

**BOOTCAMP DATA ANALYSIS**

**PATRICIO AUGUSTO PATIÑO ESPAÑA**

**CHALLENGE MODULE6**

**WeatherPy and VacationPy Analysis**

**WRITTEN REPORT**

**September 24th, 2024**

**WeatherPy**

The code provided conducts an analysis of the relationship between weather variables (temperature, humidity, cloudiness, wind speed) and geographical latitude, utilizing the OpenWeatherMap API. Here's a breakdown of the key processes in the code:

**1. City List Generation**

* The code uses the citipy library to generate a list of random cities based on randomly generated latitude and longitude coordinates.
* numpy is used to create a set of 1500 random latitude and longitude pairs within the global ranges (latitude: -90 to 90, longitude: -180 to 180). For each of these coordinates, the citipy library determines the nearest city.
* Cities are stored in the list cities to ensure uniqueness.

**2. Weather Data Retrieval**

* For each city, the OpenWeatherMap API is queried using a URL constructed with the city name and an API key stored in the weather\_api\_key.
* The following weather variables are retrieved for each city:
  + Latitude (city\_lat)
  + Longitude (city\_lng)
  + Maximum temperature (city\_max\_temp)
  + Humidity (city\_humidity)
  + Cloudiness (city\_clouds)
  + Wind speed (city\_wind)
  + Country (city\_country)
  + Date of data retrieval (city\_date)
* The weather data is stored in a dictionary format and appended to the list city\_data.

**3. Data Storage**

* The retrieved data is stored in a Pandas DataFrame and exported to a CSV file (cities.csv). This allows for easy access and further analysis.

**4. Scatter Plots**

* Scatter plots are generated to show the relationships between latitude and four weather variables:
  + **Max Temperature vs. Latitude**
  + **Humidity vs. Latitude**
  + **Cloudiness vs. Latitude**
  + **Wind Speed vs. Latitude**
* Each plot displays the data points with specific labels and titles, and the plots are saved as PNG files.

**5. Linear Regression Analysis**

* A function create\_linear\_regression\_plot is defined to compute linear regression on two variables and plot the resulting regression line along with the scatter plot.
* Linear regression is applied to the weather variables vs. latitude for both the **Northern Hemisphere** and the **Southern Hemisphere**.
* For each regression, the slope and intercept of the regression line are displayed on the plot, and the r-value (correlation coefficient) is printed. This provides a measure of how well the weather variable correlates with latitude.

**6. Northern vs. Southern Hemisphere**

* The cities are split into two dataframes:
  + northern\_hemi\_df for cities in the Northern Hemisphere (latitude ≥ 0)
  + southern\_hemi\_df for cities in the Southern Hemisphere (latitude < 0)
* Separate linear regression plots are created for the two hemispheres to examine if there are any differences in how latitude affects the weather variables across the globe.

**Key Points of Analysis:**

* **Temperature vs. Latitude**: Typically, temperature decreases as latitude increases (i.e., as you move further from the equator).
* **Humidity vs. Latitude**: The relationship might not be as strong, but areas near the equator often have higher humidity.
* **Cloudiness vs. Latitude**: There might not be a clear relationship between cloudiness and latitude.
* **Wind Speed vs. Latitude**: Wind speed may vary with latitude due to different atmospheric conditions near the poles and the equator.

Overall, this code provides a thorough way to analyze weather patterns across the globe based on latitude, using both scatter plots and linear regression to draw insights.

**VacationPy**

In this analysis, we utilized weather data to find cities with ideal weather conditions and then identified nearby hotels using the Geoapify API. The analysis involves several steps to filter the data, perform location-based queries, and visualize the results on a map.

**1. Data Preparation**

We began by importing necessary libraries and loading weather and coordinates data into a DataFrame. This included the following libraries and steps:

* Libraries Imported: hvplot.pandas, pandas, and requests.
* Data Loading: Read weather data from a CSV file into city\_data\_df.

**2. Ideal Weather Condition Filtering**

To find cities with desirable weather conditions, we filtered the DataFrame based on the following criteria:

* Max Temperature: Between 20°C and 70°C.
* Wind Speed: Less than 10 mph.
* Cloudiness: Less than 20%.

This filtering resulted in a DataFrame, ideal\_weather\_df, which contains cities meeting these criteria. We also ensured that any rows with null values were dropped to maintain data quality.

**3. Creating the Hotel DataFrame**

We created a new DataFrame, hotel\_df, from the filtered data:

* Columns Included: City, Country, Latitude, Longitude, and Humidity.
* Added Column: Hotel Name to store results from the Geoapify API.

**4. Finding Nearby Hotels**

We used the Geoapify API to find the nearest hotel within a 10,000-meter radius for each city:

* API Parameters: Included categories for accommodation (hotels), a limit of 1 result per query, and the API key.
* API Requests: Sent requests for each city, and processed the responses to extract hotel names.
* Error Handling: Managed potential errors where no hotels were found or API responses were incomplete.

**5. Visualization**

We created two key visualizations:

* Initial Map: Displayed cities with points sized by humidity, providing a geographic view of the data.
* Final Map: Included the nearest hotel's name and country in the hover message. This map visualized the filtered cities and provided additional details about nearby accommodations.

**Conclusions**

1. Ideal Weather Conditions:
   * The filtering process effectively narrowed down the cities to those with pleasant weather, suitable for travel or relocation.
   * The data analysis provided a focused list of cities with comfortable temperatures, low wind speeds, and minimal cloudiness.
2. Hotel Proximity:
   * Using the Geoapify API, we were able to find hotels near the filtered cities, offering practical options for travelers.
   * The inclusion of hotel names in the final map provides a valuable resource for planning trips or stays in cities with ideal weather.
3. Visual Insights:
   * The maps provided clear visualizations of the data, making it easy to see the distribution of cities and the relative sizes of the points based on humidity.
   * The final map with hotel details enhances the usability of the data by integrating accommodation options directly into the geographic view.

Overall, this analysis combines weather data filtering with practical location-based queries to offer a comprehensive view of travel options based on ideal weather conditions and nearby hotel availability.