Python Ka Chilla with Baba Aammar

Participant name: Abbas hyder

city : Khairpur Mirs Sindh

How to Use Jupyter Note Book

Basics Of Python

1- My first program

In [ ]:
```
#My first Program
print("hello abbas")
print(2+5)
```

```
hello abbas
7
```

02_Operators

In [ ]:
```
print(6/2)
print(10+20)
print(6//2)
print(2**3)

print(2**3/2+2-1)
```

```
3.0
30
3
8
5.0
```

PEMDAS parenthesis Exponent multiply divide add substract

03-Variables

In [ ]:
```
x=3
print(x)
x=x+2
print(x)
fruit="mangoes"
print(fruit)
print(type(fruit))
my_box="Apple","Mangoes","RedBalls"
print(my_box)
print(type(my_box))
letter='A'
print(type(letter))
```

```
3
```

```
5
mangoes
<class 'str'>
('Apple', 'Mangoes', 'RedBalls')
<class 'tuple'>
<class 'str'>
```

04_input_variable

In [ ]:

```
x=input("enter any number = ")
print(x)
name=input("plz enter your name : ")
age=input("plz enter your age : ")
greetings="hello!"
print(greetings,name, "you are clever")
```

```
5
hello! abbas you are clever
```

05_conditional_logic

In [ ]:

```
print(4<5)
max_age=10
age=input("plz enter your age")
age=int(age)
print(age==max_age)
```

```
True
False
```

06-type_conversion

In [ ]:

```
x=5                 #int
y= 5.5              #float
z="hello"           #String
print(type(x))
#implicoit type conversion
x=x+y
print(x, type(x))

#explicit type coversion
age=input("enter your age : ")
age=int(age)
print(age,type(age))

name=input("Enter Your Name")
print(name,type(str(name)))
```

```
<class 'int'>
10.5 <class 'float'>
15 <class 'int'>
abbas <class 'str'>
```

07_if_else_elif

In [ ]:

```
max_marks=80
marks=input("Enter your marks")
marks=int(marks)
```

```
if marks == max_marks:
    print("A1 Grade")
    print("congrats")
elif marks < 80:
    print("B Grade")
else:
    print("C Grade")
```

```
B Grade
```

08_function

In [ ]:

```
#defining a function
#1
# def my_function():
#     print("I am learning Python")
#     print("I am learning Python")
#     print("I am learning Python")

# my_function()

#2
# def my_function2():
#     text="Iam learning Python with ammar"
#     print(text)
#     print(text)
#     print(text)


# my_function2()


#3
def my_function3(text):
    print(text)
    print(text)
    print(text)

my_function3("Hello abbas here")



# defining a function with if else elif
def school_calculator(age):
    if age == 5:
        print("hammad can join the school")
    elif age >  5:
        print("hammad should join college")
    else:
        print("Hammad is Still a child")

school_calculator(6)

# defining a function which return vale
def future_age(age):
    new_age=age+10
    return new_age
```

```
future_predictade_age = future_age(5)
print(future_predictade_age)
```

```
Hello abbas here
Hello abbas here
Hello abbas here
hammad should join college
15
```

09_Loops

In [ ]:
```
#while loop
# x=0
# while(x<5):
#     print(x)
#     x=x+1

#for loop
# for i in range(5,10):
#     print(i)

days=["Mon","tue","Wed","Thur","fri","Sat","Sun"]
for i in days:
    if(i=="fri"):continue
    print(i)
```

```
Mon
tue
Wed
Thur
Sat
Sun
```

-indexing

In [ ]:
```
a="Samosa Pakora"
a
```

Out[ ]:
```
'Samosa Pakora'
```

In [ ]:
```
#length of string
len(a)
```

Out[ ]:
```
13
```

In [ ]:
```
a[0]
```

Out[ ]:
```
'S'
```

In [ ]:
```
a[5]
```

Out[ ]:
```
'a'
```

In [ ]:
```
a[7]
```

Out[ ]:
```
'P'
```

In [ ]:
```
a[0:5]
```

Out[ ]:
```
'Samos'
```

In [ ]:
```
a[0:5]
```

Out[ ]:
```
'Samos'
```

In [ ]:
```
a[0:5]
```

Out[ ]:
```
'Samos'
```

In [ ]:
```
a[-6:13]
```

Out[ ]:
```
'Pakora'
```

In [ ]:
```
a[0:13]
```

Out[ ]:
```
'Samosa Pakora'
```

-String Methods

In [ ]:
```
food="biryani"
food
```

Out[ ]:
```
'biryani'
```

In [ ]:
```
food.capitalize()
```

Out[ ]:
```
'Biryani'
```

In [ ]:
```
food.upper()
```

Out[ ]:
```
'BIRYANI'
```

In [ ]:
```
food.replace("b","sh")
```

Out[ ]:
```
'shiryani'
```

Counting a specific alphabet in string

In [ ]:
```
name = "Abbas hdyer soomro"
name
```

Out[ ]:
```
'Abbas hdyer soomro'
```

In [ ]:
```
name.count("o")
```

Out[ ]:
```
3
```

In [ ]:
```
name.count("b")
```

Out[ ]:
```
2
```

finding a index number in string

In [ ]:
```
b="Helllo I am Abbas Hyder"
b.find("y")
```

Out[ ]:
```
19
```

How to spilit a string

In [ ]:
```
z="hello I am Abbas I like C++, Java ,Python, C#, kotlin"
z
```

Out[ ]:
```
'hello I am Abbas I like C++, Java ,Python, C#, kotlin'
```

In [ ]:
```
z.split(",")
```

Out[ ]:
```
['hello I am Abbas I like C++', ' Java ', 'Python', ' C#', ' kotlin']
```

-Basic data structures in python

1- Tuple

2- list

3- Dictionary

4- Set

Tuple

- ordered collection of Elements
- enclosed in round braces () parenthesis
- different kind of Elements Can be stored
- once elements are store you can not change (Un Mutatable)

In [ ]:
```
tup1=(2,"python",True,5.5)
tup1
```

Out[ ]:
```
(2, 'python', True, 5.5)
```

In [ ]:
```
len(tup1)
```

Out[ ]:
```
4
```

In [ ]:
```
tup1[1]
```

Out[ ]:
```
'python'
```

In [ ]:
```
tup1[3]
```

Out[ ]:
```
5.5
```

In [ ]:
```
#last element is exclusive
tup1[0:3]
```

Out[ ]:
```
(2, 'python', True)
```

In [ ]:
```
tup1[0:4]
```

Out[ ]:
```
(2, 'python', True, 5.5)
```

In [ ]:
```
tup2=('apple',6,False,8.9)
tup2
```

Out[ ]:
```
('apple', 6, False, 8.9)
```

In [ ]:
```
#concatenate
tup1 + tup2
```

Out[ ]:
```
(2, 'python', True, 5.5, 'apple', 6, False, 8.9)
```

In [ ]:
```
tup1*2+tup2
```

Out[ ]:
```
(2, 'python', True, 5.5, 2, 'python', True, 5.5, 'apple', 6, False, 8.9)
```

In [ ]:
```
tup3=(40,50,60,70,80)
min(tup3)
```

Out[ ]:
```
40
```

In [ ]:
```
max(tup3)
```

Out[ ]:
```
80
```

-List

- ordered collection of elements
- enclosed in [] square brackets
- mutatable means you can change the values

In [ ]:
```
list1=[2,"abbas",True]
list1
```

Out[ ]:
```
[2, 'abbas', True]
```

In [ ]:
```
type(list1)
```

Out[ ]:
```
list
```

In [ ]:
```
len(list1)
```

Out[ ]:
```
3
```

In [ ]:
```
list1[1]
```

Out[ ]:
```
'abbas'
```

In [ ]:
```
list2=[3,5,"Abbas","codanics",427,5.5,False]
list2
```

Out[ ]:
```
[3, 5, 'Abbas', 'codanics', 427, 5.5, False]
```

In [ ]:
```
list1+list2
```

Out[ ]:
```
[2, 'abbas', True, 3, 5, 'Abbas', 'codanics', 427, 5.5, False]
```

In [ ]:
```
list1 *2
```

Out[ ]:
```
[2, 'abbas', True, 2, 'abbas', True]
```

In [ ]:
```
list1 *2
```

Out[ ]:
```
[2, 'abbas', True, 2, 'abbas', True]
```

In [ ]:
```
list1
```

Out[ ]:
```
[2, 'abbas', True]
```

In [ ]:
```
list1.count("adnan")
```

Out[ ]:
```
0
```

In [ ]:
```
list3=[2,5,3,66,78,4,9,66,7,88,88,99]
list3
```

Out[ ]:
```
[2, 5, 3, 66, 78, 4, 9, 66, 7, 88, 88, 99]
```

In [ ]:
```
len(list3)
```

Out[ ]:
```
12
```

In [ ]:
```
list3.sort()
list3
```

Out[ ]:
```
[2, 3, 4, 5, 7, 9, 66, 66, 78, 88, 88, 99]
```

In [ ]:
```
#repeat a list
list3*2
```

Out[ ]:
```
[2,
 3,
 4,
 5,
 7,
 9,
 66,
 66,
 78,
 88,
 88,
 99,
 2,
 3,
 4,
 5,
 7,
 9,
 66,
 66,
 78,
 88,
```

```
88,
99]
```

-Dictionaries

- an unordered collection of elements
- key and value
- enclosed in curely bracket{}
- mutatable value can be change

In [ ]:
```
#food and their prices
d1={"samosa":10,"pakora":100,"Raita":20,"salad":40,"chickenRoll":60}
d1
```

Out[ ]:
```
{'samosa': 10, 'pakora': 100, 'Raita': 20, 'salad': 40, 'chickenRoll': 60}
```

In [ ]:
```
type(d1)
```

Out[ ]:
```
dict
```

In [ ]:
```
#extract data
key1=d1.keys()
key1
```

Out[ ]:
```
dict_keys(['samosa', 'pakora', 'Raita', 'salad', 'chickenRoll'])
```

In [ ]:
```
value1=d1.values()
value1
```

Out[ ]:
```
dict_values([10, 100, 20, 40, 60])
```

In [ ]:
```
#addding a new element
d1["tiki"]=15
d1
```

Out[ ]:
```
{'samosa': 10,
 'pakora': 100,
 'Raita': 20,
 'salad': 40,
 'chickenRoll': 60,
 'tiki': 15}
```

In [ ]:
```
#update the value
d1["samosa"]=15
d1
```

Out[ ]:
```
{'samosa': 15,
 'pakora': 100,
 'Raita': 20,
 'salad': 40,
```

```
 'chickenRoll': 60,
 'tiki': 15}
```

In [ ]:
```
d2={"milk":100,"chocolate":70,"eggs":120}
d2
```

Out[ ]:
```
{'milk': 100, 'chocolate': 70, 'eggs': 120}
```

In [ ]:
```
#concatinate
d1.update(d2)
d1
```

Out[ ]:
```
{'samosa': 15,
 'pakora': 100,
 'Raita': 20,
 'salad': 40,
 'chickenRoll': 60,
 'tiki': 15,
 'milk': 100,
 'chocolate': 70,
 'eggs': 120}
```

In [ ]:
```
d1.get("milk")
```

Out[ ]:
```
100
```

In [ ]:
```
d1.items()
```

Out[ ]:
```
dict_items([('samosa', 15), ('pakora', 100), ('Raita', 20), ('salad', 40), ('chickenRol
l', 60), ('tiki', 15), ('milk', 100), ('chocolate', 70), ('eggs', 120)])
```

-Sets

- unordered and unindexed collection of elements
- curly braces are used {}
- No duplication allowed

In [ ]:
```
#boolean value not prints in set
set1={1,2.3,"abbas","adnan",True}
set1
```

Out[ ]:
```
{1, 2.3, 'abbas', 'adnan'}
```

In [ ]:
```
set1.add("Hyder")
set1
```

Out[ ]:
```
{1, 2.3, 'Hyder', 'abbas', 'adnan'}
```

In [ ]:
```
set1.add("abbas")
```

```
set1
```

Out[ ]:
```
{1, 2.3, 'Hyder', 'abbas', 'adnan'}
```

BMI

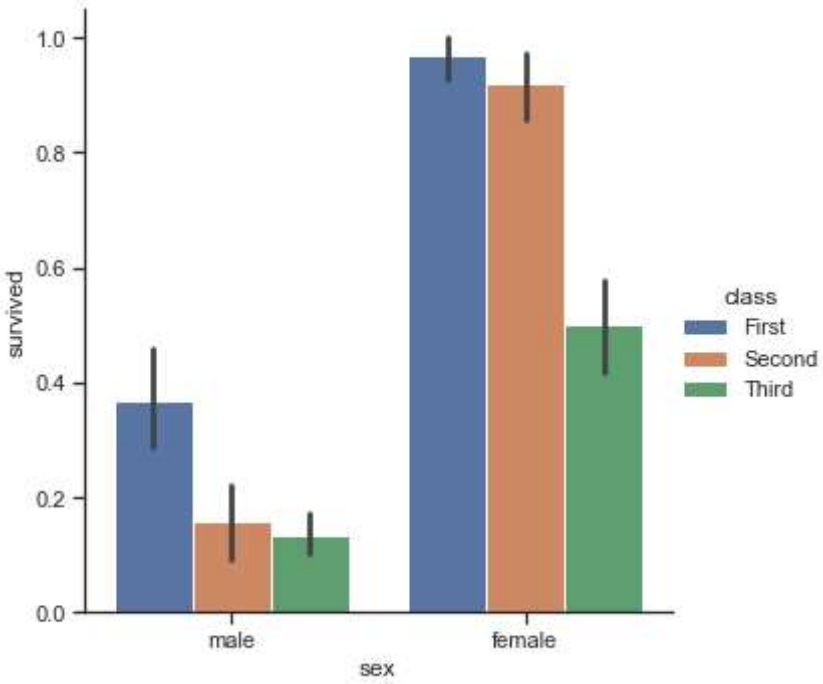
In [ ]:
```
weight=input("Enter weight in kg")
height=input("Enter height in Meters")
weight=float(weight)
height=float(height)
BMI= weight/(height**2)

print("your Bmi is = ",BMI)
```

```
your Bmi is =  5.888888888888889
```

DATA VISUALIZATION WEEK

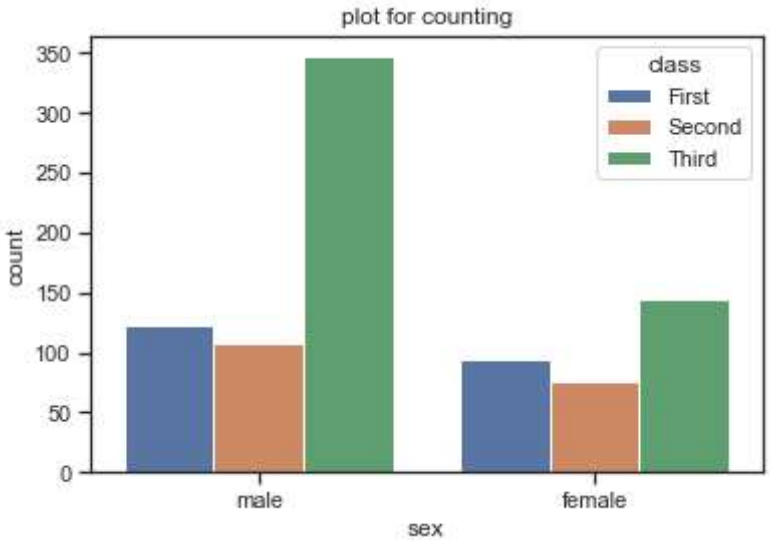
Catagorial variables

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="ticks",color_codes=True)
titanic = sns.load_dataset("titanic")
sns.catplot(x="sex",y="survived",hue="class",kind="bar", data=titanic)
plt.show()
```



In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="ticks",color_codes=True)
titanic = sns.load_dataset("titanic")
```

```
p1=sns.countplot(x='sex', data=titanic,hue="class")
p1.set_title("plot for counting")
plt.show()
```



ScatterPlot

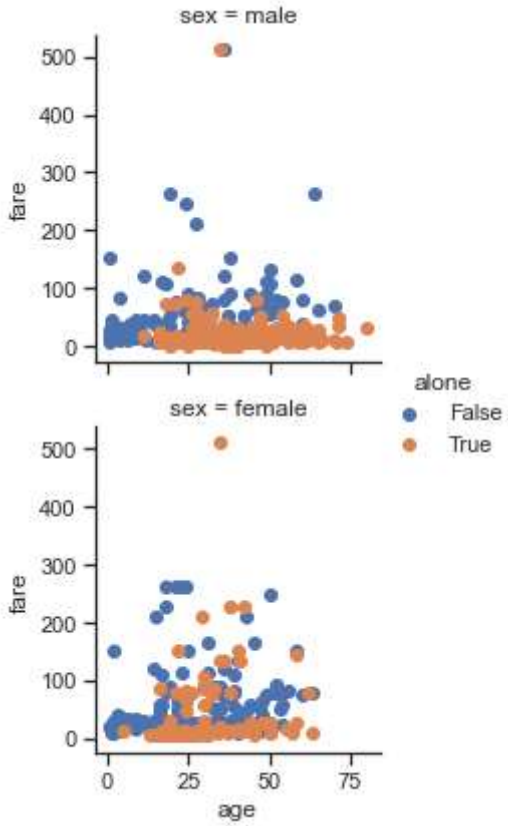
continueous variables

In [ ]:

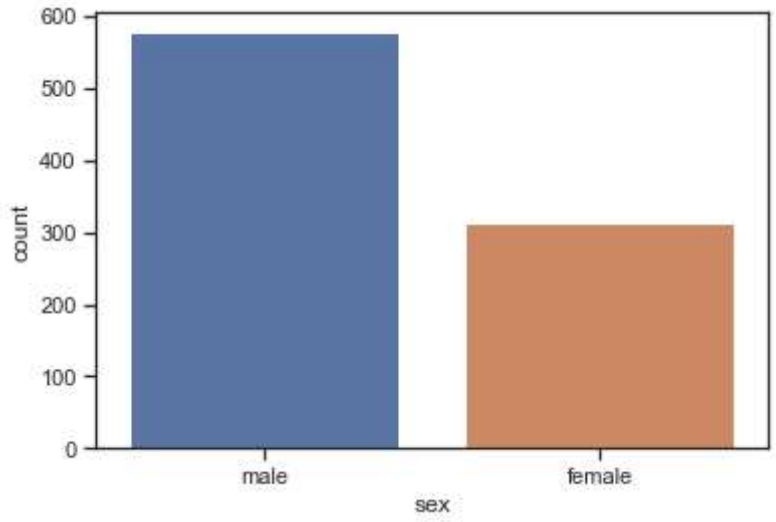
```
#scatterPlot
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="ticks",color_codes=True)
titanic=sns.load_dataset("titanic")
g=sns.FacetGrid(titanic,row="sex",hue="alone")
g=(g.map(plt.scatter,"age","fare").add_legend())
plt.show
```

Out[ ]:

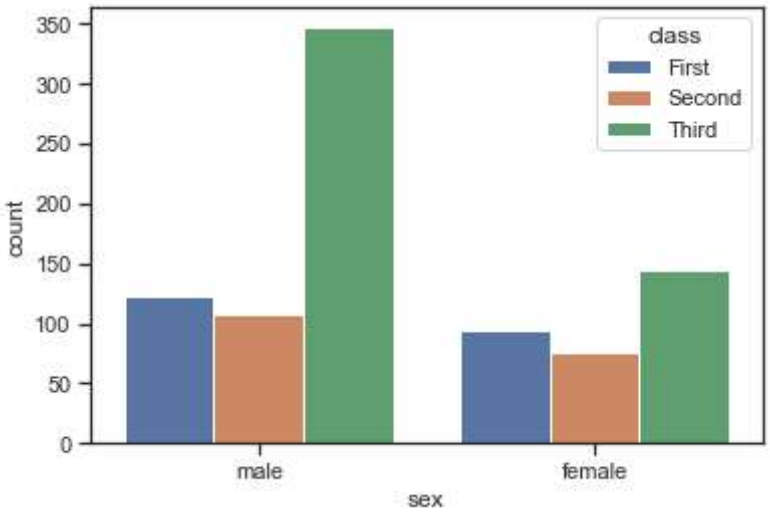
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [ ]:
```
import seaborn as sas
import matplotlib.pyplot as plt
sns.set_theme(style="ticks",color_codes=True)
kashti = sns.load_dataset("titanic")
p=sns.countplot(x="sex",data=kashti)
plt.show()
```



In [ ]:
```
import seaborn as sas
import matplotlib.pyplot as plt
sns.set_theme(style="ticks",color_codes=True)
kashti = sns.load_dataset("titanic")
p=sns.countplot(x="sex",hue="class",data=kashti)
plt.show()
```



LINE PLOTS

Import libraries

seaborn automatically install these libraries

- pandas
- scipy
- numpy
- matplotlib

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

#Load data set
phool = sns.load_dataset("iris")
#print data set
phool
```

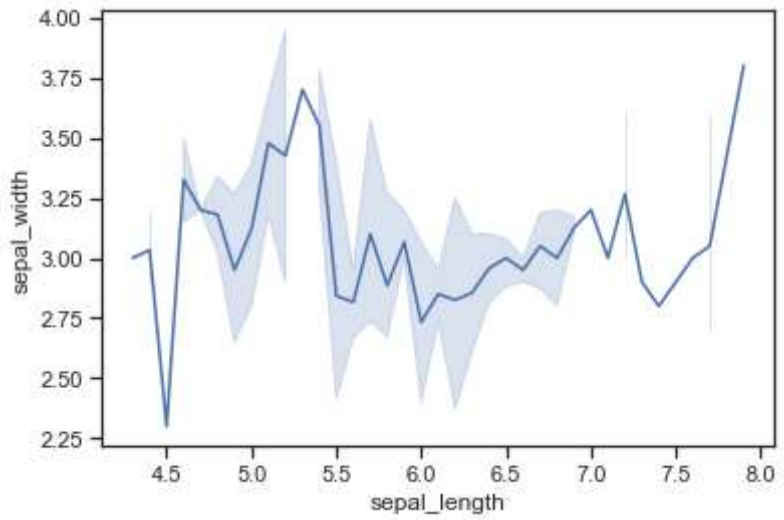
Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica

	sepal_length	sepal_width	petal_length	petal_width	species
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

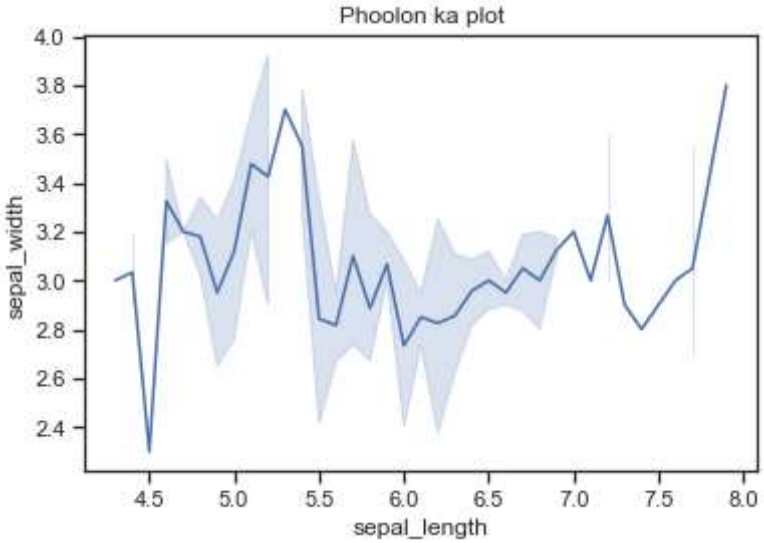
150 rows × 5 columns

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

#load data set
phool = sns.load_dataset("iris")
# draw a line plot
sns.lineplot(x="sepal_length",y="sepal_width",data=phool)
plt.show()
```



Adding titles

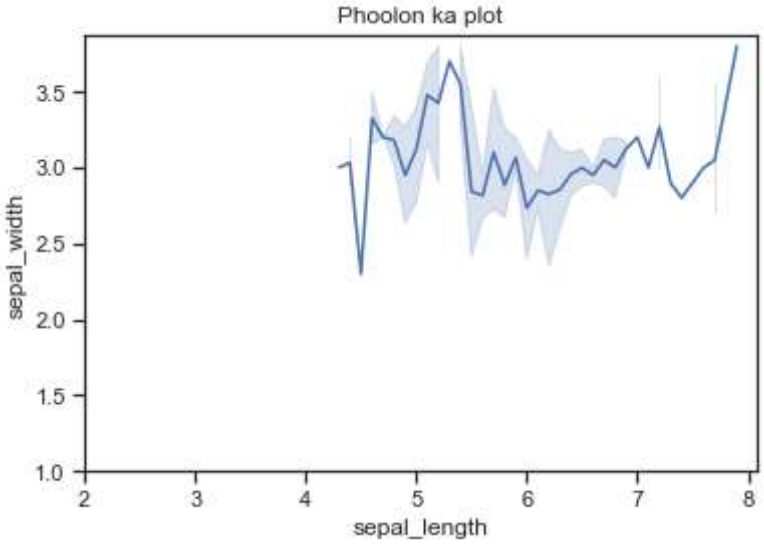
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

#load data set
phool = sns.load_dataset("iris")
# draw a line plot
sns.lineplot(x="sepal_length",y="sepal_width",data=phool)
plt.title("Phoolon ka plot")
plt.show()
```



Adding limits

In [ ]:

```
import seaborn as sns
import matplotlib.pyplot as plt

#Load data set
phool = sns.load_dataset("iris")
# draw a line plot
sns.lineplot(x="sepal_length",y="sepal_width",data=phool)
plt.title("Phoolon ka plot")
plt.xlim(2)
plt.ylim(1)
plt.show()
```



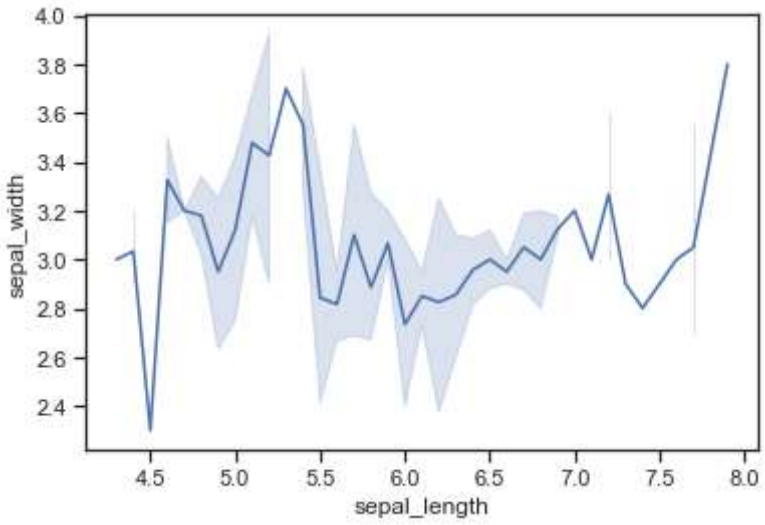
set styles

- darkgrid
- whitegrid
- dark

- white
- ticks

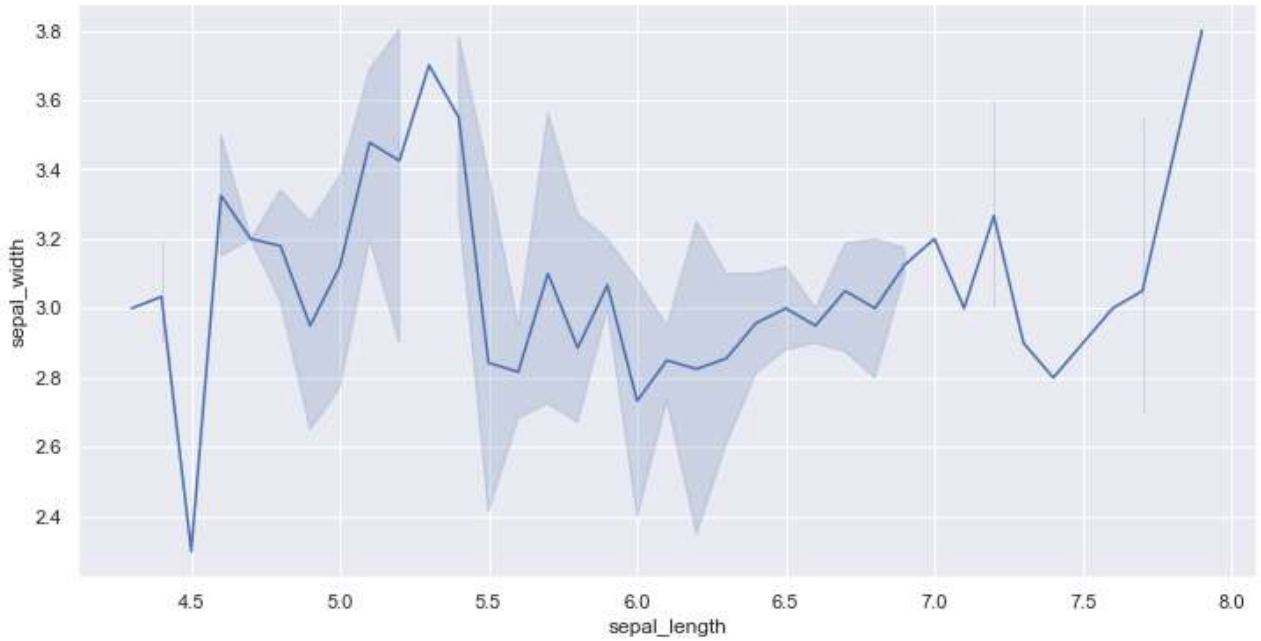
In [ ]:
```
p=sns.set_style(style=None,rc=None)
```

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as pltb


#load data set
phool = sns.load_dataset("iris")
# draw a line plot
sns.lineplot(x="sepal_length",y="sepal_width",data=phool)

p=sns.set_style("darkgrid")
plt.show()
```



size of figure

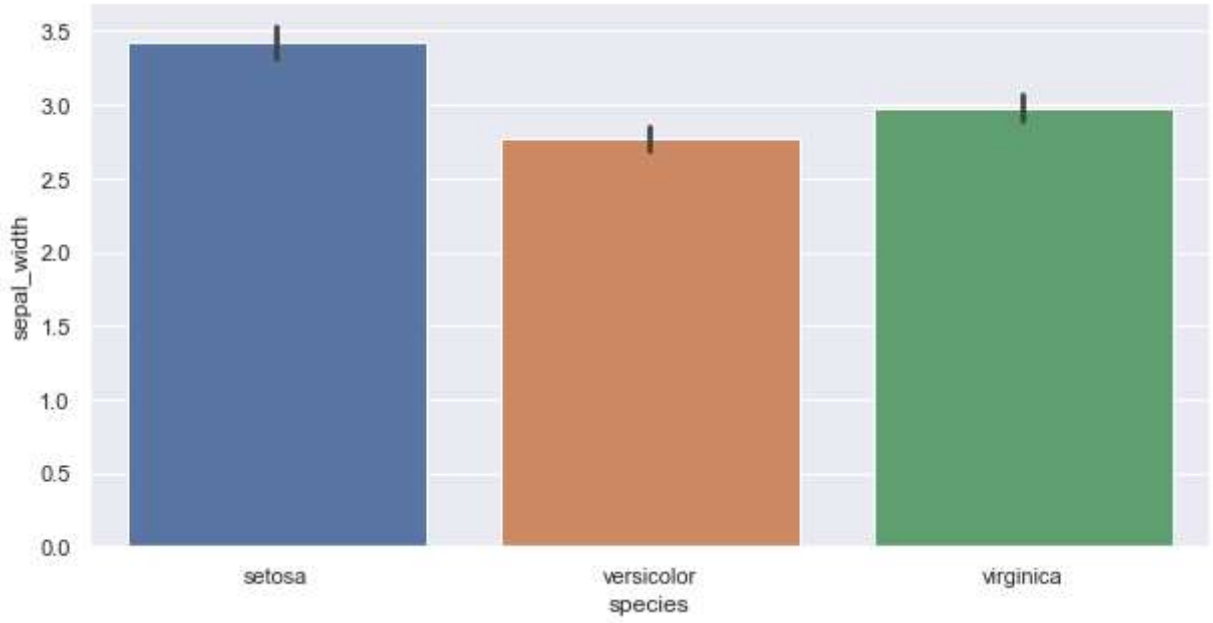
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

#load data set
phool = sns.load_dataset("iris")
# change figure size
plt.figure(figsize=(12,6))
# draw a line plot
sns.lineplot(x="sepal_length",y="sepal_width",data=phool)
plt.show()
```



02 Bar Plot(part 2)

In [ ]:

```
#import libraries
import seaborn as sns
import matplotlib.pyplot as plt


#load data set
flower = sns.load_dataset("iris")
plt.figure(figsize=(10,5))

#draw a barplot
sns.barplot(x="species",y="sepal_width",data=flower)
plt.show()
```



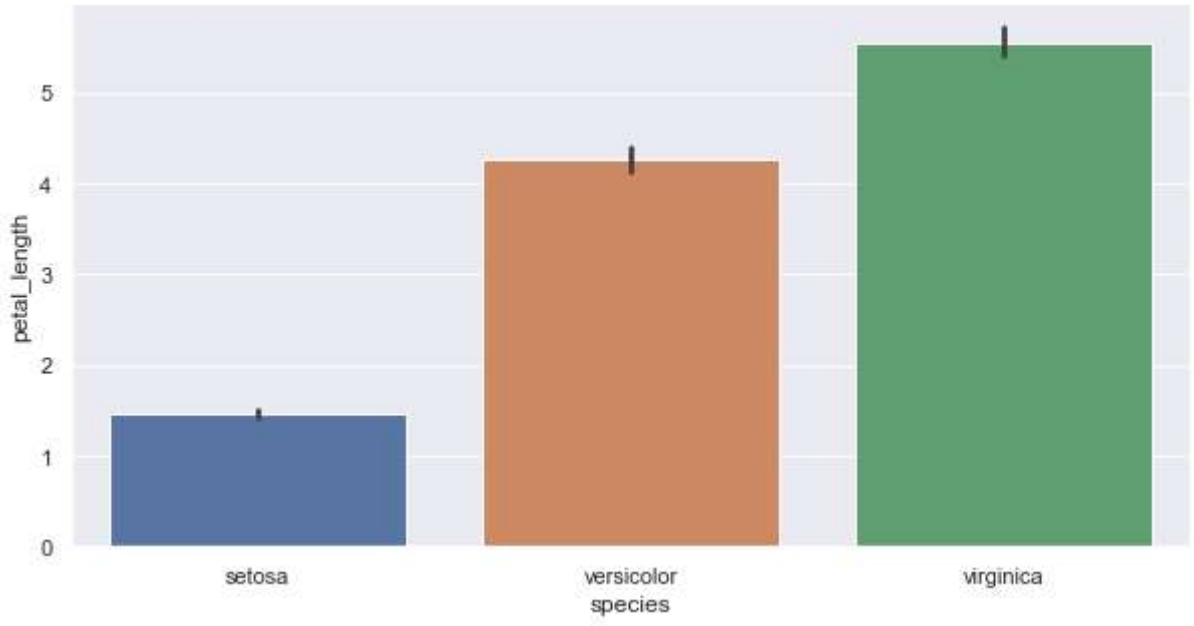
In [ ]:
```
flower
```

Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt


#Load data set
flower = sns.load_dataset("iris")
plt.figure(figsize=(10,5))

#draw a barplot
sns.barplot(x="species",y="petal_length",data=flower)
plt.show()
```



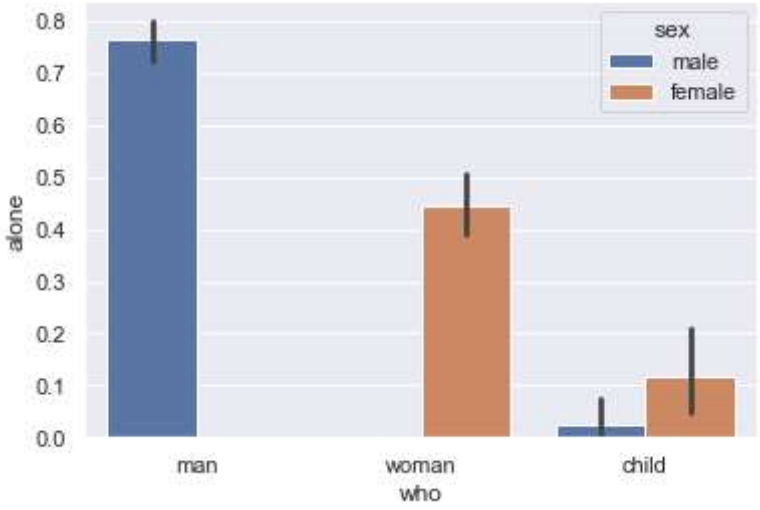
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt


#Load data set
titanic = sns.load_dataset("titanic")
titanic
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	dec
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Na
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Na
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Na
...	...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	Na
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	Na
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	Na

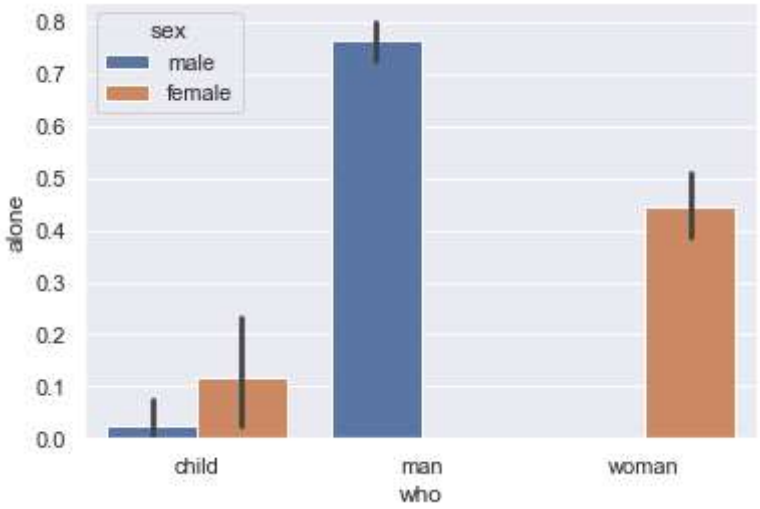
891 rows × 15 columns

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt


#Load data set
titanic = sns.load_dataset("titanic")
titanic
# draw a bar plot
sns.barplot(x="who",y="alone",hue="sex",data=titanic)
plt.show()
```



change order

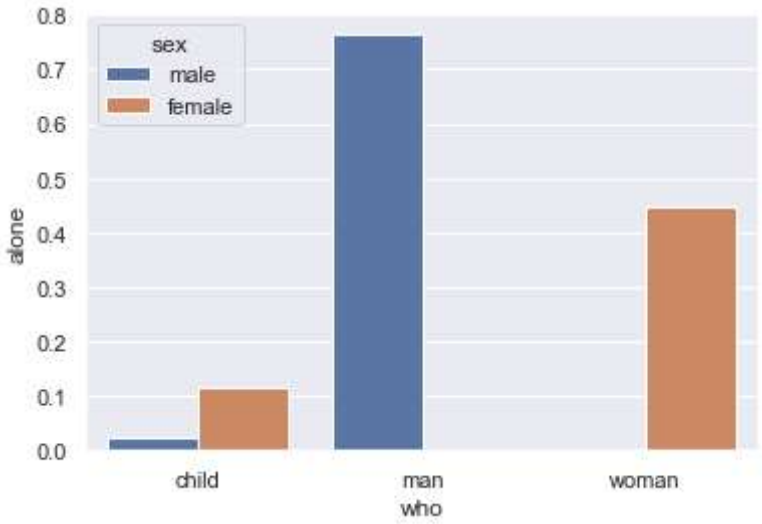
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt


#Load data set
titanic = sns.load_dataset("titanic")
titanic
# draw a bar plot
sns.barplot(x="who",y="alone",hue="sex",data=titanic,order=["child","man","woman"])
plt.show()
```



remove confidence interval (ci)

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt


#Load data set
titanic = sns.load_dataset("titanic")
titanic
```

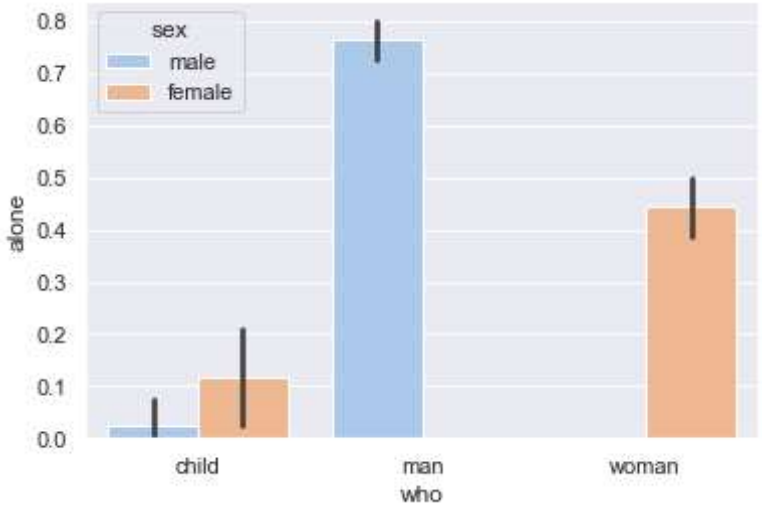
```
# draw a bar plot
sns.barplot(x="who",y="alone",hue="sex",data=titanic,order=["child","man","woman"],ci=N
plt.show()
```



using color palletes

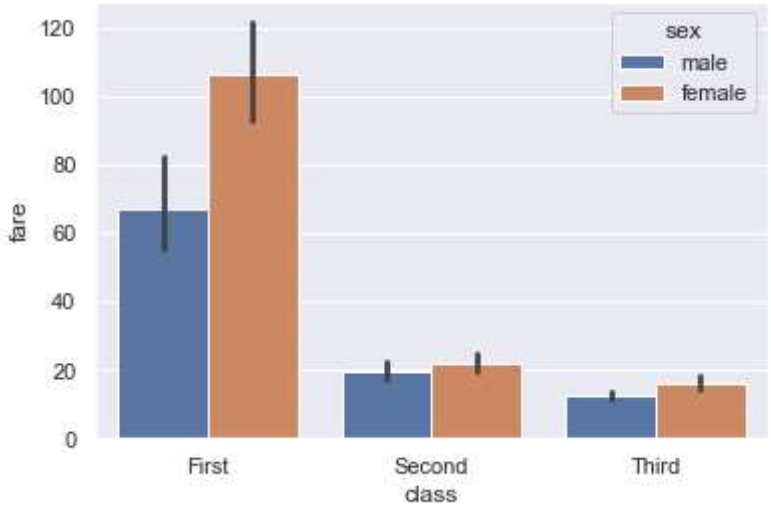
In [ ]:

```
import seaborn as sns
import matplotlib.pyplot as plt


#Load data set
titanic = sns.load_dataset("titanic")
titanic
# draw a bar plot
sns.barplot(x="who",y="alone",hue="sex",data=titanic,order=["child","man","woman"],
            palette="pastel")
plt.show()
```



using estimator import from numpy median or mean estimator

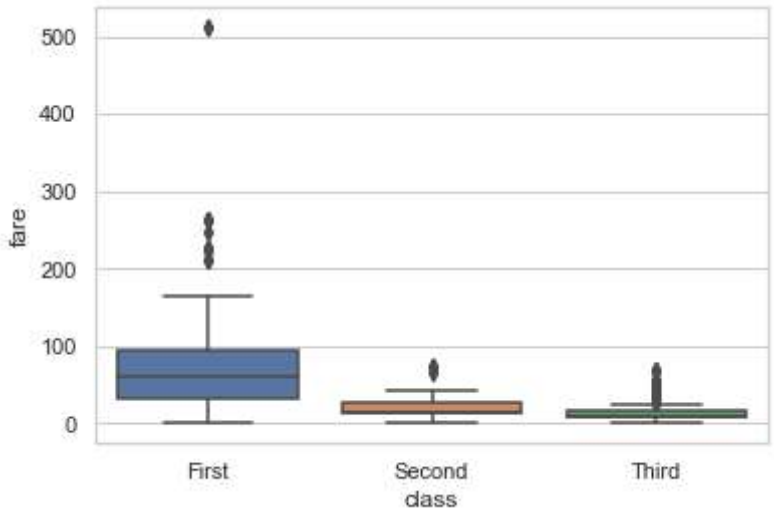
In [ ]:
```
import seaborn as sns
from numpy import mean
import matplotlib.pyplot as plt


#Load data set
titanic = sns.load_dataset("titanic")
titanic
# draw a bar plot
sns.barplot(x="class",y="fare",hue="sex",data=titanic,estimator=mean)
plt.show()
```



BOX PLOT

In [ ]:
```
import seaborn as sns
sns.set(style='whitegrid')

titanic = sns.load_dataset("titanic")
sns.boxplot(x="class",y="fare",data=titanic)
```

Out[ ]:
```
<AxesSubplot:xlabel='class', ylabel='fare'>
```



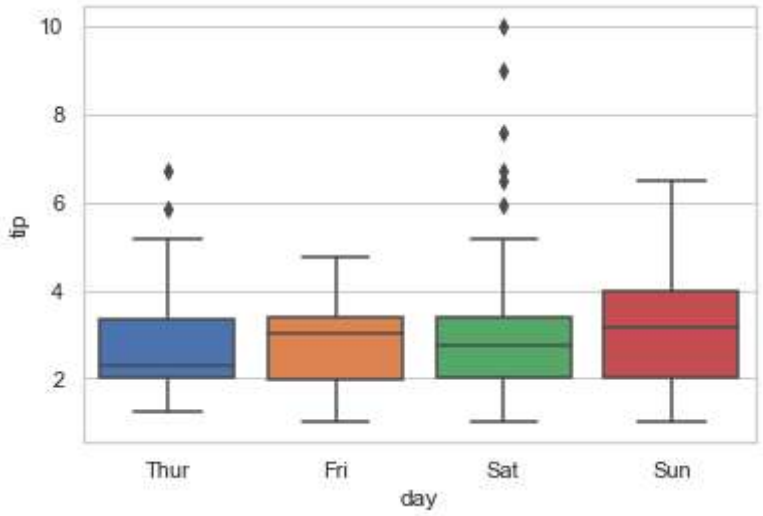
In [ ]:
```
import seaborn as sns
sns.set(style="whitegrid")

tip = sns.load_dataset("tips")
tip.head()
```

Out[ ]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [ ]:
```
import seaborn as sns
sns.set(style="whitegrid")

tip = sns.load_dataset("tips")
sns.boxplot(x="day",y="tip",data=tip,saturation=1)
```

Out[ ]:
```
<AxesSubplot:xlabel='day', ylabel='tip'>
```



In [ ]:
```
import seaborn as sns
import pandas as pd
import numpy as np

tip=sns.load_dataset("tips")
tip.describe()
```

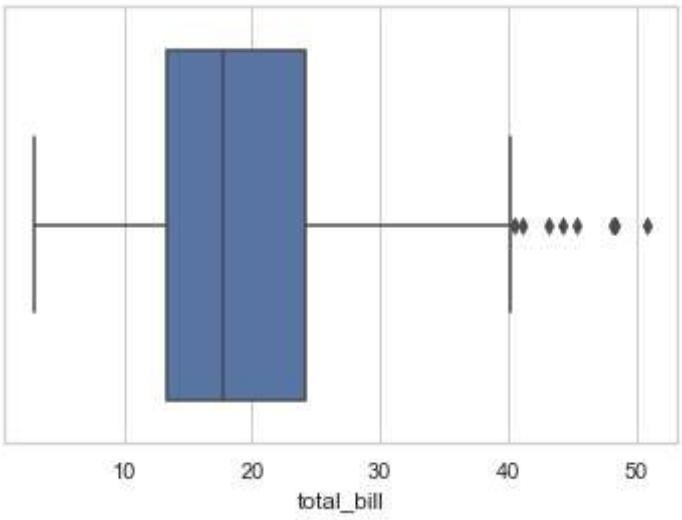
Out[ ]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100

	total_bill	tip	size
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

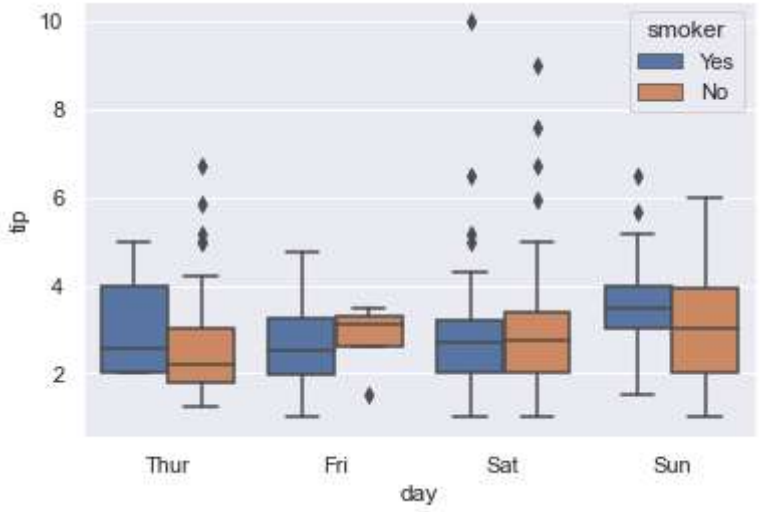
In [ ]:
```
import seaborn as sns
sns.set(style="whitegrid")

tip=sns.load_dataset("tips")
sns.boxplot(x=tip['total_bill'])
```

Out[ ]:
```
<AxesSubplot:xlabel='total_bill'>
```



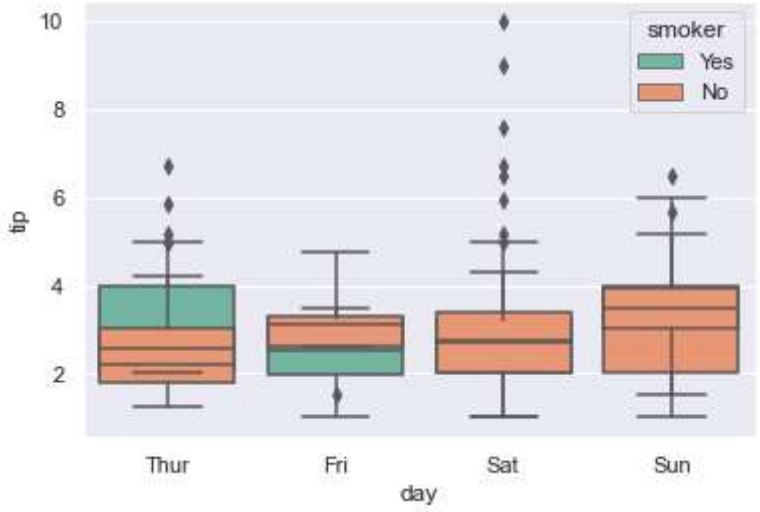
In [ ]:
```
import seaborn as sns
sns.set(style="darkgrid")

tip=sns.load_dataset("tips")
sns.boxplot(x="day",y="tip",hue="smoker",data=tip)
```

Out[ ]:
```
<AxesSubplot:xlabel='day', ylabel='tip'>
```



In [ ]:
```
import seaborn as sns
sns.set(style="darkgrid")

tip=sns.load_dataset("tips")
sns.boxplot(x="day",y="tip",hue="smoker",data=tip,palette="Set2",dodge=False)
```

Out[ ]:
```
<AxesSubplot:xlabel='day', ylabel='tip'>
```



In [ ]:
```
import seaborn as sns
import pandas as pd
import numpy as np

kashti = sns.load_dataset("titanic")
kashti.head()
```

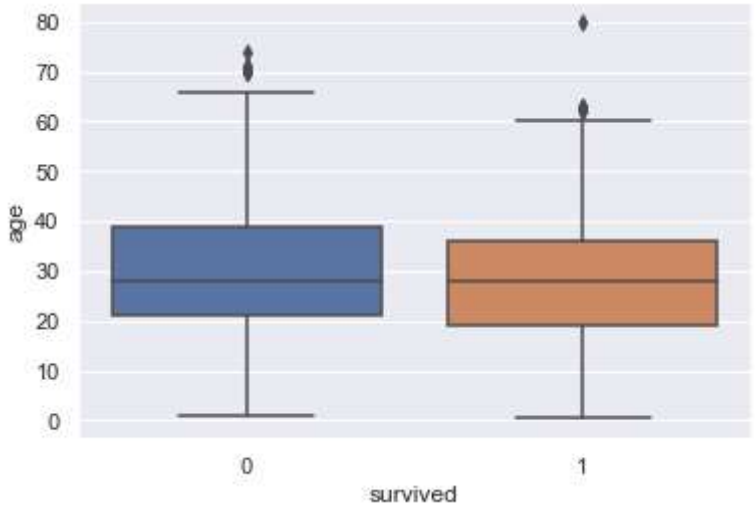
Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

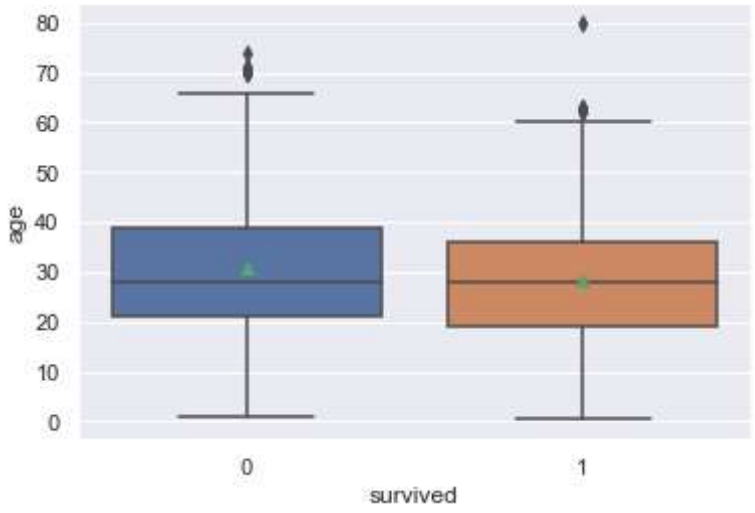
In [ ]:
```
sns.boxplot(x="survived",y="age",data=kashti)
```

Out[ ]:
```
<AxesSubplot:xlabel='survived', ylabel='age'>
```



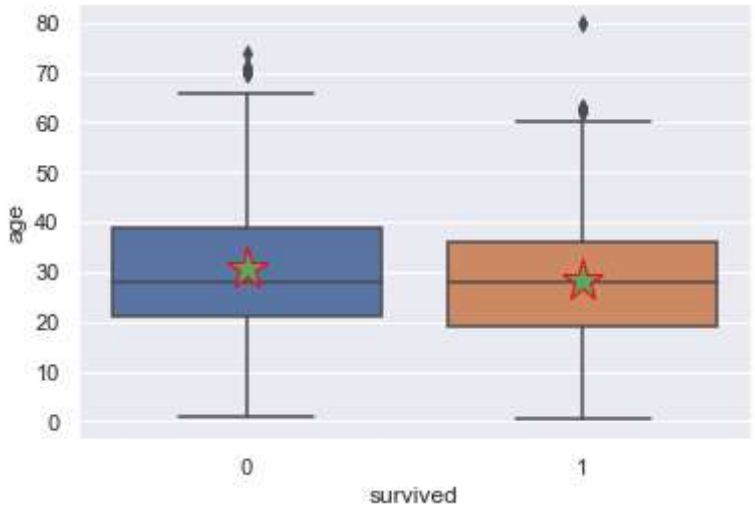
In [ ]:
```
sns.boxplot(x="survived",y="age",showmeans=True,data=kashti)
```

Out[ ]:
```
<AxesSubplot:xlabel='survived', ylabel='age'>
```



In [ ]:
```
sns.boxplot(x="survived",y="age",
            showmeans=True,
            meanprops={"marker":"*","markersize":"22","markeredgecolor":"red"},
            data=kashti)
```

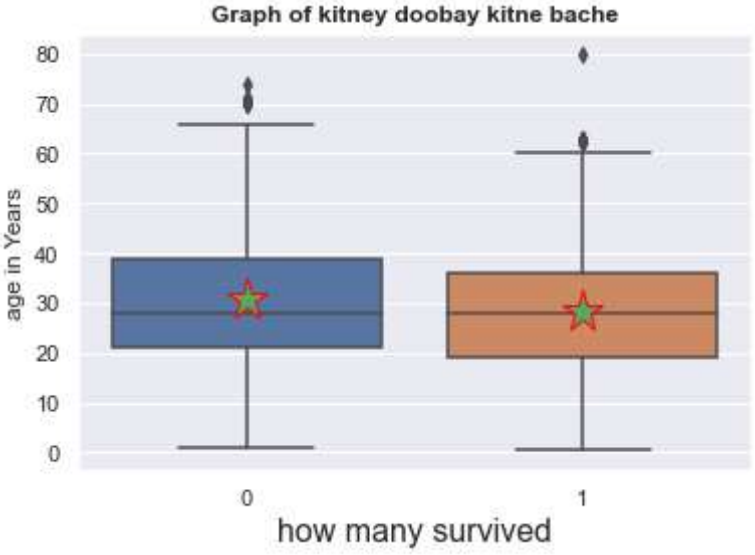
Out[ ]:
```
<AxesSubplot:xlabel='survived', ylabel='age'>
```



In [ ]:

```
import matplotlib.pyplot as plt
sns.boxplot(x="survived",y="age",
            showmeans=True,
            meanprops={"marker":"*","markersize":"22","markeredgecolor":"red"},
            data=kashti)

#set x and y label
plt.xlabel("how many survived",size=16)
plt.ylabel("age in Years")
plt.title("Graph of kitney doobay kitne bache", weight="bold")
```

Out[ ]: Text(0.5, 1.0, 'Graph of kitney doobay kitne bache')



assignment what is facet wrap & facet grid in box plot

OTHER PLOTS

line plot with multifacets

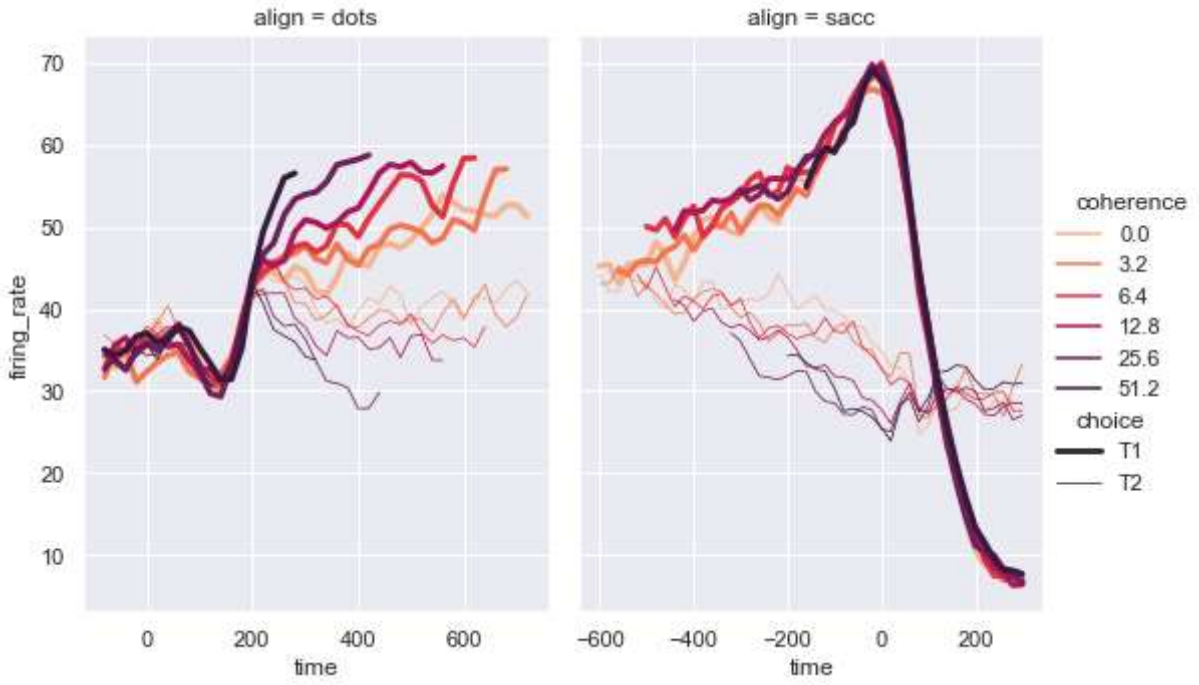
In [ ]:
```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
nuqta=sns.load_dataset("dots")
nuqta.head()
```

Out[ ]:

	align	choice	time	coherence	firing_rate
0	dots	T1	-80	0.0	33.189967
1	dots	T1	-80	3.2	31.691726
2	dots	T1	-80	6.4	34.279840
3	dots	T1	-80	12.8	32.631874
4	dots	T1	-80	25.6	35.060487

In [ ]:
```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
sns.load_dataset("titanic")

#defining a color palette
palette= sns.color_palette("rocket_r")

#plot line plot
sns.relplot(
    data=nuqta,
    x="time", y="firing_rate",
    hue="coherence", size="choice", col="align",
    kind="line", size_order=["T1", "T2"], palette=palette,
    height=5, aspect=.75, facet_kws=dict(sharex=False),
)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x19894dfc910>
```



You can pick up any graph code from seaborn website exaple gallery and customise it on your own data

Graphs On Chilla data

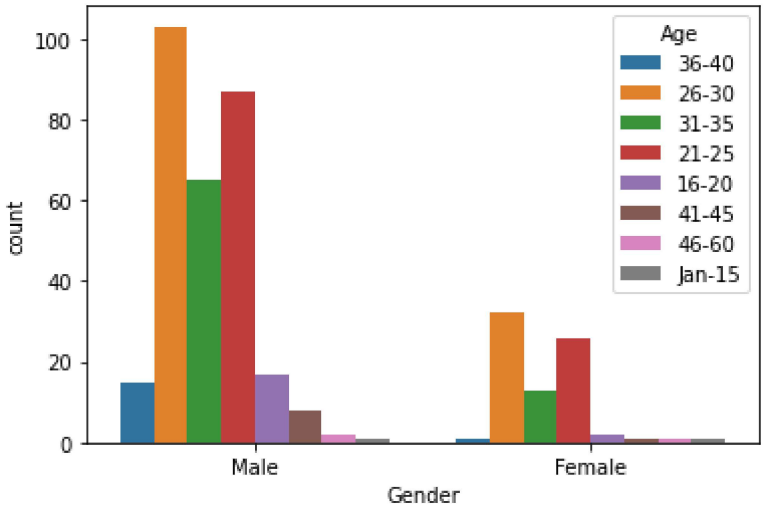
In [ ]:
```
#import libraries
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

mydata = pd.read_csv("Chilla_data2_for_plots.csv")
print(mydata.head())
```

```
   Gender  Location    Age Qualification_completed    field_of_study  \
0    Male  Pakistan  36-40                Masters  Natural Sciences
1    Male  Pakistan  26-30              Bachelors             CS/IT
2    Male  Pakistan  31-35                Masters       Enginnering
3  Female  Pakistan  31-35                Masters             CS/IT
4  Female  Pakistan  26-30                Masters       Enginnering

         Purpose_for_chilla What are you? Blood group   \
0     to boost my skill set     Unemplyed           B+
1     to boost my skill set       Student           B+
2  Switch my field of study      Employed           B+
3     to boost my skill set      Employed           O+
4     to boost my skill set       Student           A-

  Which mobile sim do you use Prepaid or Postpaid  ...  \
0                      U-fone             Prepaid  ...
1                      U-fone             Prepaid  ...
2                        Zong             Prepaid  ...
3                      U-fone            Postpaid  ...
4                    Mobilink             Prepaid  ...
```

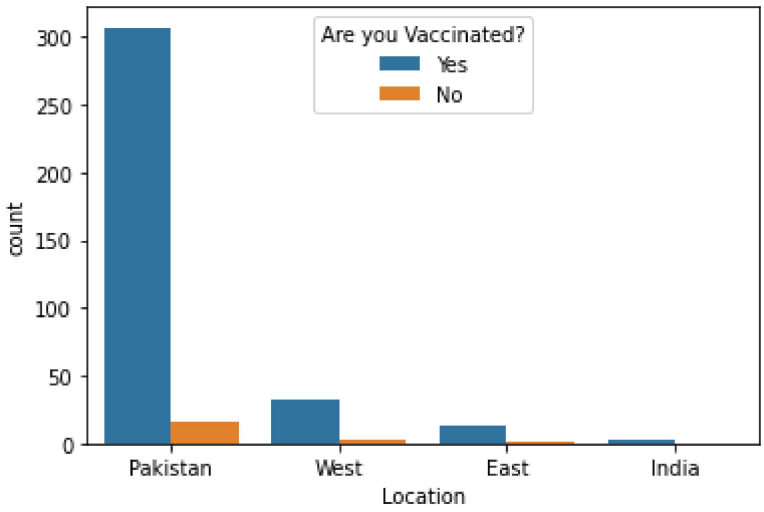
```
  Your favorite programming language? Marital Status? Are you Vaccinated?  \
0                              Python             Yes                  Yes
1                              Python              No                  Yes
2                              Python             Yes                  Yes
3                              Python             Yes                  Yes
4                          Javascript              No                  Yes

  Where do you live? Research/Working experience (Float/Int) years  \
0              Urbun                                              5
1              Urbun                                              1
2              Urbun                                            5.5
3              Urbun                                              5
4              Rural                                            3.5

  Age (years)-Float/Int Your Weight in kg? (float)  \
0                 38.00                       77.0
1                 25.00                       53.6
2                 31.34                       93.0
3                 33.00                       60.0
4                 27.00                       59.9

  Height in cm? Freelancer- (Float)  \
0                           179.000
1                           178.000
2                           173.000
3                           157.000
4                           164.544

   How many hours you code a day? (int) e.g: 5,4,3  \
0                                              3.0
1                                              2.0
2                                              2.0
3                                              3.0
4                                              6.0

   Light kitni der band hti hy? int
0                               2.0
1                               6.0
2                               0.0
3                              24.0
4                              12.0

[5 rows x 23 columns]
```

In [ ]:

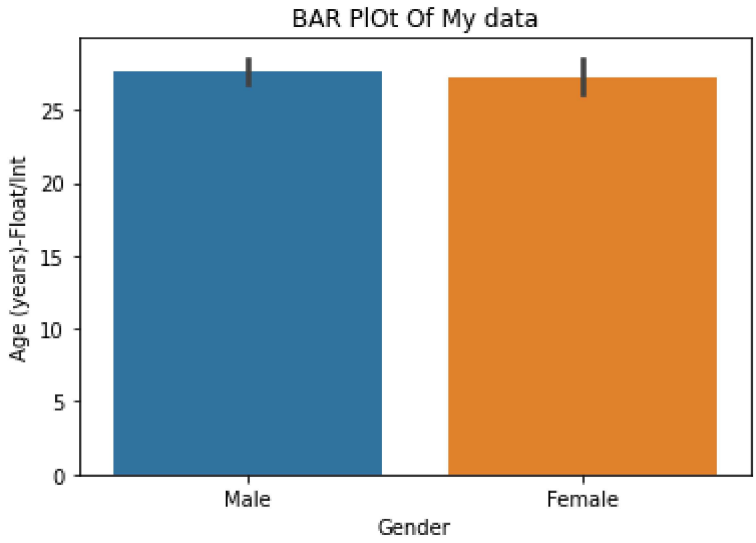
```
#Draw count plot
sns.countplot(x="Gender",hue="Age",data=mydata)
plt.show()
```



In [ ]:
```
sns.countplot(x="Location",hue="Are you Vaccinated?",data=mydata)
plt.show()
```



In [ ]:
```
#Bar plot
sns.barplot(x="Gender",y="Age (years)-Float/Int",data=mydata)
plt.title("BAR PlOt Of My data")
plt.show()
```



Numpy In Python

pip install numpy

In [ ]:
```
#import numpy librarry
import numpy as np
```

Creating an array using Numpy

In [ ]:
```
#1-D array
food = np.array(["Samosa","chicken Roll","Burger"])
food
```

Out[ ]:
```
array(['Samosa', 'chicken Roll', 'Burger'], dtype='<U12')
```

In [ ]:
```
price = np.array([10,50,150])
price
```

Out[ ]:
```
array([ 10,  50, 150])
```

In [ ]:
```
type(price)
```

Out[ ]:
```
numpy.ndarray
```

In [ ]:
```
type(food)
```

Out[ ]:
```
numpy.ndarray
```

In [ ]:
```
#Length
len(price)
```

```
3
```

Out[ ]:

In [ ]:
```
price[2]
```

Out[ ]:
```
150
```

In [ ]:
```
#from zero to n
price[0:]
```

Out[ ]:
```
array([ 10,  50, 150])
```

In [ ]:
```
food[1]
```

Out[ ]:
```
'chicken Roll'
```

In [ ]:
```
price.mean()
```

Out[ ]:
```
70.0
```

In [ ]:
```
#zeros method
np.zeros(6)
```

Out[ ]:
```
array([0., 0., 0., 0., 0., 0.])
```

In [ ]:
```
# Array of ones
np.ones(5)
```

Out[ ]:
```
array([1., 1., 1., 1., 1.])
```

In [ ]:
```
np.empty(5)
```

Out[ ]:
```
array([1., 1., 1., 1., 1.])
```

arange method

In [ ]:
```
np.arange(10)
```

Out[ ]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [ ]:
```
#specify range
np.arange(2,20)
```

Out[ ]:
```
array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
       19])
```

In [ ]:
```
#specific range with specificc interval
```

```
np.arange(2,21,2)
```

Out[ ]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

In [ ]:
```
#table of 5 (last is 55 because last is exclusive)
np.arange(5,55,5)
```

Out[ ]:
```
array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])
```

In [ ]:
```
#line space(spacific interval between given nunber)
np.linspace(0, 10, num=5)
```

Out[ ]:
```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [ ]:
```
#numbers from 1 to 100 in 10 numbers of same interval
np.linspace(1,100,num=10)
```

Out[ ]:
```
array([  1.,  12.,  23.,  34.,  45.,  56.,  67.,  78.,  89., 100.])
```

In [ ]:
```
# specify your data type
np.ones(10, dtype=np.int64)
```

Out[ ]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [ ]:
```
np.ones(5, dtype=np.float64)
```

Out[ ]:
```
array([1., 1., 1., 1., 1.])
```

array function

- sorting

In [ ]:
```
a = np.array([10,12,15,2,4,6,100,320,0.5,10.3 ])
a
```

Out[ ]:
```
array([ 10. ,  12. ,  15. ,   2. ,   4. ,   6. , 100. , 320. ,   0.5,
        10.3])
```

In [ ]:
```
a.sort()
a
```

Out[ ]:
```
array([  0.5,   2. ,   4. ,   6. ,  10. ,  10.3,  12. ,  15. , 100. ,
       320. ])
```

In [ ]:
```
b=np.array([10.2,3.4,8.3])
b
```

Out[ ]:
```
array([10.2,  3.4,  8.3])
```

In [ ]:
```
c=np.concatenate((a,b))
c
```

Out[ ]:
```
array([  0.5,   2. ,   4. ,   6. ,  10. ,  10.3,  12. ,  15. , 100. ,
       320. ,  10.2,   3.4,   8.3])
```

In [ ]:
```
c.sort()
c
```

Out[ ]:
```
array([  0.5,   2. ,   3.4,   4. ,   6. ,   8.3,  10. ,  10.2,  10.3,
        12. ,  15. , 100. , 320. ])
```

Two dimentional arrays

In [ ]:
```
y=np.array([[1,2,3],[5,4,3]])
y
```

Out[ ]:
```
array([[1, 2, 3],
       [5, 4, 3]])
```

In [ ]:
```
z=np.array([[6,7],[8,9]])
z
```

Out[ ]:
```
array([[6, 7],
       [8, 9]])
```

In [ ]:
```
z=np.array([[6,7,8],[8,9,1]])
z
```

Out[ ]:
```
array([[6, 7, 8],
       [8, 9, 1]])
```

In [ ]:
```
z = np.concatenate((y,z),axis=0)
z
```

Out[ ]:
```
array([[1, 2, 3],
       [5, 4, 3],
       [6, 7, 8],
       [8, 9, 1]])
```

In [ ]:
```
t=np.array([[7,8,6],[6,8,7]])
t
```

Out[ ]:
```
array([[7, 8, 6],
       [6, 8, 7]])
```

In [ ]:
```
np.concatenate((y,t),axis=1)
```

Out[ ]:
```
array([[1, 2, 3, 7, 8, 6],
       [5, 4, 3, 6, 8, 7]])
```

In [ ]:
```
c = np.array([[[0, 1, 2, 3],
               [4, 5, 6, 7]],
```

```
            [[0, 1, 2, 3],
            [4, 5, 6, 7]],

            [[0 ,1 ,2, 3],
            [4, 5, 6, 7]]])
c
```

Out[ ]:
```
array([[[0, 1, 2, 3],
        [4, 5, 6, 7]],

       [[0, 1, 2, 3],
        [4, 5, 6, 7]],

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

In [ ]:
```
#to find the number of dimentions
c.ndim
```

Out[ ]:
```
3
```

In [ ]:
```
c.size
```

Out[ ]:
```
24
```

In [ ]:
```
c.shape   #(3,2,4)means 3 dimentional and 2X4 row and columns
```

Out[ ]:
```
(3, 2, 4)
```

In [ ]:
```
d=np.array([[5,6,7],
           [8,9,10],
           [11,12,13]])
d
```

Out[ ]:
```
array([[ 5,  6,  7],
       [ 8,  9, 10],
       [11, 12, 13]])
```

In [ ]:
```
d.ndim
```

Out[ ]:
```
2
```

In [ ]:
```
d.size
```

Out[ ]:
```
9
```

In [ ]:
```
d.shape
```

Out[ ]:
```
(3, 3)
```

In [ ]:
```
my=np.arange(6)
my
```

Out[ ]:
```
array([0, 1, 2, 3, 4, 5])
```

In [ ]:
```
my.reshape(3,2) # hint 3X2=6
```

Out[ ]:
```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [ ]:
```
my.ndim
```

Out[ ]:
```
1
```

In [ ]:
```
np.reshape(my, newshape=(1,6),order='C') # order C= coloumn
```

Out[ ]:
```
array([[0, 1, 2, 3, 4, 5]])
```

In [ ]:
```
my.reshape(3,2)
```

Out[ ]:
```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

convert 1D array into To 2D

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

Out[ ]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [ ]:
```
n.shape
```

Out[ ]:
```
(9,)
```

In [ ]:
```
#row wise
m=n[np.newaxis, :]
m
```

Out[ ]:
```
array([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

In [ ]:
```
m.shape
```

Out[ ]:
```
(1, 9)
```

In [ ]:
```
m.ndim
```

Out[ ]:
```
2
```

In [ ]:
```
#colounm wise
o=n[ :,np.newaxis]
o
```

Out[ ]:
```
array([[1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9]])
```

In [ ]:
```
o*6
```

Out[ ]:
```
array([[ 6],
       [12],
       [18],
       [24],
       [30],
       [36],
       [42],
       [48],
       [54]])
```

PANDAS

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
# object creation
s = pd.Series([1,3,np.nan,4,5,7,8])
s
```

Out[ ]:
```
0    1.0
1    3.0
2    NaN
3    4.0
4    5.0
5    7.0
6    8.0
dtype: float64
```

In [ ]:
```
dates = pd.date_range("20200101",periods=6)
dates
```

Out[ ]:
```
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
               '2020-01-05', '2020-01-06'],
              dtype='datetime64[ns]', freq='D')
```

In [ ]:

```
df = pd.DataFrame(np.random.randn(6,4),index=dates,columns=list("ABCD"))
df
```

Out[ ]:

	A	B	C	D
2020-01-01	-0.451187	-1.268403	1.084691	1.986217
2020-01-02	0.151788	-1.207489	-0.863496	-0.684161
2020-01-03	-1.046009	0.462870	1.119754	-0.540738
2020-01-04	-0.710617	-0.199232	0.051621	-1.000899
2020-01-05	2.166396	1.097283	-0.289343	-0.570119
2020-01-06	-0.541470	1.165029	0.256686	1.271518

In [ ]:

```
# data frame with dictionaries
df2=pd.DataFrame(
    {
        "A":1.0,
        "B":pd.Timestamp(20130102),
        "C": pd.Series(1, index=list(range(4)), dtype="float32"),
        "D": np.array([3] * 4, dtype="int32"),
        "E": pd.Categorical(["test", "train", "test", "train"]),
        "F": "foo",
    }
)
df2
```

Out[ ]:

	A	B	C	D	E	F
0	1.0	1970-01-01 00:00:00.020130102	1.0	3	test	foo
1	1.0	1970-01-01 00:00:00.020130102	1.0	3	train	foo
2	1.0	1970-01-01 00:00:00.020130102	1.0	3	test	foo
3	1.0	1970-01-01 00:00:00.020130102	1.0	3	train	foo

In [ ]:

```
df2.dtypes
```

Out[ ]:

```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

In [ ]:

```
df2.head(2)
```

Out[ ]:

	A	B	C	D	E	F
0	1.0	1970-01-01 00:00:00.020130102	1.0	3	test	foo
1	1.0	1970-01-01 00:00:00.020130102	1.0	3	train	foo

In [ ]:
```
df2.tail(2)
```

Out[ ]:

	A	B	C	D	E	F
2	1.0	1970-01-01 00:00:00.020130102	1.0	3	test	foo
3	1.0	1970-01-01 00:00:00.020130102	1.0	3	train	foo

In [ ]:
```
df2.tail(2)
```

Out[ ]:

	A	B	C	D	E	F
2	1.0	1970-01-01 00:00:00.020130102	1.0	3	test	foo
3	1.0	1970-01-01 00:00:00.020130102	1.0	3	train	foo

In [ ]:
```
df2.index
```

Out[ ]:
```
Int64Index([0, 1, 2, 3], dtype='int64')
```

In [ ]:
```
df.to_numpy()
```

Out[ ]:
```
array([[-0.45118734, -1.268403  ,  1.08469145,  1.98621684],
       [ 0.15178829, -1.20748942, -0.86349574, -0.68416062],
       [-1.04600924,  0.46286967,  1.11975419, -0.54073814],
       [-0.71061714, -0.19923156,  0.0516209 , -1.00089941],
       [ 2.16639608,  1.09728341, -0.28934284, -0.57011914],
       [-0.54147003,  1.16502869,  0.25668555,  1.27151832]])
```

In [ ]:
```
df.describe()
```

Out[ ]:

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000
mean	-0.071850	0.008343	0.226652	0.076970
std	1.164578	1.084594	0.777184	1.233984
min	-1.046009	-1.268403	-0.863496	-1.000899
25%	-0.668330	-0.955425	-0.204102	-0.655650
50%	-0.496329	0.131819	0.154153	-0.555429
75%	0.001044	0.938680	0.877690	0.818454
max	2.166396	1.165029	1.119754	1.986217

In [ ]:
```
# to tanspose data
df.T
```

Out[ ]:

	2020-01-01	2020-01-02	2020-01-03	2020-01-04	2020-01-05	2020-01-06
A	-0.451187	0.151788	-1.046009	-0.710617	2.166396	-0.541470
B	-1.268403	-1.207489	0.462870	-0.199232	1.097283	1.165029
C	1.084691	-0.863496	1.119754	0.051621	-0.289343	0.256686
D	1.986217	-0.684161	-0.540738	-1.000899	-0.570119	1.271518

Again Practice

In [ ]:
```
s=pd.Series([1,3,np.nan,4,7,8,9])
s
```

Out[ ]:
```
0    1.0
1    3.0
2    NaN
3    4.0
4    7.0
5    8.0
6    9.0
dtype: float64
```

In [ ]:
```
dates=pd.date_range("20220101",periods=6)
dates
```

Out[ ]:
```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05', '2022-01-06'],
              dtype='datetime64[ns]', freq='D')
```

In [ ]:
```
df=pd.DataFrame(np.random.randn(6,4),index=dates,columns=list("ABCD"))
df
```

Out[ ]:

	A	B	C	D
2022-01-01	0.807899	0.544849	-1.187173	1.719106
2022-01-02	-1.595928	-0.264720	-1.298060	0.893769
2022-01-03	2.252807	-0.193003	-1.484574	-0.220576
2022-01-04	0.170109	-0.754622	0.442027	0.369345
2022-01-05	-0.718364	-0.092656	0.072429	-1.168935
2022-01-06	-0.052483	0.743551	-1.070496	-1.234155

In [ ]:
```
# data frame using dictionary
df2 = pd.DataFrame(
 {
    "A": 1.0,
    "B": pd.Timestamp("20130102"),
    "C": pd.Series(1, index=list(range(4)), dtype="float32"),
    "D": np.array([3] * 4, dtype="int32"),
    "E": pd.Categorical(["girl", "women", "girl", "womenn"]),
    "F": "females",
```

```
}
)
df2
```

Out[ ]:

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	girl	females
1	1.0	2013-01-02	1.0	3	women	females
2	1.0	2013-01-02	1.0	3	girl	females
3	1.0	2013-01-02	1.0	3	womenn	females

In [ ]:
```
df2.dtypes
```

Out[ ]:
```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

how to view data

In [ ]:
```
df2.head(2)
```

Out[ ]:

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	girl	females
1	1.0	2013-01-02	1.0	3	women	females

In [ ]:
```
df2.tail(2)
```

Out[ ]:

	A	B	C	D	E	F
2	1.0	2013-01-02	1.0	3	girl	females
3	1.0	2013-01-02	1.0	3	womenn	females

In [ ]:
```
df.index
```

Out[ ]:
```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05', '2022-01-06'],
              dtype='datetime64[ns]', freq='D')
```

convert data frame to numpy array

In [ ]:
```
df2.to_numpy()
```

Out[ ]:
```
array([[1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'girl', 'females'],
```

```
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'women',
        'females'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'girl', 'females'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'womenn',
        'females']], dtype=object)
```

In [ ]:
```
df.to_numpy()
```

Out[ ]:
```
array([[ 0.8078991 ,  0.54484923, -1.18717303,  1.7191055 ],
       [-1.59592762, -0.26472005, -1.29805988,  0.89376877],
       [ 2.25280708, -0.19300309, -1.48457413, -0.22057643],
       [ 0.17010901, -0.75462157,  0.44202678,  0.36934533],
       [-0.71836397, -0.09265573,  0.07242852, -1.16893518],
       [-0.0524828 ,  0.74355148, -1.07049599, -1.23415499]])
```

In [ ]:
```
df.describe()
```

Out[ ]:

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000
mean	0.144007	-0.002767	-0.754308	0.059759
std	1.318949	0.554085	0.803867	1.166819
min	-1.595928	-0.754622	-1.484574	-1.234155
25%	-0.551894	-0.246791	-1.270338	-0.931845
50%	0.058813	-0.142829	-1.128835	0.074384
75%	0.648452	0.385473	-0.213303	0.762663
max	2.252807	0.743551	0.442027	1.719106

transpose data

In [ ]:
```
df2.T
```

Out[ ]:

	0	1	2	3
A	1.0	1.0	1.0	1.0
B	2013-01-02 00:00:00	2013-01-02 00:00:00	2013-01-02 00:00:00	2013-01-02 00:00:00
C	1.0	1.0	1.0	1.0
D	3	3	3	3
E	girl	women	girl	womenn
F	females	females	females	females

In [ ]:
```
df.sort_index(axis=1, ascending=False)
```

Out[ ]:

	D	C	B	A

	D	C	B	A
2022-01-01	1.719106	-1.187173	0.544849	0.807899
2022-01-02	0.893769	-1.298060	-0.264720	-1.595928
2022-01-03	-0.220576	-1.484574	-0.193003	2.252807
2022-01-04	0.369345	0.442027	-0.754622	0.170109
2022-01-05	-1.168935	0.072429	-0.092656	-0.718364
2022-01-06	-1.234155	-1.070496	0.743551	-0.052483

In [ ]:
```
df["A"]
```

Out[ ]:
```
2022-01-01    0.807899
2022-01-02   -1.595928
2022-01-03    2.252807
2022-01-04    0.170109
2022-01-05   -0.718364
2022-01-06   -0.052483
Freq: D, Name: A, dtype: float64
```

In [ ]:
```
df["B"]
```

Out[ ]:
```
2022-01-01    0.544849
2022-01-02   -0.264720
2022-01-03   -0.193003
2022-01-04   -0.754622
2022-01-05   -0.092656
2022-01-06    0.743551
Freq: D, Name: B, dtype: float64
```

In [ ]:
```
#row wise selection
df[0:3]
```

Out[ ]:

	A	B	C	D
2022-01-01	0.807899	0.544849	-1.187173	1.719106
2022-01-02	-1.595928	-0.264720	-1.298060	0.893769
2022-01-03	2.252807	-0.193003	-1.484574	-0.220576

In [ ]:
```
df[0:]
```

Out[ ]:

	A	B	C	D
2022-01-01	0.807899	0.544849	-1.187173	1.719106
2022-01-02	-1.595928	-0.264720	-1.298060	0.893769
2022-01-03	2.252807	-0.193003	-1.484574	-0.220576
2022-01-04	0.170109	-0.754622	0.442027	0.369345

	A	B	C	D
2022-01-05	-0.718364	-0.092656	0.072429	-1.168935
2022-01-06	-0.052483	0.743551	-1.070496	-1.234155

In [ ]:
```
df[1:3]
```

Out[ ]:

	A	B	C	D
2022-01-02	-1.595928	-0.264720	-1.298060	0.893769
2022-01-03	2.252807	-0.193003	-1.484574	-0.220576

dates wise data

In [ ]:
```
df.loc[dates[0]]
```

Out[ ]:
```
A    0.807899
B    0.544849
C   -1.187173
D    1.719106
Name: 2022-01-01 00:00:00, dtype: float64
```

In [ ]:
```
df.loc[dates[2]]
```

Out[ ]:
```
A    2.252807
B   -0.193003
C   -1.484574
D   -0.220576
Name: 2022-01-03 00:00:00, dtype: float64
```

In [ ]:
```
df.loc[:,["A","B"]]
```

Out[ ]:

	A	B
2022-01-01	0.807899	0.544849
2022-01-02	-1.595928	-0.264720
2022-01-03	2.252807	-0.193003
2022-01-04	0.170109	-0.754622
2022-01-05	-0.718364	-0.092656
2022-01-06	-0.052483	0.743551

In [ ]:
```
df.loc["20220102":"20220104",["A","B"]]
```

Out[ ]:

	A	B
2022-01-02	-1.595928	-0.264720

	A	B
2022-01-03	2.252807	-0.193003
2022-01-04	0.170109	-0.754622

In [ ]:
```
df.loc["20220104",["A","B","C"]]
```

Out[ ]:
```
A    0.170109
B   -0.754622
C    0.442027
Name: 2022-01-04 00:00:00, dtype: float64
```

In [ ]:
```
df.at[dates[0],"A"]
```

Out[ ]:
```
0.807899102811283
```

In [ ]:
```
df.iloc[3]
```

Out[ ]:
```
A    0.170109
B   -0.754622
C    0.442027
D    0.369345
Name: 2022-01-04 00:00:00, dtype: float64
```

In [ ]:
```
df.iloc[2:4]
```

Out[ ]:

	A	B	C	D
2022-01-03	2.252807	-0.193003	-1.484574	-0.220576
2022-01-04	0.170109	-0.754622	0.442027	0.369345

In [ ]:
```
df.iloc[0:5,0:2]
```

Out[ ]:

	A	B
2022-01-01	0.807899	0.544849
2022-01-02	-1.595928	-0.264720
2022-01-03	2.252807	-0.193003
2022-01-04	0.170109	-0.754622
2022-01-05	-0.718364	-0.092656

Boolean indexing

In [ ]:
```
df[df["A"] > 0]
```

Out[ ]:

	A	B	C	D

	A	B	C	D
2022-01-01	0.807899	0.544849	-1.187173	1.719106
2022-01-03	2.252807	-0.193003	-1.484574	-0.220576
2022-01-04	0.170109	-0.754622	0.442027	0.369345

In [ ]:
```
df[df["A"] > 1.5]
```

Out[ ]:

	A	B	C	D
2022-01-03	2.252807	-0.193003	-1.484574	-0.220576

In [ ]:
```
df[df>0]
```

Out[ ]:

	A	B	C	D
2022-01-01	0.807899	0.544849	NaN	1.719106
2022-01-02	NaN	NaN	NaN	0.893769
2022-01-03	2.252807	NaN	NaN	NaN
2022-01-04	0.170109	NaN	0.442027	0.369345
2022-01-05	NaN	NaN	0.072429	NaN
2022-01-06	NaN	0.743551	NaN	NaN

In [ ]:
```
df2 = df.copy()

df2["E"] = ["one", "one", "two", "three", "four", "three"]
```

PANDAS CASE STUDY WITH TITANIC DATASET

In [ ]:
```
#import libaraies
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
kashti = sns.load_dataset('titanic')
kashti.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [ ]:
```
#saving data set into csv file
kashti.to_csv('kashti.csv')
```

In [ ]:
```
# Basic Statistic or summary
kashti.describe()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:
```
kashti.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [ ]:
```
# droping few coloumns and make a new data set
new_kashti = kashti.drop(['sibsp','embarked','deck'],axis=1)
new_kashti.head()
```

Out[ ]:

	survived	pclass	sex	age	parch	fare	class	who	adult_male	embark_town	alive	alone
0	0	3	male	22.0	0	7.2500	Third	man	True	Southampton	no	False
1	1	1	female	38.0	0	71.2833	First	woman	False	Cherbourg	yes	False

	survived	pclass	sex	age	parch	fare	class	who	adult_male	embark_town	alive	alone
2	1	3	female	26.0	0	7.9250	Third	woman	False	Southampton	yes	True
3	1	1	female	35.0	0	53.1000	First	woman	False	Southampton	yes	False
4	0	3	male	35.0	0	8.0500	Third	man	True	Southampton	no	True

In [ ]:
```
kashti.mean()
```

```
C:\Users\Abbas\AppData\Local\Temp\ipykernel_4944\3332994036.py:1: FutureWarning: Droppin
g of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated;
in a future version this will raise TypeError.  Select only valid columns before calling
the reduction.
  kashti.mean()
```

Out[ ]:
```
survived       0.383838
pclass         2.308642
age           29.699118
sibsp          0.523008
parch          0.381594
fare          32.204208
adult_male     0.602694
alone          0.602694
dtype: float64
```

In [ ]:
```
kashti.value_counts('survived')
```

Out[ ]:
```
survived
0    549
1    342
dtype: int64
```

In [ ]:
```
kashti.groupby(['sex','class']).mean()
```

Out[ ]:

		survived	pclass	age	sibsp	parch	fare	adult_male	alone
sex	**class**								
female	**First**	0.968085	1.0	34.611765	0.553191	0.457447	106.125798	0.000000	0.361702
	Second	0.921053	2.0	28.722973	0.486842	0.605263	21.970121	0.000000	0.421053
	Third	0.500000	3.0	21.750000	0.895833	0.798611	16.118810	0.000000	0.416667
male	**First**	0.368852	1.0	41.281386	0.311475	0.278689	67.226127	0.975410	0.614754
	Second	0.157407	2.0	30.740707	0.342593	0.222222	19.741782	0.916667	0.666667
	Third	0.135447	3.0	26.507589	0.498559	0.224784	12.661633	0.919308	0.760807

In [ ]:
```
kashti[kashti['age']<18].mean()
```

```
C:\Users\Abbas\AppData\Local\Temp\ipykernel_4944\3332001198.py:1: FutureWarning: Droppin
g of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated;
in a future version this will raise TypeError.  Select only valid columns before calling
```

```
the reduction.
  kashti[kashti['age']<18].mean()
```

Out[ ]:
```
survived       0.539823
pclass         2.584071
age            9.041327
sibsp          1.460177
parch          1.053097
fare          31.220798
adult_male     0.159292
alone          0.203540
dtype: float64
```

In [ ]:
```
#children whose age is < 18
kashti[kashti['age']<18].groupby(['sex','class']).mean()
```

Out[ ]:

		survived	pclass	age	sibsp	parch	fare	adult_male	alone
sex	**class**								
female	**First**	0.875000	1.0	14.125000	0.500000	0.875000	104.083337	0.000000	0.125000
	Second	1.000000	2.0	8.333333	0.583333	1.083333	26.241667	0.000000	0.166667
	Third	0.542857	3.0	8.428571	1.571429	1.057143	18.727977	0.000000	0.228571
male	**First**	1.000000	1.0	8.230000	0.500000	2.000000	116.072900	0.250000	0.000000
	Second	0.818182	2.0	4.757273	0.727273	1.000000	25.659473	0.181818	0.181818
	Third	0.232558	3.0	9.963256	2.069767	1.000000	22.752523	0.348837	0.232558

EXPLORATORY DATA ANALYSIS

This will show us how can we do EDA using Python

three important steps to keep in mind are:

1- UNDERSTAND THE DATA\ 2- CLEAN THE DATA\ 3- FIND RELATIONSHIP BETWEEN THE DATA

In [ ]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
#load data set
kashti = sns.load_dataset('titanic')
```

In [ ]:
```
# download data set
kashti.to_csv('kashti.to_csv')
```

In [ ]:
```
kashti.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [ ]:
```
#short name to kashti
ks = kashti
ks.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [ ]:
```
#numbers of row and coloumns in data set
ks.shape
```

Out[ ]:
```
(891, 15)
```

In [ ]:
```
ks.tail()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
886	0	2	male	27.0	0	0	13.00	S	Second	man	True	NaN
887	1	1	female	19.0	0	0	30.00	S	First	woman	False	B
888	0	3	female	NaN	1	2	23.45	S	Third	woman	False	NaN

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
889	1	1	male	26.0	0	0	30.00	C	First	man	True	C
890	0	3	male	32.0	0	0	7.75	Q	Third	man	True	NaN

In [ ]:
```
#gives information about numeric variables
ks.describe()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:
```
# unique values
ks.nunique()
```

Out[ ]:
```
survived         2
pclass           3
sex              2
age             88
sibsp            7
parch            7
fare           248
embarked         3
class            3
who              3
adult_male       2
deck             7
embark_town      3
alive            2
alone            2
dtype: int64
```

In [ ]:
```
# coloumn names
ks.columns
```

Out[ ]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

In [ ]:
```
ks['sex'].unique()
```

Out[ ]:
```
array(['male', 'female'], dtype=object)
```

In [ ]:
```
ks['who'].unique()
```

Out[ ]:
```
array(['man', 'woman', 'child'], dtype=object)
```

In [ ]:
```
ks[['sex','who']].nunique()
```

Out[ ]:
```
sex    2
who    3
dtype: int64
```

cleaning and filtering the data

In [ ]:
```
# finding out the misssing Values inside
ks.isnull().sum()
```

Out[ ]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [ ]:
```
# droping down the deck coloumn beacuse too much missing values are there
ks_clean = ks.drop(['deck'], axis=1)
ks_clean.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southar
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cher
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southar
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southar
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southar

In [ ]:
```
ks_clean.isnull().sum()
```

```
survived         0
```

Out[ ]:
```
pclass          0
sex             0
age           177
sibsp           0
parch           0
fare            0
embarked        2
class           0
who             0
adult_male      0
embark_town     2
alive           0
alone           0
dtype: int64
```

In [ ]:
```
ks_clean.shape
```

Out[ ]:
```
(891, 14)
```

In [ ]:
```
891-177
```

Out[ ]:
```
714
```

In [ ]:
```
# droping all the rows having missing values
ks_clean = ks_clean.dropna()
```

In [ ]:
```
ks_clean.shape
```

Out[ ]:
```
(712, 14)
```

In [ ]:
```
ks_clean.isnull().sum()
```

Out[ ]:
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    0
alive          0
alone          0
dtype: int64
```

In [ ]:
```
ks_clean.shape
```

Out[ ]:
```
(712, 14)
```

In [ ]:
```
ks.shape
```

Out[ ]:
```
(891, 15)
```

In [ ]:
```
ks_clean['sex'].value_counts()
```

Out[ ]:
```
male      453
female    259
Name: sex, dtype: int64
```

In [ ]:
```
ks_clean['survived'].value_counts()
```

Out[ ]:
```
0    424
1    288
Name: survived, dtype: int64
```

In [ ]:
```
ks.describe()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:
```
ks_clean.describe()
```

Out[ ]:

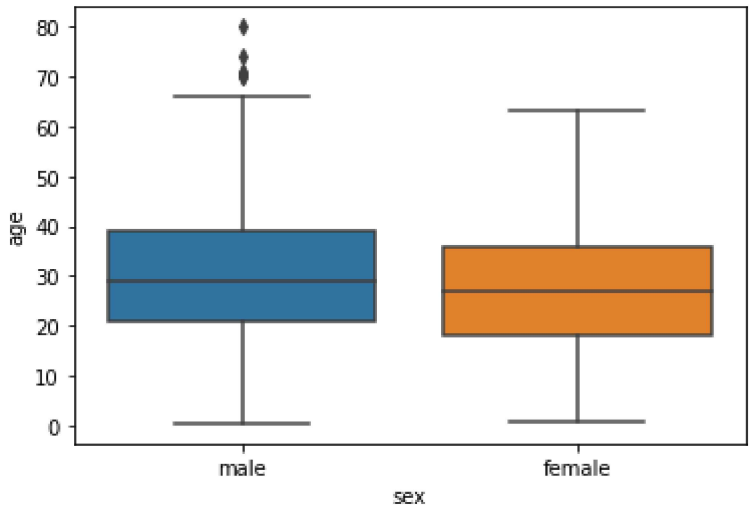
	survived	pclass	age	sibsp	parch	fare
count	712.000000	712.000000	712.000000	712.000000	712.000000	712.000000
mean	0.404494	2.240169	29.642093	0.514045	0.432584	34.567251
std	0.491139	0.836854	14.492933	0.930692	0.854181	52.938648
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	20.000000	0.000000	0.000000	8.050000
50%	0.000000	2.000000	28.000000	0.000000	0.000000	15.645850
75%	1.000000	3.000000	38.000000	1.000000	1.000000	33.000000
max	1.000000	3.000000	80.000000	5.000000	6.000000	512.329200

In [ ]:
```
ks_clean.columns
```

Out[ ]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'embark_town', 'alive',
       'alone'],
      dtype='object')
```

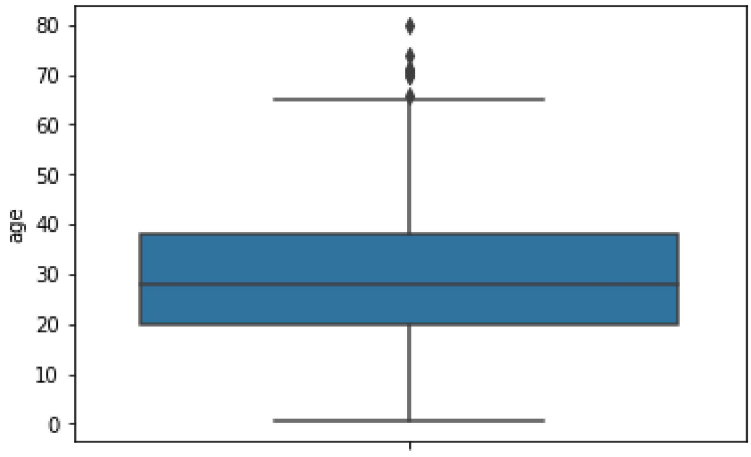
In [ ]:
```
sns.boxplot(x="sex",y="age",data=ks_clean)
```

Out[ ]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



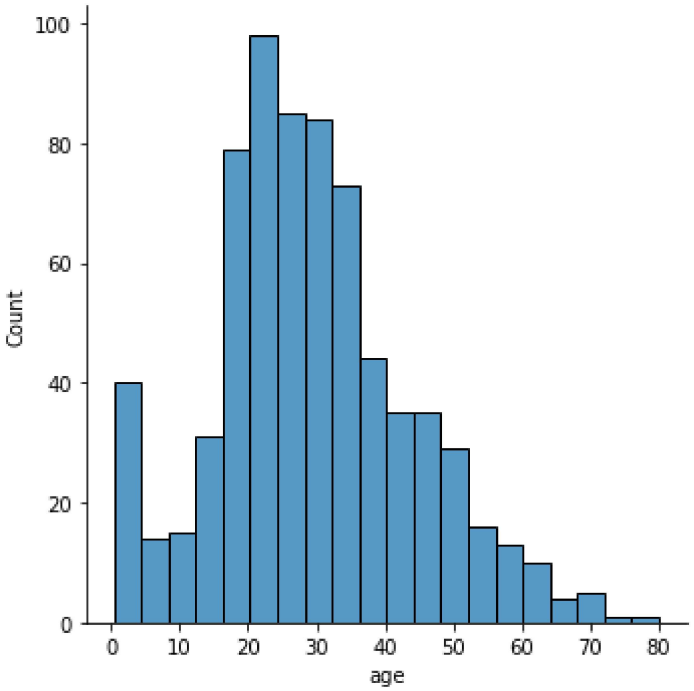
In [ ]:
```
sns.boxplot(y="age",data=ks_clean)
```

Out[ ]:
```
<AxesSubplot:ylabel='age'>
```



In [ ]:
```
sns.displot(ks_clean['age'])
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x12b598f9930>
```



In [ ]:
```
# outliers removal
ks_clean['age'].mean()
```

Out[ ]:
```
29.64209269662921
```

In [ ]:
```
ks_clean = ks_clean[ks_clean['age']<66]
ks_clean.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southar
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cher
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southar
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southar
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southar

In [ ]:
```
ks_clean.shape
```

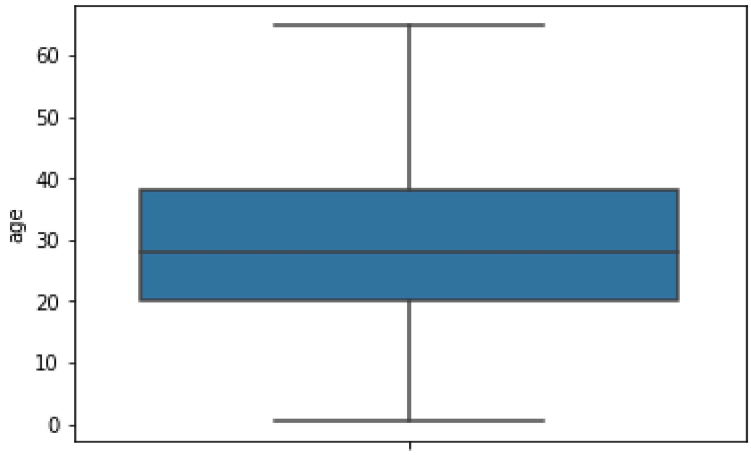
Out[ ]:
```
(704, 14)
```

In [ ]:
```
ks_clean['age'].mean()
```

Out[ ]:
```
29.16572443181818
```

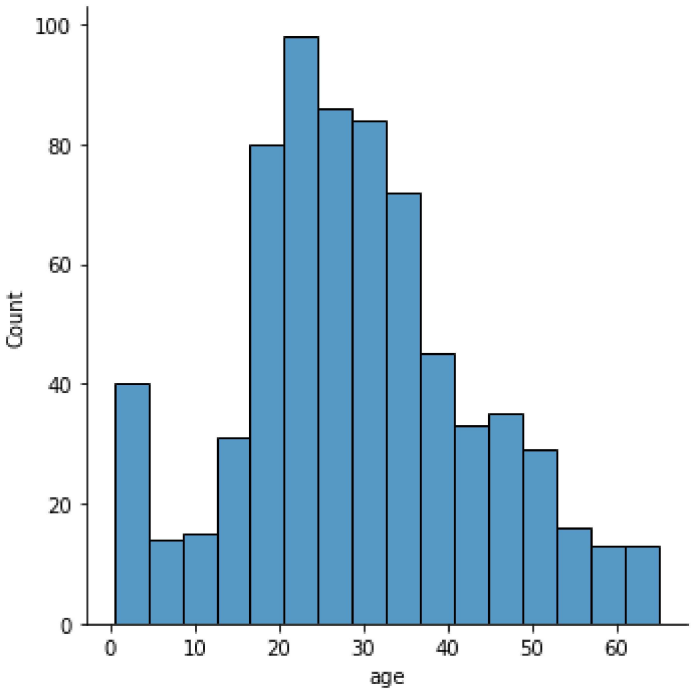
In [ ]:
```
sns.boxplot(y='age',data=ks_clean)
# now check the outliers
```

Out[ ]:
```
<AxesSubplot:ylabel='age'>
```



In [ ]:
```
sns.displot(ks_clean['age'])
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x12b59bb6320>
```



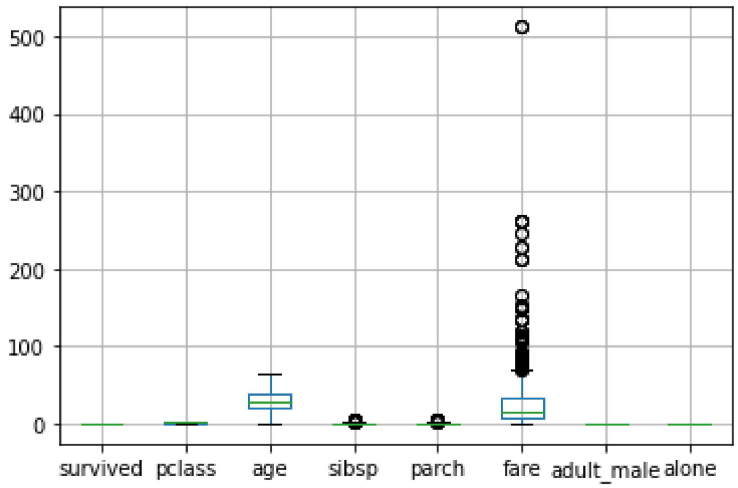
In [ ]:
```
ks_clean.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southar
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cher
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southar
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southar
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southar

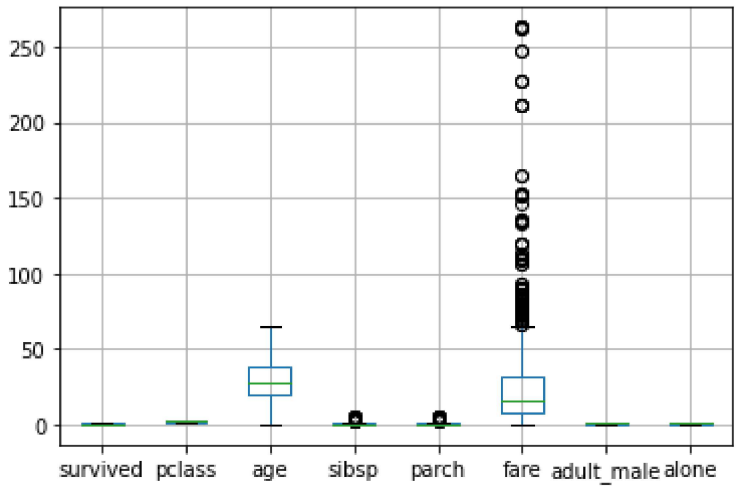
In [ ]:
```
ks_clean.boxplot()
```

Out[ ]:
```
<AxesSubplot:>
```



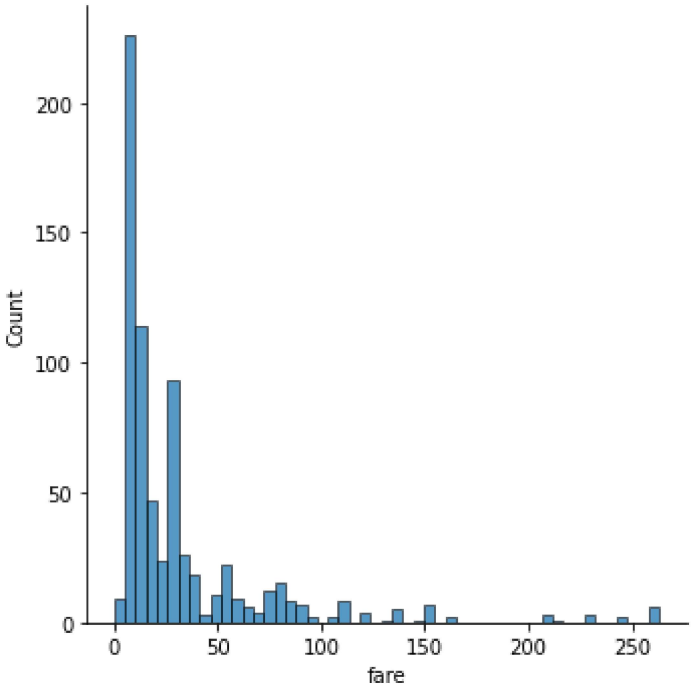
In [ ]:
```
ks_clean = ks_clean[ks_clean['fare']<300]
ks_clean.boxplot()
```

Out[ ]:
```
<AxesSubplot:>
```



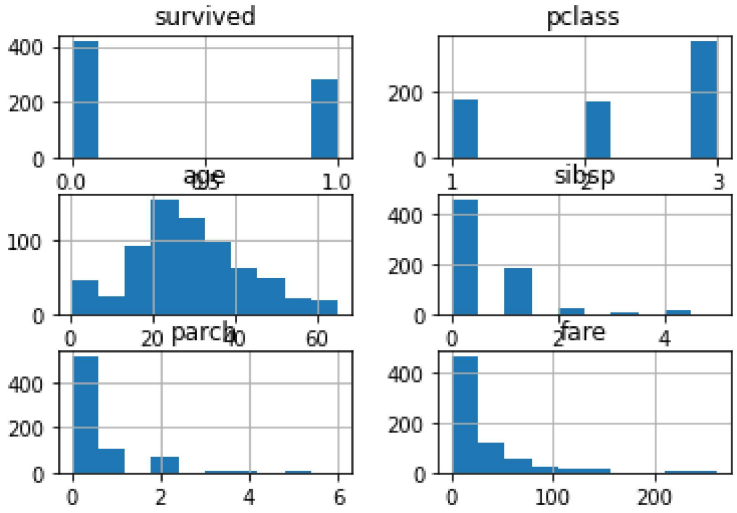
In [ ]:
```
sns.displot(ks_clean['fare'])
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x12b59e992a0>
```



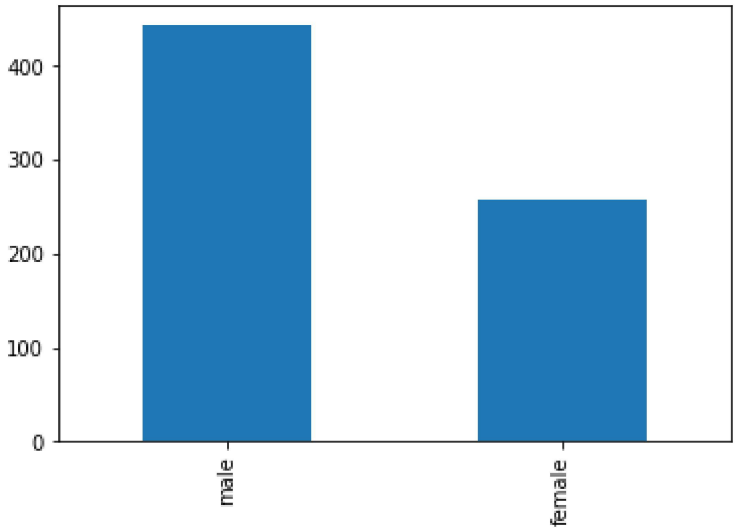
In [ ]:
```
ks_clean.hist()
```

Out[ ]:
```
array([[<AxesSubplot:title={'center':'survived'}>,
        <AxesSubplot:title={'center':'pclass'}>],
       [<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sibsp'}>],
       [<AxesSubplot:title={'center':'parch'}>,
        <AxesSubplot:title={'center':'fare'}>]], dtype=object)
```



In [ ]:
```
pd.value_counts(ks_clean['sex']).plot.bar()
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
ks_clean.groupby(['sex','class']).mean()
```

Out[ ]:

sex	class	survived	pclass	age	sibsp	parch	fare	adult_male	alone
female	First	0.963415	1.0	34.231707	0.560976	0.512195	103.696393	0.000000	0.353659
	Second	0.918919	2.0	28.722973	0.500000	0.621622	21.951070	0.000000	0.405405
	Third	0.460784	3.0	21.750000	0.823529	0.950980	15.875369	0.000000	0.372549
male	First	0.389474	1.0	40.067579	0.389474	0.336842	62.901096	0.968421	0.526316
	Second	0.154639	2.0	29.972474	0.381443	0.247423	21.331959	0.907216	0.628866
	Third	0.151394	3.0	26.143108	0.494024	0.258964	12.197757	0.888446	0.737052

In [ ]:
```
ks.groupby(['sex','class']).mean()
```

Out[ ]:

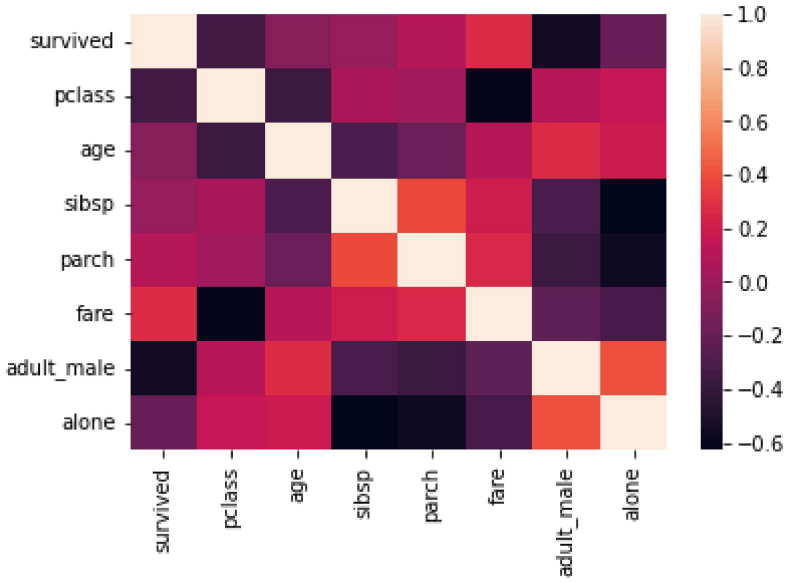
sex	class	survived	pclass	age	sibsp	parch	fare	adult_male	alone
female	First	0.968085	1.0	34.611765	0.553191	0.457447	106.125798	0.000000	0.361702
	Second	0.921053	2.0	28.722973	0.486842	0.605263	21.970121	0.000000	0.421053
	Third	0.500000	3.0	21.750000	0.895833	0.798611	16.118810	0.000000	0.416667
male	First	0.368852	1.0	41.281386	0.311475	0.278689	67.226127	0.975410	0.614754
	Second	0.157407	2.0	30.740707	0.342593	0.222222	19.741782	0.916667	0.666667
	Third	0.135447	3.0	26.507589	0.498559	0.224784	12.661633	0.919308	0.760807

Relationship

In [ ]:
```
corr_ks_clean = ks_clean.corr()
```

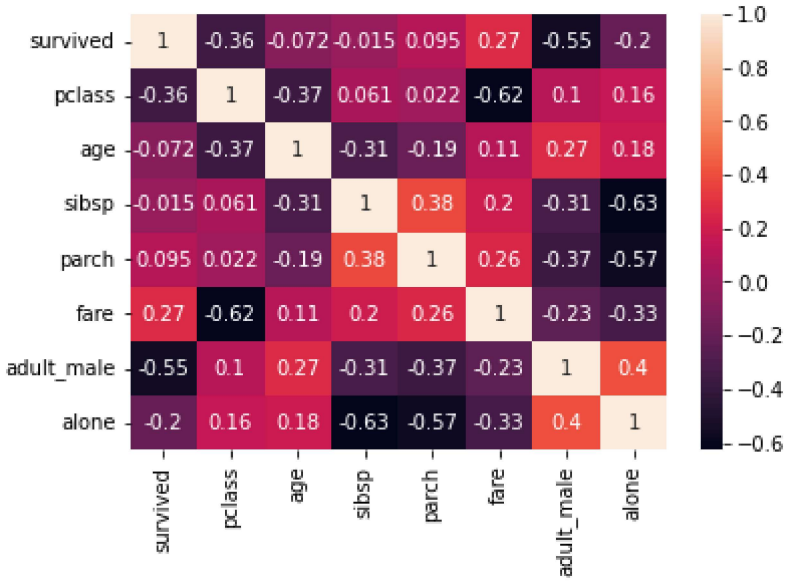
In [ ]:
```
sns.heatmap(corr_ks_clean)
```

Out[ ]:
```
<AxesSubplot:>
```



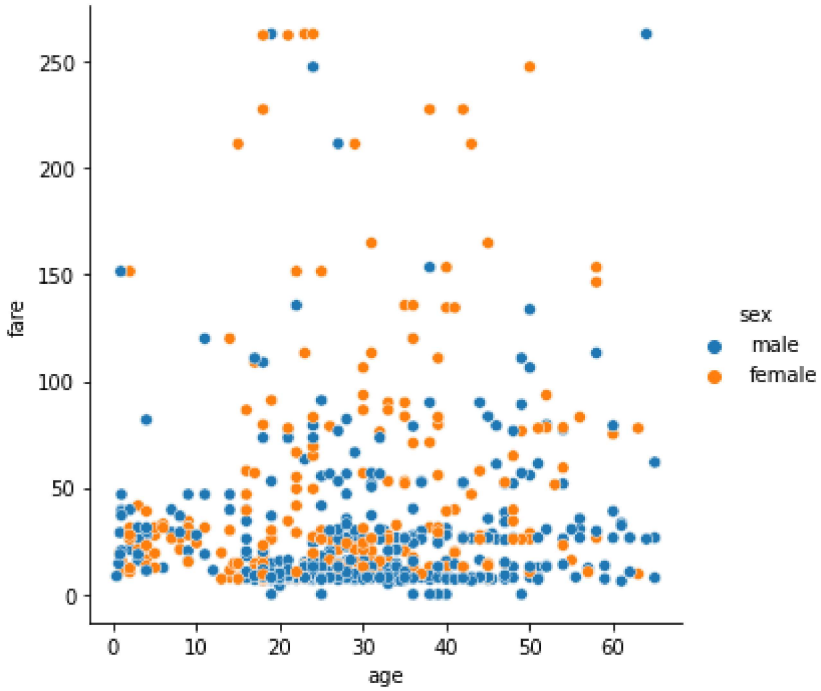
In [ ]:
```
sns.heatmap(corr_ks_clean, annot=True)
```

Out[ ]:
```
<AxesSubplot:>
```



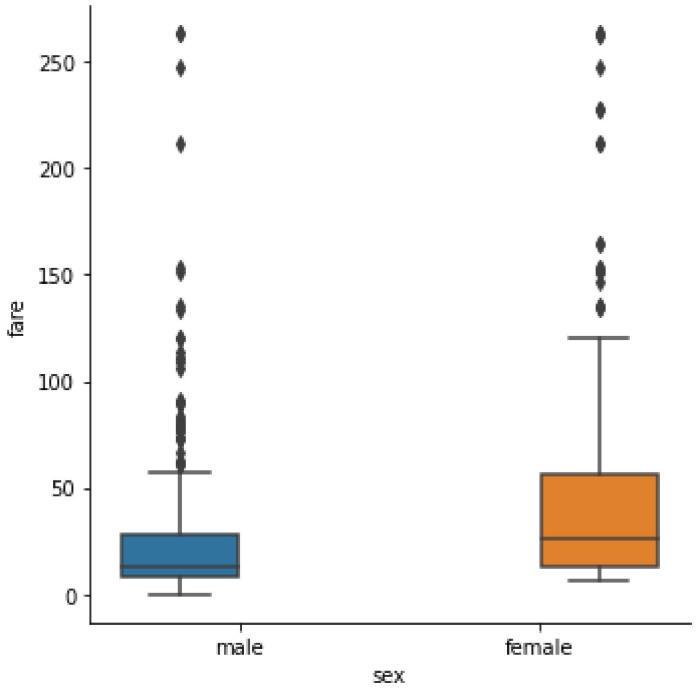
In [ ]:
```
sns.relplot(x='age',y='fare',hue='sex',data=ks_clean)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x12b5a14a800>
```



In [ ]:
```
sns.catplot(x='sex',y='fare',hue='sex',data=ks_clean,kind='box')
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x12b5b200970>
```



In [ ]:
```
# Log transformztion

ks_clean['fare_log']=np.log(ks_clean['fare'])
ks_clean.head()
```

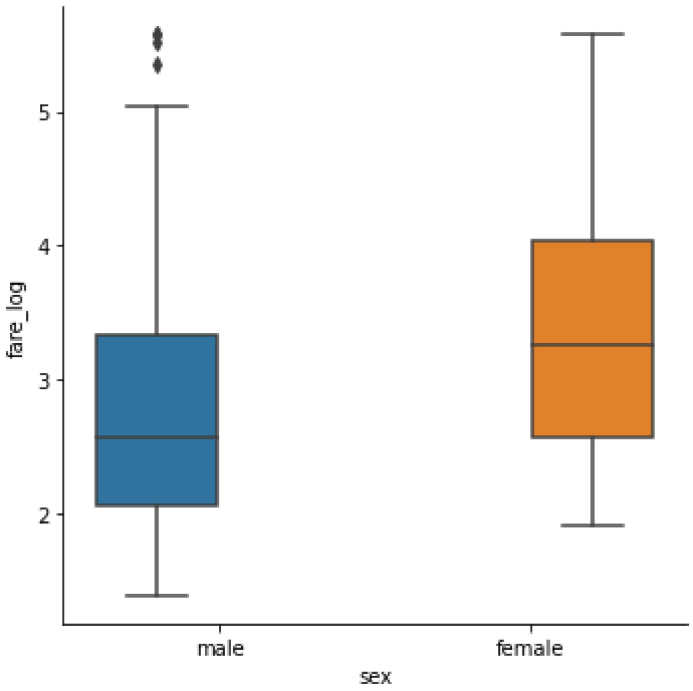
```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\arr
aylike.py:364: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southar
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cher
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southar
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southar
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southar

In [ ]:
```
sns.catplot(x='sex',y='fare_log',hue='sex',data=ks_clean,kind='box')
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x12b5b370ac0>
```



DATA WRANGLING

In [ ]:
```
#import libraries
import pandas as pd
import numpy as np
import seaborn as sns
```

In [ ]:
```
kashti = sns.load_dataset('titanic')
ks1 = kashti
ks2 = kashti
ks = sns.load_dataset('titanic')
```

In [ ]:
```
kashti.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [ ]:

```
# simple operation along the series
# series hum pandas mai kehte hai excel k sheet mai column kehte hain
# hum yahan pr age k column mai aik number add subtract ya koi b operation perform kr s
(kashti['age']+2).head()
```

Out[ ]:

```
0    24.0
1    40.0
2    28.0
3    37.0
4    37.0
Name: age, dtype: float64
```

In [ ]:

```
# an other example
# q k hum ne data set ko update nhi kia to ye operation origina kashti k data set pai a

(kashti['age']-2).head()
```

Out[ ]:

```
0    20.0
1    36.0
2    24.0
3    33.0
4    33.0
Name: age, dtype: float64
```

Dealing with missing Values

- in a data set missing values or either ? or N/A or NaN or zero or blank cell
- Jab kabhi b data na ho kisi aik row mai kisi b aik parameter ka

Steps:

- Koshish kren dubara data collect kar len ya dekh len agr kahin galti hai
- Missing Value wala variable (coloumn) hiii Nikal dein agr data pai effect nahi hota ya simple row or data entry remove kr dein
- Replace the missing Values
  - How?
    - Average Value of Entire variable or similiar data point. for example agr age ka column hai to us ki average (mean) lein or missing values ki jaga pr put kren
    - frequency ya MODE se replace kr dein

- Replace based on other functions(data sampler knows that) , mtlb jis bande ne data collect kia hai wo hi behtr bata skhta hai yaar konsa value add krna hai
- ML algorithms can also be used to find the missing values like linear regression as we used in ML workshop
- Leave it like that(kbhi kbhi khali b chor dete hain)

- Why?
  - it is better because no data is lost
  - Less accurate

In [ ]:
```
# where exactly missing values are
# this function will count k kitne variable mai kitni missing values hain
kashti.isnull().sum()
```

Out[ ]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [ ]:
```
# to see number of rows and columns
kashti.shape
```

Out[ ]:
```
(891, 15)
```

In [ ]:
```
# Use drop.na method which will drop all the missing values rows
kashti=kashti.dropna(subset=['deck'], axis=0) # This function remove specifically from
```

In [ ]:
```
kashti.isnull().sum()
# Now check there is No empty values in Deck column and also it removes some missing va
```

Out[ ]:
```
survived        0
pclass          0
sex             0
age            19
sibsp           0
parch           0
fare            0
embarked        2
class           0
who             0
adult_male      0
deck            0
```

```
embark_town     2
alive           0
alone           0
dtype: int64
```

In [ ]:
```
kashti.shape
```

Out[ ]:
```
(203, 15)
```

In [ ]:
```
# remove na from whole dataframe
kashti = kashti.dropna()
kashti.isnull().sum()
```

Out[ ]:
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
dtype: int64
```

In [ ]:
```
kashti.shape
```

Out[ ]:
```
(182, 15)
```

In [ ]:
```
ks1.isnull().sum()
```

Out[ ]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

replacing missing values with the average of the column

In [ ]:
```
# finding an average (mean)
mean = ks1['age'].mean()
mean
```

Out[ ]:
```
29.69911764705882
```

In [ ]:
```
# replacing nan (missing values) with mean of the data(updating as well)
ks1['age'] = ks1['age'].replace(np.nan,mean)
```

In [ ]:
```
ks1.isnull().sum()
```

Out[ ]:
```
survived         0
pclass           0
sex              0
age              0
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [ ]:
```
Mode= ks1['deck'].mode()
Mode
```

Out[ ]:
```
0    C
Name: deck, dtype: category
Categories (7, object): ['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

Data formatting

- Data ko aik common standard pai lana
- Ensures that data is consistent and understandable
  - Easy to gather
  - easy to work with
    - same units for each columns

In [ ]:
```
# know the data type and convert it into the known one
kashti.dtypes
```

Out[ ]:
```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
```

```
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

In [ ]:
```
# Use this method to convet data types from one to another
kashti['survived'] = kashti['survived'].astype("float64")
kashti.dtypes
```

Out[ ]:
```
survived        float64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

In [ ]:
```
kashti['survived'] = kashti['survived'].astype("int64")
kashti.dtypes
```

Out[ ]:
```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

In [ ]:
```
# here we convert age into days
ks1['age'] = ks1['age'].astype("int64")
ks1['age'] = ks1['age']*365
ks1.head(10)
```

Out[ ]:

survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	decl

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	decl
0	0	3	male	8030	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	13870	1	0	71.2833	C	First	woman	False	C
2	1	3	female	9490	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	12775	1	0	53.1000	S	First	woman	False	C
4	0	3	male	12775	0	0	8.0500	S	Third	man	True	NaN
5	0	3	male	10585	0	0	8.4583	Q	Third	man	True	NaN
6	0	1	male	19710	0	0	51.8625	S	First	man	True	E
7	0	3	male	730	3	1	21.0750	S	Third	child	False	NaN
8	1	3	female	9855	0	2	11.1333	S	Third	woman	False	NaN
9	1	2	female	5110	1	0	30.0708	C	Second	child	False	NaN

In [ ]:
```
# Always rename after wards
ks1.rename(columns={"age":"age in days"},inplace=True)
ks1.head()
```

Out[ ]:

	survived	pclass	sex	age in days	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	8030	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	13870	1	0	71.2833	C	First	woman	False	C
2	1	3	female	9490	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	12775	1	0	53.1000	S	First	woman	False	C
4	0	3	male	12775	0	0	8.0500	S	Third	man	True	NaN

Data Normalization

- Uniform the data
- make sure they have same impact
- bell curve , histogram is normal data
- data dispersion equally on both sides

In [ ]:
```
ks1.head()
```

Out[ ]:

	survived	pclass	sex	age in days	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	8030	1	0	7.2500	S	Third	man	True	NaN

	survived	pclass	sex	age in days	sibsp	parch	fare	embarked	class	who	adult_male	deck
1	1	1	female	13870	1	0	71.2833	C	First	woman	False	C
2	1	3	female	9490	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	12775	1	0	53.1000	S	First	woman	False	C
4	0	3	male	12775	0	0	8.0500	S	Third	man	True	NaN

In [ ]:
```
ks4 = ks1[['age in days','fare']]
ks4.head()
```

Out[ ]:

	age in days	fare
0	8030	7.2500
1	13870	71.2833
2	9490	7.9250
3	12775	53.1000
4	12775	8.0500

- the above data is really in wide range and we need to normalize and hard to comapre
- Normalization changes the values to the range of 0-to-1(now both variables have similiar influence on our models)

Methods of Normalization

1. Simple feature scaling -x(new) = x(old)/x(max)/
2. Min - Max Method
3. Z-score(standard score) (-3 to +3)
4. log transformation

In [ ]:
```
# simple feature scaling method
ks4['fare'] = ks4['fare']/ks4['fare'].max()
ks4['age in days'] = ks4['age in days']/ks4['age in days'].max()
ks4.head()
```

```
C:\Users\Abbas\AppData\Local\Temp\ipykernel_4944\2502560593.py:2: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  ks4['fare'] = ks4['fare']/ks4['fare'].max()
C:\Users\Abbas\AppData\Local\Temp\ipykernel_4944\2502560593.py:3: SettingWithCopyWarnin
g:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  ks4['age in days'] = ks4['age in days']/ks4['age in days'].max()
```

Out[ ]:

	age in days	fare
0	0.2750	0.014151
1	0.4750	0.139136
2	0.3250	0.015469
3	0.4375	0.103644
4	0.4375	0.015713

- now fare and age range comes in the 0 to 1 range data ka difference utna ee hai lekin range kam kr di hum ne

In [ ]:

```
# Min - Max Method
ks4['fare'] = (ks4['fare']-ks4['fare'].min())/(ks4['fare'].max()-ks4['fare'].min())
ks4.head()
```

```
C:\Users\Abbas\AppData\Local\Temp\ipykernel_4944\3915730884.py:2: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  ks4['fare'] = (ks4['fare']-ks4['fare'].min())/(ks4['fare'].max()-ks4['fare'].min())
```

Out[ ]:

	age in days	fare
0	0.2750	0.014151
1	0.4750	0.139136
2	0.3250	0.015469
3	0.4375	0.103644
4	0.4375	0.015713

In [ ]:

```
# Z- score method -3 to +3
ks4['fare']=(ks4['fare']-ks4['fare'].mean())/ks4['fare'].std()
ks4.head()
```

```
C:\Users\Abbas\AppData\Local\Temp\ipykernel_4944\1688616397.py:2: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  ks4['fare']=(ks4['fare']-ks4['fare'].mean())/ks4['fare'].std()
```

Out[ ]:

	age in days	fare
0	0.2750	-0.502163
1	0.4750	0.786404
2	0.3250	-0.488580
3	0.4375	0.420494
4	0.4375	-0.486064

In [ ]:
```
ks.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [ ]:
```
# Log Transformation
ks['fare']=np.log(ks['fare'])
ks.head()
```

```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\arr
aylike.py:364: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	1.981001	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	4.266662	C	First	woman	False	C
2	1	3	female	26.0	0	0	2.070022	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	3.972177	S	First	woman	False	C
4	0	3	male	35.0	0	0	2.085672	S	Third	man	True	NaN

Binning

- Grouping of values into smaller number of values (bins)
- Convert numeric into categories(jawan, bacahy, bhoray) or 1-16,17-30 etc
- to have better understanding of groups _low vs mid vs high price -

converting categories into dummies for example sex k ndr male or female hain to ap male ko 1 keh de or female ko 0 keh

DeprecationWarning

- easy to use for computation
- MALe female(0,1)

In [ ]:
```
pd.get_dummies(ks['sex'])

# how to use grt dummies to change data inside a data frame
```

Out[ ]:

	female	male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
...	...	...
886	0	1
887	1	0
888	1	0
889	0	1
890	0	1

891 rows × 2 columns

MACHINE LEARNING WEEK

Machine learning

as name suggest, Machine learning allows machines to learn and make decesions smartly.

In Machine learning, machines can learn from the data provided or their own experiences. it depends upon the type of machine learning

Types Of ML

- SUPERVISED
  - Works under supervison
  - teacher teaches
  - Prediction
  - Outcome
- Types of supervised learning

    - Classification
        - For CAtegories
    - Regression
        - For numerical Data
            - Supervised Learning Algorithms
            - Logistic Regression
            - K- Nearest Neighbors(K-NN)
            - Support Vector MAchines(SVM)
            - Kernel SVM
            - NAvie Bayes
            - Decesicion Tree Classsification
            - Random forest Classification

- UNSUPERVISED/CLUSTRING
    - No supervison
    - No teacher
    - Self Learning
    - No labeling Of data
    - Find pattern by itself
- Unsupervise Learning Algorithms
    - K-Means Clustering
    - Hierarchical Clustering
    - probablistic Clustering

- SEMISUPERVISED
    - Mixture of supervised and unsupervised
    - Some data is labeled most is not

- REINFORCEMENT
    - hit and trial learning
    - Learn from mistakes
    - REward and Punishment rule
    - Prediction based on reward and punishment
    - Depends on feed back
- Reinforcement Learning Algorithms
    - Model Free reinforcement learning
        - Policy Optimizatin
        - Q-Learning
            - Model BAsed Reinforcement Learning
            - Learn the Model
            - Give the Model

- Supervised Learning Algorithms

- Logistic Regression : Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on.
- KNN : kNN classifier determines the class of a data point by majority voting principle. If k is set to 5, the classes of 5 closest points are checked. Prediction is done according to the majority class. Similarly, kNN regression takes the mean value of 5 closest points.

LINEAR REGRESSION

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('mldata.csv')
```

In [ ]:
```
dataset.head()
```

Out[ ]:

	age	weight	gender	likeness	height
0	27	76.0	Male	Biryani	170.688
1	41	70.0	Male	Biryani	165
2	29	80.0	Male	Biryani	171
3	27	102.0	Male	Biryani	173
4	29	67.0	Male	Biryani	164

In [ ]:
```
# data filtering
mydata = dataset[['age','weight','height']]
mydata.head()
```

Out[ ]:

	age	weight	height
0	27	76.0	170.688
1	41	70.0	165
2	29	80.0	171
3	27	102.0	173
4	29	67.0	164

In [ ]:
```
X=mydata[['age','weight']].values  #get a copy of data set excude last column
y=mydata['height'].values # get array of dataset in column 1
```

In [ ]:
```
X
```

Out[ ]:
```
array([[ 27.  ,  76.  ],
       [ 41.  ,  70.  ],
       [ 29.  ,  80.  ],
       [ 27.  , 102.  ],
       [ 29.  ,  67.  ],
       [ 28.  ,  46.  ],
       [ 27.  ,  64.3 ],
       [ 34.  ,  98.  ],
       [ 32.  ,  87.5 ],
       [ 22.  ,  80.  ],
       [ 29.  ,  65.  ],
       [ 29.  ,  78.  ],
       [ 23.  ,  62.  ],
       [ 28.  ,  74.  ],
       [ 25.  , 102.  ],
       [ 27.  ,  78.  ],
       [ 33.  ,  93.  ],
       [ 28.  ,  68.  ],
       [ 21.  ,  71.5 ],
       [ 25.  , 135.  ],
       [ 24.  ,  82.  ],
       [ 30.  ,  70.  ],
       [ 30.  ,  60.  ],
       [ 32.  ,  84.5 ],
       [ 30.  ,  65.  ],
       [ 25.  ,  70.  ],
       [ 34.  ,  59.  ],
       [ 32.  ,  78.  ],
       [ 24.  ,  68.  ],
       [ 30.  ,  62.  ],
       [ 21.  ,  58.  ],
       [ 31.  , 102.3 ],
       [ 22.  ,  64.  ],
       [ 19.  ,  50.  ],
       [ 30.  ,  75.  ],
       [ 27.  ,  77.  ],
       [ 34.  ,  82.72],
       [ 24.  ,  93.  ],
       [ 26.  ,  55.  ],
       [ 22.  ,  50.  ],
       [ 35.  ,  75.  ],
       [ 27.  ,  64.  ],
       [ 27.  ,  85.  ],
       [ 27.  ,  65.  ],
       [ 30.  ,  84.  ],
       [ 44.  ,  85.  ],
       [ 23.  ,  53.  ],
       [ 48.  ,  90.  ],
       [ 28.  ,  79.  ],
       [ 24.  ,  68.  ],
       [ 40.  ,  85.  ],
       [ 32.  ,  79.  ],
       [ 29.  ,  50.9 ],
       [ 27.  ,  68.5 ],
       [ 26.  , 130.  ],
       [ 30.  ,  63.  ],
       [ 27.  ,  93.  ],
       [ 23.  ,  59.  ],
       [ 28.  ,  76.  ],
       [ 22.  ,  76.  ],
```

```
       [ 23.  ,  60.  ],
       [ 26.  ,  63.  ],
       [ 30.  ,  60.  ],
       [ 31.  ,  75.  ],
       [ 30.  ,  70.  ],
       [ 21.  ,  65.  ],
       [ 20.  ,  70.  ],
       [ 26.  ,  55.  ],
       [ 35.  ,  65.  ],
       [ 38.  ,  72.6 ],
       [ 27.  ,  55.  ],
       [ 27.  ,  46.  ],
       [ 38.  ,  62.  ],
       [ 25.  ,  72.5 ],
       [ 25.  ,  55.5 ],
       [ 24.  ,  70.  ],
       [ 24.  ,  86.  ],
       [ 22.  ,  52.  ],
       [ 25.  ,  43.  ],
       [ 32.  ,  61.5 ],
       [ 22.  ,  76.  ],
       [ 27.  ,  60.  ],
       [ 22.  ,  62.  ],
       [ 37.  ,  77.  ],
       [ 30.  ,  65.  ],
       [ 27.  ,  50.  ],
       [ 33.  ,  91.  ],
       [ 25.  ,  75.  ],
       [ 38.  ,  81.5 ],
       [ 25.  ,  88.  ],
       [ 21.  ,  60.  ],
       [ 31.  ,  86.  ],
       [ 24.  ,  87.  ],
       [ 25.  ,  70.  ],
       [ 18.  ,  55.6 ],
       [ 24.  ,  86.  ],
       [ 35.  ,  82.  ],
       [ 44.  ,  70.  ],
       [ 30.  ,  60.  ],
       [ 32.  ,  50.  ],
       [ 25.  ,  62.  ],
       [ 29.  ,  71.  ],
       [ 24.  ,  55.  ],
       [ 29.  ,  78.  ],
       [ 35.  ,  72.  ],
       [ 46.  ,  75.  ],
       [ 32.  ,  10.92],
       [ 27.  ,  57.  ],
       [ 25.  ,  90.  ],
       [ 29.  ,  65.  ],
       [ 37.  ,  68.  ],
       [ 32.  ,  74.  ],
       [ 20.  ,  52.  ],
       [ 32.  ,  51.  ],
       [ 28.  ,  76.  ],
       [ 27.  ,  82.  ],
       [ 30.  ,  75.  ],
       [ 31.  ,  86.  ],
       [ 27.  ,  80.  ],
       [ 27.  ,  51.  ],
```

```
       [ 29.  ,  78.  ],
       [ 20.  ,  54.  ],
       [ 31.  ,  85.  ],
       [ 27.  ,  58.  ],
       [ 68.  ,  68.  ],
       [ 40.  ,  97.  ],
       [ 24.  ,  66.  ],
       [ 25.  ,  47.  ],
       [ 24.  ,  65.  ],
       [ 26.  ,  60.  ],
       [ 30.  ,  90.  ],
       [ 31.  ,  40.  ],
       [ 36.  ,  62.  ],
       [ 30.  ,  65.  ],
       [ 19.  ,  55.2 ],
       [ 23.  ,  71.  ],
       [ 25.  ,  55.  ],
       [ 27.  ,  54.  ],
       [ 28.  , 161.  ],
       [ 39.  , 102.  ],
       [ 43.  ,  85.  ],
       [ 25.  ,  59.  ],
       [ 28.  ,  83.  ],
       [ 28.  ,  46.  ],
       [ 20.  ,  53.  ],
       [ 23.  ,  62.  ],
       [ 25.  ,  55.  ],
       [ 26.  ,  76.  ],
       [ 31.  ,  68.  ],
       [ 32.  ,  57.  ],
       [ 21.  ,  81.  ],
       [ 31.  ,  80.  ],
       [ 28.  ,  68.  ],
       [ 28.  ,  63.4 ],
       [ 27.  ,  67.  ],
       [ 24.  ,  50.  ],
       [ 19.  ,  50.4 ],
       [ 34.  ,  58.  ],
       [ 22.  ,  67.  ],
       [ 27.  ,  75.  ],
       [ 31.  ,  86.  ],
       [ 33.  ,  86.  ],
       [ 33.  ,  46.7 ],
       [ 21.  ,  45.  ],
       [ 29.  ,   6.  ],
       [ 21.  ,  62.  ],
       [ 26.  ,  90.  ],
       [ 27.  , 101.  ],
       [ 42.  , 179.  ],
       [ 25.  ,  88.  ],
       [ 29.  ,  73.  ],
       [ 21.  ,  50.  ],
       [ 44.  ,  78.  ],
       [ 28.  ,  84.  ],
       [ 30.  ,  67.  ],
       [ 40.  ,  71.  ],
       [ 28.  ,  96.  ],
       [ 26.  ,  70.  ],
       [ 30.  ,  88.  ],
       [ 29.  ,  84.  ],
```

```
       [ 26.  ,  90.  ],
       [ 50.  ,  88.  ],
       [ 35.  ,  75.  ],
       [ 31.  ,  86.  ],
       [ 24.  ,  82.  ],
       [ 21.  ,  68.  ],
       [ 39.  ,  53.  ],
       [ 30.  ,  73.8 ],
       [ 32.  ,  80.  ],
       [ 29.  ,  74.  ],
       [ 27.  ,  80.  ],
       [ 31.  ,  54.  ],
       [ 27.  ,  62.  ],
       [ 25.  ,  64.  ],
       [ 34.  ,  82.  ],
       [ 40.  ,  80.  ],
       [ 34.  ,  79.  ],
       [ 20.  ,  40.  ],
       [ 32.  ,  74.  ],
       [ 34.  ,  98.  ],
       [ 31.  ,  86.  ],
       [ 24.  ,  57.  ],
       [ 28.  ,  65.  ],
       [ 30.  ,  75.  ],
       [ 28.  ,  56.  ],
       [ 20.  ,  78.  ],
       [ 27.  ,  88.  ],
       [ 28.  ,  62.  ],
       [ 24.  ,  75.  ],
       [ 27.  ,  85.  ],
       [ 37.  ,  80.  ],
       [ 28.  ,  74.  ],
       [ 24.  ,  45.  ],
       [ 33.  ,  74.  ],
       [ 29.  ,  88.2 ],
       [ 28.  ,  77.  ],
       [ 26.  ,  60.  ],
       [ 23.  ,  68.  ],
       [ 30.  ,  78.  ],
       [ 43.  ,  85.  ],
       [ 26.  ,  62.  ],
       [ 25.  ,  70.  ],
       [ 21.  ,  60.  ],
       [ 26.  ,  74.  ],
       [ 20.  ,  48.  ],
       [ 22.  ,  53.  ],
       [ 22.  ,  64.  ],
       [ 20.  ,  26.  ],
       [ 27.  ,  59.  ],
       [ 27.  ,  96.  ],
       [ 35.  ,  64.  ],
       [ 27.  ,  68.  ],
       [ 38.  ,  72.2 ],
       [ 25.  ,  66.  ],
       [ 36.  ,  68.  ],
       [ 20.  ,  48.  ],
       [ 30.  ,  50.  ],
       [ 24.  ,  60.  ],
       [ 22.  ,  50.  ],
       [ 27.  ,  63.  ],
```

```
[ 31.  ,  60.  ],
[ 26.  ,  70.  ],
[ 40.  ,  80.  ],
[ 25.  ,  65.  ],
[ 33.  ,  56.  ]])
```

Next, we have to split our dataset (total 244 observations)into 2 sets: training set which used for training and test set which used for testing:

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0
```

In [ ]:
```
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[ ]:
```
LinearRegression()
```

In [ ]:
```
# Predicting the test set result
y_pred = regressor.predict(X_test)
```

In [ ]:
```
regressor.predict([[20,70]])
```

Out[ ]:
```
array([148.918992])
```

In [ ]:
```
regressor.predict([[27,76]])
```

Out[ ]:
```
array([151.56540304])
```

MULTIPLE LINEAR REGRESSION

In [ ]:
```
# import libraries
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
# import your data
df=pd.read_csv('ml_data_salary.csv')
```

In [ ]:
```
df.head()
```

Out[ ]:

	age	distance	YearsExperience	Salary
0	31.1	77.75	1.1	39343

	age	distance	YearsExperience	Salary
1	31.3	78.25	1.3	46205
2	31.5	78.75	1.5	37731
3	32.0	80.00	2.0	43525
4	32.2	80.50	2.2	39891

In [ ]:
```
X=df[['age','distance','YearsExperience']]
y=df['Salary']
```

In [ ]:
```
X.head()
```

Out[ ]:

	age	distance	YearsExperience
0	31.1	77.75	1.1
1	31.3	78.25	1.3
2	31.5	78.75	1.5
3	32.0	80.00	2.0
4	32.2	80.50	2.2

In [ ]:
```
y.head()
```

Out[ ]:
```
0    39343
1    46205
2    37731
3    43525
4    39891
Name: Salary, dtype: int64
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

In [ ]:
```
#  fit your model
from sklearn.linear_model import LinearRegression
model=LinearRegression().fit(X_train,y_train)
model
```

Out[ ]:
```
LinearRegression()
```

In [ ]:
```
model.coef_
```

Out[ ]:
```
array([-1.24318092e+16, -2.60535815e+14,  1.30831487e+16])
```

In [ ]:
```
model.intercept_
```

Out[ ]:
```
3.924944608971833e+17
```

In [ ]:
```
model.predict(X_test)
```

Out[ ]:
```
array([ 40640., 122688.,  64832.,  63040., 115136., 107584.])
```

In [ ]:
```
model.predict([['31.1','77.75','1.1']])
```

```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.p
y:450: UserWarning: X does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.p
y:566: FutureWarning: Arrays of bytes/strings is being converted to decimal numbers if d
type='numeric'. This behavior is deprecated in 0.24 and will be removed in 1.1 (renaming
of 0.26). Please convert your data to numeric values explicitly instead.
  X = check_array(X, **check_params)
```

Out[ ]:
```
array([36864.])
```

DECESION TREE

In [ ]:
```
# import libraries
import pandas as pd
df=pd.read_csv('mldata.csv')
df.head()
```

Out[ ]:

	age	weight	gender	likeness	height
0	27	76.0	Male	Biryani	170.688
1	41	70.0	Male	Biryani	165
2	29	80.0	Male	Biryani	171
3	27	102.0	Male	Biryani	173
4	29	67.0	Male	Biryani	164

In [ ]:
```
#convert gender into dummies data
df['gender'] = df['gender'].replace("Male",1)
df['gender'] = df['gender'].replace("Female",0)
df.tail()
```

Out[ ]:

	age	weight	gender	likeness	height
240	31	60.0	1	Pakora	160
241	26	70.0	1	Biryani	172
242	40	80.0	1	Biryani	178
243	25	65.0	1	Biryani	5.7
244	33	56.0	0	Samosa	157

In [ ]:
```
#selection of input and output variables
X =df[['weight','gender']]
y=df['likeness']
X.head()
```

Out[ ]:

	weight	gender
0	76.0	1
1	70.0	1
2	80.0	1
3	102.0	1
4	67.0	1

In [ ]:
```
y.head()
```

Out[ ]:
```
0    Biryani
1    Biryani
2    Biryani
3    Biryani
4    Biryani
Name: likeness, dtype: object
```

In [ ]:
```
# machine learning algorithm
from sklearn.tree import DecisionTreeClassifier

#create and fit our model

model=DecisionTreeClassifier().fit(X,y)

# prediction
model.predict([[80,1]])
```

Out[ ]:
```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.p
y:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was
fitted with feature names
  warnings.warn(
array(['Biryani'], dtype=object)
```

In [ ]:
```
# how to measure the accurancy of our model
##split data into test and train

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)

# create a  model
model = DecisionTreeClassifier()
model.fit(X_train,y_train)
```

```
predicted_values = model.predict(X_test)
predicted_values
```

Out[ ]:
```
array(['Biryani', 'Biryani', 'Pakora', 'Biryani', 'Samosa', 'Biryani',
       'Pakora', 'Biryani', 'Biryani', 'Biryani', 'Samosa', 'Samosa',
       'Samosa', 'Pakora', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Pakora', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Biryani', 'Biryani', 'Samosa', 'Biryani', 'Samosa',
       'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Biryani', 'Samosa', 'Biryani', 'Biryani', 'Biryani',
       'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani', 'Biryani',
       'Biryani'], dtype=object)
```

In [ ]:
```
# checking score
actual_values = y_test
score = accuracy_score(actual_values,predicted_values)
score
#if > 0.75 then its good
```

Out[ ]:
```
0.6122448979591837
```

In [ ]:
```
# how to train and save your model
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import joblib

model = DecisionTreeClassifier().fit(X,y)
joblib.dump(model,"foodie.joblib")
```

Out[ ]:
```
['foodie.joblib']
```

In [ ]:
```
# how to import/run a stored or saved model on your data assignment
from sklearn.metrics import accuracy_score

loaded_model = joblib.load('foodie.joblib')
score2 = loaded_model.predict([[80,1]])
score2
```

Out[ ]:
```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
array(['Biryani'], dtype=object)
```

In [ ]:
```
#graph
from sklearn import tree
model=DecisionTreeClassifier().fit(X,y)
#graphical evaluation look into what happen
tree.export_graphviz(model,
                     out_file="foodie.dot",
                     feature_names=['age','gender'],
                     class_names=sorted(y.unique()),
                     label='all',
                     rounded=True,
                     filled=True)
```

DECESION TREE CLASSIFIER PART 2

In [ ]:
```
#iport libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df = sns.load_dataset('iris')
df.head()
```

Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier

X = df.iloc[:,:-1]
y = df.iloc[:,-1:]
X.head()
```

Out[ ]:

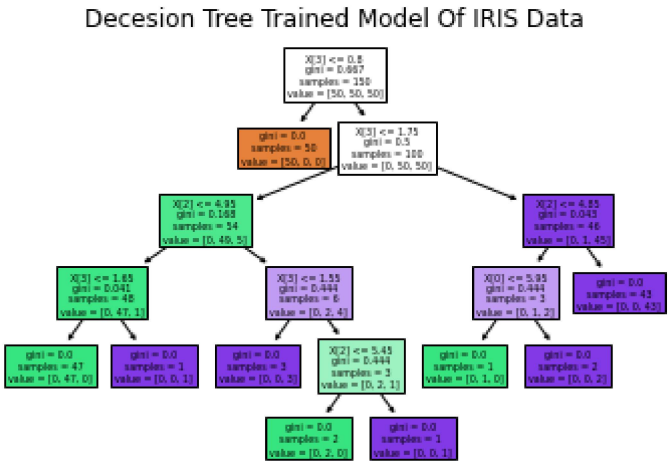
	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [ ]:
```
y.head()
```

Out[ ]:

	species
0	setosa
1	setosa
2	setosa
3	setosa
4	setosa

In [ ]:
```
from sklearn.tree import plot_tree

model=DecisionTreeClassifier().fit(X,y)
plot_tree(model,filled=True)
plt.title("Decesion Tree Trained Model Of IRIS Data")

#how to save this file in tif,png and pdf files in HD Quality?

#plt.savefig('decesiontree.png',dpi=200)

plt.savefig("tiff_compressed.tiff",dpi=600,format="tiff",
facecolor='white',edgecolor='none',pil_kwargs={"compression":"tiff_lzw"})
plt.show()
```



In [ ]:
```
#assignment spilit in 80 20, 70,30,90,10 data and check accuracy
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
model=DecisionTreeClassifier().fit(X_train,y_train)
```

In [ ]:
```
prdicted_values=model.predict(X_test)
prdicted_values
```

Out[ ]:
```
array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
       'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'setosa', 'setosa', 'versicolor', 'versicolor',
       'setosa', 'virginica', 'versicolor', 'setosa', 'virginica',
       'virginica', 'versicolor', 'setosa', 'virginica', 'versicolor',
       'versicolor', 'virginica', 'setosa', 'virginica', 'setosa',
       'setosa'], dtype=object)
```

In [ ]:
```
score = accuracy_score(prdicted_values,y_test)
score
```

Out[ ]:
```
0.9777777777777777
```

In [ ]:

```
# at spilit 80 20 and 90,10 accuracy is 100% =1
# and at spilit 70,30 accuracy is 97 % =0.9777777

#prdicted unknown samples

unknown_samples = [[4.1,5.5,1.5,0.3],[4.2,5.5,1.8,0.5],[6.1,7.5,5.5,0.5]]
model.predict(unknown_samples)
```

```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.p
y:450: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was
fitted with feature names
  warnings.warn(
array(['setosa', 'setosa', 'setosa'], dtype=object)
```

Out[ ]:

RANDOM FOREST CLASSIFICATION

In [ ]:

```
# Load Sample data
import pandas as pd
import numpy as np
import seaborn as sns
df = sns.load_dataset('iris')
df.head()
```

Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [ ]:

```
X = df.iloc[:,:-1]
y = df.iloc[:,-1:]
X.head()
```

Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [ ]:

```
y.head()
```

Out[ ]:

	species

	species
0	setosa
1	setosa
2	setosa
3	setosa
4	setosa

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100) # assignment what is n_estimator
model.fit(X,y)
model.predict([[10,4,5,6]])
```

```
C:\Users\Abbas\AppData\Local\Temp\ipykernel_10908\1582739845.py:3: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples,), for example using ravel().
  model.fit(X,y)
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.p
y:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was
fitted with feature names
  warnings.warn(
```

Out[ ]:
```
array(['virginica'], dtype=object)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
predicted_values = model.predict(X_test)
predicted_values
```

Out[ ]:
```
array(['virginica', 'setosa', 'virginica', 'versicolor', 'virginica',
       'setosa', 'virginica', 'versicolor', 'versicolor', 'versicolor',
       'setosa', 'versicolor', 'virginica', 'versicolor', 'setosa',
       'setosa', 'virginica', 'virginica', 'versicolor', 'setosa',
       'setosa', 'versicolor', 'virginica', 'versicolor', 'virginica',
       'virginica', 'virginica', 'setosa', 'versicolor', 'versicolor'],
      dtype=object)
```

In [ ]:
```
# accuracy score
score = model.score(X_test,y_test)
print("the accuracy score is ",score)
```

```
the accuracy score is  1.0
```

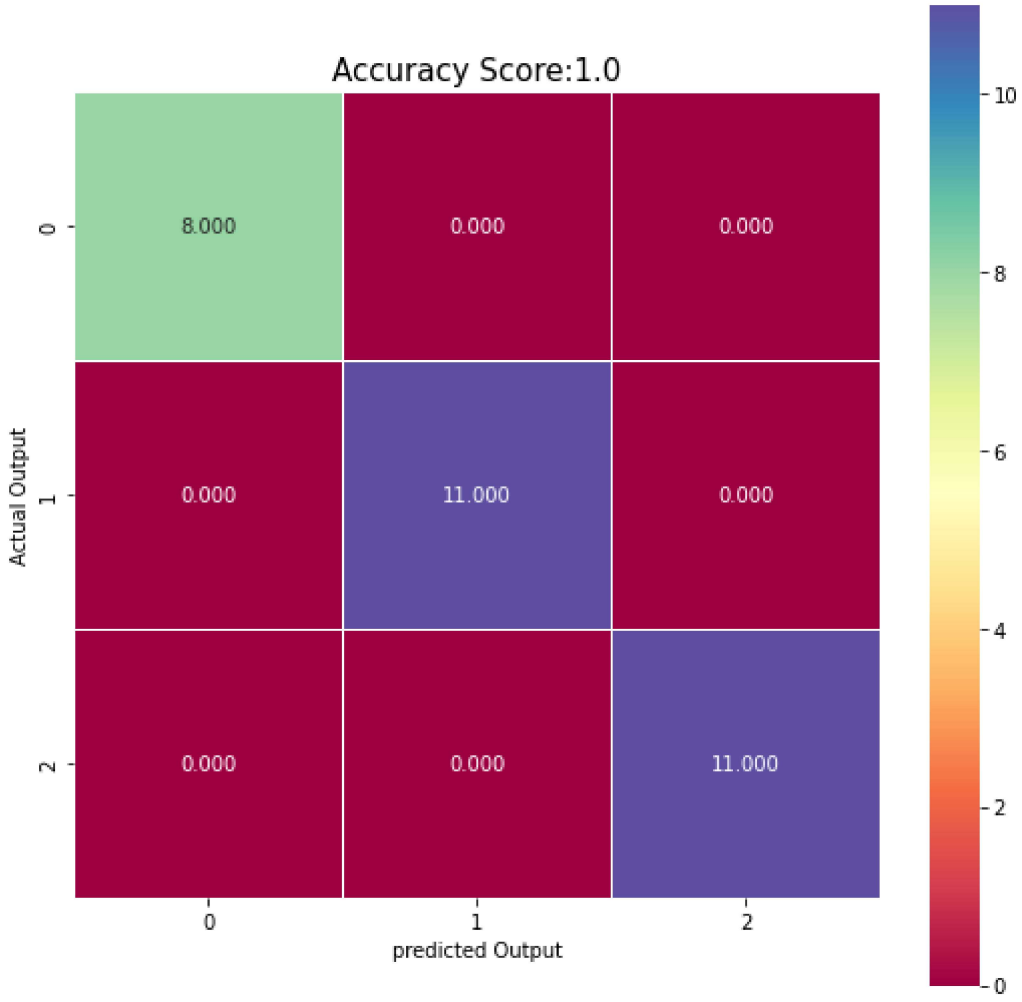
In [ ]:
```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test,predicted_values))
```

```
Accuracy: 1.0
```

In [ ]:
```
# confusion matrix
cm = metrics.confusion_matrix(y_test,predicted_values)
cm
```

Out[ ]:
```
array([[ 8,  0,  0],
       [ 0, 11,  0],
       [ 0,  0, 11]], dtype=int64)
```

In [ ]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(9,9))
sns.heatmap(cm,annot=True,fmt=".3f",linewidths=.5,square=True,cmap="Spectral");
plt.ylabel("Actual Output")
plt.xlabel("predicted Output")
all_sample_title = 'Accuracy Score:{0}'.format(score)
plt.title(all_sample_title,size=15);
```



K NEAREST NEIGHBOUT KNN

K nearest neighbor accuracy measurement

important

- Jaccard index
- F1-score
- log loass
- many more
    - Classification accuracy

```
- Confusion matrix
- Area under Curve
- Mean Absolute Error
- Mean squared error
```

K - nearest neighbour

Pros

- Training phase is faster
- instance based learning alogrith
- can be used with non linear data ### cons
- Testing phase is slower
- costly for memory and Computation
- Not suitable for large dimensions ### How to Improve
  - Data Wrangling & Scaling
  - Missing Valyes
  - Normalization on same scale for every thing
  - Reduce dimensions to improve performance

In [ ]:
```
 # import libraries annd data set
import pandas as pd
df=pd.read_csv('mldata.csv')
df.head()
```

Out[ ]:

	age	weight	gender	likeness	height
0	27	76.0	Male	Biryani	170.688
1	41	70.0	Male	Biryani	165
2	29	80.0	Male	Biryani	171
3	27	102.0	Male	Biryani	173
4	29	67.0	Male	Biryani	164

In [ ]:
```
df['gender']=df['gender'].replace("Male",1)
df['gender']=df['gender'].replace("Female",0)
df.head()
```

Out[ ]:

	age	weight	gender	likeness	height
0	27	76.0	1	Biryani	170.688
1	41	70.0	1	Biryani	165
2	29	80.0	1	Biryani	171
3	27	102.0	1	Biryani	173
4	29	67.0	1	Biryani	164

In [ ]:
```
# Selection of input and output variable
X=df[["age","gender"]]
y=df['likeness']
```

In [ ]:
```
# model and predict
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
#train the model
model.fit(X,y)
#predict output
predicited = model.predict([[70,1]]) #70 weight and male
predicited
```

```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.p
y:450: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fi
tted with feature names
  warnings.warn(
```
Out[ ]:
```
array(['Biryani'], dtype=object)
```

In [ ]:
```
# metrice for evaluation
## spilit data into test and train
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)

#create a model
model = KNeighborsClassifier(n_neighbors=5).fit(X_train,y_train)

predicted_value = model.predict(X_test)
predicted_value

#checking score
actualValues = y_test
score = accuracy_score(actualValues, predicted_value)
print("The accuracy score of our model is =",score)
```

```
The accuracy score of our model is = 0.5306122448979592
```

LOGISTIC REGRESSION

IMAGE CLASSIFICATION

In [ ]:
```
# import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
# import online data
from sklearn.datasets import load_digits
```

```
digits = load_digits()
```

In [ ]:
```
# input variable or features (X)
X=digits.data
X.shape

# means 1797 pictures each of size 64=8*8
```

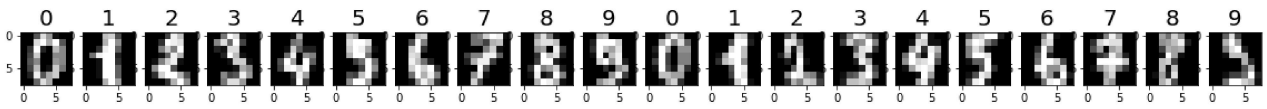
Out[ ]:
```
(1797, 64)
```

In [ ]:
```
# output labels (y)
y=digits.target
y.shape
```

Out[ ]:
```
(1797,)
```

In [ ]:
```
plt.figure(figsize=(20,4))
for index,(image,label)in enumerate(zip(digits.data[0:10], digits.target[0:10])):
    plt.subplot(1,10,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title('Training: %i\n' % label, fontsize = 20)
```

In [ ]:
```
plt.figure(figsize=(20,4))
for index,(image,label)in enumerate(zip(digits.data[0:20], digits.target[0:20])):
    plt.subplot(1,20,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title( label, fontsize = 20)
```



In [ ]:
```
#split the data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.2, random_state=0)
```

In [ ]:
```
print('train input data', X_train.shape)
print('test input data', X_test.shape)
print('train output data', y_train.shape)
print('test outnput data', y_test.shape)
```

```
train input data (1437, 64)
test input data (360, 64)
```

```
train output data (1437,)
test outnput data (360,)
```

In [ ]:
```
# train model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression().fit(X_train,y_train)
model
```

```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_
model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[ ]:
```
LogisticRegression()
```

In [ ]:
```
# prediction
model.predict(X_test[0:5]) #5 rows
```

Out[ ]:
```
array([2, 8, 2, 6, 6])
```

In [ ]:
```
model.predict(X_test[0:20]) #20 rows
predictions = model.predict(X_test)
```

In [ ]:
```
# accuracy test
score  = model.score(X_test,y_test) # X-test  ko dekha or jo pridicted values aai usko
print("the accuracy score is ",score)
```

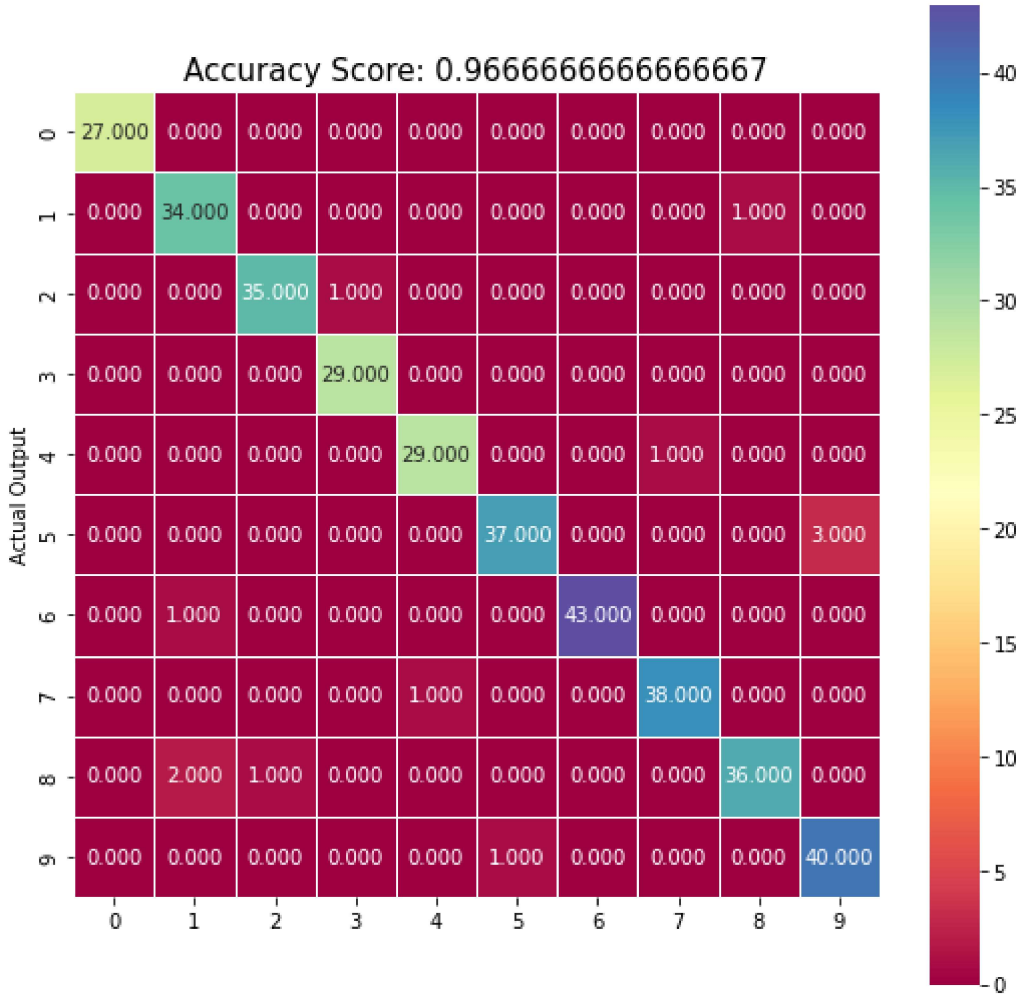
```
the accuracy score is  0.9666666666666667
```

In [ ]:
```
# confusion matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,predictions)
cm
```

Out[ ]:
```
array([[27,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 34,  0,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  0, 35,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 29,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 29,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 37,  0,  0,  0,  3],
       [ 0,  1,  0,  0,  0,  0, 43,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  0, 38,  0,  0],
       [ 0,  2,  1,  0,  0,  0,  0,  0, 36,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  0,  0, 40]], dtype=int64)
```

In [ ]:
```
plt.figure(figsize=(9,9))
sns.heatmap(cm , annot=True, fmt=".3f",linewidths=.5,square=True,cmap='Spectral');
plt.ylabel('Actual Output');
```

```
all_sample_title = "Accuracy Score: {0}".format(score)
plt.title(all_sample_title,size=15);
```

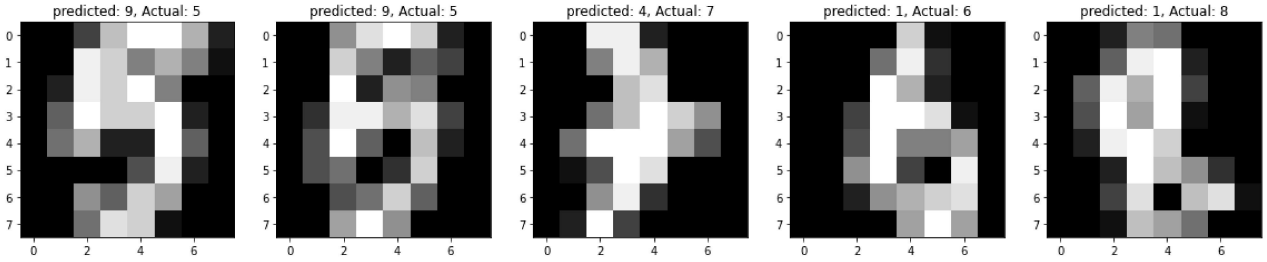


In [ ]:

```
# Getting miss classified labels
index=0
misclassifiedIndexes=[]
for label, predict in zip(y_test,predictions):
    if label != predict:
        misclassifiedIndexes.append(index)
    index+=1
```

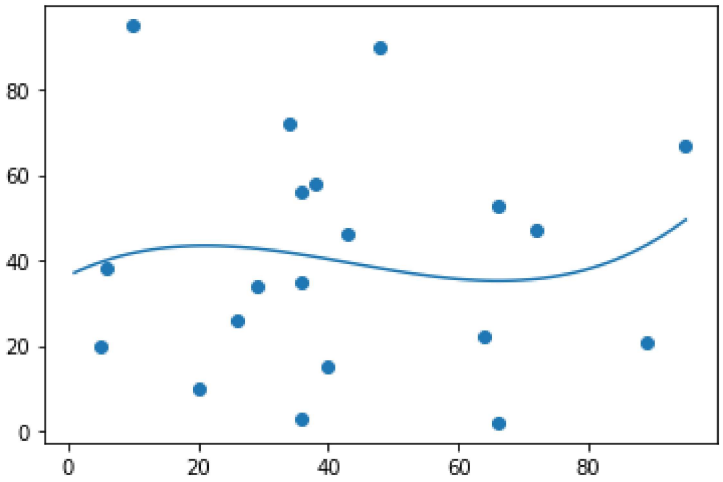
In [ ]:

```
# plotiing missclassified labels with known labels
plt.figure(figsize=(20,4))
for plotIndex, badIndex in enumerate(misclassifiedIndexes[0:5]):
    plt.subplot(1,5,plotIndex+1)
    plt.imshow(np.reshape(X_test[badIndex],(8,8)),cmap=plt.cm.gray)
    plt.title("predicted: {}, Actual: {}".format(predictions[badIndex],y_test[badIndex]
```



POLYNOMIAL REGRESSION

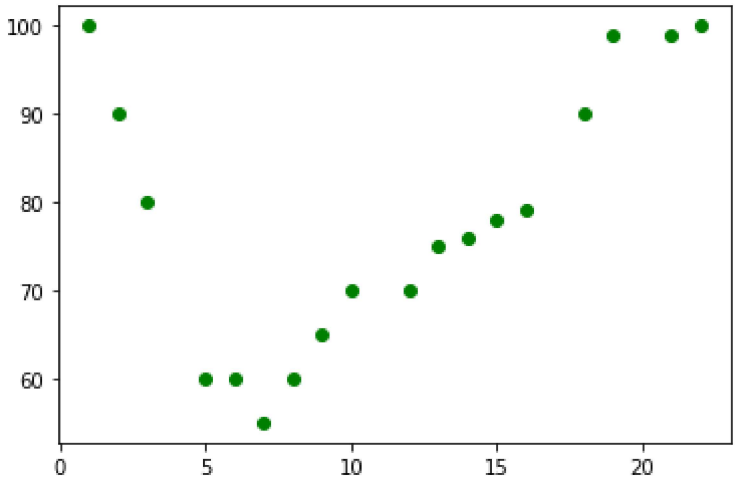
In [ ]:
```
# Example of bad fit
import numpy as np
import matplotlib.pyplot as plt

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y =[21,46,3,35,67,95,53,72,58,10,26,34,90,22,38,20,56,2,47,15]

mymodel = np.poly1d(np.polyfit(x,y,3))

myline = np.linspace(1,95,100)

plt.scatter(x,y)
plt.plot(myline,mymodel(myline))
plt.show()
```



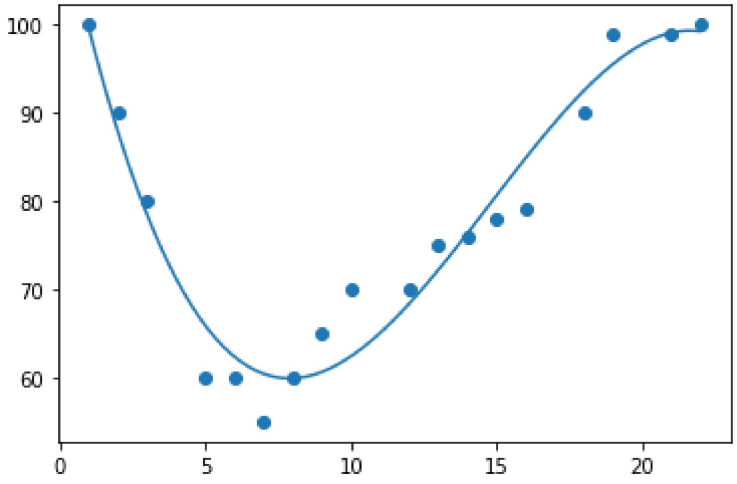
In [ ]:
```
# R-squared for bad fit
from sklearn.metrics import r2_score
print(r2_score(y,mymodel(x)))
```

```
0.016749723671793948
```

In [ ]:
```
# step 1 Data
import matplotlib.pyplot as plt
x=[1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y=[100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]
plt.scatter(x,y,color="green")
plt.show()
```



In [ ]:
```
# Step 2 draw line
import matplotlib.pyplot as plt
x=[1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y=[100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

model = np.poly1d(np.polyfit(x,y,3)) #here 3 means degree 3 line

myline=np.linspace(1,22,100)

plt.scatter(x,y)

plt.plot(myline,model(myline))
plt.show()
```



In [ ]:
```
# Step 3 : R squared
from sklearn.metrics import r2_score

print(r2_score(y,model(x)))  #Best fit
```

```
0.9432150416451026
```

In [ ]:
```
# Prediction
speed =model(18)
print(speed)
```

```
92.48673749579999
```

hand on Example

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('https://s3.us-west-2.amazonaws.com/public.gamelab.fun/dataset/positio
df.head()
```

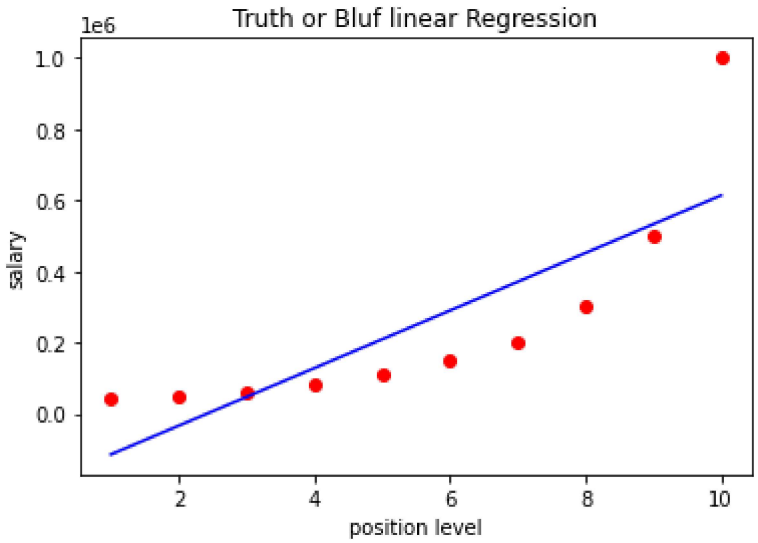
Out[ ]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

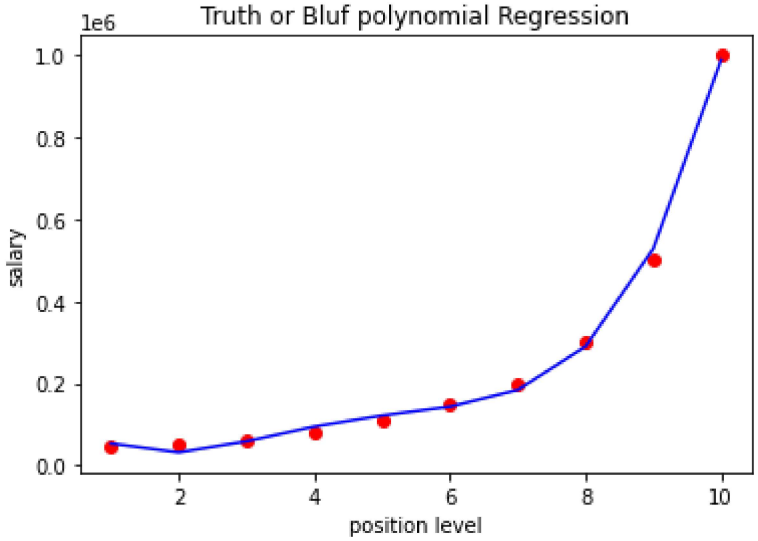
In [ ]:
```
X = df.iloc[:,1:2].values
y = df.iloc[:,2].values
```

In [ ]:
```
# Spiliting data into train and test
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

In [ ]:
```
# fitting linear regression to the daa set
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(X,y)
# visualizing the linear regression result
def viz_linear():  #function
    plt.scatter(X,y,color="red")
    plt.plot(X,model.predict(X),color="Blue")
    plt.title('Truth or Bluf linear Regression')
    plt.xlabel('position level')
    plt.ylabel("salary")
    plt.show()
    return
viz_linear()
```



In [ ]:
```
# fit polynomial regression to the data set
from sklearn.preprocessing import PolynomialFeatures
polymodel = PolynomialFeatures(degree=4)
X_poly = polymodel.fit_transform(X)
poly_reg = LinearRegression()
poly_reg.fit(X_poly,y)

def viz_poly():  #function
    plt.scatter(X,y,color="red")
    plt.plot(X, poly_reg.predict(polymodel.fit_transform(X)),color='blue')
    plt.title('Truth or Bluf polynomial Regression')
    plt.xlabel('position level')
    plt.ylabel("salary")
    plt.show()
    return
viz_poly()
```



In [ ]:
```
#predicting a new result with linear regeression
model.predict([[11]])
```

Out[ ]:
```
array([694333.33333333])
```

In [ ]:
```
# predict a new result with polynomial regression
poly_reg.predict(polymodel.fit_transform([[11]]))
```

Out[ ]:
```
array([1780833.33333359])
```

NAIVE_BAYES_CLASSIFICATION

In [ ]:
```
#import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:
```
df = sns.load_dataset('iris')
df.head()
```

Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [ ]:
```
# input and output
X = df.iloc[:,:-1]  # features data
y = df.iloc[:,-1:]  # labels
X.head()
```

Out[ ]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [ ]:
```
y.head()
```

Out[ ]:

	species
0	setosa
1	setosa

	species
2	setosa
3	setosa
4	setosa

In [ ]:
```
#training and fitting the model
from sklearn.naive_bayes import GaussianNB #assignment what is gaussianNb and other typ
model = GaussianNB().fit(X,y)
model
```

```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\v
alidation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array wa
s expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[ ]:
```
GaussianNB()
```

In [ ]:
```
# Train test and spilit and testing the accuracy
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

In [ ]:
```
# train model
model = GaussianNB().fit(X_train,y_train)
#making prediction
predicted_values=model.predict(X_test)
predicted_values
```

```
C:\Users\Abbas\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\v
alidation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array wa
s expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[ ]:
```
array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
       'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'setosa', 'setosa', 'versicolor', 'versicolor',
       'setosa'], dtype='<U10')
```

In [ ]:
```
from sklearn import metrics
score= metrics.accuracy_score(y_test,predicted_values)
print("GAussian naive byes Model Accuracy (in %) = ",(score)*100,"%")
```

```
GAussian naive byes Model Accuracy (in %) =  96.66666666666667 %
```

SUPPORT VECTOR MACHINES (SVM)

In [ ]:
```
#import scikit-learn dataset librarry
from sklearn import datasets
```

```
#load data set
cancer = datasets.load_breast_cancer()
```

In [ ]:
```
#print the name of features
print("Features",cancer.feature_names)

#print the label type of cancer('maligant','benign')
print("labels",cancer.target_names)
```

```
Features ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
labels ['malignant' 'benign']
```

In [ ]:
```
cancer.data.shape
```

Out[ ]:
```
(569, 30)
```

In [ ]:
```
print(cancer.data[0:5])
```

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.414e-01
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.723e+01
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.491e+01
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.575e-01
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.980e-01
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.444e+01
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.254e+01
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.625e-01
  2.364e-01 7.678e-02]]
```

In [ ]:
```
#print the cancer labels(0:milgnant, 1:benign)
print(cancer.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
1 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(cancer.data, cancer.target,test_size=0
```

In [ ]:
```
#import SVM
from sklearn import svm
#create a svm classifier
model = svm.SVC(kernel='linear') #linear kernal

#train the model
model.fit(X_train,y_train)

#predict the responce for the test data

y_pred = model.predict(X_test)
```

In [ ]:
```
#import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
score = metrics.accuracy_score(y_test,y_pred)
print(score)
```

```
0.97
```

In [ ]:
```
# model precision: what percentage of positive tuples are labeled as such?
print("precision",metrics.precision_score(y_test,y_pred))

# model recall: what percentage of positive tuples are labeled as such?
print("Recall",metrics.recall_score(y_test,y_pred))
```

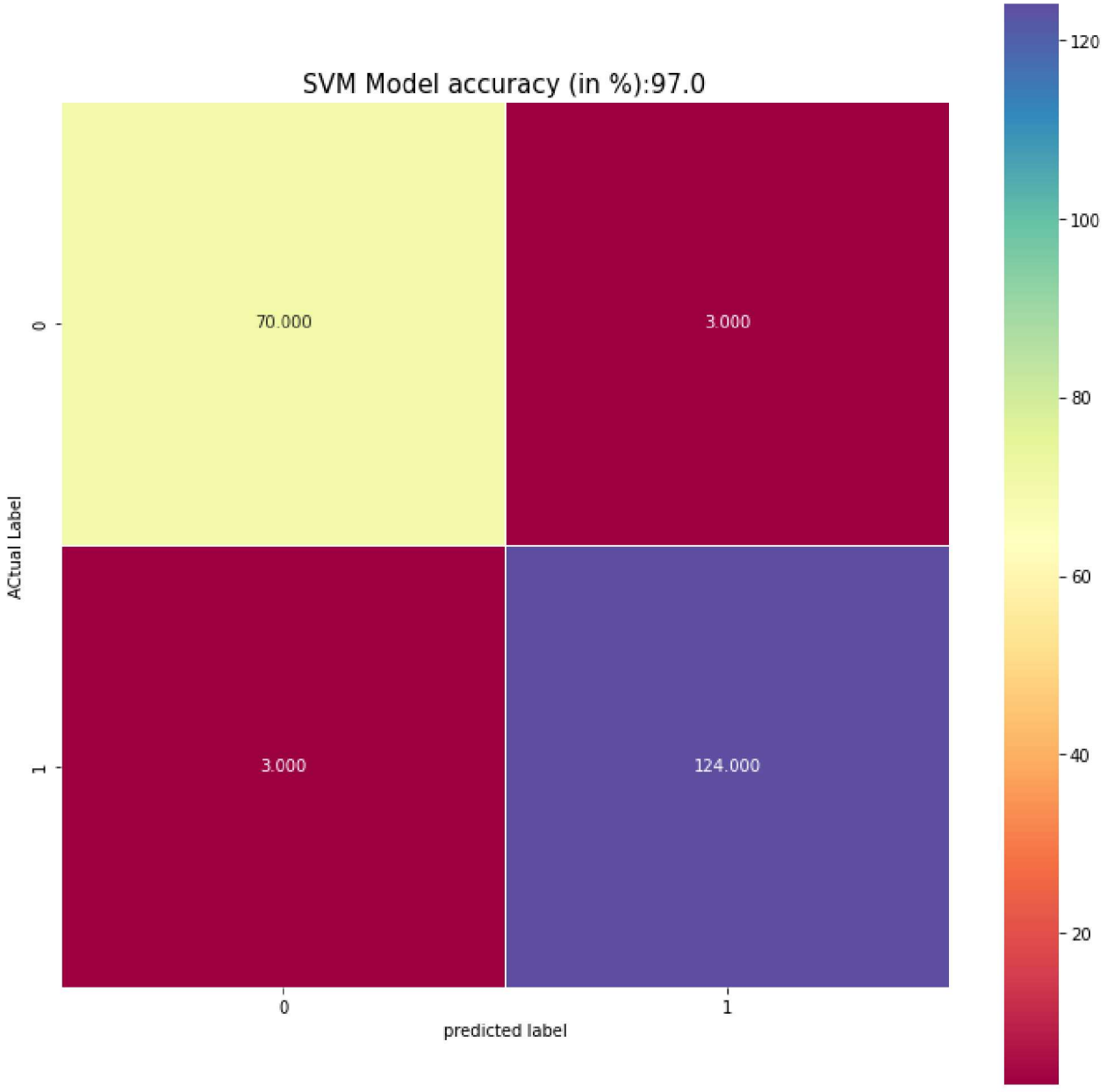
```
precision 0.9763779527559056
Recall 0.9763779527559056
```

In [ ]:
```
# Confusion Matrix
cm = metrics.confusion_matrix(y_test,y_pred)
cm
```

Out[ ]:
```
array([[ 70,   3],
       [  3, 124]], dtype=int64)
```

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
sns.heatmap(cm,annot=True,fmt=".3f",linewidths=.5,square=True,cmap='Spectral')
plt.ylabel("ACtual Label")
plt.xlabel("predicted label")
all_sample_title = 'SVM Model accuracy (in %):{0}'.format(score*100)
plt.title(all_sample_title,size=15)
```

Out[ ]:
```
Text(0.5, 1.0, 'SVM Model accuracy (in %):97.0')
```



COMPUTER VISION

01-Reading and displaying an image

In [ ]:
```
#import library
import cv2 as cv
```

```
mypic = cv.imread("resources/mypic.jpeg")
cv.imshow("pehli image",mypic)
cv.waitKey(0)
```

Out[ ]: 81

02-Resizing the image

In [ ]:
```
## Resizing the image

#import library
import cv2 as cv
from cv2 import imshow
mypic = cv.imread("resources/mypic.jpeg")
mypic2 = cv.resize(mypic,(500,400))

# display
cv.imshow("original",mypic)
cv.imshow("resized image",mypic2)
cv.waitKey(0)
cv.destroyAllWindows()
```

03-GREY SCALE CONVERSION

In [ ]:
```
##GREY SCALE CONVERSION
import cv2 as cv
mypic = cv.imread("resources/mypic.jpeg")
mypic2 = cv.resize(mypic,(500,400))

grey_img = cv.cvtColor(mypic2,cv.COLOR_BGR2GRAY)

# display code
cv.imshow("grey",grey_img)
cv.imshow("resized image",mypic2)

#delay code
cv.waitKey(0)
cv.destroyAllWindows()
```

04-CONVERT AN IMAGE INTO BLACK AND WHITE IMAGE

In [ ]:
```
## CONVERT AN IMAGE INTO BLACK AND WHITE IMAGE

import cv2 as cv
from cv2 import threshold
from cv2 import imshow
img = cv.imread("resources/mypic.jpeg")
img1 = cv.resize(img,(500,400))

gray = cv.cvtColor(img1,cv.COLOR_BGR2GRAY)
```

```
(thresh, binary) = cv.threshold(gray, 127,255,cv.THRESH_BINARY)

cv.imshow("original",img1)
cv.imshow("gray",gray)
cv.imshow("Black and white",binary)

cv.waitKey(0)
cv.destroyAllWindows()
```

05-WRITING IMAGE OR SAVING AN IMAGE

In [ ]:

```
## WRITING IMAGE OR SAVING AN IMAGE

import cv2 as cv
img = cv.imread("resources/mypic.jpeg")
img1 = cv.cvtColor(img,cv.COLOR_BGR2GRAY)

(thresh, binary) = cv.threshold(img1, 127,255,cv.THRESH_BINARY)

cv.imwrite("resources/gray_img.png",img1)
cv.imwrite("resources/black_white.png",binary)
```

Out[ ]:

```
True
```

06-reading a video

In [ ]:

```
## reading a video
import cv2 as cv
cap=cv.VideoCapture("resources/myvideo.mp4")

#indicati=or to show video is working
if(cap.isOpened()==False):
    print("Error in video")
#reading and playing
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == True:
        cv.imshow("video",frame)
        if cv.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
cv.destroyAllWindows()
```

07-converting video to gray or black and white

In [ ]:

```
#converting video to gray or black and white

## reading a video
import cv2 as cv
cap=cv.VideoCapture("resources/myvideo.mp4")
```

```
while(True):
    ret, frame = cap.read()
    grayframe = cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
    #(thresh, binary) = cv.threshold(grayframe, 127,255,cv.THRESH_BINARY) # black and w
      # to show in player
    if ret == True:
        cv.imshow("video",grayframe)
       #cv.imshow("black and white",binary) #for black and white display
        # to quit with q key
        if cv.waitKey(1) & 0xFF==ord('q'):
            break
    else:
        break


cap.release()
cv.destroyAllWindows()
```

08-converting video to gray or black and white And Saving

In [ ]:

```
#converting video to gray or black and white And Saving

## reading a video
import cv2 as cv
from matplotlib.colors import is_color_like

cap=cv.VideoCapture("resources/myvideo.mp4")

#writing format, codec, video writer object and file output
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
out = cv.VideoWriter("resources/OutVideo.avi",cv.VideoWriter_fourcc('M',"j",'P','G'),10
while(True):
    ret, frame = cap.read()
    grayframe = cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
      # to show in player
    if ret ==True:
        out.write(grayframe)
        cv.imshow("video",grayframe)
        #cv.imshow("black and white",binary) #for black and white display
        # to quit with q key
        if cv.waitKey(1) & 0xFF==ord('q'):
            break
    else:
        break


cap.release()
out.release()
cv.destroyAllWindows()
```

09-How to connect web cam to python

In [ ]:

```
# How to connect web cam to python
```

```
#step 1 import libraries
import cv2 as cv
import numpy as np

#step 2 Read the frames from camera
cap = cv.VideoCapture(0) # zero means web cam number one

# read cam until the end
# step 3: dsiplay frame by frame

while(cap.isOpened()):
    #read frame by frame
    ret, frame = cap.read()
    if ret == True:
        #to display frames
        cv.imshow("mycam",frame)
        #to quit the loop with q
        if cv.waitKey(1) & 0xFF==ord('q'):
            break
    else:
      break
cap.release()
cv.destroyAllWindows()
```

10-convert into different shades of gray and black&white

In [ ]:

```
# convert into different shades of gray and black&white
import cv2 as cv
import numpy as np

cap = cv.VideoCapture(0)

while(True):
    ret, frame = cap.read()
    gray_frame = cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
    (thresh, binary) = cv.threshold(gray_frame, 127,255,cv.THRESH_BINARY) # black and w
    cv.imshow("original cam",frame)
    cv.imshow("gray cam",gray_frame)
    cv.imshow("B&W",binary)
    if cv.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

11-Writing or saving Videos from Camera

In [ ]:

```
# Writing or saving Videos from Camera
import cv2 as cv
import numpy as np

cap=cv.VideoCapture(0)

#writing format, codec, video writer object and file output
frame_width = int(cap.get(3))
```

```
frame_height = int(cap.get(4))
out = cv.VideoWriter("resources/Cam_Video.avi",cv.VideoWriter_fourcc('M',"j",'P','G'),5
while(True):
    ret, frame = cap.read()
    gray_frame = cv.cvtColor(frame,cv.COLOR_BGR2GRAY) #if convert into gray frame than
      # to show in player
    if ret ==True:
        out.write(gray_frame)
        cv.imshow("video",gray_frame)
        # to quit with q key
        if cv.waitKey(1) & 0xFF==ord('q'):
            break
    else:
        break


cap.release()
out.release()
cv.destroyAllWindows()
```

12-Setting of camera or video

In [ ]:

```
# Setting of camera or video

import cv2 as cv
import numpy as np
cap = cv.VideoCapture(0)

cap.set(10,100) # 10 is the key to set t6he brightness
cap.set(3,300)  # 3 is key of width
cap.set(4,100)  # 4 is key of height
while(True):
    ret, frame = cap.read()
    if ret == True:
        cv.imshow("frame",frame)
        if cv.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
cv.destroyAllWindows()
```

13-Basic functions or Manipulation in opencv

In [ ]:

```
# Basic functions or Manipulation in opencv
import cv2 as cv
import numpy as np
img = cv.imread("resources/mypic.jpeg")
print("the size of my image = ",img.shape)

# resize function
resized_img = cv.resize(img,(150,250))

# Gray_Image
```

```
gray_img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

# Black & White image
(thresh, binary) = cv.threshold(gray_img, 127,255,cv.THRESH_BINARY)
# Blurred Image
blur_img = cv.GaussianBlur(img,(7,7), 0) # (7,7) is image intensity always odd you can

# Edge detection
edge_img = cv.Canny(img, 48,48) # 48,48 threshould1 and threshould 2

# thikness of lines
mat_kernal = np.ones((3,3),np.uint8)
dilated_img = cv.dilate(edge_img, (mat_kernal),iterations=1) #dilated means change the

# make thiner outline
ero_img = cv.erode(dilated_img,(mat_kernal),iterations=1)

# for croping image we will use numpy not opencv
croped_img = img[0:500,0:350] #0:500 height & 0:350 width



cv.imshow("original",img)
# cv.imshow("Resized",resized_img)
# cv.imshow("Gray Image",gray_img)
# cv.imshow("B&G",binary)
# cv.imshow("Blured Img",blur_img)
# cv.imshow("Edge",edge_img)
# cv.imshow("dilated img",dilated_img)
# cv.imshow("Erosion",ero_img)
cv.imshow("Crop Image",croped_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

```
the size of my image =  (717, 720, 3)
```

14-how to draw lines and shapes in python

In [ ]:

```
# how to draw lines and shapes in python

import cv2 as cv
import numpy as np


#draw a canvs 0 is for black
img  = np.zeros((600,600))
img1 = np.ones((600,600))

#print size
print("The size of our canvas is  =",img.shape)


#adding colors to the canvas

colored_img = np.zeros((600,600,3),np.uint8) #color channel addition
colored_img[:] = 255,0,255 #color complete image
```

```
colored_img[150:230,100:207]=255,0,0 #color part of image

#adding line
cv.line(colored_img,(0,0),(colored_img.shape[0],colored_img.shape[1]),(255,0,0),3)

cv.line(colored_img,(100,100),(600,600),(255,0,240),3 )
#adding rectangle
cv.rectangle(colored_img,(50,100),(300,400),(205,240,0),3) #Draw rectangle
cv.rectangle(colored_img,(50,100),(300,400),(205,240,0),cv.FILLED) #Fill Rectangle

#Adding circle
cv.circle(colored_img,(300,300),50,(255,100,0),5)
cv.circle(colored_img,(300,300),50,(255,100,0),cv.FILLED)

# adding text
cv.putText(colored_img,"python opencv",(200,50),cv.FONT_HERSHEY_DUPLEX,1,(255,255,0),2)
#assignment how to break above text into two lines




# cv.imshow("black canvas",img)
# cv.imshow("white camvas",img1)
cv.imshow("colored ",colored_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

```
The size of our canvas is  = (600, 600)
```

15-Resolution of camera

In [ ]:

```
# Resolution of cam
import cv2 as cv
import numpy as np

cap = cv.VideoCapture(0)
#resolution HD(1280x720)

#creating function for hd resolution
def hd_resolution():
    cap.set(3,1280) # width
    cap.set(4,720)  # height

def sd_resolution():
    cap.set(3,640) # width
    cap.set(4,480)  # height

# Assignment how to change and fix the frame rate to 30fps
def fhd_resolution():
    cap.set(3,1920) # width
    cap.set(4,1000)  # height
    cap.set(cv.CAP_PROP_FPS,15)

fhd_resolution()

# hd_resolution()
# sd_resolution()
```

```
while(True):
    ret,frame = cap.read()
    if ret==True:
        cv.imshow("camera",frame)
        if cv.waitKey(1) & 0XFF == ord('q'):
            break
    else:
        break

cap.release()
cv.destroyAllWindows()
```

16-saving HD Recording of cam stream

In [ ]:

```
#saving HD Recording of cam stream

import cv2 as cv
import numpy as np

cap = cv.VideoCapture(0)
#resolution HD(1280x720)

#creating function for hd resolution
def hd_resolution():
    cap.set(3,1280) # width
    cap.set(4,720)  # height

def sd_resolution():
    cap.set(3,640) # width
    cap.set(4,480)  # height

# Assignment how to change and fix the frame rate to 30fps
def fhd_resolution():
    cap.set(3,1920) # width
    cap.set(4,1000)  # height
    cap.set(cv.CAP_PROP_FPS,15)

fhd_resolution()

# hd_resolution()
# sd_resolution()

frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
out = cv.VideoWriter("resources/Cam2_Video.avi",cv.VideoWriter_fourcc('M',"j",'P','G'),
while(True):
    ret, frame = cap.read()
      # to show in player
    if ret ==True:
        out.write(frame)
        cv.imshow("video",frame)
        # to quit with q key
        if cv.waitKey(1) & 0xFF==ord('q'):
            break
    else:
```

```
        break

cap.release()
out.release()
cv.destroyAllWindows()
```

17-Joining two images

In [ ]:

```
# Joining two images

import cv2 as cv
import numpy as np

# read mage
img = cv.imread("resources/mypic.jpeg")
img1 = cv.resize(img,(100,150))
#stacking same image

# 1- Horizontal stack
hor_img = np.hstack((img1,img1))

# 2-Vertical stack
ver_img = np.vstack((img1,img1))


cv.imshow("Horizontal",hor_img)
cv.imshow("Vertical",ver_img)
cv.waitKey(0)
cv.destroyAllWindows()

# You can only stack images with same shapes (width,height,color channel)
```

18-how to change the perspective of an image

In [ ]:

```
# how to change the perspective of an image
import cv2 as cv
import numpy as np

img = cv.imread("resources/warp.jpg")
img = cv.resize(img, (500,800))
print(img.shape)

#defining points

point1 = np.float32([[102,106],[400,94],[20,549],[468,555]])  # first argument for heigh
width = 500
height = 800


point2 = np.float32([[0,0],[width,0],[0,height],[width,height]])
matrix = cv.getPerspectiveTransform(point1,point2)
out_img = cv.warpPerspective(img,matrix,(width,height))

# stack both images
hor_img = np.hstack((img,out_img))
```

```
# cv.imshow("original",img)
# cv.imshow("transformed",out_img)
cv.imshow("transformed",hor_img)
cv.waitKey(0)
cv.destroyAllWindows()
```

```
(800, 500, 3)
```

19-coordinates of an image or video

In [ ]:

```
# coordinates of an image or video
# bgr color codes from an image

# step-1 Import librraries
import cv2 as cv
import numpy as np

# step-2 define function
def find_coord(event,x,y,flags,params):
    if event == cv.EVENT_LBUTTONDOWN:
       #left mouse click
        print(x, '',y)
      #how to define or print on the same image or window
        font = cv.FONT_HERSHEY_PLAIN
        cv.putText(img,str(x) + ','+ str(y),(x,y),font,1,(255,0,0),thickness=2)
      #show the text on image and image itself
        cv.imshow("image",img)
    #for color finding
    if event == cv.EVENT_RBUTTONDOWN:
        print(x,'',y)
        font = cv.FONT_HERSHEY_PLAIN
        b= img[y,x,0]
        g=img[y,x,1]
        r=img[y,x,2]

        cv.putText(img, str(b)+',' + str(g) +',' + str(r),(x,y),font,1,(255,0,255),thic
        cv.imshow("image",img)

# step-3 final function to read and display
if __name__=="__main__":
    #reading an image
    img = cv.imread("resources/warp.jpg",1)
    #resize image
    img = cv.resize(img,(500,800))
    #display the image
    cv.imshow("image",img)
    #seting call back function
    cv.setMouseCallback("image",find_coord)
    cv.waitKey(0)
    cv.destroyAllWindows()
```

20-spilit your videos into frames

In [ ]:

```
# spilit your videos into frames
import cv2 as cv
cap = cv.VideoCapture("resources/myvideo.mp4")
```

```
frameNumber=0
while(True):
    ret,frame =cap.read()
    if ret:
        cv.imwrite(f"resources/frames/frames_{frameNumber}.jpg",frame)
    else:
        break
    frameNumber = frameNumber+1
cap.realease()
```

21-how to detect specific colors inside python

In [ ]:

```
# how to detect specific colors inside python
import cv2 as cv
from matplotlib.pyplot import hsv
import numpy as np

# img = cv.imread("resources/mypic.jpeg")

#convert into hsv (Hue, Saturation, Value)
# hsv_img = cv.cvtColor(img, cv.COLOR_BGR2HSV)

# sliders
def slider():
    pass

path = "resources/mypic.jpeg"

cv.namedWindow("Bars")
cv.resizeWindow("Bars",700,300)

cv.createTrackbar("Hue Min","Bars",0,179,slider)
cv.createTrackbar("Hue Max","Bars",179,179,slider)

cv.createTrackbar("Sat Min","Bars",0,255,slider)
cv.createTrackbar("Sat Max","Bars",255,255,slider)

cv.createTrackbar("val Min","Bars",0,255,slider)
cv.createTrackbar("val Max","Bars",255,255,slider)

# img = cv.imread(path)
# hsv_img = cv.cvtColor(img, cv.COLOR_BGR2HSV)

# hue_min = cv.getTrackbarPos("Hue Min","Bars")
# print(hue_min)
while(True):
    img = cv.imread(path)
    hsv_img = cv.cvtColor(img, cv.COLOR_BGR2HSV)
    hue_min = cv.getTrackbarPos("Hue Min","Bars")
    hue_max = cv.getTrackbarPos("Hue Max","Bars")
    Sat_min = cv.getTrackbarPos("Sat Min","Bars")
    Sat_max = cv.getTrackbarPos("Sat Max","Bars")
    Val_min = cv.getTrackbarPos("val Min","Bars")
    Val_max = cv.getTrackbarPos("val Max","Bars")
    print(hue_min,hue_max,Sat_min,Sat_max,Val_min,Val_max)
```

```
    # to see these changes inside image
    lower = np.array([hue_min,Sat_min,Val_min])
    upper = np.array([hue_max,Sat_max,Val_max])

    mask_img = cv.inRange(hsv_img,lower,upper)
    out_img = cv.bitwise_and(img,img,mask=mask_img)


    cv.imshow("original",img)
    cv.imshow("hsv",hsv_img)
    cv.imshow("mask",mask_img)
    cv.imshow("final output",out_img)
    if cv.waitKey(1) & 0xFF == ord('q'):
        break
cv.destroyAllWindows()
```

In [ ]: