Exploring Data With Pandas

Data exploration is an important step in the data analysis process. It involves investigating the dataset to gain insights into its characteristics, such as its size, shape, and content. Python is a powerful programming language that provides a variety of tools for data exploration. In this tutorial, we will cover the following topics:

1. Importing the dataset
2. Understanding the dataset
3. Visualizing the dataset
4. Cleaning the dataset
5. Handling missing data

Let's get started!

1. Importing the dataset

The first step in data exploration is to import the dataset into Python. Python provides a variety of libraries for reading different types of datasets, such as CSV, Excel, JSON, HTML, and SQL. In this tutorial, we will use the Pandas library, which provides a powerful data structure called DataFrame that allows us to manipulate and analyze the data easily.

Here's an example of how to import a CSV dataset using Pandas:

```python
import pandas as pd

# Reading a CSV file
df = pd.read_csv("path/to/file.csv")
```

Filepaths can be relative or absolute. If relative, the file will be read in relation to the current working directory, which typically is the directory in which the Python script is located. If absolute, the file will be read in relation to the root directory. The `read_csv()` function returns a `DataFrame` object.

The `read_csv()` function takes a number of optional arguments. For a complete list of arguments, refer to the official `pandas` documentation.

Reading in Data from a URL

The `read_csv()` function can also be used to read data from a URL. The following example demonstrates how to read in CSV data from a URL:

```python
import pandas as pd

# Reading a CSV file from a URL
df = pd.read_csv(
    "https://raw.githubusercontent.com/alexeygrigorev/datasets/master/AB_NYC_2019.csv"
)
```

The above example acquires the New York City Airbnb Open Data from Kaggle. The data contains information about Airbnb listings in New York City. The data is stored in a CSV file on the web. The `read_csv()` function is used to read the data from the URL and store it in a DataFrame.

2. Understanding the dataset

Once we have imported the dataset, we need to understand its characteristics, such as its size, shape, and content. Here are some useful methods for exploring the dataset:

- `df.head(n)` : displays the first n rows of the dataset (by default, n=5)
- `df.tail(n)` : displays the last n rows of the dataset (by default, n=5)
- `df.shape` : displays the number of rows and columns in the dataset
- `df.describe()` : displays statistical information about the dataset, such as mean, standard deviation, and quartiles
- `df.info()` : displays information about the dataset, such as column names, data types, and missing values
- `df["column_name"].value_counts()` : displays the unique values and their counts in a column
- `df["column_name"].unique()` : displays the unique values in a column
- `df["column_name"].nunique()` : displays the number of unique values in a column

Here's an example of how to use these methods:

```python
df.head()
```

```python
df.tail()
```

```python
df.shape
```

```python
df.describe()
```

```python
df.info()
```

```
df["neighbourhood"].value_counts()
```

```
df["room_type"].unique()
```

```
df["neighbourhood"].nunique()
```

3. Visualizing the dataset

Visualizing the dataset can help us gain insights into its characteristics and relationships between variables. Python provides a variety of libraries for data visualization, such as Matplotlib, Seaborn, and Plotly. In this tutorial, we will use Matplotlib, which is a widely used library for creating basic visualizations.

Here's an example of how to create a histogram using Matplotlib:

```
import matplotlib.pyplot as plt
import numpy as np

plt.hist(df["price"], bins=np.linspace(0, 1000, 100))
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.title("Histogram of Price")
plt.show()
```

This code creates a histogram of a column named column_name in the dataset. You can replace column_name with the name of your own column.

Alternatively, Pandas provides built-in methods for plotting. For example, creating a histogram using Pandas is as simple as:

```
df.hist(column="price", bins=np.linspace(0, 1000, 100))
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.title("Histogram of Price")
```

4. Cleaning the dataset

Before analyzing the dataset, we need to clean it by removing irrelevant or redundant information, dealing with missing data, and transforming data if necessary. Here are some common data cleaning tasks:

- Removing duplicates: `df.drop_duplicates()`
- Removing irrelevant columns: `df.drop(['column_name'], axis=1)`
- Renaming columns: `df.rename(columns={'old_name': 'new_name'})`
- Transforming data: `df['new_column'] = df['old_column'] * 2`

Here's an example of how to remove duplicates and irrelevant columns:

```
df.drop_duplicates(subset=["name"])
```

```
df.drop(["host_id"], axis=1)
```

5. Handling missing data

Handling missing data is a crucial step in data exploration as missing data can cause errors and bias in analysis. There are different strategies for handling missing data, such as removing missing values, imputing missing values, or using advanced techniques like data interpolation. Pandas provides various functions for handling missing data, including:

- `df.isnull()`: returns a Boolean DataFrame indicating which values are missing
- `df.dropna()`: removes rows or columns with missing values
- `df.fillna(value)`: fills missing values with a specified value
- `df.interpolate()`: performs linear interpolation to fill missing values

Here's an example of how to fill missing values:

```
df["reviews_per_month"].fillna(0)
```

This code fills missing values in a column named `column_name` with the mean of the column. You can replace `column_name` with the name of your own column.

In conclusion, data exploration is a crucial step in the data analysis process, and Python provides a powerful set of tools for exploring, visualizing, cleaning, and handling missing data. By following these steps, you can gain insights into your dataset and prepare it for further analysis.

Project: Exploring Airbnb listings in New York City

The goal of this project is to explore a dataset of Airbnb listings in New York City using Pandas. We will perform various data exploration tasks to gain insights into the dataset and answer some questions about Airbnb in NYC.

Dataset

We will be using an unclean version of the New York City Airbnb Open Data dataset from Kaggle. The data contains information about Airbnb listings in New York City. The data is stored in a CSV file on the GitHub repository under the `data` folder. The `read_csv()` function is used to read the data from the URL and store it in a DataFrame.

```python
import pandas as pd

# Reading a CSV file from a URL
df = pd.read_csv(
    "../../data/AB_NYC_2019_unclean.csv"  # Replace with your own path if necessary.
)
```

Tasks

The tasks below are merely suggestions for exploring the dataset. Feel free to explore and analyze the dataset in any way you like.

1. Import the dataset into a Pandas DataFrame.
2. Explore the dataset, for example using the `head()`, `tail()` and `info()` methods.
3. Explore the basic statistics of the dataset using the `describe()` method.
4. Clean the dataset. For example:
    - Remove unnecessary columns
    - Handle missing data
    - Remove duplicate records
    - Fix typos
    - Deal with outliers
    - Ensure the data types are correct
5. Visualize the distribution of prices using a histogram.
6. Identify the top 10 neighborhoods with the highest number of listings and create a bar chart to visualize the results.
7. Analyze the relationship between price and availability by creating a scatter plot.
8. Use the method to calculate the average price by neighborhood and room type.
9. Create a heatmap to visualize the availability of listings by neighborhood.
    - Hint: Use the `seaborn` library to create a heatmap. You can install the library using the command `pip install seaborn`. For more information, refer to the official documentation.
10. Identify the top 10 hosts with the most listings and create a bar chart to visualize the results.

11. Analyze the relationship between the number of reviews and price by creating a scatter plot.