pandas

August 31, 2024

Getting familiar with PANDAS

Series: A one-dimensional labeled array of data. It's essentially a single column from a DataFrame.
DataFrame: A two-dimensional labeled data structure with columns and rows. It's analogous to a spreadsheet or SQL table.

[12]:
```
import pandas as pd
#creating series with list
data = [1, 2, 3, 4, 5]
series = pd.Series(data)
print("Series")
print(series)
#creating dataframe with list
data = [[1, 2], [3, 4]]
df = pd.DataFrame(data, columns=["A", "B"])
print("DataFrame")
print(df)
#creating dataframes with dictionary
data = {"Name": ["Rakesh", "anil", "lohith"], "Age": [19,20,20],"City":␣
 ↪["srikakulam", "vizianagaram", "godavari"]}
df = pd.DataFrame(data)
print(df)
```

```
Series
0    1
1    2
2    3
3    4
4    5
dtype: int64
DataFrame
   A  B
0  1  2
1  3  4
     Name  Age          City
0  Rakesh   19    srikakulam
1    anil   20  vizianagaram
2  lohith   20      godavari
```

Common Operations

Selecting Data:

By column: df[“Column_name”] By row: df.iloc[row_index] or df.loc[row_label] By condition: df[df[“Column_name”] > value] Filtering Rows:

df[condition] Modifying Data:

Assign new values: df[“Column_name”][index] = new_value Create new columns: df[“New_column”] = expression

[15]:
```
# Select data
print(df["Name"])
print(df.iloc[1])
# Filter rows
filtered_df = df[df["Age"] > 19]
print(filtered_df)
# Modify data
df.loc[0, "City"] = "vishakapatnam"
print(df)
# Create a new column
df['branch'] = ['CSD','CSD','CSD']
print(df)
```

```
0    Rakesh
1      anil
2    lohith
Name: Name, dtype: object
Name              anil
Age                 20
City      vizianagaram
branch             CSD
Name: 1, dtype: object
     Name  Age          City branch
1    anil   20  vizianagaram    CSD
2  lohith   20      godavari    CSD
     Name  Age           City branch
0  Rakesh   19  vishakapatnam    CSD
1    anil   20   vizianagaram    CSD
2  lohith   20       godavari    CSD
     Name  Age           City branch
0  Rakesh   19  vishakapatnam    CSD
1    anil   20   vizianagaram    CSD
2  lohith   20       godavari    CSD
```

2. Data Handling with Pandas

Handling Missing Data:

[26]:
```
import numpy as np
data = {
    'Name': ['lohith', 'rakesh', np.nan, 'bunny'],
    'Age': [20, 19, 21, np.nan],
    'City': ['vizianagaram', 'srikakulam', 'godavari', 'vishakapatnam'],
    'Salary': [50000, 60000, np.nan, 70000]
}

df = pd.DataFrame(data)
#Identifying missing values:
missing_values = df.isnull().sum()
print(missing_values)
#Filling missing values:
df.fillna(method='ffill', inplace=True)  # Forward fill
print(df)
#Dropping rows with missing values:
df.dropna(inplace=True)
print(df)
df['Age'] = df['Age'].astype(int)
#creating new column
df['Is_Adult'] = df['Age'] >= 18
#filtering the data
adult_df = df[df['Is_Adult']]
print(adult_df)
# Remove duplicates
df.drop_duplicates(inplace=True)
```

```
Name      1
Age       1
City      0
Salary    1
dtype: int64
     Name   Age           City   Salary
0  lohith  20.0   vizianagaram  50000.0
1  rakesh  19.0     srikakulam  60000.0
2  rakesh  21.0       godavari  60000.0
3   bunny  21.0  vishakapatnam  70000.0
     Name   Age           City   Salary
0  lohith  20.0   vizianagaram  50000.0
1  rakesh  19.0     srikakulam  60000.0
2  rakesh  21.0       godavari  60000.0
3   bunny  21.0  vishakapatnam  70000.0
     Name  Age           City   Salary  Is_Adult
0  lohith   20   vizianagaram  50000.0      True
1  rakesh   19     srikakulam  60000.0      True
2  rakesh   21       godavari  60000.0      True
3   bunny   21  vishakapatnam  70000.0      True
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_22028\3250273382.py:14:
FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a
future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method='ffill', inplace=True)  # Forward fill
```

Create two DataFrames (df1 and df2). Generate summary statistics using describe(). Group data by Store and aggregates Sales and Revenue. Merge, join, and concatenate the DataFrames using different methods. Display the results for each operation.

[30]:
```
# DataFrame 1: Sales data
data1 = {
    'Store': ['Store A', 'Store B', 'Store A', 'Store B', 'Store C'],
    'Product': ['Apples', 'Apples', 'Oranges', 'Oranges', 'Bananas'],
    'Sales': [100, 150, 200, 250, 300],
    'Revenue': [1000, 1500, 2000, 2500, 3000]
}

df1 = pd.DataFrame(data1)

# DataFrame 2: Store information
data2 = {
    'Store': ['Store A', 'Store B', 'Store C', 'Store D'],
    'City': ['palasa', 'tekkali', 'narsannapeta', 'srikakulam'],
    'Manager': ['raghu', 'ravi', 'raja', 'prasad']
}

df2 = pd.DataFrame(data2)

#Generate Summary Statistics
summary_statistics = df1.describe()

#Grouping Data and Applying Aggregate Functions
grouped_data = df1.groupby('Store').agg({
    'Sales': 'sum',
    'Revenue': 'sum'
})

#Advanced Data Manipulation
# Merging DataFrames
merged_data = pd.merge(df1, df2, on='Store', how='inner')

# Joining DataFrames
df1.set_index('Store', inplace=True)
df2.set_index('Store', inplace=True)
joined_data = df1.join(df2, how='inner')

# Concatenating DataFrames
concatenated_data = pd.concat([df1.reset_index(), df2.reset_index()], axis=1)
```

```python
# Step 6: Display Results
print("Summary Statistics:\n", summary_statistics)
print("\nGrouped Data:\n", grouped_data)
print("\nMerged Data:\n", merged_data)
print("\nJoined Data:\n", joined_data)
print("\nConcatenated Data:\n", concatenated_data)
```

```
Summary Statistics:
             Sales      Revenue
count    5.000000     5.000000
mean   200.000000  2000.000000
std     79.056942   790.569415
min    100.000000  1000.000000
25%    150.000000  1500.000000
50%    200.000000  2000.000000
75%    250.000000  2500.000000
max    300.000000  3000.000000

Grouped Data:
          Sales  Revenue
Store
Store A    300     3000
Store B    400     4000
Store C    300     3000

Merged Data:
      Store  Product  Sales  Revenue          City Manager
0  Store A   Apples    100     1000        palasa   raghu
1  Store B   Apples    150     1500       tekkali    ravi
2  Store A  Oranges    200     2000        palasa   raghu
3  Store B  Oranges    250     2500       tekkali    ravi
4  Store C  Bananas    300     3000  narsannapeta    raja

Joined Data:
          Product  Sales  Revenue          City Manager
Store
Store A   Apples    100     1000        palasa   raghu
Store B   Apples    150     1500       tekkali    ravi
Store A  Oranges    200     2000        palasa   raghu
Store B  Oranges    250     2500       tekkali    ravi
Store C  Bananas    300     3000  narsannapeta    raja

Concatenated Data:
      Store  Product  Sales  Revenue    Store          City Manager
0  Store A   Apples    100     1000  Store A        palasa   raghu
1  Store B   Apples    150     1500  Store B       tekkali    ravi
```

```
2  Store A  Oranges    200     2000  Store C  narsannapeta    raja
3  Store B  Oranges    250     2500  Store D    srikakulam  prasad
4  Store C  Bananas    300     3000      NaN           NaN     NaN
```

Advantages of using Pandas

- Efficient data handling: Pandas provides optimized data structures (DataFrames, Series) for fast data manipulation and analysis.
- Convenient data analysis: Pandas offers various functions for data filtering, grouping, merging, and reshaping, making data analysis easier.
- Data cleaning and preprocessing: Pandas provides functions for handling missing data, removing duplicates, and data type conversions.
- Integration with other libraries: Pandas seamlessly integrates with popular data science libraries like NumPy, Matplotlib, and Scikit-learn.

Real-world examples where Pandas is essential

- Data cleaning: Handling missing values, removing duplicates, and data type conversions.
- Exploratory data analysis (EDA): Generating summary statistics, visualizing distributions, and identifying correlations.
- Data transformation: Pivoting, melting, and reshaping data for analysis.
- Data merging and joining: Combining data from multiple sources.

Example use cases

- Data analysis in finance: Analyzing stock prices, portfolio optimization, and risk analysis.
- Data analysis in healthcare: Analyzing patient outcomes, disease trends, and treatment efficacy.
- Data analysis in marketing: Analyzing customer behavior, market trends, and campaign effectiveness.