HAWA MAHAL
PAGE NO.
DATE :

Last Unit ie
"Society law and Ethics"

Read Completely
from
previously shared
"Must Read" Notes
They are useful for not only Exams but also day to day activities of Cyber world.

These summarised Notes are to be read after doing one Revision from book. Then only these pages will work like Handy Quick Review Capsule.

Best wishes
from

Ravita Pathak
PGT CS
KV No. 1 Rewa

23rd

Unit 1
topic 1 Dataframes → A Quick Review

Important Statements

→ import numpy as np
→ import pandas as pd
→ from pandas import Dataframe

→ pivot()
→ needs 3 arguement
index, col, values
→ index becomes row
→ col becomes column
→ value shown as data

pivot_table()
→ need 4 parameters
index, col, values, agg fun
→ aggregate function of value shown as data

Ex df-P = df.pivot_table(index='item', column='company', values='Rupees', aggfunc=np.mean)

Dataframes are size mutable and also value mutable

properties of Dataframe
- index = row
- columns = cols
- dtypes = dtypes of data
- size = total no. of elements in df
- shape = dimension (r, c) (m, n)
- ndim = tot axes → 2 for every dataframe
- T = transpose of df → row to col, col to row

* leave this & you will get pivot

→ Notation in Dataframe

df.columnname
df["colname"]
df[['col1', 'col2', 'col3', ...]]

→ use of loc & iloc → to select a subset from DF

df.loc[start row : end row , start col : end col]

df.iloc[start index Row : end index Row , start index col : end index col] → end index not included

Dataframes

portion from book
page 59 Add, delete

→ Add a col ⇒ df['col'] = [val1, val2, ... valn]

→ delete a col ⇒

df.drop (['col1', 'col2'], axis = 1) } drop is dataframe function.
df.drop (columns = ['col1', 'col2']) }
del df['colname'] → del is command of python

delete a Row

df.drop (['row1', 'row2']) by default axis is 0.

* Dataframe functions

df.min() } similarly mode(), sum(), count(), median(), corr(), std(), mean(), var(), quantile()
df.max() }

by default axis = 0 ⇒ and calculation done on each column
write axis = 1 to calculate along rows.

Sorting of dataframe (By Value, By index) sort_value()

ex ⇒ df1.sort_value(col ["col"], ascending = True/false)

df1.sort_values (by = ['col', 'col'])

df1.sort_values (['col', 'col'])

df1.sort_values

⇒ df1.sort_value (['col'], axis = 0/1, asc_=T/F, na_position=)

* Pipe function ⇒ df.pipe (func1).pipe (func2).pipe (func3)

ex> how it works (sandwiching of functions.

(fun3 (fun2 (fun1)))
3 2 1 order of execution

Dataframes Contd...

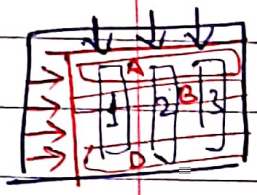
⇒ apply & applymap()

apply() → applies the function given as argument to one row or col.

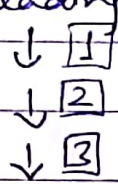
applymap() → applies the function to all individual element.

ex → df.apply(functnm, axis=0/1) → every
0 - index/row
1 - column

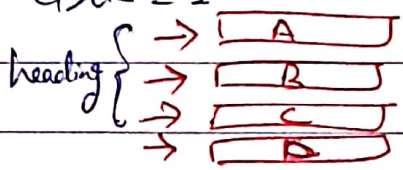
df.applymap(function) → → no need to give axis.



axis = 0



axis = 1



applymap() → result spread over every element (cumsum)

Pg 86

Groupby() → df.groupby(by = label or list of labels, axis=0/1)

ex → df1.groupby('tutor') or df1.groupb(['col1', 'col2'])

one column — multi column

then group attributes can be used after saving groupby results in some Dataframe.

ex ⇒ df1 = df.groupby('tutor') then we can use following attributes.

→ 1) groups → give the list of all items in the group

imp * 2) get_group(grpnm) → show the data of particular group
get_group('x')

→ 3) size → shows the group size (How many items in grp)

→ 4) count → count of non-Na value of each column in group

→ 5) head → list any specified col in detail
→ df1[col1].head().

→ Aggregation can be done in a grouped dataframe as you do in normal dataframe, using agg()

df1.agg([np.mean(), np.median(), np.mode()])

⇒ aggregation & groupby can be combined to get result in one single command.

df1.groupby('col').agg([np.mean, ...])

Transform → agg function reduces the data by giving summary. Transform() function returns the transformed version of data by adding repeated rows of result so that shape of original data & tranformed data matches.

ex:- if aggregate data has only 4 rows & transformed data will convert agg data also to original data has 12 rows then 12 rows by repeating result at same groupby col data.

```
dfnew = df1.groupby([Tutor]).
        transform(np.mean)
```

reindex_like() → df.reindex_like(dfnew) will copy index from dfnew

* reinden, reshape, rename, reindex_like() (→ small letters)

⇒ **Rename** → rename() function renames the existing index/columns provide the data in form of dictionary where

{Key1: value1, Key2: value2, Key3: value3 ... }

(Key1 → old index, value1 → new index)

ex⇒ df.rename({dict data}, axis=0/1, inplace=True/False)

→ df.rename({Q1: A1, Q2: A2, Q3: A3}, inplace=True) All changes in df.

→ df1=df.rename({Q1: A1, Q2: A2, Q3: A3}, axis=1, inplace=False)

change in column, changes saved in df1. df is as it is.

⇒ **reindex()** ⇒ change the order of existing index or recreate index

for row { df.reindex(['q4', 'q1', 'q3', 'q2']) or ~~df.reindex[]~~ df.reindex(index=[new index])

for col → df.reindex(columns=[new values]) (for index)

→ if new col/row is defined then use fill_value to give new value

ex original df has only 2 col, new df after reindex has 4

→ df.reindex(columns=[1, 2, 3, 4], fill_value=50)

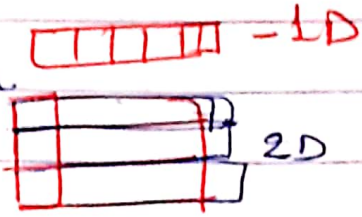
(3, 4 → new cols → filled by 50)

Numpy (Unit-1 topic 2)

import numpy as np → library linking

→ Array: [1D - vectors - single row/col
2D → multi dimension MR/MC if M=2, 2D Array

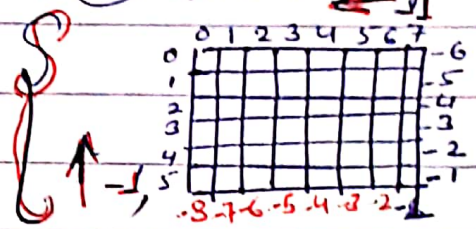
→ homogenous, continous set of element.
→ 2D arrays stored in Row Major/Col Major pattern



→ Creating 2D Array

a = np.array([[, ,], [, ,]]) → ①
b = np.array([, , , , , ,])
c = np.reshape(b, (m,n)) → ② → new shape of 2D Array

→ Slicing A[start:stop:step]
B[start:stop:step, start:stop:step]
Row , col



default → start=0, stop=dimension, step=1
→ X = A[::-1] → reverse of A (1D array)
Y = A[::-1, ::-1] → revers both row & col in any 2D array → bottom-up, Right-left
Z = A[0:m] ⇒ traversal from 0 to m-1

* step = -1 (negative) means reverse traverse

properties → shape, dtype, itemsize
a.shape → (2,3)
a.dtype → type of data of Array
a.itemsize → size in bytes of each elements.

* difference b/w numpy Array & list *

Array	List
⇒ size not changeable	⇒ size can be change
⇒ homogenous items	⇒ different type of datatypes item
⇒ less space	⇒ more space in memory
⇒ vectorized operation effects each element A*2 ⇒ A=[1,2] then [2,4] elements multiplied	⇒ not allowed vector operation L*2 ⇒ L, L → L=[1,2] → [1,2,1,2] elements extended

Split: subset of Arrays
* split(A, (m,n))
* hsplit(A, n)
* vsplit(A, n)

Stack - hstack() → A1 A2 →
vstack() →
Concatenate()
np.concatenate((A1, A2), axis=0/1)
axis=0 axis=1

hsplit → splits like this in equal parts
vsplit → like this in equal parts
split → 0-1 | 2-5 | 6 → 0,1 / 2-5 / 6- → 2, 6 means
split(A, [2,6]), axis=0/1)

HAWA MAHAL
PAGE NO.
DATE :

1D, 2D Array
Array slices, joins, subset
Arithmetic operation on 2D
Covariance, correlation, Linear Regression
Plot → bar, histogram, frequency polygons
boxplot, scatter plots

Numpy Contd...

★ → Arithmetic operations +, −, *, /, % etc works like mathematical operation in vector sequence. ex

$$A = \begin{bmatrix} A1 & A2 \\ A3 & A4 \end{bmatrix} + B = \begin{bmatrix} B1 & B2 \\ B3 & B4 \end{bmatrix} = \begin{bmatrix} A1+B1 & A2+B2 \\ A3+B3 & A4+B4 \end{bmatrix}$$

'+' is distributed → A+B

★ Covariance ⇒ np.cov( ) method

→ np.cov(arr1, arr2) ⇒ 02 arrays needed to find covariance by elements of arr1 & arr2. (1D)

→ if array is 2D or multidimension → np.cov(ARR) ND array

np.cov(a,b)

cov (a,a)	cov (a,b) *
cov (b,a) *	cov (b,b)

cov(a,b) = cov(b,a) both are equal.

cov(a,a) ⇒ is called variance of a

Cov(b,b) ⇒ is variance of b

covariance zero (0) means not related

−ve means → negatively related, +ve → positively related

★ Correlation → it is also called as normalised covariance

np.corrcoef(ar1, ar2) → correlation coefficient is calculated

⇒ both matrix (array) must have same shape.

output

corrcoef a,a *	corrcoef a,b
corrcoef b,a	corrcoef b,b *

* ⇒ is always 1

→ positive correlation or 1 means highly related data

→ negative means when one data increases other decreases.

★ → Linear regression ⇒ A method to find relationship b/w dependant variable and a set of independant variables.

① → np.polyfit(x, y, deg) ↳ degree of freedom.

★ Random numbers random module → import random

random( ) → any random number b/w 0 to 1 (less than 1) <1.0

randint(a,b) → generated integer N such that a <= N <= b (a,b included)

randrange( ) → randrange(start, stop, step) (a, b, s)

Any number between a & b with step value s.

for math function → import math

Plots and Graphs
Data Visualisation.

HAWA MAHAL
PAGE NO.
DATE :

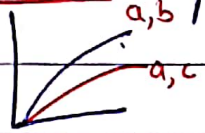
Library needed → import matplotlib.pyplot as pl
install it using pip install matplotlib

pyplot → collection of method to plot 2D graphs/plots, available under matplotlib.

matplotlib → in idle do a pip installation.
↳ in Anaconda, it is pre installed
↳ It is a high Quality plotting library of python.

plotting

1) line chart → pl.plot(a,b) a,b are arrays.



* (not mentioned in syllabus) still study it too.

in same plot if you want to plot another line use pl.plot(a,c)

attributes :→ you can change color of line, type of marker, its size etc

→ pl.plot(a,b, color = color code / color name, marker = marker code, markersize = integer,
markeredgecolor = color code / color name) ← also linestyle, linewidth etc

similarly for every graph you can add xlabel, ylabel, title etc

→ pl.xlabel("your label"), pl.ylabel("your label")

pl.title("title of chart")

→ these attributes and method are generally for all plots

2) scatter plots [using plot() / using scatter()] → scatter chart is a graph of scattered points b/w 02 data sets.

in plot method use "" linecolor and marker style in following manner ⇒ "r+", "b+", "b0" or any marker type without giving linestyle attributes you will get scattered chart. ex- pl.plot(a,b,"0")
ex- pl.plot(a,b,"r+")

pl.show() to show the plot

in scatter method → pl.scatter(a,b s=__, c=__, marker=__)

s → marker size
c → marker color
marker → marker style

ex → pl.scatter(a,b, marker="+")

✱ Creating Bar / histograms

Bar graph → Bars of different height are used to display the data → pl.bar(a,b)
first sequence forms x axis. second sequence forms y axis. } use xlabel, ylabel title methods to draw better.

→ pl.bar(a,b, width = floatval) → for common width
pl.bar(a,b, width = [0.5, .6, .8, .9]) for variable width.

plotting 02 bar in same plots, [a small trick]

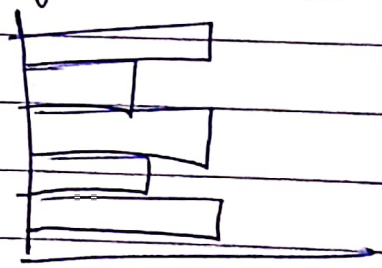
A = [2, 4, 6, 8] B = [3, 5, 7, 9] X = [1, 2, 3, 4]

plt.bar(X, A, color = 'red', width = 0.35) first bar

normally we plot → plt.bar(X, B, color = 'blue', width = 0.35) 2nd bar.
but change this as ⟹ plt.bar(X+0.35, B, color='b', width=0.35)

✱ if we plot in normal way bars will overlap to add the width of 1st graph in X array of 2nd graph

pl.bar(a,b) → now → pl.barh(a,b) will create a horizontal bar plot



✱ Histogram → df.hist(column = 'ar1') → simplest statement
continuous data set / no gaps in bar
↳ it will creates histogram of bin size = 10.

df.hist() is available under Dataframe and creates histogram for all numeric column if given as df.hist()

in pyplot it has following syntax

pl.hist(x, bins = ___, weight = ___, cumulative = T/F, color = ___, edgecolor = ___, histtype = ___)

when histtype → bar = histogram
→ step ⇒ frequency polygon.

pyplot contd...

few examples of histograms

Histogram ⭐ → pl.hist(x, bins = 30, cumulative = True)

frequency polygon → pl.hist (x, bins = 20, histtype = step)

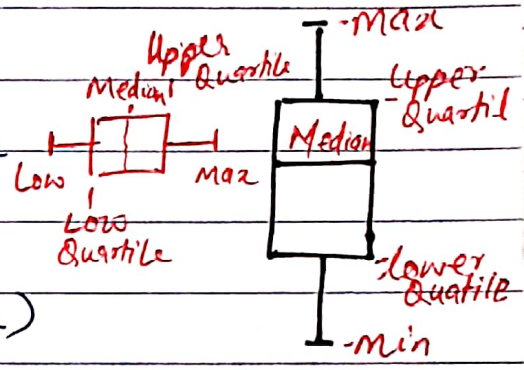
two ndarray x, Y in same plot → pl.hist ([x, Y])

~~horizontal orientation~~ → pl.hist (x, bins = 50, orientation = 'horizontal')

↳ horizontal orientation

Box Plot

Box plot displays
- min range value
- max range value
- upper Quartiles
- lower Quartiles
- median



pl.boxplot(Data, showmeans = True)

↑ Array ↑ will show Mean point also

properties

notch - <True/false>
- when True → notched box
- when False → rectangular box

Vertical
- vert = True
- vert = False

Meanline → shows mean as a line

Showbox → by default True →
- T
- ⊥

No box is shown when showbox is False

Showmeans

SQL and Database → import mysql.connector

install → pip install mysql-connetor-python

1 how to connect.

2 import package — import mysql.connector as sqltor

3 Open a connection — Mycon = sqltor.connect (host = "localhost", user = "root", passwd = "yourMySQLpassword", database = "yourdb")

4 create cursor — to test connection → if mycon.is_connected(): suitable msg
→ cursor1 = mycon.cursor()

5 Execute a Query → cursor1.execute("SQL Query as per Question")

It contains methods like - fetchall(), fetchone(), fetchmany(<n>), cur.rowcount

6 Extract data → works after fetching

data = cursor1.fetchall()
cursor1.fetchmany(5)
cursor1.fetchone()

data = cursor1.fetchall()
for row in data:
print (row)
if row

and proceed like tuple of python →

7 clean Environment → cursor1.close()
↳ mycon.close()

DJANGO pronounced Jango-h
OSS, web application framework
→ Instagram, pinterest
→ deep, dynamic sites with web enabled application.

Advantages of Using Django (It is written in Python)
- Easy and fast web application development
- Compatible with major OS and databases
- less coding required
- Easy to extend
- Possesses scalability

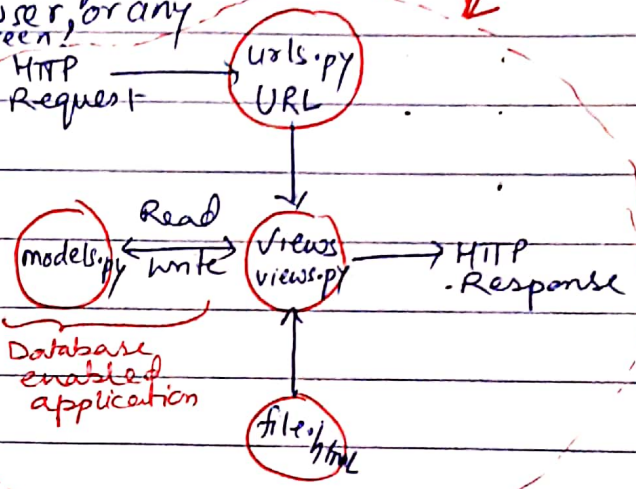
Backend: End which responds to the user's request usually any DBMS or RDBMS.

Frontend → user end
It send request
ex. Browser, or any user interface screen

HTTP Get Request :- a way of retrieving information from a web server through some URL.

HTTP Post Request :- a way to send data to server, may be via HTML forms.

How Django works.



→ Install the django in system → pip install django
→ Creating a project in Django (Proj is project) → django-admin startproject proj
→ run the Django server (127.0.0.1:8000)* or (localhost:8000) → manage.py runserver
→ Creating any application (myapp) → manage.py startapp myapp

Virtual environment, is not necessary but if any user creates it ensures co-existence of multiple python environment of different Django projects in isolated manner.
→ To create Virtual Environment → pip install virtualenv
Activate Virtual Environment → (venv\Scripts\activate)
virtualenv venv ↵ create venv as Virtual environment

file in outer project folder → manage.py

files in inner project folder
- __init__.py
- settings.py → most of settings
- urls.py → stores info of URLs
- wsgi.py

files in app folder
- admin.py → Configuration file for Django Admin
- apps.py → configuration file for app
- models.py → DB model defined here
- tests.py → app specific tests
- views.py → resolve http requests
- migrations/ → updates DB.

Get & Post Method

DJANGO. contd.

HAWA MAHAL
PAGE NO.
DATE :
11

02 Types of HTTP Request

Get	Post
→ limited data can be sent	→ large data can be sent
→ data sent in header	→ data sent in body of page
→ Not secure as data is visible in header	→ Secure as data not shown in URL.
→ Can be bookmarked	→ Can't be bookmarked
→ Request remains in browser history	→ do not remains in browser history

* CSV ⇒ comma seperated Values *

↓

Read & write in CSV

May be helpful in few programs

Django follows Model View Template MVT paradigm

app gets data from Model, View works on data and template displays the final processed data.

Society, law and Ethics

→ IPR- Intellectual Property Rights : Rights of the Owner of Information to decide how much info to be shared, exchanged or distributed & to decide cost/price for the share/distribute/Exchange info. Ensures distributor

why to protect →

Why ↑ It has to be protected for following reason
① It encourages creation of new s/w & ideas, improves.
② Ensures wide distribution of innovative idea/Techno.
③ promotes investment in National Economy

Solutions → Patent, Copyright, TradeMarks,

-Stealing -cheating -academic offence

Plagiarism → Stealing & pretending own property → It is unauthorized use of someone elses thoughts, Intellectual work and representing it as own work, without citing sources or original generator name.

→ To avoid plagiarism → GIVE DUE CREDIT TO OWNER.

* Plagiarism can be planned or may happen accidently.

→ DRM → Digital Rights Management :- Assets you own in digital form is called digital property/Estate.
DRM is a scheme that protects or controls the access to the copyrighted materials by using Technology. It can be done by encrypting, coding, password protections, time bound sharing, obtaining license, putting legal clauses, limiting the sharing, Anti-temper solutions.
Few threaths to DRM :- ① Digital s/w penetration tools.
② plagiarism or stealing.

→ Licencing :- It is a document that provides legal bindings and guidelines to the person to perform Digital Right mgmt and secure IPR. It includes rules like :-
① fair use of s/w ② Limitation of liability, warranty disclaimers & protections.
GPL → (GNU) → General public licence → free s/w licence means freedom for use, not necessarily free from price.
ex- (wordpress),