14/10/25

pandas :

Libraries :

They are collection of modules and functions to perform specific tasks.

pandas : Data manipulation.

→ pandas stands for "python Data Analysis Library".

It's used for :

- Data cleaning.
- Data analysis.
- Data manipulation (sorting, filtering, grouping, merging etc).
- Reading / writing data from files (CSV, Excel, etc)

1. Importing pandas :

```
import pandas as pd
```

→ pd is a common alias used for pandas.

2 main Data structures in pandas.

a) Series :

One-dimensional data (like a list or column)

Ex :

```
import pandas as pd
data = [10, 20, 30, 40]
s = pd.series(data)
print(s)
```

Op :

```
0    10
1    20
2    30
3    40
dtype : int64.
```

b) Data frame :

Two-Dimensional (rows and columns), like an Excel sheet or SQL table.

Ex:

```
data = {
    "Name": ["Alice", "Bob", "charlie"],
    "Age" : [25, 30, 35],
    "City" : ["Delhi", "Mumbai", "chennai"]
}
df = pd.DataFrame(data)
print(df)
```

O/p:

	Name	Age	City
0	Alice	25	Delhi
1	Bob	30	Mumbai
2	charlie	35	Chennai

Null Value Imputation Methods:

Null value imputation means filling or replacing those missing values with meaningful data instead of deleting them.

→ Extract particular rows we have two functions iloc and loc functions are used to extract particular rows and columns.

15/10/25

1. Data Creation and Basic info:

1. pd.DataFrame( ) → Create a DataFrame
2. pd.Series( ) → Create a series.
3. df.head( ) → Display first 5 rows.
4. df.tail( ) → Display last 5 rows.
5. df.info( ) → Summary of DataFrame.
6. df.describe( ) → Statistical Summary of numeric columns.
7. df.shape & x → Returns (rows, columns)
8. df.columns & x → Return list of column names.
9. df.dtypes & → Returns data type of columns.

2. Data Selection:

df['col_name'] → select one column.

df[['col1', 'col2']] → select multiple column.

df.loc[ ] → select by ~~label (rows / columns)~~

df.iloc[ ] → select by index position.

df.at[ ] → Access a single value by label.

df.iat[ ] → Access a single value by position.

3. Data cleaning :

1. df.isnull() → Check for null values
2. df.notnull() → Check for non-null values.
3. df.dropna() → Remove rows with missing values.
4. df.fillna(value) → Replace missing values.
5. df.replace(old, new) → Replace specific values.
6. df.rename(columns={}) → Rename columns.
7. df.duplicated() → Check for duplicate rows.
8. df.drop_duplicates() → Remove duplicate rows.

4. Data Manipulation :

1. df.sort_values(by='col') → Sort by column values.
2. df.sort_index() → Sort by index.
3. df.reset_index() → Reset index to default.
4. df.set_index('col') → Set a column as index.
5. df.apply(func) → Apply a custom function
6. df.~~mapp~~map(func) → Apply function to Series.
7. df.assign() → Add new columns easily.
8. df.astype(type) → Convert data type.

5. Aggregation and Grouping:

1. df.groupby('col') → Group by column.
2. df.mean() → Mean of values.
3. df.sum() → Sum of values.
4. df.min() → Minimum value.
5. df.max() → Maximum value.
6. df.count() → Count of non-null values.
7. df.agg(['mean', 'sum']) → Multiple aggregations.

6. Merging, Joining & Concatenation:

pd.concat([df1, df2]) → Combine dataframes vertically or horizontally.

pd.merge(df1, df2, on='col') → Merge two dataframes on a common column.

df.join(df2) → Join on index or key column.

7. Input / Output Operations:

pd.read_csv('file.csv') → Read csv file.

df.to_csv('file.csv') → write to csv.

pd.read_excel('file.xlsx') → Read Excel file.

df.to_excel('file.xlsx') → write to Excel.

pd.read_json('file.json') → Read JSON file.

df.to_json('file.json') → write to JSON.

8. Date and Time Handling:

pd.to_datetime(df['col']) → Convert to date time.

df['col'].dt.year → Extract year.

df['col'].dt.month → Extract month.

df['col'].dt.day → Extract day.

9. Statistical and Math operations:

df.corr() → Correlation between columns.

df.var() → variance.

df.std() → Standard deviation.

df.median() → Median value.

df.mode() → Mode value.

loc[] → Label-based indexing

You use row and column labels (names).

Syntax:

df.loc[row_label, column_label]

iloc[ ] → Index - based (integer position) indexing:

You use row and column numbers (positions).

Syntax:

```
df.iloc [row_index, column_index]
```

24/10/25

Numpy:

Numpy (Numerical python) is a powerful python library used for:

* Handling large numerical data efficiently.
* performing mathematical, statistical, and linear algebra operations.
* Creating multidimensional arrays (ndarrays)

Importing Numpy:

```
import numpy as np.
```

Ex:

```
arr1 = np.array ([1, 2.5, 67, 'vasu'])
arr1
```

O/P: array (['1', '2.5', '67', 'vasu'], dtype='<U32')

```
arr1 = np.array([1, 2.5, 67, 'vasu'])
arr2 = np.array([1, 2.5, 67, 34.8])
arr1 + arr2
```

How to check dimension of array

```
a2.ndim
```

O/P : 2

Ex(2) : 3x3 Array.

```
arr = np.array([[3,6,7], [1,9,0], [2,4,5]])
arr
```

O/P :

```
array([[3, 6, 7],
       [1, 9, 0],
       [2, 4, 5]])
```

→ arr.ndim

O/P : 2

→ arr.dtype

O/P : dtype('int32')

→ arr.size

O/P : 9

→ type(arr)

O/P : numpy.ndarray.

→ arr.shape

(3, 3)

→ All the elements of first column.

→ arr [:, 0]

O/p: arr([3, 1, 2]).

2) np. zeros ((2,2))

O/p: array ([[0, 0],
[0, 0].]).

→ Integer matrix:

① np. zeros ((4,3), dtype = int)

O/p:
array ([[0, 0, 0],
[0, 0, 0],
[0, 0, 0],
[0, 0, 0]])

② np. eye (C3), dtype = int)

O/p: array ([1, 0, 0],
[0, 1, 0],
[0, 0, 1]])

③ np. full ((3, 3), 7)

O/p: array ([[7, 7, 7],
[7, 7, 7],
[7, 7, 7]])

→ a1.reshape((1,12))

Olp: array([[4,5,8,1,9,2,0,5,1,3,6,9]])

2) v1 = a1.flatten()

v.ndim

Olp: 1

27|10|25

Universal functions:

Examples:

arr = np.array([3,5,8,2,1,3])

arr

Olp: array([3,5,8,2,1,3])

1) np.sqrt(arr)

Olp: array([1.73205081, 2.23606798, 2.82842712,
1.41421356, 1. , 1.73205081])

2) np.exp(arr) → 2.71 eulers rule.

Olp:
array([2.00855369e+01, 1.48413159e+02,
7.38905610e+00, 2.71828183e+00, 2.00855369e+01])

→ empty :

* np.empty((2,2), dtype = int)

o/p: array([[-2017819648, 435],
[0, 0]]).

→ linspace

np.linspace(1, 10, 10).

o/p: array([1., 2., 3., 4., 5., 6., 7., 8., 9., 10.])

→ np.random.rand(2,2)

o/p: array([[0.5351, 0.5224],
[0.5696, 0.8227]]).

→ random.randint

np.random.randint(10, 20, (3,3))

o/p: array([[10, 13, 19],
[12, 17, 10],
[10, 16, 13]]).

→ np.arange(2, 10, 2)

o/p: array([2, 4, 6, 8])

Ex:

```
arr1 = np.array([[1,2],[8,9]])
arr2 = np.array([[3,4],[6,5]])
arr1 + arr2
```

O/p: array([[4, 6],
[14, 14]])

→ np.matmul(arr1, arr2)

O/p: array([[15, 14],
[78, 77]])

→ arr1 @ arr2

O/p: array([[15, 14],
[78, 77]])

→ arr1.T

O/p: array([[1, 8],
[2, 9]])

→ arr1

O/p: array([[1, 2],
[8, 9]])

Data visualization:

Data visualization means showing data in the form of charts, graphs, or plots instead of plain numbers - so we can understand patterns, trends, and insights easily.

Why Data visualization?

Easy understanding of data it save more time because we can quickly analyze the data.

popular libraries in python for Data visualization

1. Matplotlib → Basic plotting (bar, line, scatter, etc.)
2. Seaborn → Beautiful and advanced statistical charts.
3. plotly → Interactive graphs.
4. pandas → Simple built-in charts from DataFrames.

univariate Analysis:

To understand the quality of the one feature.

Countplot, histogram, boxplot, piechart.

Bivariate Analysis:

To understand the relationship b|w two features.

28/10/25

Data visualization:

Data visualization means showing data in the form of charts, graphs, or plots instead of plain numbers - so we can understand patterns, trends, and insights easily.

Why Data visualization?

Easy understanding of data it save more time because we can quickly analyze the data.

popular libraries in python for Data visualization

1. Matplotlib → Basic plotting (bar, line, scatter, etc.)
2. Seaborn → Beautiful and advanced statistical charts.
3. plotly → Interactive graphs.
4. pandas → Simple built-in charts from DataFrames.

univariate Analysis:

To understand the quality of the one feature.

Countplot, histogram, boxplot, piechart.

Bivariate Analysis:

To understand the relationship b|w two features.

barplot, Scatterplot.

Multivariate Analysis:

To understand the relationship blw more than 2 features.

heatmap, pairplot.

Ex:

```
import pandas as pd
mtcars_data = pd.read_csv(r"C:\Users\VAISHNAVI\
                Downloads\mtcars.csv")
mtcars_data.
```

→ # pip install matplotlib

```
from matplotlib import pyplot as plt
# import matplotlib.pyplot as plt.
```

Ex:

```
plt.figure(figsize=(5,5))
plt.hist(x='mpg', data=mtcars_data)
plt.title("MPG Distribution")
plt.xlabel("MPG")
plt.ylabel("Frequency")
plt.show.
```