

[00:00-15:16](#)

Data Cleaning Techniques in Pandas: A Comprehensive Guide

Introduction

Data cleaning is a crucial step in the data analysis process. In this article, we will explore various techniques for data cleaning using the Pandas library in Python. We will walk through the process of downloading a dataset, loading it into a Pandas DataFrame, and applying different data cleaning techniques to ensure the dataset is ready for further analysis.

Downloading and Loading the Dataset

To follow along with this tutorial, you can download the dataset from the provided link in the video description. Once downloaded, we can load the dataset into a Pandas DataFrame using the `read_csv()` function. This function reads the CSV file and stores the data in a tabular format.

```
import pandas as pd

# Load the dataset into a Pandas DataFrame
df = pd.read_csv('dataset.csv')
```

Examining the Dataset

Before diving into data cleaning, let's first examine the dataset to get a better understanding of its structure. The dataset contains various independent variables such as area type, location, size, and total square foot area. The dependent variable we are trying to predict is the price of the properties listed in the dataset.

```
# Display the shape of the dataset (number of
rows and columns)
print(df.shape)

# Display a sample of the dataset
print(df.head())
```

Handling Missing Values

Dealing with missing values is an important part of data cleaning. To identify missing values in the dataset, we can use the `isnull()` function, which returns a boolean DataFrame indicating the presence of missing values. We can then use the `dropna()` function to remove the rows containing missing values.

```
# Check for missing values
print(df.isnull().sum())

# Drop rows with missing values
df.dropna(inplace=True)
```

Data Transformation and Cleaning

In this section, we will perform further data cleaning and transformation operations on the dataset.

Dropping Unnecessary Columns

Some columns may not contribute significantly to the analysis or modeling process. In such cases, we can drop those columns from the DataFrame using the `drop()` function.

```
# Drop unnecessary columns
df.drop('availability', 'society', 'area type',
axis=1, inplace=True)
```

Handling Categorical Variables

In our dataset, the "size" column contains values such as "2 BHK" or "4 BHK." To handle these values, we can create a new column called "BHK" and extract the numeric part from the "size" column using string manipulation techniques.

```
# Create a new column for the number of bedrooms
(BHK)
df['BHK'] = df['size'].apply(lambda x:
int(x.split()[0]))
```

Handling Irregularities in the Dataset

Sometimes, the dataset may contain irregularities or errors that need to be addressed. For example, the "total square foot" column may include ranges or units other than square feet. We can apply specific data cleaning functions to handle such cases.

```

# Function to convert range values into average
numbers
def convert_range_to_avg(value):
    if '-' in value:
        tokens = value.split('-')
        return (float(tokens[0]) + float(tokens[1]))
    / 2
    elif 'sqft' in value:
        return float(value.replace('sqft', ''))
    else:
        return None

# Apply the function to the \"total square foot\"
column
df['total_sqft'] = df['total
sqft'].apply(convert_range_to_avg)

```

Conclusion

Data cleaning is a crucial step in preparing a dataset for analysis. In this article, we covered various data cleaning techniques using the Pandas library. We explored methods to handle missing values, transform and clean data, and remove unnecessary columns. By applying these techniques, we can ensure that our dataset is ready for further analysis and modeling.

In the next article, we will dive into feature engineering and dimensionality reduction techniques, which will further enhance our dataset for advanced analysis. Stay tuned!