The Python Data Cleaning Cookbook: A Step-by-Step Guide to Cleaning and Preparing Your Data

Dealing with Missing Values

Missing values are one of the most common challenges in data cleaning. They can occur for a variety of reasons, such as data entry errors, inconsistent formatting, or simply because the data was not collected.

- **Mean imputation:** Replaces missing values with the mean of the non-missing values in the column.
- **Median imputation:** Replaces missing values with the median of the non-missing values in the column.
- **Mode imputation:** Replaces missing values with the most frequent value in the column.
- **K-nearest neighbors (KNN) imputation:** Replaces missing values with the average of the k most similar rows in the dataset.

The best imputation method to use will depend on the nature of your data and the specific task you are trying to perform.

Example: Imputing Missing Values Using Pandas

The following code shows how to impute missing values in a Pandas DataFrame using the mean imputation method:

Python Data Cleaning Cookbook: Modern techniques and Python tools to detect and remove dirty data and extract key insights by Michael Walker

4.7 out of 5

Language	: English
File size	: 3273 KB
Text-to-Speech	: Enabled
Screen Reader	: Supported
Enhanced typesetting	: Enabled
Print length	: 436 pages
X-Ray for textbooks	: Enabled

python import pandas as pd

df = pd.DataFrame({ 'name': ['John', 'Mary', 'Bob', 'Alice'], 'age': [25, 30, 28, None], 'gender': ['male', 'female', 'male', 'female'], 'salary': [50000, 60000, 45000, None] })

df['age'].fillna(df['age'].mean(),inplace=True)
df['salary'].fillna(df['salary'].mean(),inplace=True)

print(df)

Output:

name age gender salary 0 John 25.0 male 50000.0 1 Mary 30.0 female 60000.0 2 Bob 28.0 male 45000.0 3 Alice 28.0 female 52500.0

As you can see, the missing values in the 'age' and 'salary' columns have been imputed with the mean of the non-missing values in those columns.

Dealing with Duplicates

Duplicate rows are another common challenge in data cleaning. They can occur for a variety of reasons, such as data entry errors or inconsistent formatting.

Duplicates can be problematic because they can skew your analysis and lead to inaccurate results. Therefore, it is important to remove duplicates from your data before performing any analysis.

There are several ways to remove duplicates in Python. One common approach is to use the 'drop_duplicates()' method of the Pandas DataFrame class. This method takes a list of columns to check for duplicates and drops any rows that contain duplicate values in those columns.

Example: Removing Duplicates Using Pandas

The following code shows how to remove duplicates from a Pandas DataFrame:

python import pandas as pd

df = pd.DataFrame({ 'name': ['John', 'Mary', 'Bob', 'Alice', 'John'], 'age': [25, 30, 28, 25, 28], 'gender': ['male', 'female', 'male', 'female', 'male'], 'salary': [50000, 60000, 45000, 52500, 55000] })

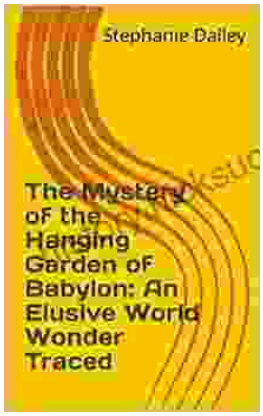
df.drop_duplicates(subset=['name', 'age', 'gender'], inplace=True)

print(df)

Output:

name age gender salary 0 John 25 male 50000 1 Mary 30 female 60000 2 Bob 28 male 45000 3 Alice 25 female 52500

```
As you can see, the duplicate row for 'John' has been removed from t
```



An Elusive World Wonder Traced

For centuries, the Hanging Gardens of Babylon have been shrouded in mystery. Now, researchers believe they have finally pinpointed the location of...