Machine Learning associate

Numpy:

- Numpy stands for numerical python which is used to perform numerical calculation
- Numpy is used to handle arrays and matrices.

What is numpy?

→ Numpy in Python is a library/package that used to work with arrays

→ It is a data structure used to store multiple values.

→

Why numpy?

- Faster and more compact than lists
- consumes less memory
- data type can be specified
- high level mathematical functions can be used
- used to solve problems related to linear algebra, fourier transform, vectors etc...

What is an array

A:

Numpy arrays commonly used data structures in python that store data as a grid or a matrix and easy to access.

```
import numpy as np
arr1 = np.array([6, 7, 8, 9])
print(arr1)
```

o/p: [6 7 8 9]

Built in function to generate numpy array

① arange: It is used to generate evenly spaced values within a given interval

Note: End value not included.

Ex: np.arange(4)
o/p: [0, 1, 2, 3]

② ones: returns an array will all ones

Ex: np.ones(5)
o/p: [1, 1, 1, 1, 1]

③ zeros: returns zeros

Ex: np.zeros(5)
o/p: [0, 0, 0, 0, 0]

④ Eye -- (Identity matrix)

Ex: np.eye(2) o/p [[1 0]
[0 1]]

(5) full :- It is used generate array with some elements

```
np.full((2,2), 10)
```

O/P
```
[[10, 10]
 [10, 10]]
```

(6) Linspace :- It is used to generate linearly spaced values

```
np.linspace(1, 10, retstep=True)
```
1 → b/w 1 & 10
retstep → different b/w each value

(7) Random :-

a) randint :- It is used to generate random integers from given interval.

Ex:
```
np.random.randint(1, 100, 10)
```
→ number of Elements

b) rand :- It generates random values b/w the range [0, 1]

Ex [uniform distribution]
```
np.random.rand(10)
```
→ 10 elements print b/w [0, 1]

c) randn :- It is used to generate random numbers from normal distribution within a range [-3, 3]

```
np.random.randn(10)
```
→ 10 element b/w [-3, 3]

Indexing with 2-D array:

```
arr = np.array([[1,2,3],[4,5,6],[7,8,9]])
arr
```

o/p:
```
[1 2 3]
[4 5 6]
[7 8 9]
```

fetching first row

1) `arr[[0]]` o/p `[1 2 3]`

fetching first column

1) `arr[:,[0]]`

o/p :
```
[[1]
 [4]
 [7]]
```

1-D Array:

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

`arr.ndim` :- dimension of an array

`arr.size` ⇒ number of elements

`arr.dtype` ⇒ data type of an array element

`arr.shape` ⇒ order of matrix

Eg: (1, 6) , (2, 2)

type casting: arr.astype("int")

It will convert to integers.

It is used to convert one data type into another data type.

→ Matrix multiplication: A @ B

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} (5+2) & (2+6) \\ (5+4) & (6+12) \end{bmatrix}$$

→ Element wise multiplication: A * B

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 5 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ 3 & 12 \end{bmatrix}$$

(VVI) Difference b/w matrix multiplication & element wise multiplication

Pandas:-

- Pandas is a python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.

Why is Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

* Import pandas as pd

pandas series:- Its an one dimensional array holding data of any type.

1) A = pd.Series()

2) B = pd.series(30)
o/p (30)

pandas data frame :

A data frame is a data structure that organizes data into a 2-dimensional table of rows and columns, much like a spreadsheet.

```
dict = { "name" : ["John", "Sam"], "age" : [21, 22]
       , "salary" : [3000, 400] }

data = pd.DataFrame(dict)
data
```

o/p :

	name	age	salary
0	John	21	3000
1	Sam	22	400

* How to create copy ? : `data1 = data.copy()`
* Sort data : `data1.sort_values("salary")` → default asc
* permanent change : use `inplace = True`
* To sort in desc : `data1.sort_values("age" ascending = False, inplace = True)`
* rename column : `data1.rename({ "name of colmn" : "Renamed by emp" }, axis = 1)`
* fetching column ? : `data[["age"]]` →

Loc and iloc

loc:- label based indexing which helps in fetching rows and columns

iloc:- index / integer based indexing

→ data.iloc [rows, column] → [:, 1:4], [[1,2,3], :]

→ data.loc [rows, column] → [:, ["age", "job"]], [[103, 105], :]

Reading file in pandas

```
data = pd.read_csv("train.csv")
```

- **shape**: It is used to find no rows & columns
  `data.shape`
- **Head**: `data.head()`
- **tail**: `data.tail()`
- **data.columns**:- To fetch all the names of column
- **data.dtype**:- used to find data type of each column

*) data.drop("column_name", axis=1, inplace=True)

*) data.select_dtypes(include=["float64", "int64"])
↳ fetching numerical columns

* data.select_dtypes(include="object")
fetching categorical data

*) for above two codes add .columns → To see only column name

* data.info()

* data.describe() → by default fetch numerical data.

*) data.describe(include='object')
→ To fetch categorical data

*) data.column_name.unique() → To fetch unique element [distinct element]

*) data.column_name.value_counts()
It is used to count of unique categories

*) data.isnull().sum() → TO check missing value

How to filter data :

① data.loc[data['sex'] == 'male']
fetch records related to male passenger

② data.loc[data["Age"] > 60]
fetching records where age is greater than 60

③ . shape → add to above see no of records
@ like count

④ data.loc[data['Aage'] < 10].shape
o/p : (62, 12)

---

data1 = data.copy() → Creating a copy

How to create new column

→ data["column name"] = "Data inside column"

Eg:- data["course"] = "Data science"

→ To combine two column to one different column

data["Family"] = data["SibSp"] + data["Parch"]

drop the columns :

```
data.drop("course", axis=1, inplace=True)
```
↳ column name

```
data.drop(["Col1", "col2"], axis=1, inplace=True)
```
↳ ↳ dropping multiple columns

Sort data

```
data.sort_values("Age", axis=0, inplace=True)
```
↳ In ascending order.

```
data.sort_values("Age", axis=0, ascending=False,
                 inplace=True, ignore_index=True)
```
↳ In decending order

Type casting in pandas :

1. int can be converted into float and vice versa
2. Object cannot be converted into int or float if data inside column is text.
3. object can be converted into int or float if data inside columns is number
4. int, float can be converted into object.
5. Type casting works only when data doesn't have missing values

① → data1.dtypes → To check data types

② → data1["Fair"] = data1["Fair"].astype("int")

③ → data1["Family"] = data1["Family"].astype("float64")

column name

To which data type to convert

How to replace missing values

① numerical → mean → ∩ → normal distribution

→ median → ∩ (skewed)

② Categorical → mode

VVI ← [If you find more than 80% data missing drop columns]

---

* How replace missing value.

① check for the distribution

```
import seaborn as sns
sns.histplot(x=data["Age"], kde=True)
```

O/P:

∩

② Since distribution of age is not normal we replace missing values in age column with median

```
data["Age"].median()    o/p: 28.0
```

③ replace missing value in age column with median

```
data.loc[data["Aage"].isnull() == True, "Age"] = 28
```

replaced successfull

④ replacing categorical column for missing value

① Embarked data as missing value

```
data["Embarked"].mode()
```

②
```
data.loc[data["Embarked"].isnull() == True,
         'Embarked'] = 'S'
```

If you get multiple mode means replace with any one of them

combine two data frame.

```
pd.concat([df1, df2], axis=0, ignore_index=True)
```

#rows manipulating axis=0

Seaborn :- [Exploratory data Analysis [EDA]]

① Univariate :- Analysis single variable

eg: To check distribution :- histplot, distplot, countplot, boxplot.

② Bivariate :- Analysis two variables :-

eg: Line plot, scatter plot, pie, barplot, Boxplot, violin, strip, swarm

③ Multivariate :- Analysing multiple variable

eg: Pair plot
heat map

seaborn has a inbuilt dataset

→ sns.get_dataset_names()

→ Loding data :-

data = sns.load_dataset("iris")

① -> Relation plot.

relation plots are used to understand relationship b/w two variables

- line plot (2 Numerical)
- scatter -"-
- relplot -"-

② Categorical plots :-

It is used to analyze one numerical one and categorical column

- count plot (1 categorical)
- box plot (1 num (or) 1 num & 1 cat)
- bar plot (1 cat and 1 num) (or) 1 num
- violin plot (1 num (or) 1 num & 1 cat)
- strip plot -"- -"-
- swarm plot -"- -"-
- cat plot -"- -"-

③ Distribution plots :-

- histplot
- dist plot

These two graphs are used to check distribution of data

④ Multivarient :-

- pairplot
- heat map

What is seaborn :-

[Seaborn is a python data visualization library based on matplotlib]. It provides a high-level interface for drawing attractive and informative statistical graphics.

- seaborn is built on top of matplotlib.