HyperionDev

Data Cleaning and Preprocessing

Model Answer Approach

Visit our website

Auto-graded task 1

The solution provided handles data cleaning and processing for a dataset. Initially, it displays unique country names before any cleaning is applied to assess the data's starting state. The `fillna()` method is used to replace missing values in the 'country' column with 'unknown', ensuring that no null entries remain.

Subsequently, the dataset is refined by converting all country names to lowercase and removing leading or trailing whitespace. This standardisation helps address inconsistencies. The `replace()` method is then employed to correct specific country names by substituting variations like 'UK' with 'United Kingdom'.

To further standardise the data, the `replace_country_names()` function uses fuzzy logic to find and replace country names that closely match predefined strings, with a minimum similarity ratio specified. This function is called multiple times to ensure various synonyms and abbreviations are replaced accurately.

Additionally, the `replace_name()` function is utilised to make general replacements across the dataset, correcting any remaining inconsistencies and ensuring uniformity. It addresses cases such as substituting 'america' with 'United States' and cleaning up extraneous characters like '/' and '.'.

Lastly, date-related columns are processed by converting 'date_measured' to a DateTime format and creating a 'days_ago' column. This new column calculates the difference between the current date and the measurement date, providing insight into how long ago each entry was recorded.

Common pitfalls include failing to address inconsistencies in text data, such as varying capitalisations or extra spaces, which can lead to inaccurate analysis. It's also crucial to verify that replacements are correct and not inadvertently replacing unintended values. Additionally, ensure that date formats are consistently handled to avoid errors in time calculations.

Auto-graded task 2

In this approach, we start by reading a CSV file using pandas' `read_csv()` function, which returns a DataFrame – a two-dimensional labelled data structure with rows and columns. We then utilise the `head()` method to retrieve the first five observations from the DataFrame. By default, the `head()` method returns the first five observations. Subsequently, we identify missing values in each column using the `isnull()` function, followed by the `sum()` function to count the number of missing values in each column of the DataFrame. Finally, we classify the given scenarios according to the categories or types of missingness of data, which include Missing Not At Random, Missing At Random, Missing Not at Random, and Missing Completely at Random.

Auto-graded task 3

In the first set of two scenarios, we determine whether standardisation or normalisation makes more sense.

Normalisation is chosen for the first scenario because the data features vary in scales. Normalisation scales the features to a range between zero and one, while preserving the relative differences. For the other scenario, standardisation works well, as the data assumes a Gaussian distribution. Standardisation transforms the features to have a mean of zero and a standard deviation of one. We then employ the `minmax()` function to normalise the “EG.ELC.ACCS.ZS” feature from the “countries” DataFrame. Subsequently, Seaborn’s `histplot()` is utilised to plot a histogram showing the distribution of our column before and after the normalisation process. Following normalisation, the scales are observed to be between zero and one, indicating the success of the normalisation process.