

Start coding or [generate](#) with AI.

Name: Divyesh Singh Rajput NU ID: 002788618

1. Theoretical Foundation

What is Data Cleaning?

Data cleaning, also known as data cleansing or scrubbing, is the process of detecting and correcting (or removing) corrupt or inaccurate records from a dataset. It involves identifying incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replacing, modifying, or deleting this dirty data.

Why is Data Cleaning Critical?

-> Dirty data can lead to inaccurate models in machine learning. Since ML algorithms learn from the given data, the quality of data directly influences the model's performance.

-> High-quality data is crucial for accurate analytics and reliable decision-making.

-> Cleaning data improves its efficiency for processing and analysis, making it faster and easier to derive insights.

Principles and Techniques of Data Cleaning

1. Accuracy: Ensuring that the data correctly represents the real-world entities or events it is supposed to depict.
2. Completeness: Filling in missing values and ensuring no critical data is missing.
3. Consistency: Resolving discrepancies in naming conventions, data formats, and measurements.
4. Timeliness: Updating data regularly and removing outdated information.
5. Validity: Ensuring that the data follows the specified schema and predefined patterns.

Standard Techniques:

-> Handling Missing Values: Techniques like imputation (replacing missing data with statistical estimates), or removing rows/columns with missing values.

-> Filtering Noise and Outliers: Identifying and removing anomalies that deviate significantly from the norm.

-> Error Correction: Rectifying wrong data entries and typos. Data Transformation: Standardizing and normalizing data to bring it into a consistent format.

Challenges in Data Cleaning:

-> Dealing with Large Datasets: As the volume of data increases, it becomes more challenging to clean and manage the data efficiently.

-> Diverse Data Formats: Data coming from various sources may have different formats, making integration and cleaning a complex task.

-> Noisy Data: Data with a lot of irregularities and inconsistencies can be challenging to clean without losing valuable information.

-> Subjectivity in Cleaning Decisions: Deciding how to handle outliers or missing data can be subjective and vary depending on the specific use case.

This theoretical foundation sets the stage for understanding the crucial role of data cleaning in data science and machine learning. It highlights the need for meticulous and thoughtful approaches to improve data quality, which is fundamental for any successful data-driven project.

2. Introduction to the problem and dataset

Dataset Overview: The NYC Airbnb dataset, commonly available on platforms like huggingface, contains information about Airbnb listings in New York City. This dataset generally includes details such as listing ID, name, host ID, host name, neighborhood groups, latitude, longitude, room type, price, minimum nights, number of reviews, last review, reviews per month, calculated host listings count, and availability.

Context and Relevance

Why Choose this Dataset?

The dataset provides a comprehensive set of features which makes it ideal for exploring various data cleaning techniques. Airbnb is a widely recognized platform, and the dataset represents real-world issues in data management. It is known to contain a variety of data quality issues, such as missing values, outliers, and inconsistencies, making it a suitable choice for demonstrating data cleaning.

Relevance in Data Science: Cleaned data can be used for predictive modeling, like forecasting rental prices or understanding factors influencing Airbnb rental popularity. It allows for in-depth analysis of market trends, customer preferences, and geographic distribution of rentals.

Urban Planning and Research: Insights from the data can assist in urban planning and tourism management research.

Objectives of Cleaning

Aim of Data Cleaning on NYC Airbnb Dataset:

Improving Data Quality: Address and correct issues like missing values, outliers, and inaccuracies to enhance the dataset's reliability.

Data Standardization: Ensure data is in a uniform format, making it easier for analysis and modeling.

Preparation for Analysis: Clean data to facilitate accurate and efficient data analysis and machine learning model development.

3. Analyzing the worked Example

```
pip install datasets
```

```
Collecting datasets
  Downloading datasets-2.16.1-py3-none-any.whl (507 kB)
    507.1/507.1 kB 9.7 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets) (3.13.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.23.5)
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (10.0.1)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-packages (from datasets) (0.6)
Collecting dill<0.3.8,>=0.3.0 (from datasets)
  Downloading dill-0.3.7-py3-none-any.whl (115 kB)
    115.3/115.3 kB 17.8 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (1.5.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from datasets) (3.4.1)
Collecting multiprocessing (from datasets)
  Downloading multiprocessing-0.70.15-py310-none-any.whl (134 kB)
    134.8/134.8 kB 20.5 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]<=2023.10.0,>=2023.1.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (2023.10.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.9.1)
Requirement already satisfied: huggingface-hub>=0.19.4 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.20.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (23.2.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.9.4)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.4.1)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.19.4->datasets) (4.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets) (2023.7.22)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2023.3.post1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->datasets) (1.16.0)
Installing collected packages: dill, multiprocessing, datasets
Successfully installed datasets-2.16.1 dill-0.3.7 multiprocessing-0.70.15
```

```
from datasets import load_dataset
```

```
dataset = load_dataset("gradio/NYC-Airbnb-Open-Data")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as :
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Downloading readme: 100%                25.0/25.0 [00:00<00:00, 155B/s]
Downloading data: 100%                  7.08M/7.08M [00:00<00:00, 7.74MB/s]
Generating train split: 48895/0 [00:00<00:00, 87867.16 examples/s]
```

```
import pandas as pd
df = pd.DataFrame(dataset['train'])
df.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nig
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK!	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   id                                    48895 non-null  int64
1   name                                48879 non-null  object
2   host_id                             48895 non-null  int64
3   host_name                           48874 non-null  object
4   neighbourhood_group                 48895 non-null  object
5   neighbourhood                       48895 non-null  object
6   latitude                           48895 non-null  float64
7   longitude                           48895 non-null  float64
8   room_type                           48895 non-null  object
9   price                               48895 non-null  int64
10  minimum_nights                      48895 non-null  int64
11  number_of_reviews                   48895 non-null  int64
12  last_review                         38843 non-null  object
13  reviews_per_month                  38843 non-null  float64
14  calculated_host_listings_count      48895 non-null  int64
15  availability_365                    48895 non-null  int64
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

we can see that the DataFrame has 48,895 rows (entries) and 16 columns. The columns have different data types, including int64, float64, and object. We can also see that some columns have missing values ('name', 'host_name', 'last_review', and 'reviews_per_month'), as indicated by the difference between the total number of entries (48,895) and the number of non-null values for those columns.

Changing datatype

We can see that 'last_review' column contains dates, but its type is object. There is a need to change it to a datetime data type. It'll allow to perform calculations on the dates.

```
df['last_review'] = pd.to_datetime(df['last_review'])
```

```
#changing data type of a few columns so that they can be visualization and EDA purpose later.
df['neighbourhood_group'] = df['neighbourhood_group'].astype('category')
df['neighbourhood'] = df['neighbourhood'].astype('category')
```

#Missing Data

#Missing data can cause problems in your analysis, such as biased results, incorrect statistical inferences, and errors in machine learning

```
df.isnull().sum()
```

```
id                0
name              16
host_id           0
host_name         21
neighbourhood_group 0
neighbourhood     0
latitude          0
longitude         0
room_type         0
```

```

price                0
minimum_nights       0
number_of_reviews    0
last_review         10052
reviews_per_month    10052
calculated_host_listings_count  0
availability_365     0
dtype: int64

```

Handling Missing Data: The primary purpose here is to handle missing data in the 'name' column of the DataFrame. Instead of leaving missing values as NaN, which can be problematic for data analysis and machine learning models, they are replaced with a placeholder string 'N/A'.

```
df.fillna({'name': 'N/A'}, inplace=True)
```

```

#drop unnecessary columns (columns which you feel like are not going to be used in future)
df.drop(['id','host_name','last_review'], axis=1, inplace=True)

```

✓ Data Standardization in Data Cleaning

Converting all values of 'private' to 'private room' in the 'room_type' column of your DataFrame is indeed a form of data cleaning. This specific task falls under the category of data standardization, which is a key aspect of data cleaning.

Consistency: Standardizing values in your dataset ensures consistency, which is a crucial aspect of data quality. Inconsistent data can lead to inaccuracies in analysis and can adversely affect the outcomes of data-driven processes or models.

Uniformity: By changing 'private' to 'private room', you are making the dataset more uniform. This uniformity is important, especially when the data will be used for analysis or in machine learning models, where consistent categories are necessary for accurate results.

Data Integrity: Standardization helps maintain the integrity of the dataset. It ensures that similar data points are represented in the same way, making it easier to aggregate, analyze, and visualize data.

Improved Data Analysis: With standardized data, performing analysis becomes more straightforward. For instance, in your case, querying the dataset for all private rooms will yield accurate results without the risk of missing out on entries labeled merely as 'private'.

```
df['room_type'] = df['room_type'].replace('private', 'private room')
```

✓ 4. Utilizing ChatGPT for In-Depth Exploration

```

#Question: What are the most effective strategies for dealing with missing data in the 'reviews per month' or 'host name' columns in the dataset?

...
Response: Dealing with missing data effectively requires a tailored approach, especially for different types of data like 'reviews per month' and 'host name'.

Handling Missing Data in 'Reviews per Month'
This column is likely numeric, representing the average number of reviews a listing receives per month. Missing values here could mean that a listing has no reviews or that the data is incomplete.

Imputation with Zero: If a listing has no reviews, it's logical to impute missing values with zero, indicating no reviews have been received.

Mean/Median Imputation: If you assume that the absence of data is random, replacing missing values with the mean or median reviews per month might be appropriate.

Using a Marker Value: In some analyses, it might be helpful to distinguish listings with no reviews from those with missing data. Using a specific value like -1 or 'No Reviews' could be useful.

Advanced Techniques: Techniques like regression imputation, where you predict missing values based on other variables in your dataset, could also be considered.

Handling Missing Data in 'Host Name'
The 'host name' is a categorical column, and missing values here can be trickier to handle. Strategies include:

Imputation with a Placeholder: Replace missing values with a placeholder value like 'Unknown' or 'N/A'. This maintains the integrity of the data while marking the missing information.

Mode Imputation: You could replace missing names with the most common host name in the dataset, although this might not be meaningful as it could mask distinct categories.

Data Deletion: If the number of missing values in 'host name' is small and if this column is not critical for your analysis, you might consider deleting the rows with missing data.

Cross-Referencing: If you have other datasets or information sources, you might cross-reference to fill in missing host names. This requires additional data and might not always be feasible.

General Considerations
Understand the Data: Before choosing a strategy, understand why data might be missing. Is it missing at random, or is there a pattern? This can influence which strategy is most appropriate.
Impact on Analysis: Consider how each approach might impact your subsequent analysis. For instance, imputing with the mean might mask the distribution of reviews.
Dataset Size and Missingness Proportion: The size of your dataset and the proportion of missing values should also guide your strategy.
...

```

```
'\nResponse: Dealing with missing data effectively requires a tailored approach, especially for different types of data like 'reviews per month' (numeric) and 'host name' (categorical/textual). Here are strategies for each:\n\nHandling Missing Data in 'Reviews per Month'\nThis column is likely numeric, representing the average number of reviews a listing receives per month. Missing values here could mean that the listing has not received any reviews. Strategies include:\n\nImputation with Zero: If a listing has no reviews, imputing zero is reasonable.\n\nQuestion: What could be the potential impact of not cleaning the dataset properly, such as leaving in extreme outliers in price or not\n...\nResponse: Not cleaning the dataset properly, particularly not addressing issues like extreme outliers or missing values, can have several impacts:\n\nSkewed Analysis and Misleading Results: Extreme outliers, especially in key variables like price, can skew the results of your analysis.\n\nReduced Model Accuracy and Reliability: In machine learning models, unaddressed outliers and missing values can reduce the accuracy and reliability of predictions.\n\nCompromised Decision Making: Decisions based on improperly cleaned data can be flawed. In the context of Airbnb data, this could lead to incorrect business strategies.\n\nQuestion: How to approach inconsistencies in textual data, like varying formats in the 'name' or 'room_type' columns?\n...\nResponse: Standardization of Text Formats:\n\nCase Normalization: Convert all text to a consistent case (e.g., all lower case or upper case) to avoid discrepancies like 'Private Room', 'private room', and 'PRIVATE ROOM' being treated as different categories.\n\nTrimming Spaces: Remove leading and trailing spaces, which can cause unique values to be treated as different (e.g., 'private room ' vs. 'private room').\n\nConsolidating Variants: Standardize variations of the same term. For example, 'suite' and 'Suite' should be treated as the same category.\n\nHandling Typos and Spelling Errors: Implement a spell-checking algorithm to correct obvious typos in text fields.\n\nFuzzy Matching: Use fuzzy matching techniques to identify and correct near-identical variants of the same words (e.g., 'privat room' vs. 'private room').\n\nQuestion: How to handle missing data in a dataset?\n...\nResponse: Handling missing data effectively requires a tailored approach, especially for different types of data like 'reviews per month' (numeric) and 'host name' (categorical/textual). Here are strategies for each:\n\nHandling Missing Data in 'Reviews per Month'\nThis column is likely numeric, representing the average number of reviews a listing receives per month. Missing values here could mean that the listing has not received any reviews. Strategies include:\n\nImputation with Zero: If a listing has no reviews, imputing zero is reasonable.\n\nQuestion: What could be the potential impact of not cleaning the dataset properly, such as leaving in extreme outliers in price or not\n...\nResponse: Not cleaning the dataset properly, particularly not addressing issues like extreme outliers or missing values, can have several significant impacts:\n\nSkewed Analysis and Misleading Results: Extreme outliers, especially in key variables like price, can skew the results of your analysis. They can affect statistical calculations like mean and standard deviation, leading to misleading conclusions.\n\nReduced Model Accuracy and Reliability: In machine learning models, unaddressed outliers and missing values can reduce the accuracy and reliability of predictions. The model might overfit or underfit, failing to generalize well to new, unseen data.\n\nCompromised Decision Making: Decisions based on improperly cleaned data can be flawed. In the context of Airbnb data, this could lead to incorrect business strategies.
```

✓ Thank you

MIT License
Copyright (c) 2024 Divyesh Rajput

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.