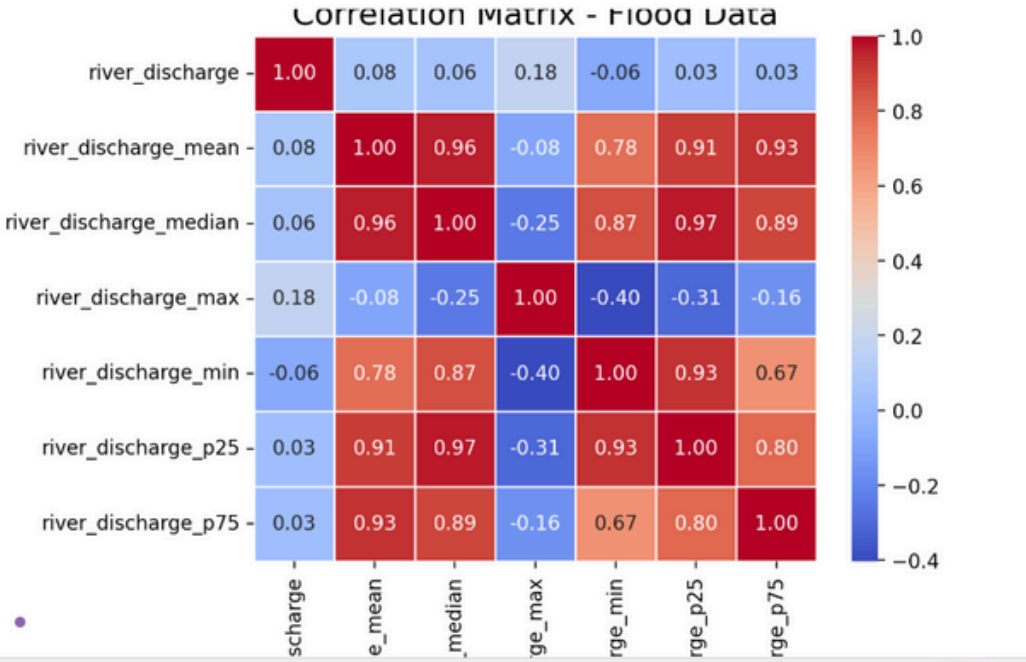
Deploy

Navigation

Select a section to explore features

Select Section

Heatmaps



White Blue Modern Project Outlir | WeatherView

localhost:8501

cheatsheet-a5.pdf | HTML Examples | New folder | New tab | W3Schools Online... | Dell | Imported From IE | Imported From IE (1) | Gmail | YouTube | Maps | Arif Malik: One of to...

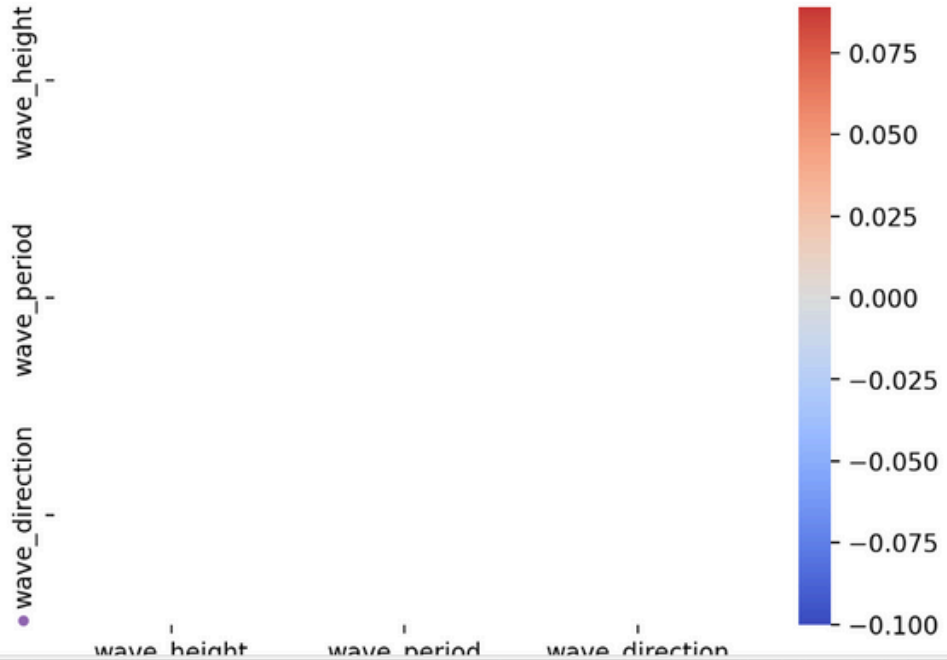
Deploy

Navigation

Select a section to explore features

Select Section

Heatmaps

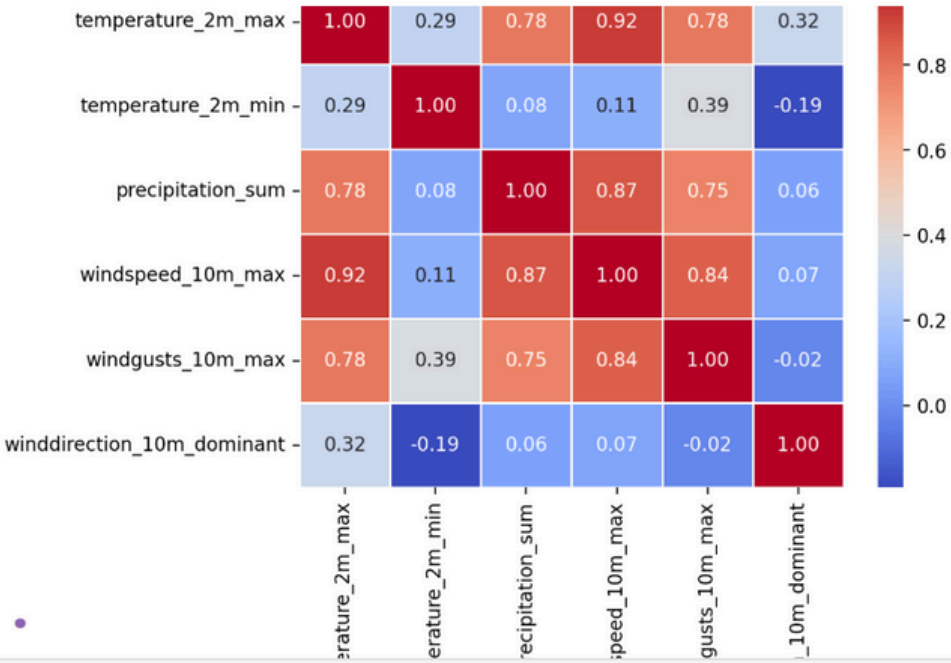


Navigation

Select a section to explore features

Select Section

Heatmaps



Deploy

Navigation

Select a section to explore features

Select Section

Time Series Analysis

Welcome to WeatherView

Dive into the world of climate insights! Analyze global weather patterns, visualize trends, and make data-driven predictions.

Time Series Analysis

Performing time series analysis on weather data is helping to identify trends, seasonal patterns, and anomalies over time for forecasting and insights, results will be shown in future engineering and climate health section. stay tuned .....

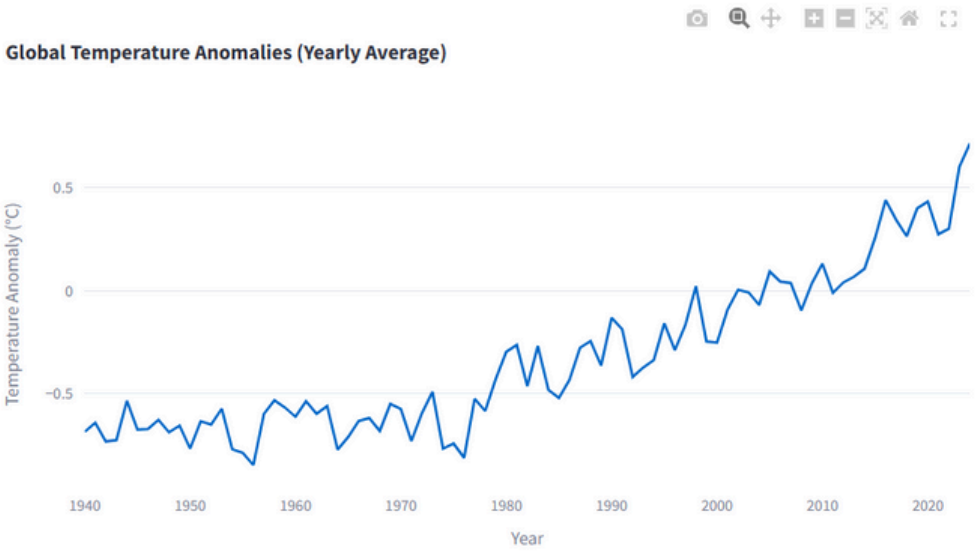
Deploy

Navigation

Select a section to explore features

Select Section

climate_health_status

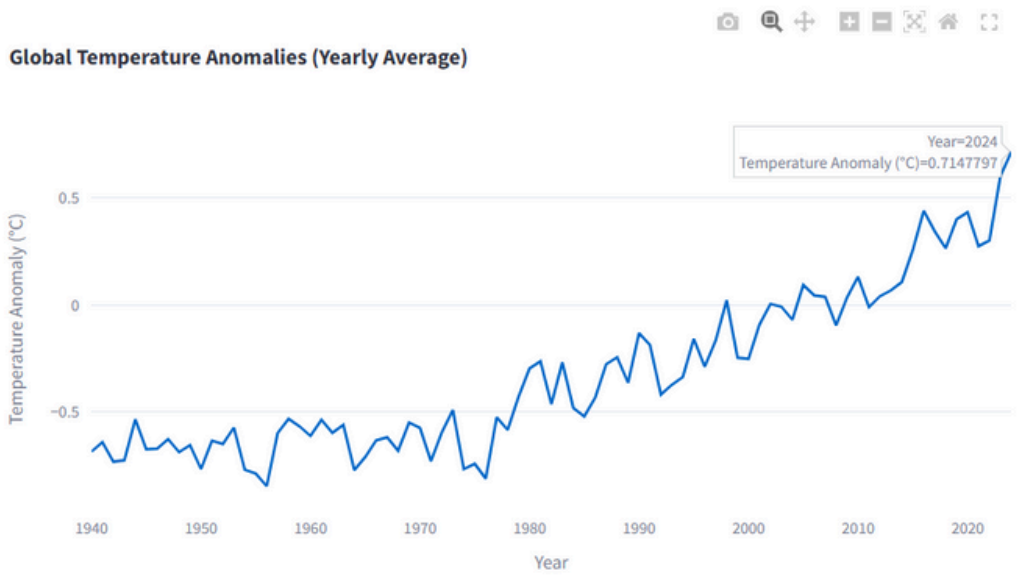


Navigation

Select a section to explore features

ect Section

limate_health_status



Predictive Analysis: Temperature Anomalies

Predicted Temperature Anomalies for 2025-2030:

	Year	Predicted_Anomaly
0	2,025	0.2976
1	2,026	0.3122
2	2,027	0.3268
3	2,028	0.3414
4	2,029	0.356
5	2,030	0.3706

Deploy

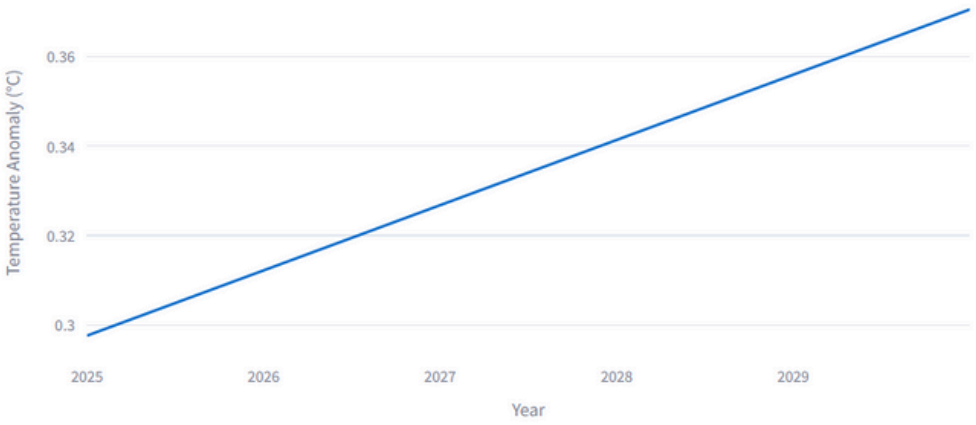
Navigation

Select a section to explore features

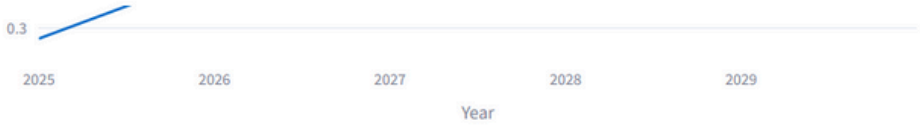
elect Section

climate_health_status

Predicted Global Temperature Anomalies (2025-2030)



Deploy



Navigation

Select a section to explore features

Select Section

climate_health_status

Insights and Suggestions

- Global temperature anomalies have been steadily increasing, indicating a warming trend.
- Yearly precipitation patterns show variability, which could impact agriculture and water resources.
- Predictive analysis suggests further warming in the next decade.
- Governments and organizations should focus on reducing greenhouse gas emissions and adopting sustainable practices.