Dataset Analysis and Cleaning

21/03/2025

stratified_history.csv
8.1 MB

Analyze the provided dataset to identify its key characteristics for data-driven decision-making. Upon receiving a specific analytical question, address it strictly using the dataset. Your deliverables must include a clear, concise answer to the question and the Python code that extracts and processes the data used to derive your answer. The dataset should be cleaned by filtering admission_period to include only values between 2006.1 and 2019.2, removing entries where enrollment_type is labeled as "Exempted", and excluding those where situation is "In Progress". The resulting cleaned dataset should be stored in a new DataFrame named df_filtered.

qwen-max-latest

To address your request, I will outline the key characteristics of the dataset and then proceed to clean it according to the specified criteria. Once the dataset is cleaned, I will store it in a new DataFrame named `df_filtered`. Below is the step-by-step process:

Key Characteristics of the Dataset

1. **Columns and Data Types** :
   - The dataset contains numerous columns, including identifiers like `id`, temporal data such as `admission_period` and `enrollment_period`, categorical data like `situation`, `enrollment_type`, and `quota`, and numerical data such as `grade`.
   - Columns like `admission_period` and `enrollment_period` appear to be in a "year.semester" format (e.g., 2006.1 for the first semester of 2006).
2. **Temporal Scope** :
   - The dataset spans multiple years, with `admission_period` ranging from the 1980s to 2019.
3. **Categorical Variables** :
   - `situation`: Indicates the student's academic status (e.g., Graduate, Dropout, Reentry).
   - `enrollment_type`: Describes the type of enrollment (e.g., Regular, Exempted).
   - `quota`: Specifies whether the student benefited from quota reservation.
4. **Numerical Variables** :
   - `grade`: Represents the grade obtained in a specific course.
   - `credits`: Indicates the number of credits associated with a course.
5. **Missing Values** :
   - Some columns contain missing values (e.g., `race`, `school_type`).
6. **Potential Use Cases** :
   - Analyzing trends in academic performance, dropout rates, or graduation rates over time.
   - Investigating the impact of quotas or enrollment types on student outcomes.

Data Cleaning and Filtering

The dataset needs to be cleaned based on the following criteria:

1. Filter `admission_period` to include only values between 2006.1 and 2019.2.
2. Remove entries where `enrollment_type` is labeled as "Exempted".
3. Exclude entries where `situation` is "In Progress".

Python Code for Data Cleaning

python

```python
import pandas as pd

# Load the dataset
df = pd.read_csv('stratified_history.csv')

# Step 1: Filter admission_period to include only values between 2006.1 and 2019.2
df_filtered = df[(df['admission_period'] >= 2006.1) & (df['admission_period'] <= 2019.2)]
```

```

# Step 2: Remove entries where enrollment_type is labeled as "Exempted"
df_filtered = df_filtered[df_filtered['enrollment_type'] != 'Exempted']

# Step 3: Exclude entries where situation is "In Progress"
df_filtered = df_filtered[df_filtered['situation'] != 'In Progress']

# Display the first few rows of the cleaned dataset
print(df_filtered.head())

# Save the cleaned dataset to a new CSV file (optional)
df_filtered.to_csv('cleaned_dataset.csv', index=False)
```

Explanation of the Code

1. **Loading the Dataset** :
   - The dataset is loaded using `pd.read_csv()` .
2. **Filtering** `admission_period` :
   - The condition `(df['admission_period'] >= 2006.1) & (df['admission_period'] <= 2019.2)` ensures that only rows with `admission_period` within the specified range are retained.
3. **Removing "Exempted" Entries** :
   - The condition `df_filtered['enrollment_type'] != 'Exempted'` filters out rows where `enrollment_type` is labeled as "Exempted".
4. **Excluding "In Progress" Situations** :
   - The condition `df_filtered['situation'] != 'In Progress'` removes rows where `situation` is "In Progress".
5. **Saving the Cleaned Dataset** :
   - The cleaned dataset is optionally saved to a new CSV file for further analysis.

Resulting DataFrame (`df_filtered`)

The resulting DataFrame, `df_filtered` , contains only the rows that meet the specified criteria. It can now be used for further analysis or to address specific analytical questions.

If you have a specific analytical question, please provide it, and I will use the cleaned dataset to derive an answer along with the corresponding Python code.

What are the key statistical properties (mean, median, variance) of students' final grades for each enrollment period?

qwen-max-latest

To analyze the key statistical properties (mean, median, and variance) of students' final grades (`final_average`) for each `enrollment_period` , we will use the cleaned dataset (`df_filtered`) generated earlier. Below is the step-by-step approach:

Approach

1. **Group Data by Enrollment Period** :
   - Use the `groupby()` function to group the data by `enrollment_period` .
2. **Calculate Statistical Properties** :
   - For each group (i.e., each `enrollment_period`), calculate:
     - **Mean** : Average of `final_average` .
     - **Median** : Middle value of `final_average` when sorted.
     - **Variance** : Measure of how much the values in `final_average` vary from the mean.
3. **Handle Missing Values** :
   - Ensure that missing or null values in `final_average` are excluded during calculations.
4. **Output Results** :
   - Create a summary DataFrame containing the calculated statistics for each `enrollment_period` .
5. **Python Code** :
   - Implement the above steps using Python and Pandas.

Python Code

```python
import pandas as pd

# Load the cleaned dataset (if not already loaded)
# df_filtered = pd.read_csv('cleaned_dataset.csv')

# Ensure 'final_average' is numeric and drop rows with NaN values in 'final_average'
df_filtered['final_average'] = pd.to_numeric(df_filtered['final_average'], errors='coerce')
df_filtered = df_filtered.dropna(subset=['final_average'])

# Group by 'enrollment_period' and calculate mean, median, and variance
grade_stats = df_filtered.groupby('enrollment_period')['final_average'].agg(
    mean_grade='mean',
    median_grade='median',
    variance_grade='var'
).reset_index()

# Display the resulting statistics
print(grade_stats)

# Save the results to a CSV file (optional)
grade_stats.to_csv('grade_statistics_by_enrollment_period.csv', index=False)
```

Explanation of the Code

1. **Data Cleaning** :
   - The column `final_average` is converted to a numeric type using `pd.to_numeric()`, with invalid values coerced to `NaN`.
   - Rows with missing `final_average` values are dropped using `dropna()`.
2. **Grouping and Aggregation** :
   - The dataset is grouped by `enrollment_period`.
   - For each group, the following aggregations are performed:
     - `mean` : Computes the average grade.
     - `median` : Finds the middle value of the grades.
     - `var` : Calculates the variance of the grades.
3. **Output** :
   - The resulting DataFrame contains three columns: `mean_grade`, `median_grade`, and `variance_grade`, along with the corresponding `enrollment_period`.
4. **Saving Results** :
   - The results are optionally saved to a CSV file for further analysis.

Example Output

The output DataFrame (`grade_stats`) might look like this:

ENROLLMENT_PERIOD	MEAN_GRADE	MEDIAN_GRADE	VARIANCE_GRADE
2006.1	7.2	7.5	1.2
2006.2	6.8	7.0	1.5
2007.1	7.0	7.2	1.3
...	...	...	...

Interpretation

- **Mean Grade** : Represents the average performance of students in each enrollment period.
- **Median Grade** : Indicates the central tendency, which is less sensitive to outliers compared to the mean.
- **Variance** : Measures the spread or variability of grades within each enrollment period. A higher variance suggests greater variability in student performance.

If you need further analysis or visualization (e.g., plotting trends over time), please let me know!

copyright © 2025