Malla Reddy College of Engineering

(Approved by AICTE(New Delhi), Permanently Affiliated to JNTUH & Accredited by NBA & NAAC
Maisammaguda, Dhulapally, post via Kompally, Secunderabad - 500100

Technical Seminar Topic:

Introduction to Numpy and Pandas for Data Analysis

PRESENTED BY:

21Q91A66D1 - CHINNA PRAVEEN

UNDER THE GUIDANCE OF :

Mr. R. Venkatesh

Introduction

■Overview of Python as a Leading Language for Data Analysis

- Easy-to-learn syntax and strong community support.
- Integration with powerful libraries like NumPy, Pandas, and Matplotlib for data manipulation, analysis, and visualization.

■Importance of NumPy and Pandas

- NumPy: Enables fast and efficient numerical computations with multi-dimensional arrays.
- Pandas: Simplifies handling and manipulation of structured data using DataFrames and Series.

What is NumPy?

- NumPy stands for Numerical Python.
- A core library for numerical computing in Python.
- It provides support for multi-dimensional arrays and matrices, enabling efficient mathematical operations.
- Used for scientific computing, data analysis, and machine learning tasks.

What is Pandas?

- Pandas stands for Python Data Analysis Library.
- It is built on top of NumPy, extending its functionality for data analysis.

  1. Pandas provides two main data structures:

  2. Series: One-dimensional labeled array.
- DataFrame: Two-dimensional labeled data structure (similar to a table or spreadsheet).
- Key Benefits: Simplifies data manipulation, cleaning, and analysis with powerful tools and methods.

Why Use NumPy and Pandas?

1. Efficient Handling of Large Datasets

- Both libraries are optimized for performance, allowing efficient storage and manipulation of large data arrays and tables.

2. Simplifies Data Manipulation

- Pandas provides powerful tools for data wrangling, including easy data cleaning, filtering, and transformation. NumPy simplifies mathematical and statistical operations on arrays.

3. Seamless Integration with Other Python Libraries

- Both integrate easily with other popular Python libraries like Matplotlib (for visualization), Scikit-learn (for machine learning), and SciPy (for scientific computing).

Installing NumPy and Pandas

- Using pip:
- To install NumPy and Pandas, use the following command in your terminal or command prompt:

  pip install numpy pandas
- Compatibility:
- Python Version: NumPy and Pandas are compatible with Python 3.x and later.
- IDEs: They work with popular IDEs like Jupyter Notebook, PyCharm, and Visual Studio Code. Make sure you have the necessary Python environment set up.

Basic NumPy Features

- N-dimensional Arrays: NumPy provides support for multi-dimensional arrays, allowing efficient storage and manipulation of large datasets. These arrays are more efficient than traditional Python lists, especially for mathematical and scientific computing.
- Mathematical Operations: NumPy supports vectorized operations, enabling element-wise arithmetic operations (addition, multiplication, etc.) on arrays without needing loops, making calculations faster and more efficient.
- Broadcasting: Broadcasting allows NumPy to perform operations on arrays of different shapes by automatically expanding the smaller array to match the dimensions of the larger one. This simplifies complex array manipulations and reduces memory consumption.
- Random Sampling: NumPy includes random sampling functions to generate random numbers and distributions, supporting operations like random integers, random floating-point numbers, and random selection from arrays for simulations and modeling.

- Creation of NumPy Array:
- python code:

```
import numpy as np
arr = np.array([1, 2, 3])
```

- NumPy arrays allow for efficient handling of numerical data and perform operations much faster than Python lists.

NumPy Array Operations

- NumPy allows performing element-wise mathematical operations on arrays.
- **Operations:**
- Addition, Subtraction, and Multiplication: Perform these operations element-wise between arrays or with a scalar.

  Example: arr * 2 (multiplies each element by 2).
- Element-wise Operations: Operations are applied element-wise between arrays. Example: arr1 + arr2 (adds corresponding elements from two arrays).

NumPy Array Indexing and Slicing

- Access specific elements or slices of a NumPy array using indexing and slicing.
- **Indexing:** Access elements using indices (starts from 0). Example: arr[2] (accesses the third element).
- **Slicing:** Extract a subset of the array. Example: arr[1:3] (accesses elements at index 1 and 2).

NumPy Functions

- Mathematical Functions: NumPy provides built-in functions for statistical calculations.
- Mean: np.mean(arr) (computes the average of elements).
- Median: np.median(arr) (computes the middle value).
- Standard Deviation: np.std(arr) (measures data spread).
- Random Functions: Generate random numbers or sample data.

  Example: np.random.rand(3) (generates 3 random numbers between 0 and 1).

Introduction to Pandas

- Pandas is a Python library providing Series and DataFrame objects for data manipulation.
- It’s designed for handling structured data (e.g., tables, CSV files) and time series data.
- Key features: Easy indexing, data cleaning, and manipulation.

Pandas Series

- A Series is a one-dimensional labeled array in Pandas.
- Each element is associated with an index, allowing for quick access.
- Example:

```
#code
import pandas as pd
s = pd.Series([1, 2, 3])
```

Pandas DataFrame

- A DataFrame is a two-dimensional tabular data structure.
- It consists of rows and columns, where each column can have a different data type.
- Example:

```
#code
data = {'Name': ['A', 'B'], 'Age': [21, 22]}
df = pd.DataFrame(data)
```

Loading Data with Pandas

- Pandas allows easy data loading from different file formats.
- CSV:

  #code

  df = pd.read_csv('file.csv')
- Excel:

  #code

  df = pd.read_excel('file.xlsx')

Data Cleaning with Pandas

- Handle missing data using fillna():

  #code

  df.fillna(0)

- Drop missing rows or columns with dropna():

  #code

  df.dropna()

Data Selection and Filtering

- Access columns in a DataFrame:

  #code

  df['Column']

- Filter rows based on conditions:

  #code

  df[df['Age'] > 21]

Pandas DataFrame Operations

- Sorting data by column values:

#code

df.sort_values('Age')

- Aggregation to compute statistical measures:

#code

df['Age'].mean()

Merging and Joining DataFrames

- Merge two DataFrames based on a common key:

```
#code
pd.merge(df1, df2, on='key')
```

- Concatenate DataFrames along a specific axis:

```
#code
pd.concat([df1, df2])
```

NumPy vs Pandas

- NumPy: Uses arrays that store homogeneous data types. Best suited for numerical computations like scientific calculations and matrix operations.
- Pandas: Provides Series (1D) and DataFrame (2D) structures, supporting heterogeneous data types. Primarily used for data analysis, manipulation, and handling structured data.

Real-World Applications

- NumPy: Widely used in scientific computing, simulation, image processing, and machine learning tasks where numerical computations are essential.
- Pandas: Ideal for data cleaning, financial analysis, handling time-series data, and performing complex data manipulations.

Integrating with Other Libraries

- Example: NumPy and Pandas are commonly integrated with Matplotlib for visualizations. NumPy handles numerical data, while Pandas manages the structure, enabling seamless plotting of graphs and charts.

Limitations of NumPy and Pandas

- NumPy: May have performance overhead for smaller datasets due to its dependence on low-level operations.
- Pandas: Faces scalability issues when working with very large datasets, particularly compared to distributed data frameworks like Dask or Spark.

Conclusion

- NumPy is crucial for efficient numerical tasks, providing fast array operations and mathematical functions, while Pandas excels in handling and manipulating structured data, making data cleaning, analysis, and transformation much easier.
- These tools are foundational for data science, enabling quick and efficient data handling.
- Encouraging practice with real-world datasets is key to mastering these libraries for effective data analysis.

THANK YOU