pandas

```
#Numpy is ideal for numerical data and array operations
#better for structured data (like tables) and provides more advanced
tools for data manipulation and analysis.
import pandas as pd#for the use of tables (data frames)

mydata1=["alice","bob","ram","charlie"]
ser1=pd.Series(mydata1)#series of with data gives according to the
index
print(ser1)
```

```
0      alice
1        bob
2        ram
3    charlie
dtype: object
```

```
mydata1=["alice","bob","ram","charlie"]
roll=[1,2,3,4]
ser2=pd.Series(mydata1,index=roll)#here we can specifies the index
data by using index
print(ser2)
```

```
1      alice
2        bob
3        ram
4    charlie
dtype: object
```

```
ser2[1]
```

```
'alice'
```

```
ser2.to_csv(r"C:\Users\DELL\Downloads\my_python\mydata1.csv")#which
create file and store into the given path with extention csv(if error
occurs mention r in front of it to recignize it as a path
```

daraframes

```
data={"names":["alice","bob","charlie"],
      "age":[19,19,20],
      "city":["aaa","bbb","ccc"]
     }
data#dictionary creation
```

```
{'names': ['alice', 'bob', 'charlie'],
 'age': [19, 19, 20],
 'city': ['aaa', 'bbb', 'ccc']}
```

```
d1=pd.DataFrame(data)#dataframes are used to create the table by this
data
print(d1)
```

```
     names  age city
0    alice   19  aaa
1      bob   19  bbb
2  charlie   20  ccc
```

```
d1.to_csv(r"C:\Users\DELL\Downloads\my_python\data.csv")#create a file
and save it into a give path
```

```
df=pd.read_csv(r"C:\Users\DELL\Downloads\my_python\sample.csv")#to
read the file use function read_csv
```

```
df.head()#after file reading which is used to create/veiw the table it
only gives first 5 rows only
```

```
  name   dept   sem1   sem2  sem3
0   ram    ISE    6.7   8.9   7.3
1   sam    ISE    6.7   8.9   7.3
2   mam    ISE    6.7   8.9   7.3
3   oam    ISE    6.7   NaN   7.3
4   bam    ISE    6.7   8.9   7.3
```

loading large files

```
d1.to_csv(r"C:\Users\DELL\Downloads\my_python\diabetcsvsmall.csv")

df1=pd.read_csv(r"C:\Users\DELL\Downloads\my_python\
diabetcsvsmall.csv")#dataframe name df1
df1.head()#by head() we get only the first 5 rows only
```

```
   preg  plas  pres  skin  insu  mass   pedi  age            class
0   6.0   148  72.0  35.0     0  33.6  0.627   50  tested_positive
1   1.0    85  66.0  29.0     0  26.6  0.351   31  tested_negative
2   8.0   183  64.0   0.0     0  23.3  0.672   32  tested_positive
3   1.0    89  66.0  23.0    94  28.1  0.167   21  tested_negative
4   0.0   137  40.0  35.0   168  43.1  2.288   33  tested_positive
```

```
df1.tail()#which gives the last 5 row in data set
```

```
     preg  plas  pres  skin  insu  mass   pedi  age            class
97    1.0    71  48.0   NaN    76  20.4  0.323   22  tested_negative
98    6.0    93  50.0  30.0    64  28.7  0.356   23  tested_negative
99    NaN   122  90.0  51.0   220  49.7  0.325   31  tested_positive
100   1.0   163  72.0   0.0     0  39.0  1.222   33  tested_positive
101   1.0   151  60.0   0.0     0  26.1  0.179   22  tested_negative
```

accessing

```
df1.loc[1:10]#location ,if we want to get more then 5 row use loc and menton range of rows
```

	preg	plas	pres	skin	insu	mass	pedi	age	class
1	1.0	85	66.0	29.0	0	26.6	0.351	31	tested_negative
2	8.0	183	64.0	0.0	0	23.3	0.672	32	tested_positive
3	1.0	89	66.0	23.0	94	28.1	0.167	21	tested_negative
4	0.0	137	40.0	35.0	168	43.1	2.288	33	tested_positive
5	5.0	116	74.0	0.0	0	25.6	0.201	30	tested_negative
6	3.0	78	50.0	32.0	88	31.0	0.248	26	tested_positive
7	10.0	115	0.0	0.0	0	35.3	0.134	29	tested_negative
8	2.0	197	70.0	45.0	543	30.5	0.158	53	tested_positive
9	8.0	125	96.0	0.0	0	0.0	0.232	54	tested_positive
10	4.0	110	92.0	0.0	0	37.6	0.191	30	tested_negative

```
df1.loc[1:10,"age"]#also we can access the data we want by specifing the column name
```

```
1     31
2     32
3     21
4     33
5     30
6     26
7     29
8     53
9     54
10    30
Name: age, dtype: int64
```

```
df1.iloc[1:10,3:8]#whict gives the inedx wise data ,3-column:8rows is index range iloc[row_range:column_range]
```

	skin	insu	mass	pedi	age
1	29.0	0	26.6	0.351	31
2	0.0	0	23.3	0.672	32
3	23.0	94	28.1	0.167	21
4	35.0	168	43.1	2.288	33
5	0.0	0	25.6	0.201	30
6	32.0	88	31.0	0.248	26
7	0.0	0	35.3	0.134	29
8	45.0	543	30.5	0.158	53
9	0.0	0	0.0	0.232	54

feature engineering

insu ,age ,mass ,pedi,preg==>independent(features)
class(positive /nagative)===>dependent (target)

```
df1.rename(columns={"plas":"glucose"})#by rename it will not change
the original but show the duplicate of it
```

```
     preg  glucose  pres  skin  insu  mass   pedi  age
class
0     6.0      148  72.0  35.0     0  33.6  0.627   50
tested_positive
1     1.0       85  66.0  29.0     0  26.6  0.351   31
tested_negative
2     8.0      183  64.0   0.0     0  23.3  0.672   32
tested_positive
3     1.0       89  66.0  23.0    94  28.1  0.167   21
tested_negative
4     0.0      137  40.0  35.0   168  43.1  2.288   33
tested_positive
..    ...      ...   ...   ...   ...   ...    ...  ...              ..
.
97    1.0       71  48.0   NaN    76  20.4  0.323   22
tested_negative
98    6.0       93  50.0  30.0    64  28.7  0.356   23
tested_negative
99    NaN      122  90.0  51.0   220  49.7  0.325   31
tested_positive
100   1.0      163  72.0   0.0     0  39.0  1.222   33
tested_positive
101   1.0      151  60.0   0.0     0  26.1  0.179   22
tested_negative

[102 rows x 9 columns]
```

df1.head()

```
df1.rename(columns={"plas":"glucose"},inplace=True)#by rename with the
use of inplace=true to change original data column name
df1.head()
```

```
   preg  glucose  pres  skin  insu  mass   pedi  age            class
0   6.0      148  72.0  35.0     0  33.6  0.627   50  tested_positive
1   1.0       85  66.0  29.0     0  26.6  0.351   31  tested_negative
2   8.0      183  64.0   0.0     0  23.3  0.672   32  tested_positive
3   1.0       89  66.0  23.0    94  28.1  0.167   21  tested_negative
4   0.0      137  40.0  35.0   168  43.1  2.288   33  tested_positive
```

```
df1['glc_mmol']=df1['glucose']/18.01#create a new column by existing
column
df1.head()
```

```
   preg  glucose  pres  skin  insu  mass   pedi  age            class
\
```

```
0   6.0      148  72.0  35.0     0  33.6  0.627   50  tested_positive

1   1.0       85  66.0  29.0     0  26.6  0.351   31  tested_negative

2   8.0      183  64.0   0.0     0  23.3  0.672   32  tested_positive

3   1.0       89  66.0  23.0    94  28.1  0.167   21  tested_negative

4   0.0      137  40.0  35.0   168  43.1  2.288   33  tested_positive


    glc_mmol
0   8.217657
1   4.719600
2  10.161022
3   4.941699
4   7.606885
```

filter and group

```
fil_age30=df1[df1['age']<30]#this which just a cop of the df1 with
some condition
fil_age30.head()

    preg  glucose  pres  skin  insu  mass   pedi  age            class
\
3    1.0       89  66.0  23.0    94  28.1  0.167   21  tested_negative

6    3.0       78  50.0  32.0    88  31.0  0.248   26  tested_positive

7   10.0      115   0.0   0.0     0  35.3  0.134   29  tested_negative

20   3.0      126  88.0  41.0   235  39.3  0.704   27  tested_negative

23   9.0      119  80.0  35.0     0  29.0  0.263   29  tested_positive


    glc_mmol
3   4.941699
6   4.330927
7   6.385341
20  6.996113
23  6.607440

gluc100=df1[df1['glucose']>100]
gluc100.head(5)#we can also specify the data size what we want

   preg  glucose  pres  skin  insu  mass   pedi  age            class
\
0   6.0      148  72.0  35.0     0  33.6  0.627   50  tested_positive
```

```
2   8.0      183  64.0   0.0     0  23.3  0.672   32  tested_positive

4   0.0      137  40.0  35.0   168  43.1  2.288   33  tested_positive

5   5.0      116  74.0   0.0     0  25.6  0.201   30  tested_negative

7  10.0      115   0.0   0.0     0  35.3  0.134   29  tested_negative


    glc_mmol
0   8.217657
2  10.161022
4   7.606885
5   6.440866
7   6.385341
```

create a filter dataset which as only the rows with age between 20 and 30

```
age20_30=df1[(df1['age']>20) & (df1['age']<30)]#to combine the tw
condition to copy the
age20_30.head()
```

```
    preg  plas  pres  skin  insu  mass   pedi  age            class
3    1.0    89  66.0  23.0    94  28.1  0.167   21  tested_negative
6    3.0    78  50.0  32.0    88  31.0  0.248   26  tested_positive
7   10.0   115   0.0   0.0     0  35.3  0.134   29  tested_negative
20   3.0   126  88.0  41.0   235  39.3  0.704   27  tested_negative
23   9.0   119  80.0  35.0     0  29.0  0.263   29  tested_positive
```

grouping and deriving result

```
g1=df1.groupby('class')['age'].mean()#by grouping we can also analyse
the data set with mean [target--features(mean)]
g1
```

```
class
tested_negative    31.238095
tested_positive    40.589744
Name: age, dtype: float64
```

```
g1=df1.groupby('class')['age'].max()#by grouping we can also analyse
the data set with max
g1
```

```
class
tested_negative    60
tested_positive    60
Name: age, dtype: int64
```

```
g=df1.groupby('class')['age'].min()#by grouping we can also analyse
the data set with min
g
```

```
class
tested_negative    21
tested_positive    25
Name: age, dtype: int64
```

```
g1=df1.groupby('class')['insu'].mean()#by grouping we can also analyse
the data set with mean
g1
```

```
class
tested_negative     52.571429
tested_positive    114.692308
Name: insu, dtype: float64
```

cleaning data

handling null

```
df1.isnull().sum()#to find null value & how many null values
init,isnull-true/false,sum-total null values in row
```

```
preg     1
plas     0
pres     1
skin     1
insu     0
mass     1
pedi     1
age      0
class    0
dtype: int64
```

```
df1.info()#which give how many not null rows/information about it
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 96 entries, 0 to 101
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   preg    96 non-null     float64
 1   plas    96 non-null     int64
 2   pres    96 non-null     float64
 3   skin    96 non-null     float64
 4   insu    96 non-null     int64
 5   mass    96 non-null     float64
 6   pedi    96 non-null     float64
 7   age     96 non-null     int64
```

```
 8   class   96 non-null     object
dtypes: float64(5), int64(3), object(1)
memory usage: 7.5+ KB
```

```
df1.dropna(inplace=True)#which is used to drop/remove the null values
not in original,it presents by removing null values in table
```

```
df1.isnull().sum()
```

```
preg     0
plas     0
pres     0
skin     0
insu     0
mass     0
pedi     0
age      0
class    0
dtype: int64
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 98 entries, 0 to 101
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   preg    98 non-null     float64
 1   plas    98 non-null     int64
 2   pres    98 non-null     float64
 3   skin    98 non-null     float64
 4   insu    98 non-null     int64
 5   mass    98 non-null     float64
 6   pedi    98 non-null     float64
 7   age     98 non-null     int64
 8   class   98 non-null     object
dtypes: float64(5), int64(3), object(1)
memory usage: 7.7+ KB
```

handle the duplicate

```
df1.drop_duplicates(inplace=True)#which removes the duplicates in data
affects only inplace=True
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 96 entries, 0 to 101
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
```

```
 0   preg    96 non-null     float64
 1   plas    96 non-null     int64
 2   pres    96 non-null     float64
 3   skin    96 non-null     float64
 4   insu    96 non-null     int64
 5   mass    96 non-null     float64
 6   pedi    96 non-null     float64
 7   age     96 non-null     int64
 8   class   96 non-null     object
dtypes: float64(5), int64(3), object(1)
memory usage: 7.5+ KB
```

```
df2=pd.read_excel(r"C:\Users\DELL\Downloads\my_python\
diabetes.xlsx",sheet_name="Hello")#we can aslo read the other file
format by specifing name
df2
```

```
Empty DataFrame
Columns: [hello, guys, how, are ]
Index: []
```

```
df2=pd.read_excel(r"C:\Users\DELL\Downloads\my_python\
diabetes.xlsx",sheet_name="dora")#when the file have two or more page
is we want to get the 2nd sheet then we use thi form
df2
```

```
  Dead Alive
0  yes    no
1  yes    no
2  yes    no
3  yes    no
4  yes    no
```

loading text file

```
df3=pd.read_csv(r"C:\Users\DELL\Downloads\my_python\
grades.txt")#without seperator it will treat as all data are same word
df3
```

```
   Names Initials SEM1 SEM2 SEM3 Grade
0                  Joe K 9.8 10 9.9 A+
1               Rajesh M 8.9 9.1 9.3 A
2               Kissan V 9.9 9.3 9.2 A
3                   Mary N 7.7 8 7.1 B
4                Jeen K 9.8 9.1 9.9 A+
5                  Raj M 8.9 9.1 9.3 A
6                Hassan V 9.9 9 9.2 A
7                  Mari N 7.7 8 7.1 B
8                Jess K 9.8 9.1 9.9 A+
9                 Rajini M 7 9.1 9.3 A
```

```
10               Kiran V 9.9 9.3 9.2 A
11                  Maya N 7.7 8 7.1 B
12              Jolin K 9.8 9.1 9.9 A+
13                  Riya M 8 9.1 9.3 A
14                Sana V 9.9 9.3 9.2 A
15                   Mark N 7.7 8 7 B
```

```
df3=pd.read_csv(r"C:\Users\DELL\Downloads\my_python\grades.txt",sep='
')#by using sep(seperator) it will grouped it
df3
```

	Names	Initials	SEM1	SEM2	SEM3	Grade
0	Joe	K	9.8	10.0	9.9	A+
1	Rajesh	M	8.9	9.1	9.3	A
2	Kissan	V	9.9	9.3	9.2	A
3	Mary	N	7.7	8.0	7.1	B
4	Jeen	K	9.8	9.1	9.9	A+
5	Raj	M	8.9	9.1	9.3	A
6	Hassan	V	9.9	9.0	9.2	A
7	Mari	N	7.7	8.0	7.1	B
8	Jess	K	9.8	9.1	9.9	A+
9	Rajini	M	7.0	9.1	9.3	A
10	Kiran	V	9.9	9.3	9.2	A
11	Maya	N	7.7	8.0	7.1	B
12	Jolin	K	9.8	9.1	9.9	A+
13	Riya	M	8.0	9.1	9.3	A
14	Sana	V	9.9	9.3	9.2	A
15	Mark	N	7.7	8.0	7.0	B

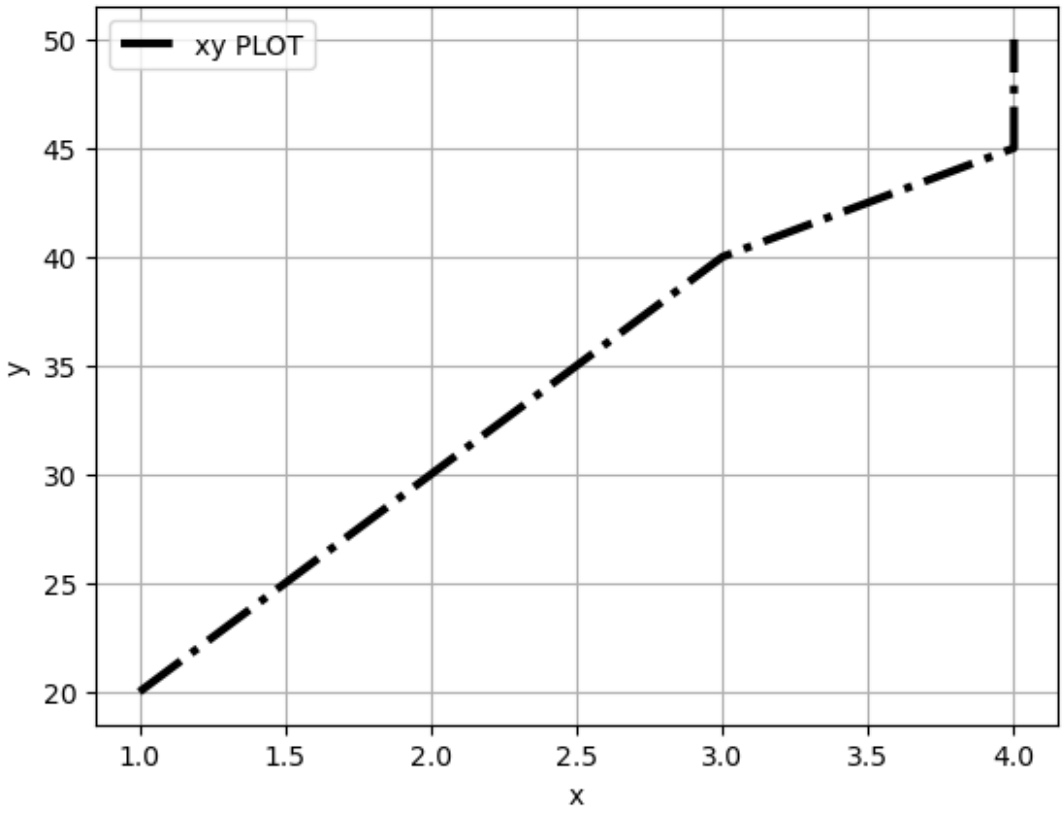
```
df3['sem1_int']=df3['SEM1'].astype(int)#to convert the datatype
df3
```

	Names	Initials	SEM1	SEM2	SEM3	Grade	sem1_int
0	Joe	K	9.8	10.0	9.9	A+	9
1	Rajesh	M	8.9	9.1	9.3	A	8
2	Kissan	V	9.9	9.3	9.2	A	9
3	Mary	N	7.7	8.0	7.1	B	7
4	Jeen	K	9.8	9.1	9.9	A+	9
5	Raj	M	8.9	9.1	9.3	A	8
6	Hassan	V	9.9	9.0	9.2	A	9
7	Mari	N	7.7	8.0	7.1	B	7
8	Jess	K	9.8	9.1	9.9	A+	9
9	Rajini	M	7.0	9.1	9.3	A	7
10	Kiran	V	9.9	9.3	9.2	A	9
11	Maya	N	7.7	8.0	7.1	B	7
12	Jolin	K	9.8	9.1	9.9	A+	9
13	Riya	M	8.0	9.1	9.3	A	8
14	Sana	V	9.9	9.3	9.2	A	9
15	Mark	N	7.7	8.0	7.0	B	7

matplotlib

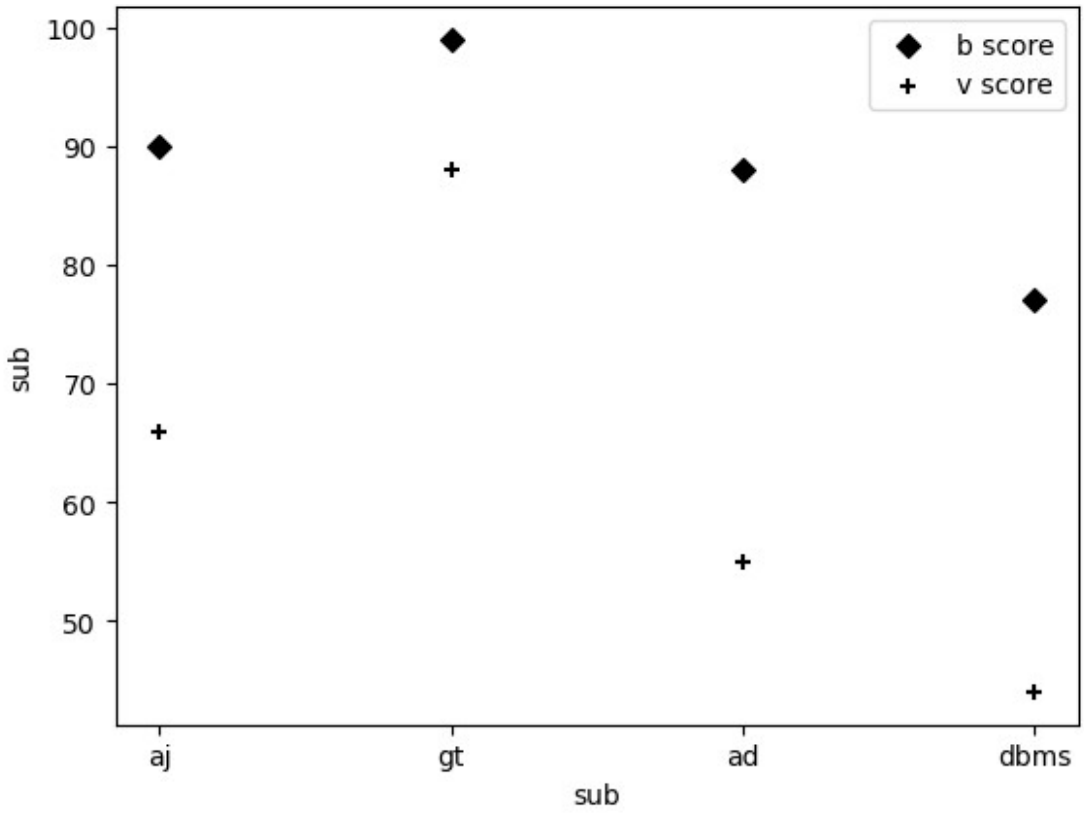
```
import matplotlib.pyplot as plt
x=[1,2,3,4,4]
y=[20,30,40,45,50,]
plt.plot(x,y,color='k',label="xy PLOT",linestyle="-.",linewidth=3)
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.legend()
```

```
<matplotlib.legend.Legend at 0x24b573849d0>
```



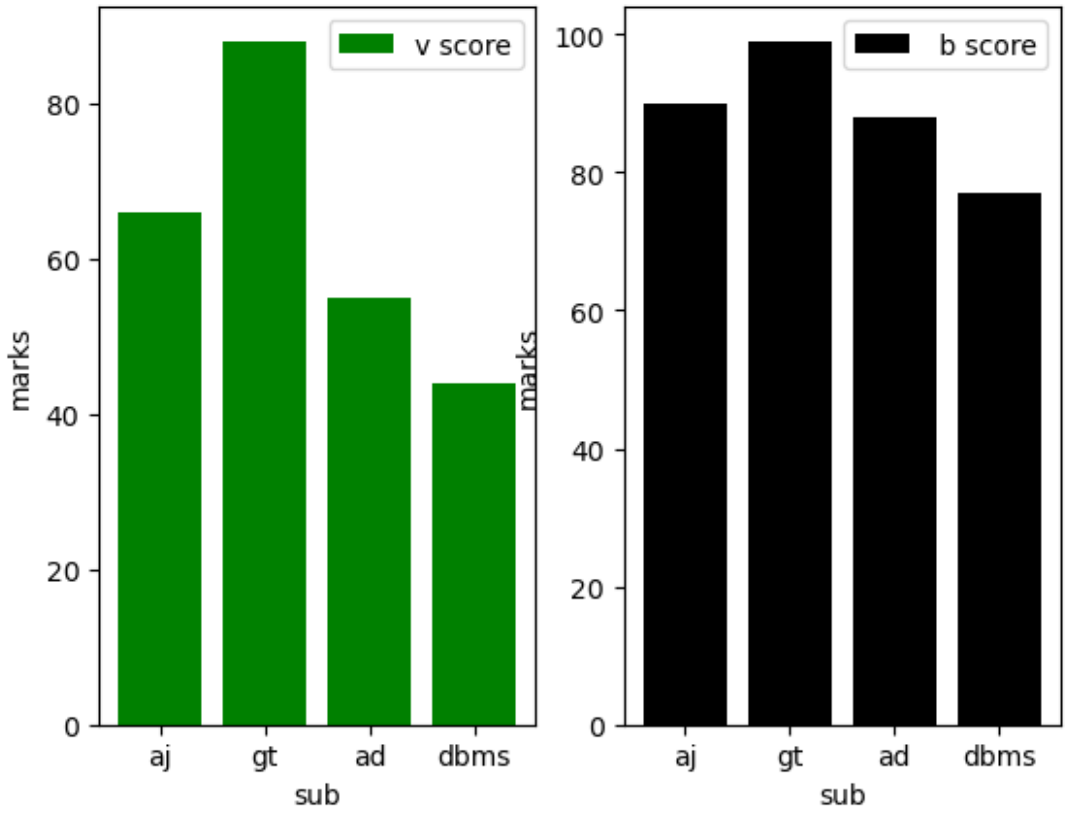

```
sub=["aj","gt","ad","dbms"]
b=[90,99,88,77]
v=[66,88,55,44]
plt.scatter(sub,b,color='k',label="b score",marker="D")
plt.scatter(sub,v,color='k',label="v score",marker="+")
plt.xlabel("sub")
plt.ylabel("sub")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x24b573b4fd0>
```



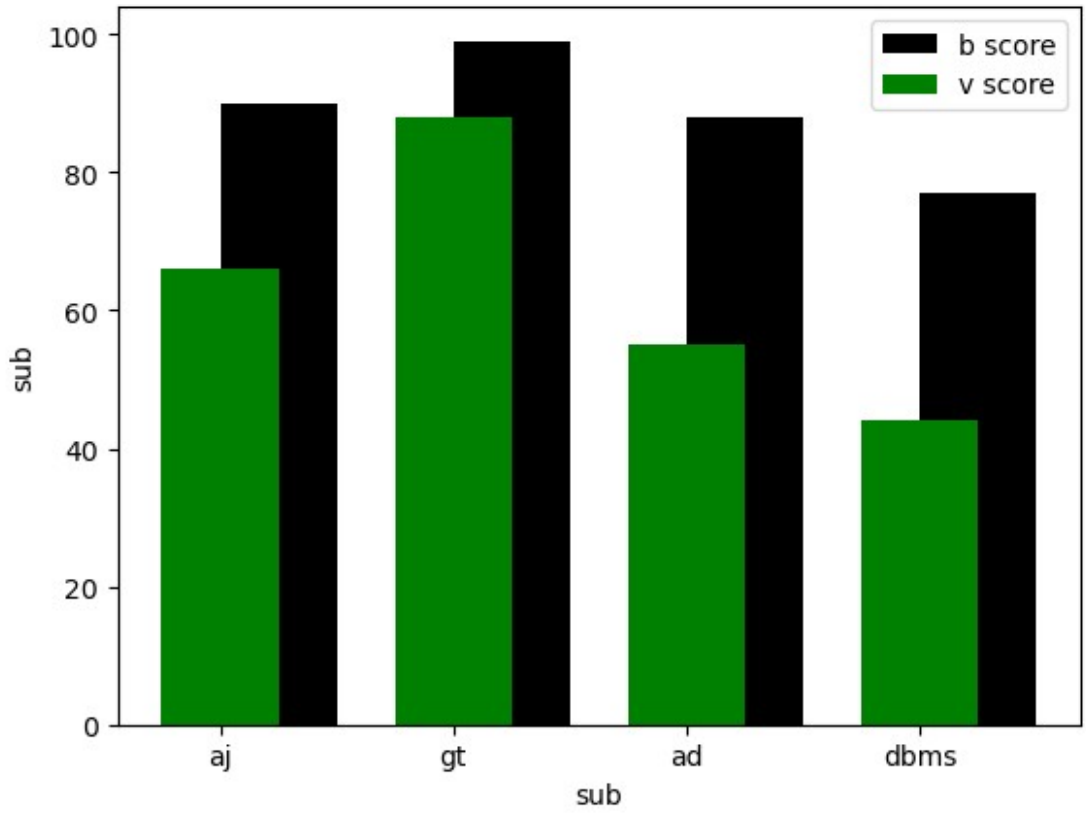

```
sub=["aj","gt","ad","dbms"]
b=[90,99,88,77]
v=[66,88,55,44]
plt.subplot(1,2,1)
plt.bar(sub,v,color='green',label="v score")
plt.xlabel("sub")
plt.ylabel("marks")
plt.legend()
plt.subplot(1,2,2)
plt.bar(sub,b,color='k',label=" b score")
plt.xlabel("sub")
plt.ylabel("marks")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x24b5ececa90>
```




```
sub=["aj","gt","ad","dbms"]
b=[90,99,88,77]
v=[66,88,55,44]
plt.bar(sub,b,color='k',label="b score",width=0.5,align="edge")
plt.bar(sub,v,color='green',label="v score",width=0.5,align="center")
plt.xlabel("sub")
plt.ylabel("sub")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x24b5db84590>
```




```
import numpy as np
a=np.array([25,60,5,10])
labe=["a","b","c","d"]
color=["pink",'k',"coral","yellow"]
plt.pie(a,labels=labe,colors=color)
plt.
```