**Data Cleanup Strategy**

**Overview**

This code performs web scraping to gather election data, processes and cleans the collected data, and then saves it to CSV files. The cleanup strategy involves handling missing values, cleaning specific columns, removing duplicates, and ensuring data consistency. Below is a detailed explanation of the data cleanup strategy used in the code.

**Functions and Their Use**

1. \*\*`get\_data\_and\_next\_page(response\_text: str)`\*\*:

- Extracts the election data and the URL for the next page from the response text.

- Converts the extracted JSON string to a Python dictionary.

- Returns the data and the next page URL.

2. \*\*`navigate\_next\_page(base\_url, next\_page)`\*\*:

- Constructs the full URL for the next page if it exists.

- Returns `None` if there is no next page.

3. \*\*`scrape\_election\_data(base\_url, state\_entries)`\*\*:

- Iterates through the state entries and fetches data for each entry.

- Utilizes the `get\_data\_and\_next\_page` function to parse the response.

- Appends the fetched data to a dictionary that will be used to create a DataFrame.

4. \*\*`clean\_and\_save\_data(state\_name, ac\_name, election\_data)`\*\*:

- Converts the election data dictionary into a DataFrame.

- Replaces empty strings with NaN values.

- Removes rows where the 'VOTES' column is "None".

- Cleans specific columns ('AC', 'CANDIDATE') by removing numeric prefixes and non-alphabetic characters.

- Strips leading and trailing whitespaces from all string columns.

- Drops duplicate rows to ensure data consistency.

- Saves the cleaned DataFrame to CSV files, both for the entire state and for a specific Assembly Constituency (AC).

5. \*\*`find\_state\_entries(start\_url, state\_name, n\_pages=10)`\*\*:

- Crawls the specified number of pages to find entries related to the given state.

- Utilizes the `get\_data\_and\_next\_page` function to parse the response.

- Collects state entries and navigates to the next page until the specified number of pages is reached or no more pages are available.

6. \*\*`main\_scraper(state\_name, ac\_name)`\*\*:

- Entry point of the script.

- Initializes the scraping process by calling `find\_state\_entries`.

- Calls `scrape\_election\_data` to fetch and clean the data.

- Saves the cleaned data to CSV files using `clean\_and\_save\_data`.

**Data Cleanup Steps**

1. \*\*Handling Missing Values\*\*:

- Rows with missing or non-numeric 'VOTES' values are filtered out during the data collection phase.

- Empty strings are replaced with NaN values using `df.replace('', np.nan, inplace=True)`.

2. \*\*Cleaning Specific Columns\*\*:

- The 'AC' (Assembly Constituency) and 'CANDIDATE' columns are cleaned to remove numeric prefixes and non-alphabetic characters using regular expressions:

```python

df['AC'] = df['AC'].str.replace(r'^\d+\s+', '', regex=True)

df['AC'] = df['AC'].str.replace(r'[^a-zA-Z\s\.]', '', regex=True)

df['CANDIDATE'] = df['CANDIDATE'].str.replace(r'^\d+\s+', '', regex=True)

df['CANDIDATE'] = df['CANDIDATE'].str.replace(r'[^a-zA-Z\s\.]', '', regex=True)

```

- All string columns are stripped of leading and trailing whitespaces using `df.applymap(lambda x: x.strip() if isinstance(x, str) else x)`.

3. \*\*Removing Duplicates\*\*:

- Duplicate rows are removed using `df.drop\_duplicates()`.

- Additional duplicate removal is performed based on specific columns to ensure uniqueness:

4. \*\*Saving Data\*\*:

- The cleaned DataFrame is saved to a CSV file for the entire state.

- A filtered DataFrame for a specific Assembly Constituency (AC) is created and saved to a separate CSV file.

**Summary**

The data cleanup strategy ensures that the collected election data is accurate, consistent, and ready for analysis. By filtering out invalid data, cleaning specific columns, and removing duplicates, the script prepares the data for reliable use in further analysis or reporting.

**link to scripts used for data transformation:**

https://colab.research.google.com/drive/1PCwaHXeOzR5tSPm62pqF8DTmjPeN9l5U?usp=sharing