pandas

August 31, 2024

Getting Familiar with Pandas:

Pandas is a powerful and flexible Python library used for data manipulation and analysis. It is built on top of NumPy, and its key data structures, DataFrame and Series, are specifically designed for handling structured data like tabular datasets, which makes Pandas an indispensable tool for data scientists.

1. Understanding Pandas Data Structures: DataFrames and Series

a. Series:

A Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, etc.). The labels, also known as the index, can be used to access and manipulate the data in a Series. It's similar to a column in a spreadsheet or a database table. Key Characteristics:

Indexing: Like NumPy arrays, Series can be indexed by position. Additionally, they can also be indexed by a label, which is unique to each element. Homogeneous Data: A Series holds data of a single type. Automatic Indexing: When a Series is created without specifying an index, Pandas automatically assigns a default integer index starting from 0. b. DataFrame:

A DataFrame is a two-dimensional labeled data structure with columns of potentially different types. Think of it as a table in a database, a spreadsheet, or a dictionary of Series objects. Key Characteristics:

Labeled Axes: A DataFrame has both row and column labels (index and columns). Heterogeneous Data: Unlike a Series, a DataFrame can hold data of multiple types across different columns. Flexible Data Alignment: DataFrame operations align on both row and column labels, which simplifies handling missing data and merging datasets. Data Handling: DataFrames are excellent for handling and manipulating large datasets, making them ideal for data wrangling tasks.

[3]:
```
pip install pandas
```

```
Collecting pandas
  Downloading pandas-2.2.2-cp312-cp312-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.26.0 in
c:\users\sivasai\appdata\local\programs\python\python312\lib\site-packages (from
pandas) (2.1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\sivasai\appdata\local\programs\python\python312\lib\site-packages (from
pandas) (2.9.0.post0)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2024.1-py2.py3-none-any.whl.metadata (22 kB)
```

```
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in
c:\users\sivasai\appdata\local\programs\python\python312\lib\site-packages (from
python-dateutil>=2.8.2->pandas) (1.16.0)
Downloading pandas-2.2.2-cp312-cp312-win_amd64.whl (11.5 MB)
   ---------------------------------------- 0.0/11.5 MB ? eta -:--:--
   ---------------------------------------- 0.0/11.5 MB 1.4 MB/s eta 0:00:09
   ---------------------------------------- 0.1/11.5 MB 1.3 MB/s eta 0:00:09
    --------------------------------------- 0.2/11.5 MB 2.0 MB/s eta 0:00:06
   - -------------------------------------- 0.5/11.5 MB 3.2 MB/s eta 0:00:04
   --- ------------------------------------ 1.0/11.5 MB 4.8 MB/s eta 0:00:03
   ----- ---------------------------------- 1.6/11.5 MB 6.5 MB/s eta 0:00:02
   -------- ------------------------------- 2.4/11.5 MB 8.0 MB/s eta 0:00:02
   ---------- ----------------------------- 3.1/11.5 MB 9.0 MB/s eta 0:00:01
   ----------- ---------------------------- 3.4/11.5 MB 9.3 MB/s eta 0:00:01
   ----------- ---------------------------- 3.4/11.5 MB 9.3 MB/s eta 0:00:01
   ---------------- ----------------------- 4.8/11.5 MB 9.8 MB/s eta 0:00:01
   ------------------- -------------------- 5.6/11.5 MB 10.5 MB/s eta 0:00:01
   ---------------------- ----------------- 6.3/11.5 MB 11.0 MB/s eta 0:00:01
   ------------------------ --------------- 7.2/11.5 MB 11.6 MB/s eta 0:00:01
   --------------------------- ------------ 7.9/11.5 MB 12.1 MB/s eta 0:00:01
   ----------------------------- ---------- 8.6/11.5 MB 12.0 MB/s eta 0:00:01
   ------------------------------- -------- 9.4/11.5 MB 12.2 MB/s eta 0:00:01
   --------------------------------- ------ 9.8/11.5 MB 12.1 MB/s eta 0:00:01
   ---------------------------------- ----- 10.3/11.5 MB 12.4 MB/s eta 0:00:01
   ------------------------------------ --- 10.8/11.5 MB 13.9 MB/s eta 0:00:01
   ---------------------------------------  11.5/11.5 MB 14.2 MB/s eta 0:00:01
   ---------------------------------------- 11.5/11.5 MB 13.6 MB/s eta 0:00:00
Downloading pytz-2024.1-py2.py3-none-any.whl (505 kB)
   ---------------------------------------- 0.0/505.5 kB ? eta -:--:--
   ---------------------------------------- 505.5/505.5 kB 15.5 MB/s eta 0:00:00
Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
   ---------------------------------------- 0.0/345.4 kB ? eta -:--:--
   ---------------------------------------- 345.4/345.4 kB 20.9 MB/s eta 0:00:00
Installing collected packages: pytz, tzdata, pandas
Successfully installed pandas-2.2.2 pytz-2024.1 tzdata-2024.1
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Creating DataFrames and Series a. Creating a Series: You can create a Pandas Series from a variety of data sources, such as lists, NumPy arrays, or dictionaries.

[5]:
```
import pandas as pd
```

```
# Creating a Series from a list
data_list = [10,20,30,40,50,60,70,80]
series_from_list = pd.Series(data_list)

# Creating a Series from a dictionary
data_dict = {'a': 11, 'b': 12, 'c': 13, 'd':14, 'e':15}
series_from_dict = pd.Series(data_dict)

print("Series from List:\n", series_from_list)
print("Series from Dictionary:\n", series_from_dict)
```

```
Series from List:
 0    10
1    20
2    30
3    40
4    50
5    60
6    70
7    80
dtype: int64
Series from Dictionary:
 a    11
b    12
c    13
d    14
e    15
dtype: int64
```

b. Creating a DataFrame: A Pandas DataFrame can be created from various data sources such as lists of lists, dictionaries of lists, NumPy arrays, or directly from external data files like CSVs.

[20]:
```
# Creating a DataFrame from a dictionary of lists
data_dict = {
    'Name': ['siva', 'lokesh', 'freak', 'yesh'],
    'Age': [25, 30, 56, 35],
    'City': ['vzm', 'visaka', 'kadapa', 'nellor']
}
data_dict= pd.DataFrame(data)

print("DataFrame from Dictionary of Lists:\n", data_dict)
```

```
DataFrame from Dictionary of Lists:
      Name  Age    City
0    siva   25     vzm
1  lokesh   30  visaka
2   freak   56  kadapa
```

```
3    yesh   35  nellor
```

[11]:
```
#From a CSV File
df_from_csv = pd.read_csv('C:/Users/sivasai/Downloads/Titanic-Dataset (2).csv')

print("DataFrame from CSV:\n", df_from_csv)
```

```
DataFrame from CSV:
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
5              6         0       3
6              7         0       1
7              8         0       3
8              9         1       3
9             10         1       2
10            11         1       3
11            12         1       1
12            13         0       3
13            14         0       3
14            15         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
5                                    Moran, Mr. James    male   NaN      0
6                             McCarthy, Mr. Timothy J    male  54.0      0
7                      Palsson, Master. Gosta Leonard    male   2.0      3
8   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                 Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1
10                    Sandstrom, Miss. Marguerite Rut  female   4.0      1
11                           Bonnell, Miss. Elizabeth  female  58.0      0
12                     Saundercock, Mr. William Henry    male  20.0      0
13                        Andersson, Mr. Anders Johan    male  39.0      1
14                Vestrom, Miss. Hulda Amanda Adolfina  female  14.0      0

    Parch            Ticket     Fare Cabin Embarked
0       0         A/5 21171   7.2500   NaN        S
1       0          PC 17599  71.2833   C85        C
2       0  STON/O2. 3101282   7.9250   NaN        S
3       0            113803  53.1000  C123        S
4       0            373450   8.0500   NaN        S
```

```
5       0            330877   8.4583   NaN        Q
6       0             17463  51.8625   E46        S
7       1            349909  21.0750   NaN        S
8       2            347742  11.1333   NaN        S
9       0            237736  30.0708   NaN        C
10      1          PP 9549  16.7000    G6        S
11      0            113783  26.5500  C103        S
12      0         A/5. 2151   8.0500   NaN        S
13      5            347082  31.2750   NaN        S
14      0            350406   7.8542   NaN        S
```

[22]:
```
# Creating a DataFrame from a list of lists
data_list = [
    ['siva', 25,'vzm'],['lokesh',30,'visaka'],[ 'freak',56,'kadapa'],
    ['yesh', 35,'nellor']
]
df_from_list = pd.DataFrame(data_list, columns=['Name', 'Age', 'City'])

print("DataFrame from List of Lists:\n", df_from_list)
```

```
DataFrame from List of Lists:
      Name  Age    City
0    siva   25     vzm
1  lokesh   30  visaka
2   freak   56  kadapa
3    yesh   35  nellor
```

Common Operations in DataFrames a. Selecting Data:

Data selection in Pandas can be done using column names, row indices, or conditions.

[31]:
```
# Selecting a single column
data_list = {
    'Name': ['siva', 'lokesh', 'freak', 'yesh'],
    'Age': [25, 30, 56, 35],
    'City': ['vzm', 'visaka', 'kadapa', 'nellor']
}
df_from_dict=pd.DataFrame(data_list)
age_column = df_from_dict['Age']

# Selecting multiple columns
name_age_columns = df_from_dict[['Name', 'Age']]

# Selecting rows by index
first_row = df_from_dict.iloc[0]  # By position
first_row_label = df_from_dict.loc[0]  # By label if available

print("Age Column:\n", age_column)
print("Name and Age Columns:\n", name_age_columns)
```

```
print("First Row (by position):\n", first_row)
```

```
Age Column:
 0    25
1    30
2    56
3    35
Name: Age, dtype: int64
Name and Age Columns:
      Name  Age
0    siva   25
1  lokesh   30
2   freak   56
3    yesh   35
First Row (by position):
 Name    siva
Age       25
City     vzm
Name: 0, dtype: object
```

Filtering Rows:

Rows can be filtered based on conditions. This is particularly useful for data cleaning and preparation.

[32]:
```
age_filtered_df = df_from_dict[df_from_dict['Age'] > 28]

print("Rows where Age > 28:\n", age_filtered_df)
```

```
Rows where Age > 28:
      Name  Age    City
1  lokesh   30  visaka
2   freak   56  kadapa
3    yesh   35  nellor
```

Modifying Data:

Data in a DataFrame can be modified by assigning new values to specific elements, rows, or columns.

[34]:
```
# Modifying a single element
df_from_dict.at[0, 'Age'] = 26

# Modifying a whole column
df_from_dict['Age'] = df_from_dict['Age'] + 1  # Increment age by 1

print("Modified DataFrame:\n", df_from_dict)
```

```
Modified DataFrame:
      Name  Age    City
0    siva   27     vzm
```

```
1  lokesh   31  visaka
2   freak   57  kadapa
3    yesh   36  nellor
```

Data Handling with Pandas: o Write a Python program to demonstrate data handling using Pandas. Focus on tasks like reading data from files, handling missing data, and transforming data. o Practice using Pandas functions to clean and preprocess data, such as handling missing values, removing duplicates, and data type conversions.

Pandas is a powerful tool for handling and preprocessing data in Python. This section will guide you through various tasks such as reading data from files, handling missing data, and transforming data. We'll also cover data cleaning and preprocessing techniques like removing duplicates and converting data types.

1. Reading Data from Files Pandas provides convenient functions for reading data from various file formats, such as CSV, Excel, and SQL databases. The most commonly used function is pd.read_csv() for reading CSV files.

[35]:
```
import pandas as pd

# Reading data from a CSV file
df = pd.read_csv('C:/Users/sivasai/Downloads/Titanic-Dataset (2).csv')

# Displaying the first few rows of the DataFrame
print("First few rows of the DataFrame:\n", df.head())
```

```
First few rows of the DataFrame:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

Handling Missing Data Missing data is a common issue in datasets. Pandas provides several methods to handle missing data, such as filling missing values or dropping rows/columns that

contain them.

[36]:
```
# Handling missing data by filling with a specific value
df_filled = df.fillna(0)

# Dropping rows with missing data
df_dropped_rows = df.dropna()

# Dropping columns with missing data
df_dropped_columns = df.dropna(axis=1)

print("DataFrame with missing values filled:\n", df_filled.head())
print("DataFrame with rows with missing values dropped:\n", df_dropped_rows.
 ↪head())
print("DataFrame with columns with missing values dropped:\n",␣
 ↪df_dropped_columns.head())
```

```
DataFrame with missing values filled:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500     0        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250     0        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500     0        S
DataFrame with rows with missing values dropped:
     PassengerId  Survived  Pclass  \
1              2         1       1
3              4         1       1
6              7         0       1
10            11         1       3
11            12         1       1

                                                 Name     Sex   Age  SibSp  \
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
```

```
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
6                             McCarthy, Mr. Timothy J    male  54.0      0
10                   Sandstrom, Miss. Marguerite Rut  female   4.0      1
11                          Bonnell, Miss. Elizabeth  female  58.0      0

    Parch    Ticket     Fare Cabin Embarked
1       0  PC 17599  71.2833   C85        C
3       0    113803  53.1000  C123        S
6       0     17463  51.8625   E46        S
10      1   PP 9549  16.7000    G6        S
11      0    113783  26.5500  C103        S
DataFrame with columns with missing values dropped:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex  SibSp  Parch  \
0                             Braund, Mr. Owen Harris    male      1      0
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female      1      0
2                             Heikkinen, Miss. Laina  female      0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female      1      0
4                           Allen, Mr. William Henry    male      0      0

             Ticket     Fare Embarked
0         A/5 21171   7.2500        S
1          PC 17599  71.2833        C
2  STON/O2. 3101282   7.9250        S
3            113803  53.1000        S
4            373450   8.0500        S
```

Transforming Data Data transformation is an essential step in data preprocessing. This includes operations like changing data types, renaming columns, and applying functions to columns.

[43]:
```
# Renaming columns
df_renamed = df.rename(columns={'OldName': 'NewName'})

print("DataFrame with renamed columns:\n", df_renamed.head())
```

```
DataFrame with renamed columns:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
```

```
                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

Data Cleaning and Preprocessing Data cleaning involves removing duplicates, converting data types, and other preprocessing tasks to prepare the data for analysis.

[46]:
```
# Removing duplicates
df_no_duplicates = df.drop_duplicates()

# Converting data types
df['Fare'] = pd.to_datetime(df['Fare'])

print("DataFrame with duplicates removed:\n", df_no_duplicates.head())
print("DataFrame with converted data types:\n", df.dtypes)
```

```
DataFrame with duplicates removed:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket                        Fare Cabin Embarked
0      0         A/5 21171 1970-01-01 00:00:00.000000007   NaN        S
1      0          PC 17599 1970-01-01 00:00:00.000000071   C85        C
2      0  STON/O2. 3101282 1970-01-01 00:00:00.000000007   NaN        S
3      0            113803 1970-01-01 00:00:00.000000053  C123        S
4      0            373450 1970-01-01 00:00:00.000000008   NaN        S
```

```
DataFrame with converted data types:
 PassengerId              int64
Survived                 int64
Pclass                   int64
Name                    object
Sex                     object
Age                    float64
SibSp                    int64
Parch                    int64
Ticket                  object
Fare           datetime64[ns]
Cabin                   object
Embarked                object
dtype: object
```

Data Analysis with Pandas: o Use Pandas to perform data analysis, including generating summary statistics, grouping data, and applying aggregate functions. o Explore advanced data manipulation techniques like merging, joining, and concatenating DataFrames.

To perform data analysis using Pandas, you can use the Titanic dataset as an example. Below are the steps for generating summary statistics, grouping data, applying aggregate functions, and exploring advanced data manipulation techniques like merging, joining, and concatenating DataFrames.

[47]:
```
#Importing Pandas and Reading the CSV File
import pandas as pd

# Reading the Titanic dataset from the specified file path
file_path = 'C:/Users/sivasai/Downloads/Titanic-Dataset (2).csv'
df = pd.read_csv(file_path)

# Displaying the first few rows of the DataFrame to understand its structure
print("First few rows of the Titanic dataset:\n", df.head())
```

```
First few rows of the Titanic dataset:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0
```

```
   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

[48]:
```
# Getting summary statistics of the numerical columns
summary_statistics = df.describe()

# Generating summary statistics for a specific column (e.g., 'Age')
age_statistics = df['Age'].describe()

print("Summary Statistics of the Titanic dataset:\n", summary_statistics)
print("Summary Statistics of the 'Age' column:\n", age_statistics)
```

```
Summary Statistics of the Titanic dataset:
        PassengerId   Survived     Pclass        Age      SibSp      Parch  \
count    15.000000  15.000000  15.000000  14.000000  15.000000  15.000000
mean      8.000000   0.466667   2.400000  27.714286   0.600000   0.600000
std       4.472136   0.516398   0.910259  16.739766   0.828079   1.352247
min       1.000000   0.000000   1.000000   2.000000   0.000000   0.000000
25%       4.500000   0.000000   1.500000  15.500000   0.000000   0.000000
50%       8.000000   0.000000   3.000000  26.500000   0.000000   0.000000
75%      11.500000   1.000000   3.000000  37.250000   1.000000   0.500000
max      15.000000   1.000000   3.000000  58.000000   3.000000   5.000000

            Fare
count  15.000000
mean   24.042493
std    20.235219
min     7.250000
25%     8.050000
50%    16.700000
75%    30.672900
max    71.283300
Summary Statistics of the 'Age' column:
 count    14.000000
mean     27.714286
std      16.739766
min       2.000000
25%      15.500000
50%      26.500000
75%      37.250000
max      58.000000
Name: Age, dtype: float64
```

[54]:
```
# Creating another DataFrame with additional information
additional_info = pd.DataFrame({
    'PassengerId': [1, 2, 3, 4, 5],
    'CabinClass': ['A', 'B', 'C', 'B', 'A']
})

# Merging the Titanic dataset with the additional information on 'PassengerId'
merged_df = pd.merge(df, additional_info, on='PassengerId', how='left')

print("Merged DataFrame:\n", merged_df.head())
```

```
Merged DataFrame:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked CabinClass
0      0         A/5 21171   7.2500   NaN        S          A
1      0          PC 17599  71.2833   C85        C          B
2      0  STON/O2. 3101282   7.9250   NaN        S          C
3      0            113803  53.1000  C123        S          B
4      0            373450   8.0500   NaN        S          A
```

[55]:
```
# Joining DataFrames using set_index() and join()
df1 = df.set_index('PassengerId')
df2 = additional_info.set_index('PassengerId')

# Performing a join on the index
joined_df = df1.join(df2, how='left')

print("Joined DataFrame:\n", joined_df.head())
```

```
Joined DataFrame:
              Survived  Pclass  \
PassengerId
1                    0       3
2                    1       1
3                    1       3
```

```
4                    1       1
5                    0       3

                                                     Name     Sex   Age  \
PassengerId
1                                  Braund, Mr. Owen Harris    male  22.0
2        Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0
3                                   Heikkinen, Miss. Laina  female  26.0
4             Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
5                                 Allen, Mr. William Henry    male  35.0

             SibSp  Parch            Ticket     Fare Cabin Embarked CabinClass
PassengerId
1                1      0         A/5 21171   7.2500   NaN        S          A
2                1      0          PC 17599  71.2833   C85        C          B
3                0      0  STON/O2. 3101282   7.9250   NaN        S          C
4                1      0            113803  53.1000  C123        S          B
5                0      0            373450   8.0500   NaN        S          A
```

[56]:
```
# Concatenating two DataFrames vertically
concatenated_df = pd.concat([df.head(), df.tail()], axis=0)

# Concatenating two DataFrames horizontally
concatenated_df_horizontal = pd.concat([df.head(), additional_info], axis=1)

print("Vertically Concatenated DataFrame:\n", concatenated_df)
print("Horizontally Concatenated DataFrame:\n", concatenated_df_horizontal)
```

```
Vertically Concatenated DataFrame:
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
10            11         1       3
11            12         1       1
12            13         0       3
13            14         0       3
14            15         0       3

                                                Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
10                   Sandstrom, Miss. Marguerite Rut  female   4.0      1
```

```
11                           Bonnell, Miss. Elizabeth  female  58.0      0
12                     Saundercock, Mr. William Henry    male  20.0      0
13                        Andersson, Mr. Anders Johan    male  39.0      1
14                Vestrom, Miss. Hulda Amanda Adolfina  female  14.0      0

    Parch            Ticket     Fare Cabin Embarked
0       0         A/5 21171   7.2500   NaN        S
1       0          PC 17599  71.2833   C85        C
2       0  STON/O2. 3101282   7.9250   NaN        S
3       0            113803  53.1000  C123        S
4       0            373450   8.0500   NaN        S
10      1           PP 9549  16.7000    G6        S
11      0            113783  26.5500  C103        S
12      0         A/5. 2151   8.0500   NaN        S
13      5            347082  31.2750   NaN        S
14      0            350406   7.8542   NaN        S
Horizontally Concatenated DataFrame:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked  PassengerId CabinClass
0      0         A/5 21171   7.2500   NaN        S            1          A
1      0          PC 17599  71.2833   C85        C            2          B
2      0  STON/O2. 3101282   7.9250   NaN        S            3          C
3      0            113803  53.1000  C123        S            4          B
4      0            373450   8.0500   NaN        S            5          A
```

Application in Data Science: o Conclude your program by explaining how the use of Pandas in your program can help a data science professional. Discuss the advantages of using Pandas over traditional Python data structures for data handling and analysis. o Provide real-world examples where Pandas is essential, such as in data cleaning, exploratory data analysis (EDA)

Application in Data Science: The Role of Pandas

Pandas is an essential tool in the toolkit of a data science professional, offering robust capabilities for data handling, analysis, and manipulation. Here's an overview of how using Pandas in your program can significantly enhance data science workflows and why it's favored over traditional Python data structures like lists, dictionaries, and tuples.

1. Advantages of Using Pandas Over Traditional Python Data Structures

- **Efficiency in Handling Large Datasets: Pandas DataFrames and Series are optimized for performance, allowing efficient handling of large datasets. Unlike lists or dictionaries, Pandas structures are built on top of NumPy, making them much faster for operations involving numerical data.
- **Ease of Use:** Pandas provides high-level abstractions and a rich set of functions that make data manipulation and analysis more intuitive. Tasks that would require complex loops and conditionals with traditional data structures can often be accomplished with a single line of Pandas code.
- **Integrated Data Handling:** Pandas allows seamless reading from and writing to various data formats like CSV, Excel, SQL databases, and more. It also offers powerful tools for handling missing data, filtering, and cleaning data, which are not as straightforward with native Python structures.
- **Data Alignment and Labeling:** Unlike NumPy arrays, Pandas DataFrames and Series are designed with labeled axes (rows and columns), making data alignment and indexing more intuitive. This labeling is crucial for tasks like joining datasets, where column names serve as keys.
- **Powerful Grouping and Aggregation:** Pandas provides built-in functions for grouping data and performing aggregate operations. While it's possible to achieve similar results with dictionaries and lists, the process would be far less efficient and more error-prone.

2. Real-World Examples Where Pandas is Essential**

- Data Cleaning: In real-world data science projects, datasets often contain missing values, duplicates, or inconsistent data types. Pandas provides a suite of tools for handling these issues, such as `fillna()` for imputing missing values, `drop_duplicates()` for removing duplicates, and `astype()` for converting data types. This data cleaning process is a crucial first step in preparing data for analysis or modeling.

  Example: In financial analysis, datasets often have missing values for certain days or securities. Pandas can quickly identify and fill these gaps using domain-specific logic, ensuring that the analysis or predictive modeling is not biased by incomplete data.

Exploratory Data Analysis (EDA): EDA is a critical phase in data science where data scientists explore the dataset to uncover patterns, anomalies, and insights. Pandas makes EDA straightforward with functions like `describe()` for summary statistics, `groupby()` for segmenting data, and `plot()` for quick visualizations. These capabilities allow data scientists to understand the dataset's structure, identify relationships between variables, and form hypotheses for further analysis.

Example: In healthcare analytics, EDA using Pandas can help in identifying patterns in patient data, such as the correlation between certain demographics and health outcomes. By grouping data and calculating summary statistics, data scientists can quickly draw meaningful conclusions that guide the direction of more in-depth analysis.

- **Data Transformation and Feature Engineering:** Pandas is also crucial for transforming raw data into formats suitable for modeling. This includes creating new features, scaling or normalizing data, and reshaping datasets. These transformations are often necessary to improve the performance of machine learning models.

Example: In machine learning, especially in tasks like customer segmentation, Pandas is used to engineer features from transactional data, such as calculating the frequency and recency of purchases, which can then be used as inputs to clustering algorithms.

0.0.1 Conclusion:

Pandas is indispensable in data science due to its powerful, efficient, and intuitive data structures and functions. It simplifies the complex processes of data handling, cleaning, and analysis, enabling data science professionals to focus more on extracting insights and building models rather than wrestling with data preparation. Whether it's for initial data exploration or final data transformation before modeling, Pandas is a critical tool that vastly outperforms traditional Python data structures in both capability and efficiency.

By mastering Pandas, data scientists can streamline their workflows, handle larger and more complex datasets, and deliver more accurate and insightful analyses, making it a cornerstone of modern data science practices.