final_notebook

February 15, 2022

1 Participant

Title= “Mr”

Name= “Syed Saad ul Hassan”

email = “saadulhassanis@gmail.com”

whatsapp = “+491729024676”

2 Python Basics

2.1 Basic Operations and Logic

[ ]:
```
# my_first prog in python
print(2+3)
print("Hello world")
print("Learning python with Ammar")
```

```
5
Hello world
Learning python with Ammar
```

02- Operators

[ ]:
```
print(2+34)
print(13%2)
print(6/2)
print(6//2)
x = 2 ** 3
print("the power is ",2**3)
```

```
36
1
3.0
3
the power is  8
```

03- Strings

[ ]:
```
print('Test  for single quote')
print("Test for double quote")
print('''Test for triple quote''')
print("What's up   ?")
```

```
Test  for single quote
Test for double quote
Test for triple quote
What's up   ?
```

04- Comments

[ ]:
```
# print(2+3) print  operator functions
# print("Hello world")
print("Learning python with Ammar") # Comment with Ctrl + /
```

```
Learning python with Ammar
```

05- Variables

[ ]:
```
#variables: Object containing specific values
x = 5

print("x is ",x)
x=20
print("The value of updated x is now",x)

y2 = ("We are learning Python with Ammar")
print(y2)

# types/class of variables
type(x)
print(type(x))

print(type(y2))

# Rules to assign a variable
# 1. the variable should contain only from letters,numbers or underscores
# 2. Donot start with number like 2y. It is wrong. you can use y2
# 3. Donot use spaces
# 4. Donot use keywords for eg Python k keywords used in functions for eg (␣
 ↪Break, mean, median, test)
# 5. variable should be short and descriptive
# 6. Case sensitivity (try using lower case letters)

fruit_basket = "Mangoes", "Oranges"
fruit_basket2 = "Mangoes, Oranges"

print(fruit_basket)
```

```
print(fruit_basket2)
del x
```

```
x is  5
The value of updated x is now 20
We are learning Python with Ammar
<class 'int'>
<class 'str'>
('Mangoes', 'Oranges')
Mangoes, Oranges
```

06- Input variables

[ ]:
```
# fruit_basket="Mangoes"
# print(fruit_basket)

# #input  fn
# fruit_basket = input("Which is your favourite fruit?   ")
# print(fruit_basket)

# # Input function of second stage

# name= input("What is your name ? ")
# greetings = "Hello"

# print(greetings,  name)

# Another way of stage 2 input function
# name= input("What is your name ? ")
# print("Hello!",  name)

# Input function of third stage

name = input ("What is your name? ")
age = input ("Wie alt sind Sie? ")
greetings = "Hello!"

print(greetings,name,age," \n Oh So you are 28 , You are still young",)
```

```
Hello!
 Oh So you are 28 , You are still young
```

07- Conditional Logic

[ ]:
```
# equal to ==
# not equal to  !=
# less than     <
# greater than  >
# less than and equal to <=
```

```
# greater than and equal to >=

# logic oprators are boolean like TRUE/FALSE, YES/NO

x = 4 != 4
print(x)
print(3>4)
print(3<=3)

#applications of logical operator includes
hammad_age = 4
age_at_school=5
print("The eligibility says",hammad_age==age_at_school)

#input fns and logical operator
age_at_school=5
print(type(age_at_school))
hammad_age= input("How old is Hammad")
hammad_age = int (hammad_age)
print(type(hammad_age))
print("The eligibility now says because of type conversion to int is␣
 ↪",hammad_age==age_at_school)
```

```
False
False
True
The eligibility says False
<class 'int'>
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_7008/1265173658.py in <module>
     22 print(type(age_at_school))
     23 hammad_age= input("How old is Hammad")
---> 24 hammad_age = int (hammad_age)
     25 print(type(hammad_age))
     26 print("The eligibility now says because of type conversion to int is␣
 ↪",hammad_age==age_at_school)

ValueError: invalid literal for int() with base 10: ''
```

08- Type Conversion

[ ]:
```
x = 10 # int
y=10.2 # float
z = "Hello" # string
```

```
x = x*y

print(type(x))

x = 33
str(x)
print(x==3)

name= input("Enter your name \n")
name = int(name) # no compile cuz int me tune str dedia hay
print("My name is  ",name)
```

```
<class 'float'>
False
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_7008/3074063657.py in <module>
     12
     13 name= input("Enter your name \n")
---> 14 name = int(name) # no compile cuz int me tune str dedia hay
     15 print("My name is  ",name)

ValueError: invalid literal for int() with base 10: 'Saad'
```

09- if/else

[ ]:
```
x = 10 # int
y=10.2 # float
z = "Hello" # string

x = x*y

print(type(x))

x = 33
str(x)
print(x==3)

name= input("Enter your name \n")
name = int(name) # no compile cuz int me tune str dedia hay
print("My name is  ",name)
```

```
<class 'float'>
False
Enter your name
2
My name is   2
```

10- Functions

[ ]:
```
print("Wie alt sind sie")
print("We are learnin with Ammar")
print("We are learnin with Ammar")
print("We are learnin with Ammar")

# #functions definition
# #1
# def print_codanics():
#     print("We are learning with Ammar")
#     print("We are learning with Ammar")
#     print("We are learning with Ammar")
#     print("We are learning with Ammar")

# print_codanics()

#functions definition
#2
# def print_codanics():
#     text = "We are learning with Ammar"
#     print(text)
#     print(text)
#     print(text)

# print_codanics()

#function def
#3
def print_codanics(text):
    print(text)
    print(text)
    print(text)

print_codanics("We are learning Python with ammar 3rd method")

#function def elif
#4

def school_calculator(age,text):
    if age == 5:
        print("Hammad can join the school")

    elif age >5:
        print("Hammad should go to bigger school")
    else:
        print("hammad is still a baby boy")
```

```
school_calculator(8,"Hammad")

#function def of future (return scene)
#5

def future_age(age):
    new_age = age +20
    return new_age

# z = future_age(18)
print(future_age(18))
```

```
Wie alt sind sie
We are learnin with Ammar
We are learnin with Ammar
We are learnin with Ammar
We are learning Python with ammar 3rd method
We are learning Python with ammar 3rd method
We are learning Python with ammar 3rd method
Hammad should go to bigger school
38
```

11- Loops

[ ]:
```
# While loops
x =0
while (x<5):
     print(x)
     x=x+2

#for loops
for x in range(4,10):
    x=x+2
    print(x)

# arrays
days = ["Mon","Tues","Wed","Thu","Fri","Sat","Sun"]

for d in days:
    if (d=="Wed"): break
    print(d)
    print(days)
```

```
0
2
4
```

```
6
7
8
9
10
11
Mon
['Mon', 'Tues', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
Tues
['Mon', 'Tues', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
```

12- Import Libraries

[ ]:
```
# if u want to print the value of pi

import math

x= math.pi
x=int(x)
print("The value of pi is",x)

x = [3,4,5,6,7,8,8,9]
import statistics
print("The mean of the no is ",statistics.mean(x)," \n Clear!!!")
```

```
The value of pi is 3
The mean of the no is  6.25
 Clear!!!
```

13- Troubleshooting

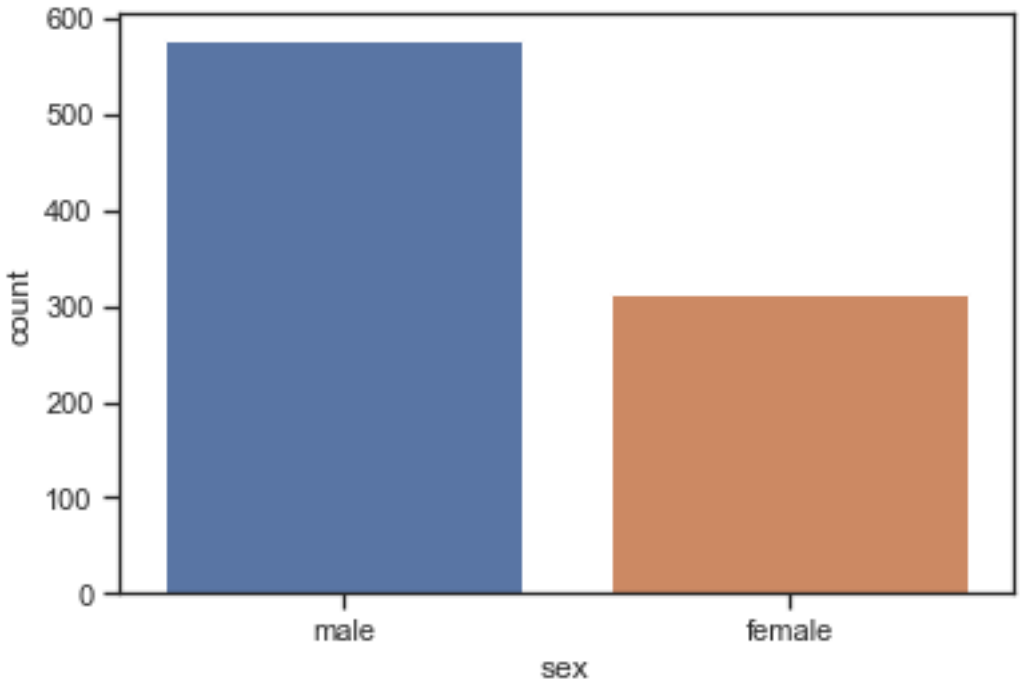
[ ]:
```
print(25/0)
```

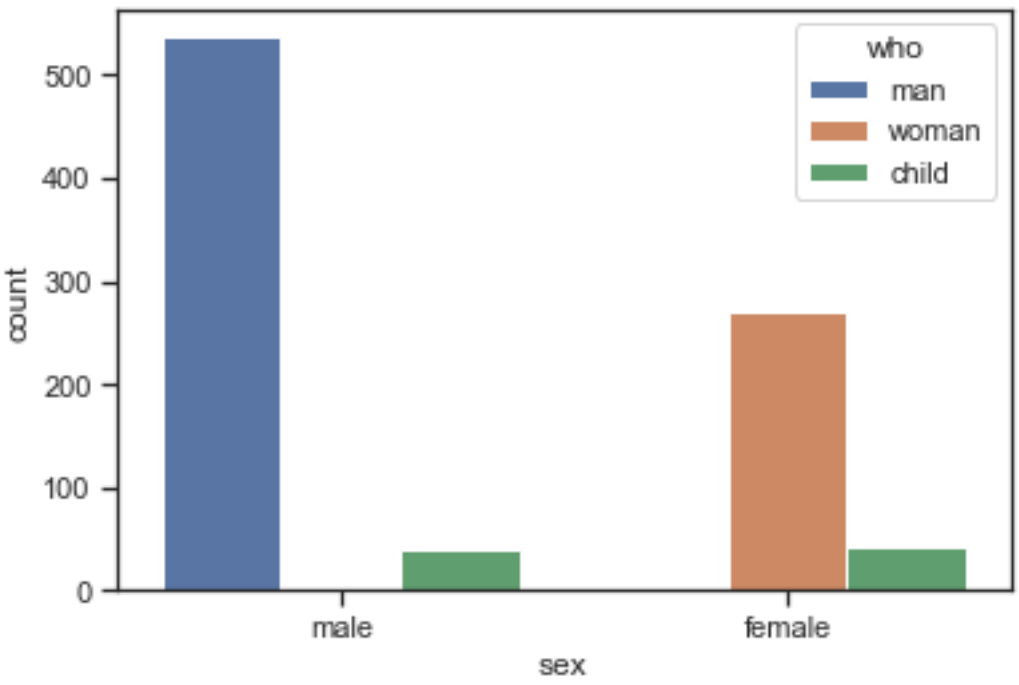
```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-14-a08611be9ae0> in <module>
----> 1 print(25/0)

ZeroDivisionError: division by zero
```

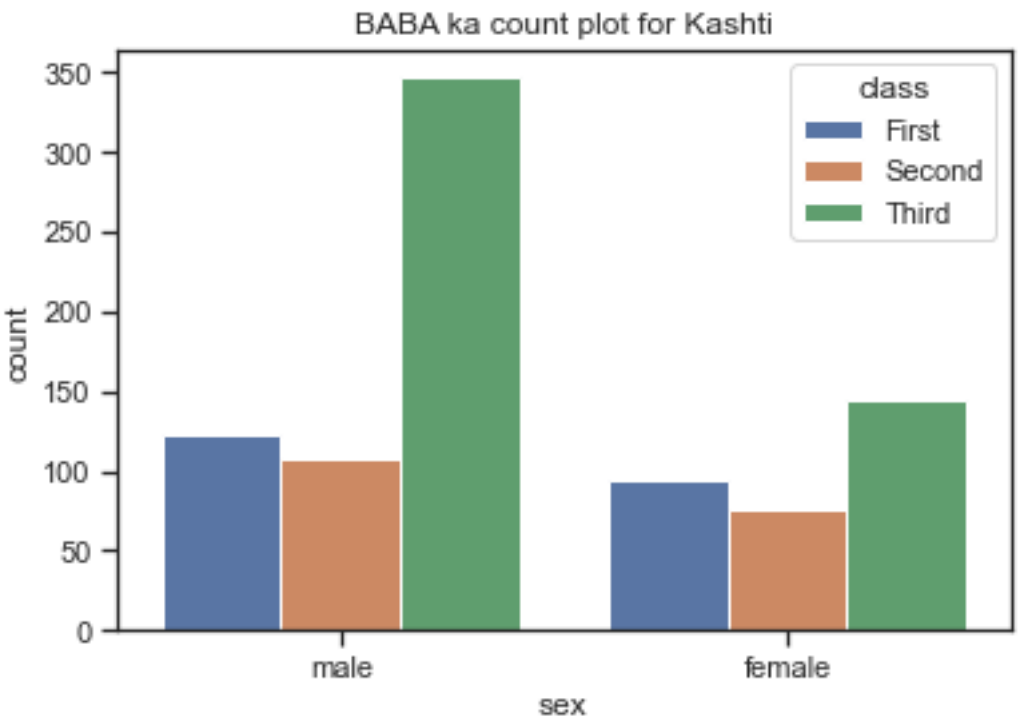
14- Data VIZ

[ ]:
```
# Steps involved in Data Viz
# Step1 Import libraries
import seaborn as sns
import matplotlib.pyplot as plt

# Step2 Set a theme
```

```
sns.set_theme(style="ticks",color_codes=True)

# Step3 Import Dataset ( You can also import own data)
kashti = sns.load_dataset("titanic")
#print(kashti)  # pura data ajega

# Step4 Plot basic graph with 1 variable (COunt plot me y axis par count auto␣
 ↪ata hay)
p=sns.countplot(x='sex',data=kashti)
plt.show()

# Step5 plot Basic  graph with 2 variable (hue means color)
# jese hue me class agae male female ki ticket k hisab se mtlb x ko tor rahaa␣
 ↪hay hue apka (LAzmi smjh)
p=sns.countplot(x='sex',hue='who',data=kashti)
plt.show()

# Step6 plot Basic  graph with 2 variable (count plot) with Titles
p=sns.countplot(x='sex',hue='class',data=kashti)
p.set_title("BABA ka count plot for Kashti")
plt.show()
```




who
man
woman
child
500
400
300
200
100
0
count
male
female
sex


BABA ka count plot for Kashti
class
First
Second
Third
350
300
250
200
150
100
50
0
count
male
female
sex

2.1.1 Assignment : BMI Calculator

```
[ ]: def BMI_calc(weightt,height,name):
        BMI = float(weightt)/float(height)**2
        return BMI
x=input("Please enter your weight")
#float(x)
w=input("Please enter your height in m")
#float(w)
y=input("Please enter your name \n ")
type(y)
y
z=print("The BMI of",y,"is",BMI_calc(x,w,"y"))
```

```
The BMI of Syed is 24.535123966942148
```

BMI Calculator with Dr Ammar Method

```
[ ]: height=input("Please enter your height in metres")
```

```
[ ]: height = float(height)
```

```
[ ]: weight=input("Please enter your weight in kilogram \n ")
```

```
[ ]: weight=float(weight)
```

```
[ ]: name=input("Please tell me your name")
```

```
[ ]: BMI = weight/height**2
     BMI
```

```
[ ]: 28.08626033057851
```

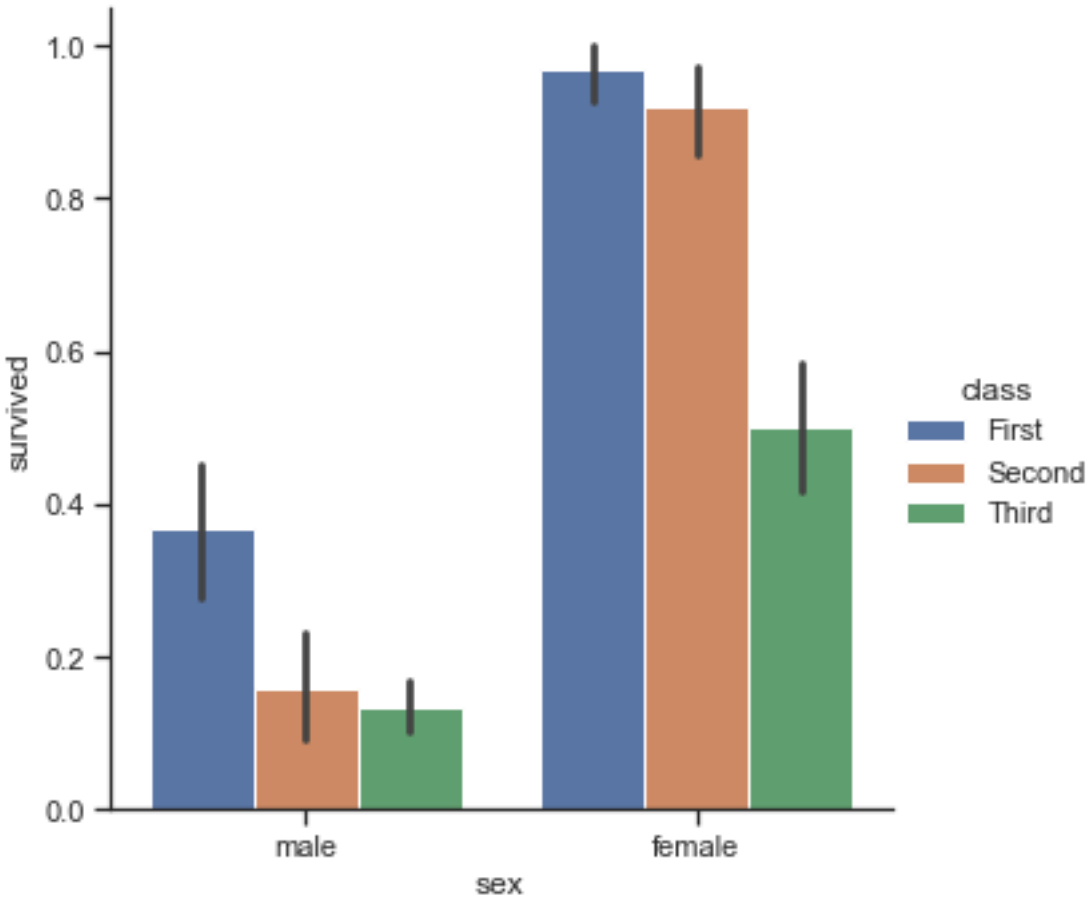
```
[ ]: print("The BMI of",name,"is",BMI)
```

```
The BMI of Hassan is 28.08626033057851
```

2.2 Plots Basics

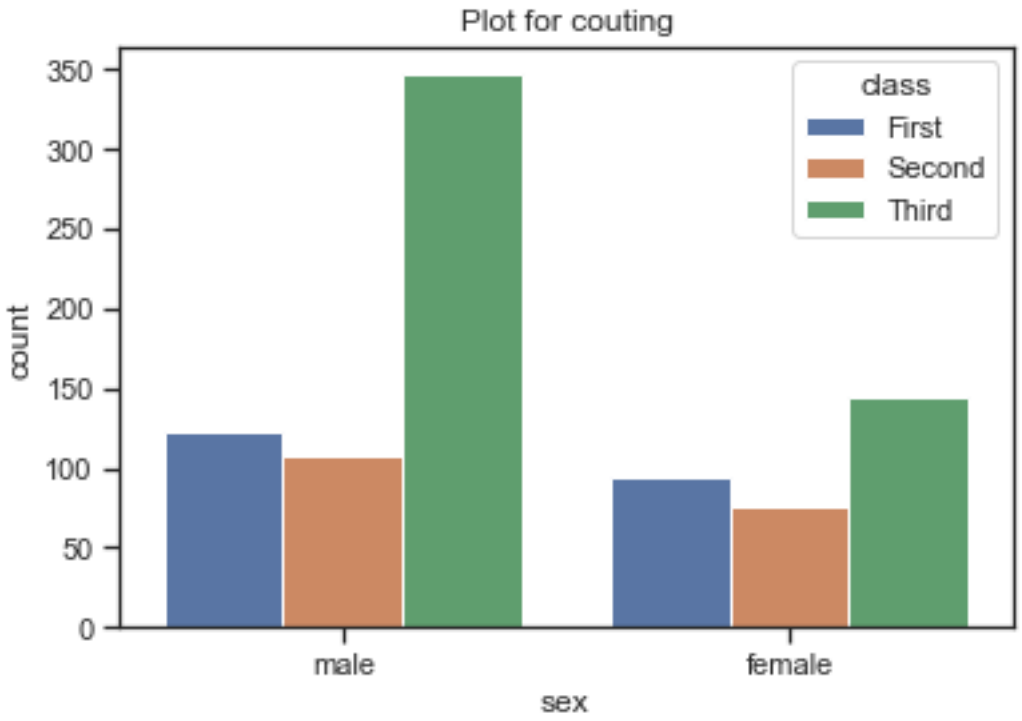
Catplot

```
[ ]: 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()
```



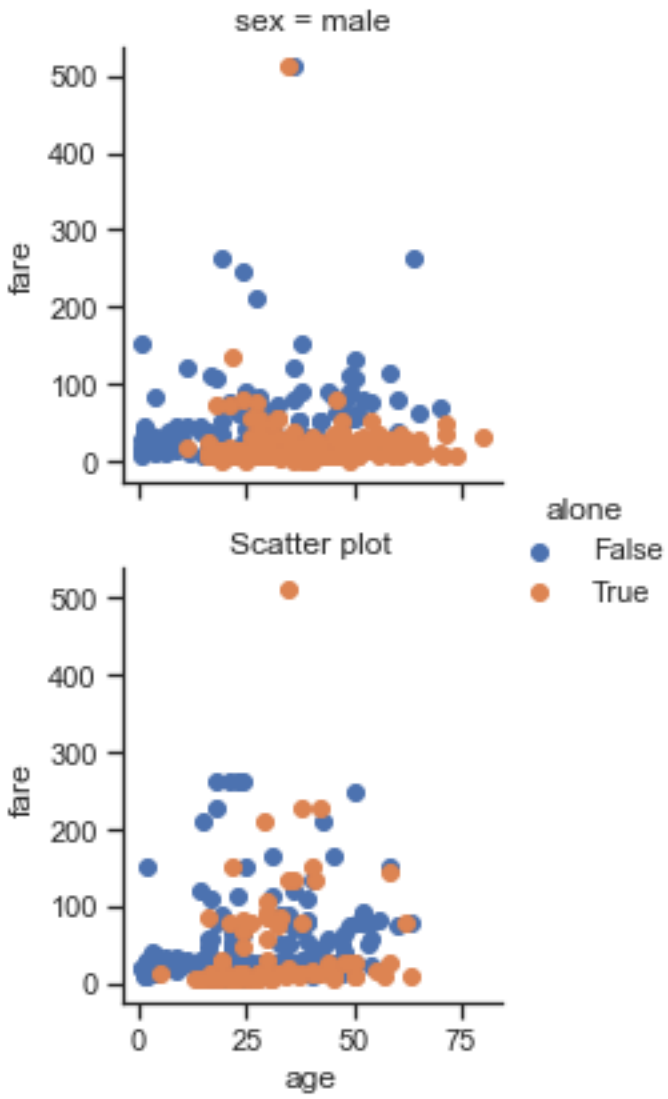
Count plot

[ ]:
```
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',hue='class',data=titanic)
p1.set_title("Plot for couting")
plt.show()
```



Scatter plot

[ ]:
```
# scatter plot
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())
#g.set_title("Plot for couting")
plt.title("Scatter plot")
plt.show()
```



2.3 Indexing

[ ]:
```
# make  a string

a = "Samosa Pakora"
a
```

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

[ ]:
```
a[0]
a[1]
```

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

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

[ ]: 'P'

[ ]:
```
#length of indice
len(a)
```

[ ]: 13

[ ]:
```
a[0:6]
```

[ ]: 'Samosa'

[ ]:
```
a[-1:-6]
```

[ ]: ''

[ ]:
```
# reverse array
a[-1]
```

[ ]: 'a'

[ ]:
```
# reverse me array print
a[-6:13]
```

[ ]: 'Pakora'

[ ]:
```
food = "Biryani"
food
```

[ ]: 'Biryani'

2.3.1 String Indexing

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

[ ]: 'Biryani'

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

[ ]: 'BIRYANI'

[ ]:
```
food.replace("i","Sh")
```

[ ]: 'BShryanSh'

[ ]:
```
#Counting specific alphabet in a string
name = "Baba Ammar with Doctor Ammar"
name
```

[ ]:
```
'Baba Ammar with Doctor Ammar'
```

[ ]:
```
#name.count("baba")
name.count("a")
```

[ ]:
```
4
```

2.3.2 Assignment: How to find Index Number in string

[ ]:
```
name = "Baba Ammar with Doctor Ammar"
name.find("t")
```

[ ]:
```
13
```

Split String

[ ]:
```
# how to split a string
food = " I love samosa, pakora, raita, biryani and karahi"
food.split(",")
```

[ ]:
```
[' I love samosa', ' pakora', ' raita', ' biryani and karahi']
```

2.4 Basic Data structures in Python

1- Tuples

a. Ordered collection of elements
b. enclosed in round braces (). Can store different type of data (Rust scene)
c. Once stored elements cannot be changed. (Immutable)

2- List

a. Ordered collection of elements
b. stored in square braces
c. Mutable

3- Dictionaries

a. Unordered Collection of elements
b. Key and Value (like Hashmap of Rust)
c. Curly braces or bracket {}
d. Mutable / changeable
e. duplicates allowed

4- Set

a. Unordered/ unindexed
b. Used with curly braces
c. No duplicates allowed

2.5 Tuples

[ ]:
```
tup1 = (1,"python",True,2.5)
tup1.count(2.5)
```

[ ]:
```
1
```

[ ]:
```
tup1.index(2.5)
```

[ ]:
```
3
```

[ ]:
```
#type of  a tuple
type(tup1)
```

[ ]:
```
tuple
```

2.5.1 Indexing in Tuple

[ ]:
```
tup1[1] #python
```

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

[ ]:
```
tup1[0:6]
```

[ ]:
```
(1, 'python', True, 2.5)
```

[ ]:
```
## last element will not print because it is exclusive ##
tup1[0:3]
```

[ ]:
```
(1, 'python', True)
```

[ ]:
```
#length of tuple
len(tup1)
```

[ ]:
```
4
```

[ ]:
```
tup2 = (2,"babaAmmar",3.5,False)
tup2
```

[ ]:
```
(2, 'babaAmmar', 3.5, False)
```

[ ]:
```
tup1 + tup2
```

[ ]:
```
(1, 'python', True, 2.5, 2, 'babaAmmar', 3.5, False)
```

[ ]:
```
tup1*3+tup2
```

[ ]:
```
(1,
 'python',
```

```
True,
2.5,
1,
'python',
True,
2.5,
1,
'python',
True,
2.5,
2,
'babaAmmar',
3.5,
False)
```

[ ]:
```
tup3= (20,30,40,59,40)
tup3
```

[ ]:
```
(20, 30, 40, 59, 40)
```

[ ]:
```
min(tup3)
```

[ ]:
```
20
```

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

[ ]:
```
59
```

[ ]:
```
z="34"
tup3 + tup2
#tup3+z # tuple can only be added or concatenated to a tuple
```

[ ]:
```
(20, 30, 40, 59, 40, 2, 'babaAmmar', 3.5, False)
```

2.6 List

[ ]:
```
list1 = [2,"BabaAmmar",False]
list1
```

[ ]:
```
[2, 'BabaAmmar', False]
```

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

[ ]:
```
list
```

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

[ ]:
```
3
```

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

[ ]:
```
'BabaAmmar'
```

[ ]:
```
#list1[3]
```

[ ]:
```
list2=[3,5,"Ammar","Codanics",478,53.2,False]
list2
```

[ ]:
```
[3, 5, 'Ammar', 'Codanics', 478, 53.2, False]
```

[ ]:
```
list1 + list2 #concatenated
```

[ ]:
```
[2, 'BabaAmmar', False, 3, 5, 'Ammar', 'Codanics', 478, 53.2, False]
```

[ ]:
```
#list1 + 2     # not possible
```

[ ]:
```
list1.reverse() # braces laga bachay
list1
```

[ ]:
```
[False, 'BabaAmmar', 2]
```

[ ]:
```
list1.append("codanics yt channel") # ek tarah se concatenate jitni bar run
list1
```

[ ]:
```
[False,
 'BabaAmmar',
 2,
 'codanics yt channel',
 'codanics yt channel',
 'codanics yt channel',
 'codanics yt channel']
```

[ ]:
```
list1.count(2) # count a element occurance
```

[ ]:
```
1
```

[ ]:
```
list3 = [10,20,30,340,50,60]
list3
```

[ ]:
```
[10, 20, 30, 340, 50, 60]
```

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

[ ]:
```
6
```

[ ]:
```
#sorting a list
list3.sort()
list3
```

[ ]:
```
[10, 20, 30, 50, 60, 340]
```

[ ]:
```
list3 *2 # two times list is printed
```

[ ]:
```
[10, 20, 30, 50, 60, 340, 10, 20, 30, 50, 60, 340]
```

[ ]:
```
list4 = list1 + list2
list4
```

[ ]:
```
[False,
 'BabaAmmar',
 2,
 'codanics yt channel',
 'codanics yt channel',
 'codanics yt channel',
 'codanics yt channel',
 3,
 5,
 'Ammar',
 'Codanics',
 478,
 53.2,
 False]
```

[ ]:
```
list1.remove(2) #element remove karta hay list ka function
list1
```

[ ]:
```
[False,
 'BabaAmmar',
 'codanics yt channel',
 'codanics yt channel',
 'codanics yt channel',
 'codanics yt channel']
```

2.7 Dictionaries

[ ]:
```
# Food and their prices Samsoa key and value 30
food1 = {"Samosa":30,"Pakora":100,"Raita":20,"Salad":50,"Chicken Roll":30}
food1
```

[ ]:
```
{'Samosa': 30, 'Pakora': 100, 'Raita': 20, 'Salad': 50, 'Chicken Roll': 30}
```

[ ]:
```
type(food1)
```

[ ]:
```
dict
```

[ ]:
```
# extract data
```

```
keys = food1.keys()
keys
```

[ ]:
```
dict_keys(['Samosa', 'Pakora', 'Raita', 'Salad', 'Chicken Roll'])
```

[ ]:
```
values= food1.values()
values
```

[ ]:
```
dict_values([30, 100, 20, 50, 30])
```

[ ]:
```
food1.update({"Tikki":3}) # how to add new key value method1
food1
```

[ ]:
```
{'Samosa': 30,
 'Pakora': 100,
 'Raita': 20,
 'Salad': 50,
 'Chicken Roll': 30,
 'Tikki': 3}
```

[ ]:
```
food1["Tikki"]= 10 # Method 2 for updating
food1
```

[ ]:
```
{'Samosa': 30,
 'Pakora': 100,
 'Raita': 20,
 'Salad': 50,
 'Chicken Roll': 30,
 'Tikki': 10}
```

[ ]:
```
# update values
food1.update({"Samosa":20})
food1
```

[ ]:
```
{'Samosa': 20,
 'Pakora': 100,
 'Raita': 20,
 'Salad': 50,
 'Chicken Roll': 30,
 'Tikki': 10}
```

[ ]:
```
food2 = {"Dates":20,"Chocolates":200,"Sewayyan":1000}
food2
```

[ ]:
```
{'Dates': 20, 'Chocolates': 200, 'Sewayyan': 1000}
```

[ ]:
```
# food1 + food2 will give error here correct method
food1.update(food2) # do dict apas me concatenate
```

```
food1
```

```
[ ]: {'Samosa': 20,
      'Pakora': 100,
      'Raita': 20,
      'Salad': 50,
      'Chicken Roll': 30,
      'Tikki': 10,
      'Dates': 20,
      'Chocolates': 200,
      'Sewayyan': 1000}
```

```
[ ]: s1= {1,2.2,5.2,"Ammar","Codanics","Faisalabad",True} # no boolean
     s1
```

```
[ ]: {1, 2.2, 5.2, 'Ammar', 'Codanics', 'Faisalabad'}
```

```
[ ]: s1.add("Ammar") # no duplication
     s1
```

```
[ ]: {1, 2.2, 5.2, 'Ammar', 'Codanics', 'Faisalabad'}
```

```
[ ]: s1.add("Ammar1")
     s1
```

```
[ ]: {1, 2.2, 5.2, 'Ammar', 'Ammar1', 'Codanics', 'Faisalabad'}
```

```
[ ]: s1.difference("1")
```

```
[ ]: {1, 2.2, 5.2, 'Ammar', 'Ammar1', 'Codanics', 'Faisalabad'}
```

```
[ ]: s1.discard("Ammar")
     s1
```

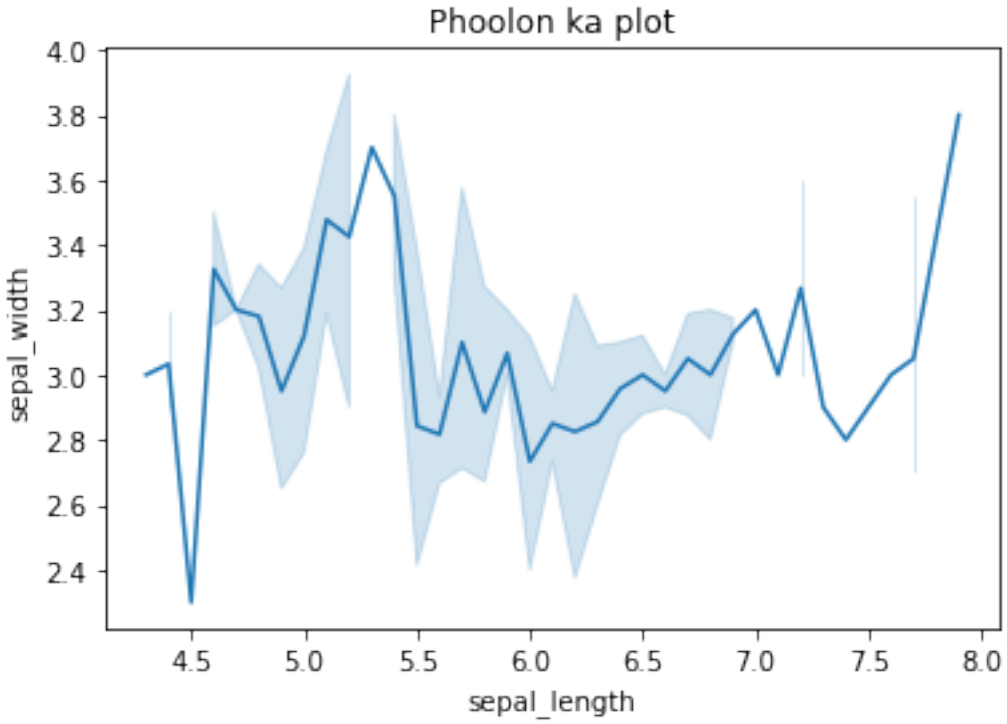
```
[ ]: {1, 2.2, 5.2, 'Ammar1', 'Codanics', 'Faisalabad'}
```

2.7.1 More Plots (Seaborn)

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

     #load data set
     phool = sns.load_dataset("iris")
     phool

     #draw a line plot
```

```
sns.lineplot(x="sepal_length",y="sepal_width",data=phool)
plt.title("Phoolon ka plot")
plt.show
```

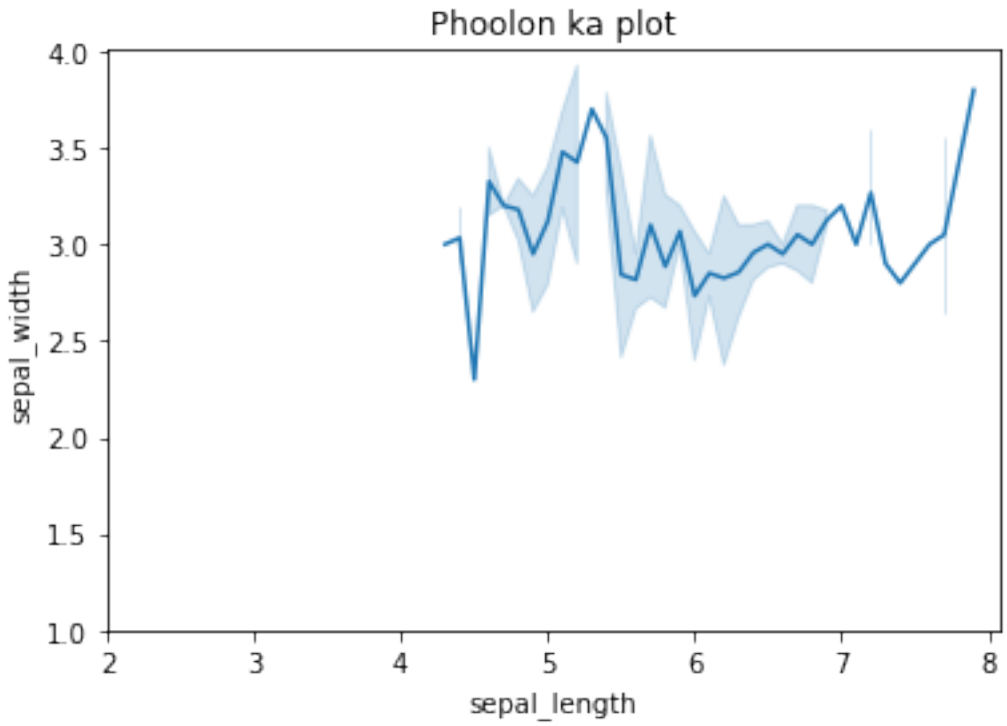
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



Line plot

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

#load data set
phool = sns.load_dataset("iris")
phool

#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
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



Styling - darkgrid - White grid - dark - white - ticks

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

#load data set
phool = sns.load_dataset("iris")
phool

#draw a line plot
sns.lineplot(x="sepal_length",y="sepal_width",data=phool)
plt.title("Phoolon ka plot")

#style
sns.set_style(style=None, rc=None)
sns.set_style("dark")

#limits x and y
plt.xlim(2)
plt.ylim(1)
```

```
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

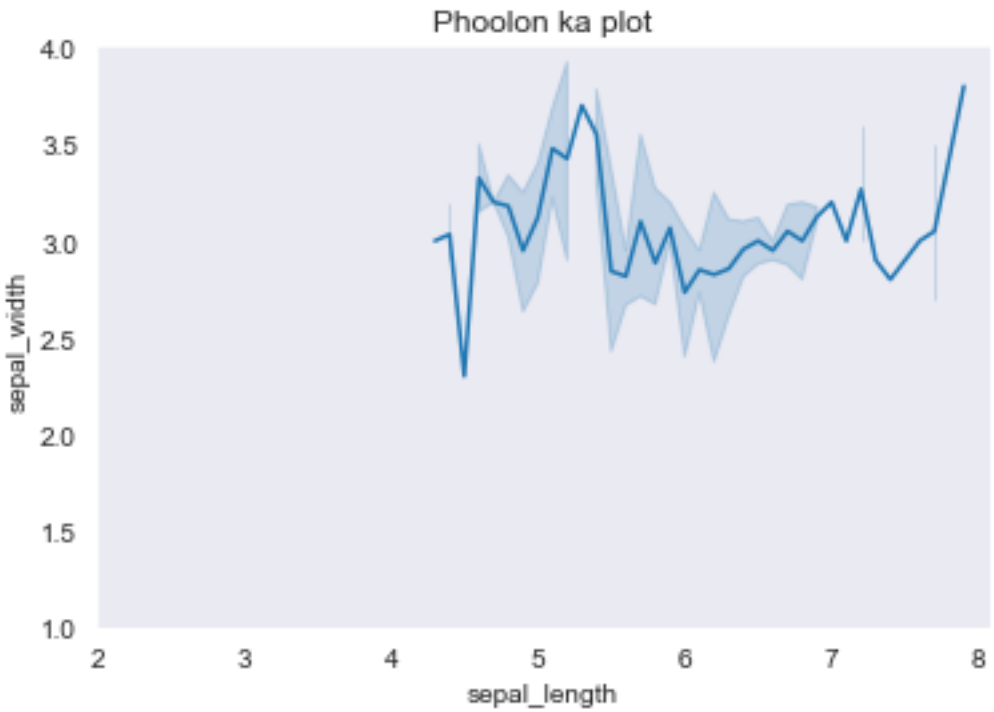
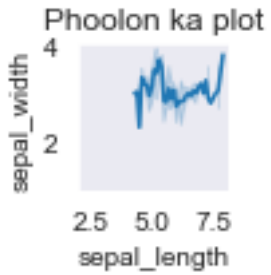


Figure Dimensions / Sizes

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

#load data set
phool = sns.load_dataset("iris")
phool

#figure size
plt.figure(figsize=(1,1))

#draw a line plot
sns.lineplot(x="sepal_length",y="sepal_width",data=phool)
plt.title("Phoolon ka plot")

#style
sns.set_style(style=None, rc=None)
```

```
sns.set_style("dark")

#limits x and y
plt.xlim(2)
plt.ylim(1)




plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



[ ]:
```
phool
```

[ ]:

	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

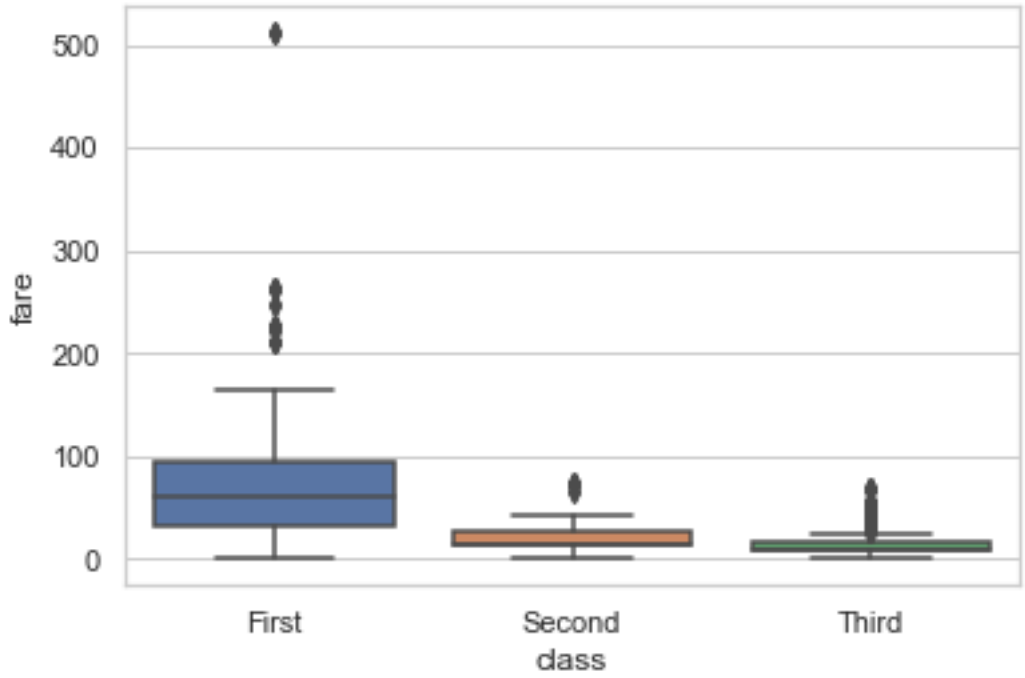
[150 rows x 5 columns]

Box PLot

[ ]:
```
import seaborn as sns

#Canvas Ballonn board
sns.set(style='whitegrid')
```

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

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

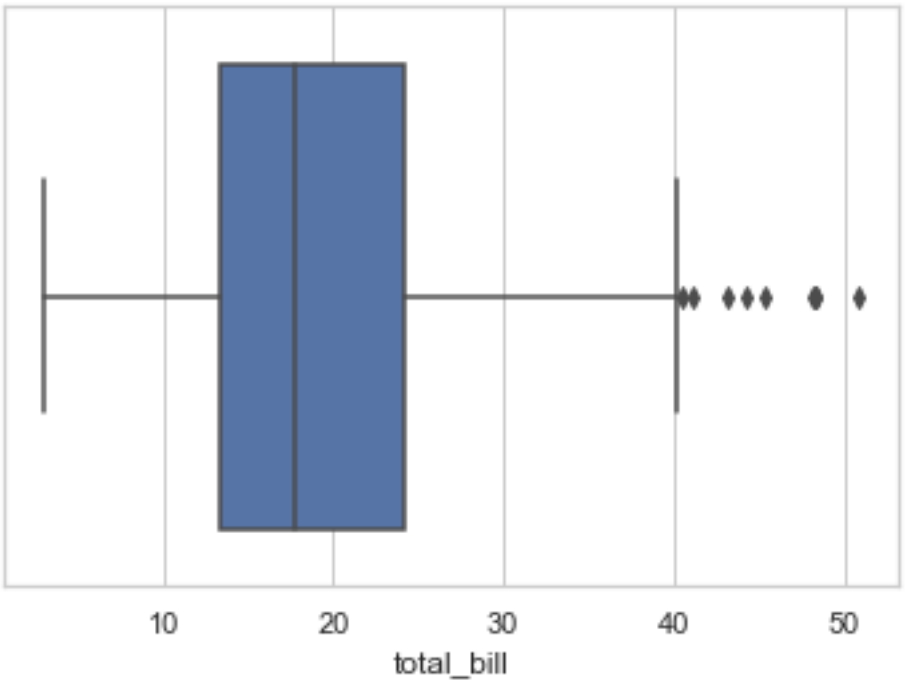
[ ]: <AxesSubplot:xlabel='class', ylabel='fare'>



Box plot on Dinner Data - (describe and indexed plot)

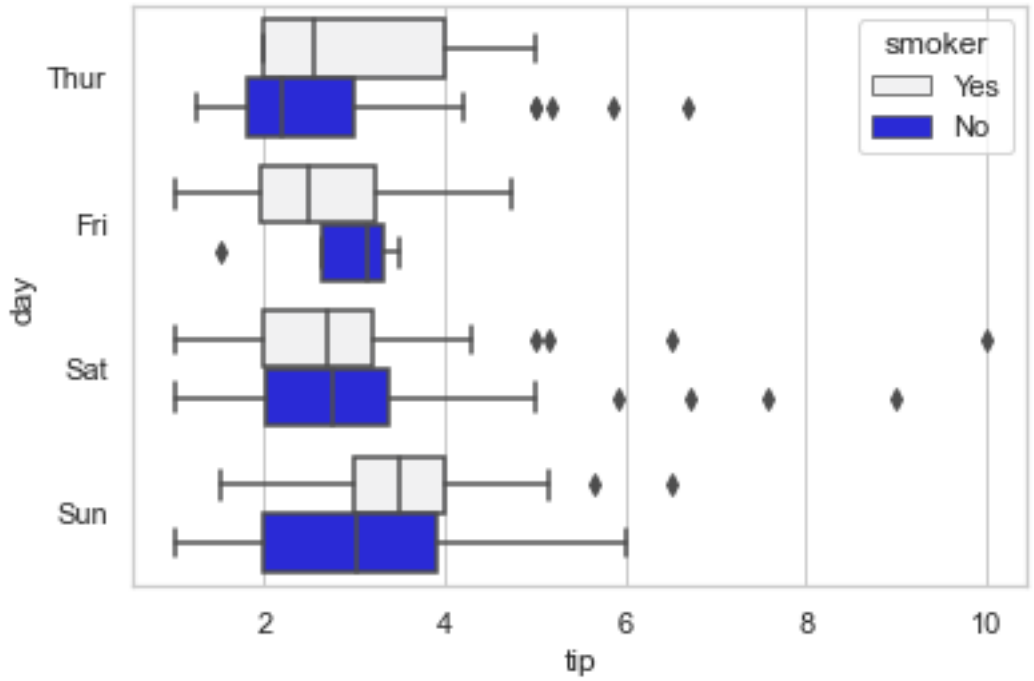
[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


#Canvas Ballonn board
sns.set(style='whitegrid')

tip = sns.load_dataset("tips")
# it is describing all numeric values
tip.describe()

sns.boxplot(x=tip["total_bill"],data=tip,saturation=0.8)
```

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



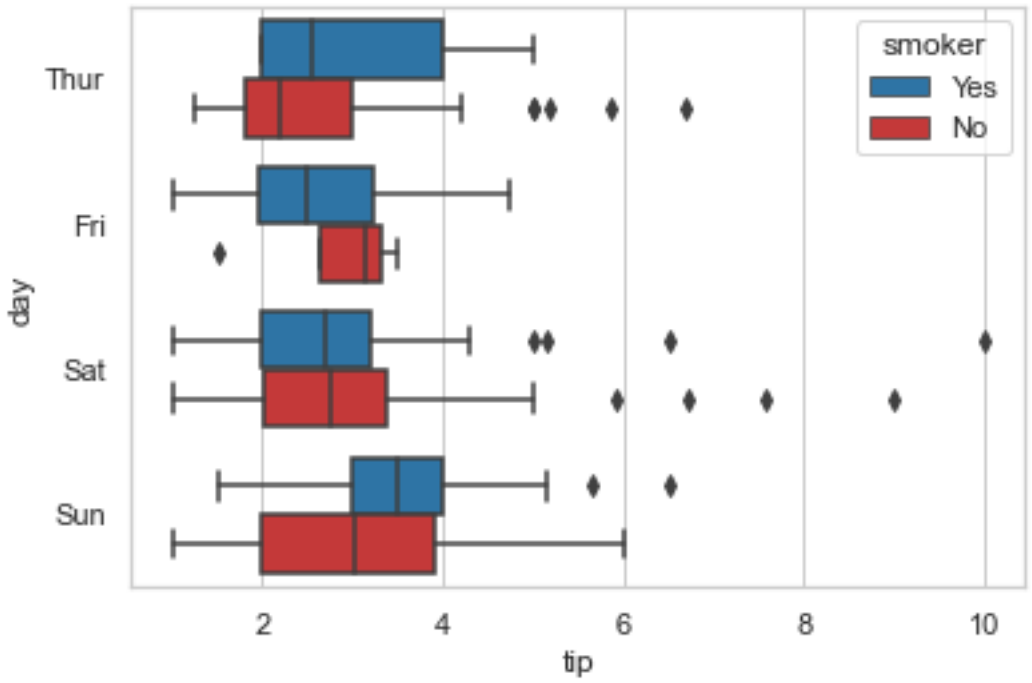
[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


#Canvas Ballonn board
sns.set(style='whitegrid')

tip = sns.load_dataset("tips")
# it is describing all numeric values
tip.describe()

sns.boxplot(x="tip",y="day",hue="smoker",data=tip,saturation=0.
 ↪8,dodge=True,color="#1515eb")


#sns.boxplot(x="tip",y="day",hue="smoker",data=tip,saturation=0.
 ↪8,palette="Set2",dodge=False)
```

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



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


#Canvas Ballonn board
sns.set(style='whitegrid')

tip = sns.load_dataset("tips")
# it is describing all numeric values
tip.describe()
palette = {
    'Yes': 'tab:blue',
    'No': 'tab:red',

}

sns.boxplot(x="tip",y="day",hue="smoker",data=tip,saturation=0.
 ↪8,dodge=True,palette=palette,color="#1515eb")
```

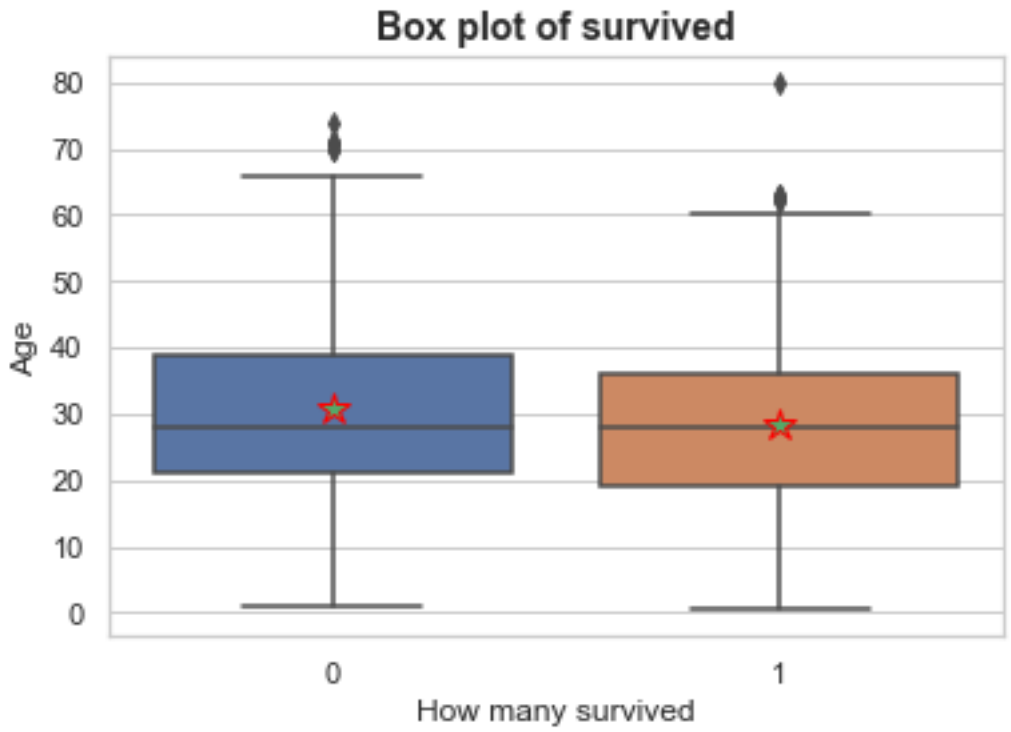
[ ]:
```
<AxesSubplot:xlabel='tip', ylabel='day'>
```



2.7.2 Assignment: Box plot more design features

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


#Canvas Ballonn board
#sns.set(style='whitegrid')

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

p1= sns.boxplot(x="survived",y="age",showmeans=True,meanprops={"marker":
 ↪"*","markersize":"12","markeredgecolor":"red"},data=kashti)
plt.xlabel("How many survived"),
plt.ylabel("Age")
plt.title("Box plot of survived",size=14,weight="bold")
```

```
plt.show()
```

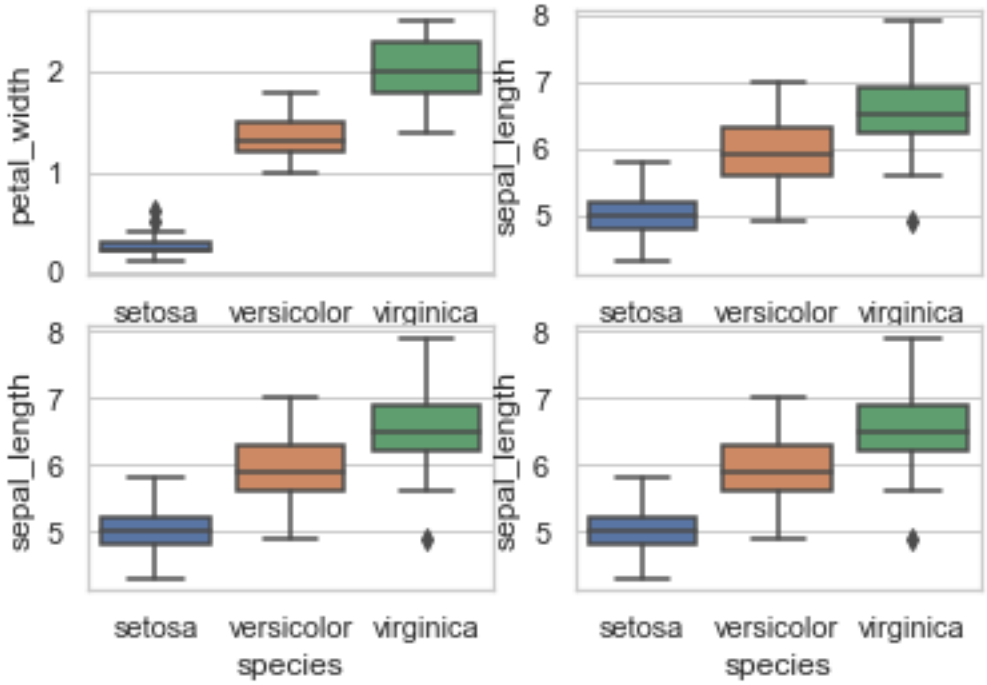


2.7.3 Assignment: Facet grid in Boxplot

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




sns.set(style="whitegrid")
iris_vis = sns.load_dataset("iris")

fig, axes = plt.subplots(2, 2)

ax = sns.boxplot(x="species", y="petal_width", data=iris_vis,
    ax=axes[0,0])
```

```
ax = sns.boxplot(x="species", y="sepal_length", data=iris_vis,
                 ax=axes[0,1])
ax = sns.boxplot(x="species", y="sepal_length", data=iris_vis,
                 ax=axes[1,0])
ax = sns.boxplot(x="species", y="sepal_length", data=iris_vis,
                 ax=axes[1,1])

#this is self created error for future learning
ax = sns.boxplot(x="species", y="petal_width", data=iris_vis, orient='v',
                  ax=axes[2,1])
```

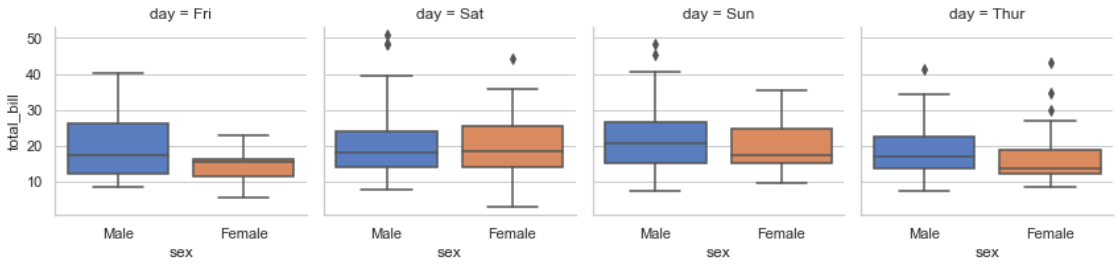
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12492/461546390.py in <module>
     23 #this is self created error for future learning
     24 ax = sns.boxplot(x="species", y="petal_width", data=iris_vis, orient='v' ,
---> 25                   ax=axes[2,1])

IndexError: index 2 is out of bounds for axis 0 with size 2
```



2.7.4 Assignment: Facet Grid Method 2

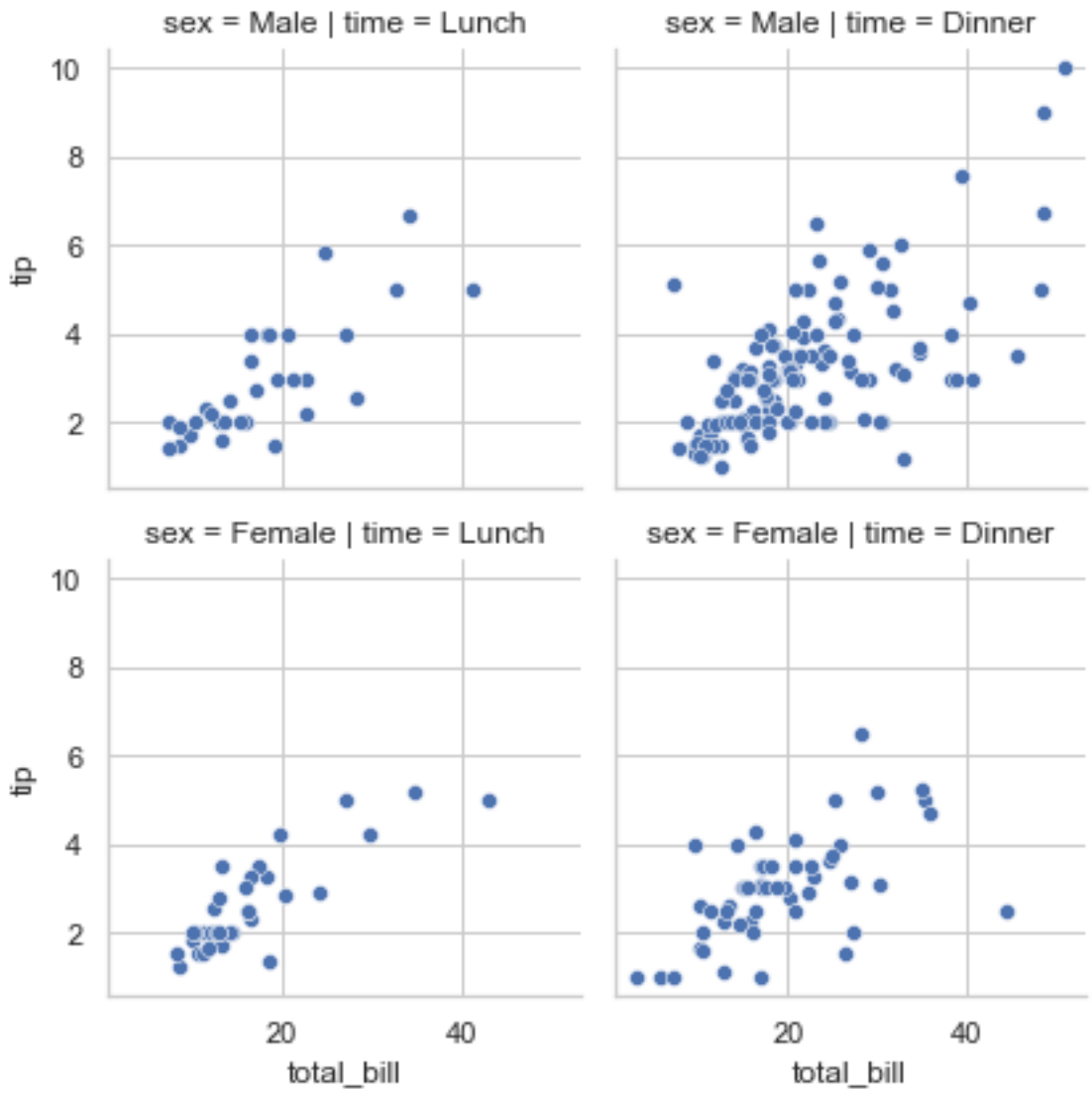
[ ]:
```
import seaborn as sns, matplotlib.pyplot as plt

tips = sns.load_dataset('tips')
ordered_days = sorted(tips['day'].unique())
g = sns.FacetGrid(tips,col='day',col_order=ordered_days,col_wrap=4)

g.map(sns.boxplot,'sex','total_bill',palette='muted')
for ax in g.axes.flatten():
    ax.tick_params(labelbottom=True)
plt.tight_layout()
plt.show()
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\seaborn\axisgrid.py:670: UserWarning: Using the boxplot function
without specifying `order` is likely to produce an incorrect plot.
  warnings.warn(warning)
```



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

tips = sns.load_dataset('tips')

g = sns.FacetGrid(tips, col="time",  row="sex")
g.map(sns.scatterplot, "total_bill", "tip")
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x22b2303d0c0>
```



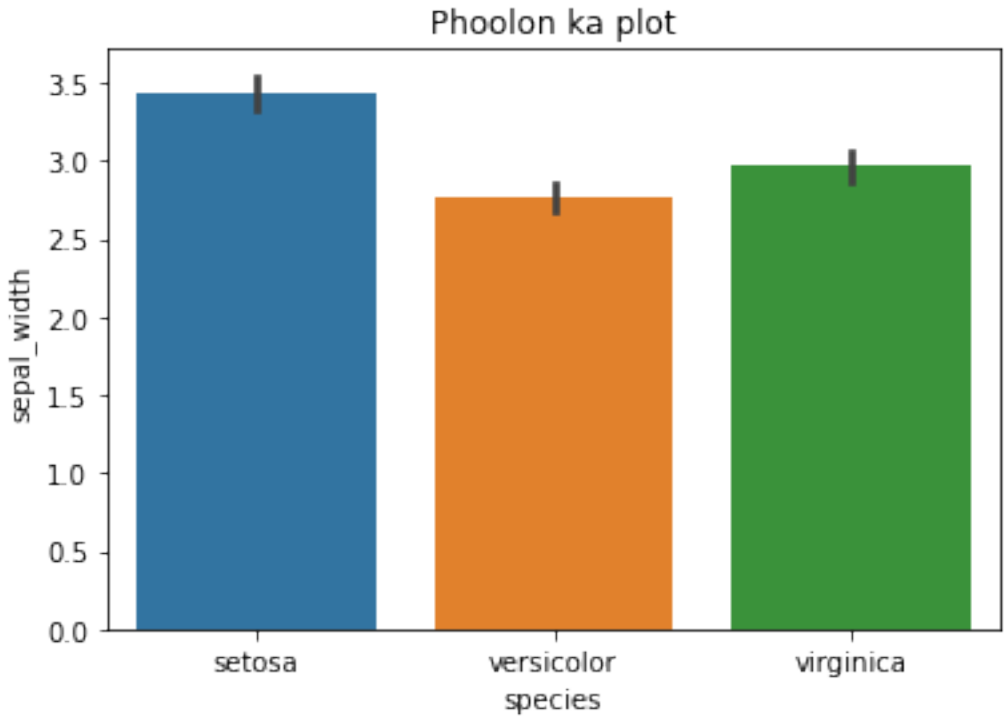
3 Facet Wrap you have to understand

Barplot on IRIS (Flower) data

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

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

```
phool

#draw a bar plot
sns.barplot(x="species",y="sepal_width",data=phool) # x categorical data ha   y␣
 ↪is numeric data
plt.title("Phoolon ka plot")
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

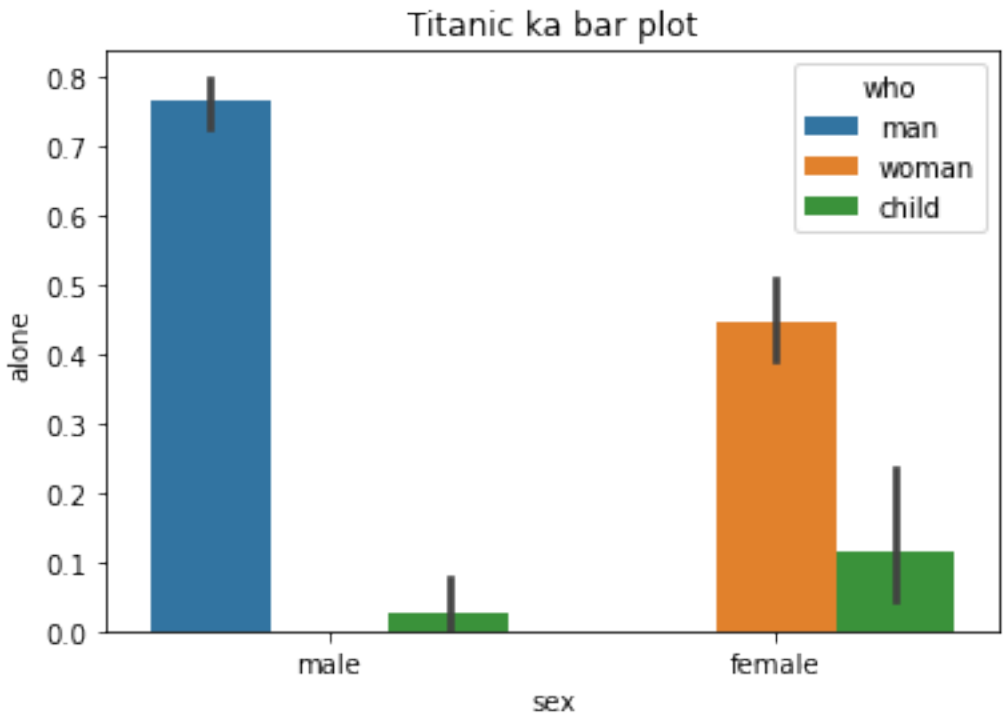


Barplot on Titanic Data

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

#load data set
kashti = sns.load_dataset("titanic")
kashti

#draw a bar plot
```

```
sns.barplot(x="sex",y="alone",hue="who",data=kashti) # x categorical data ha
 ↪y is numeric data
plt.title("Titanic ka bar plot")
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

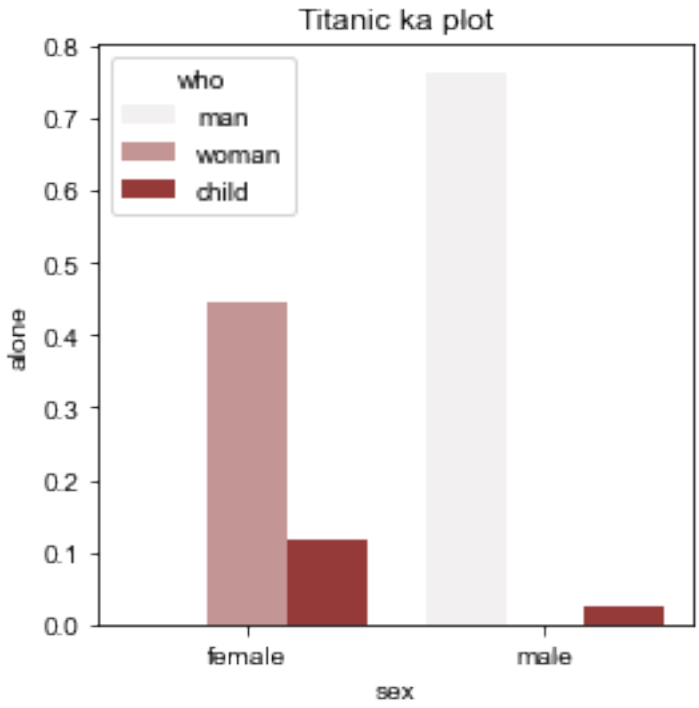


Barplot on Titanic Data with Detailing

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

#load data set
kashti = sns.load_dataset("titanic")
kashti

#figure size
plt.figure(figsize=(4,4))

#draw a bar plot
```

```
# order of data
# ci graph se dande hatane k lye
#
sns.barplot(x="sex",y="alone",hue="who",data=kashti,␣
 ↪order=["female","male"],color="brown",ci=None)
plt.title("Titanic ka plot")

#style
sns.set_style(style=None, rc=None)
sns.set_style("white")

#limits x and y
#plt.xlim(0)
#plt.ylim(0)



plt.show
```

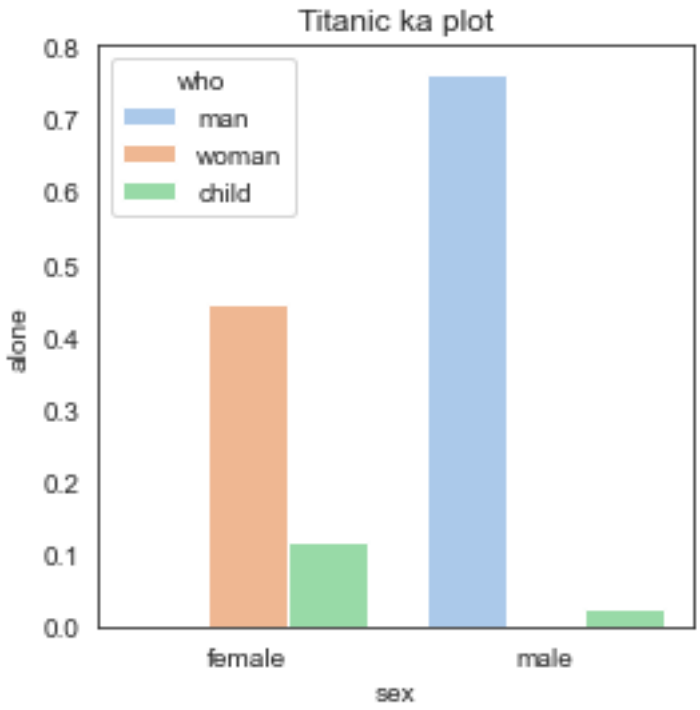
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



Palette and Dande removing (Line 17)

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


#load data set
kashti = sns.load_dataset("titanic")
kashti

#figure size
plt.figure(figsize=(4,4))

#draw a line plot
# order of data
# ci graph se dande hatane k lye
#
sns.barplot(x="sex",y="alone",hue="who",data=kashti,␣
 ↪order=["female","male"],color="brown",ci=None,palette="pastel")
plt.title("Titanic ka plot")

#style
sns.set_style(style=None, rc=None)
sns.set_style("white")

#limits x and y
#plt.xlim(0)
#plt.ylim(0)




plt.show
```

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

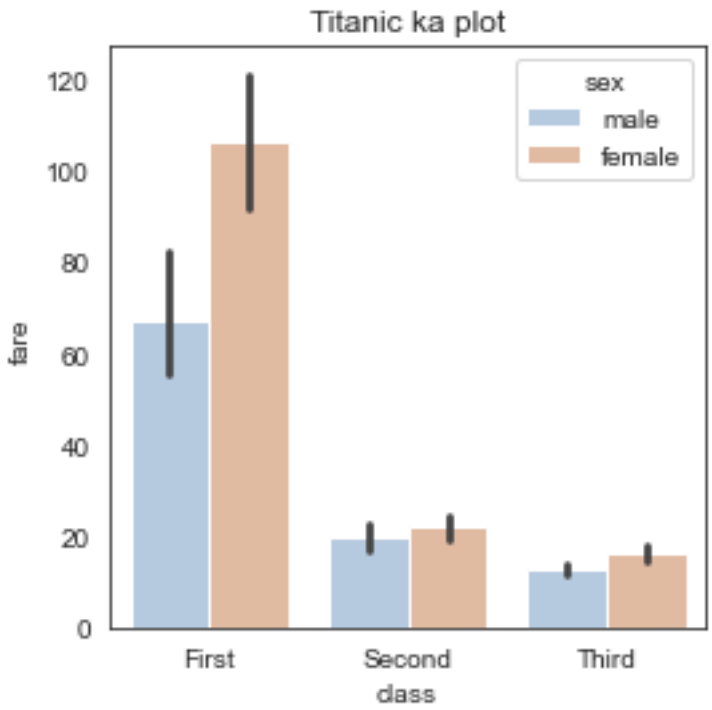


Estimator usage

[ ]:
```
# import libraries
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from numpy import mean
#load data set
kashti = sns.load_dataset("titanic")
kashti

#figure size
plt.figure(figsize=(4,4))

#draw a line plot
# order of data
# ci graph se dande hatane k lye (Confidence Interval)
# yahan tm ne order hataya tha tabhe sahe plot hua yad karlena tmhe lazmi yad␣
 ↪aega
# color saturation
sns.
 ↪barplot(x="class",y="fare",hue="sex",data=kashti,color="brown",ci=None,palette="pastel",est
 ↪5)
plt.title("Titanic ka plot")
```

```
#style
sns.set_style(style=None, rc=None)
sns.set_style("white")

#limits x and y
#plt.xlim(0)
#plt.ylim(0)




plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

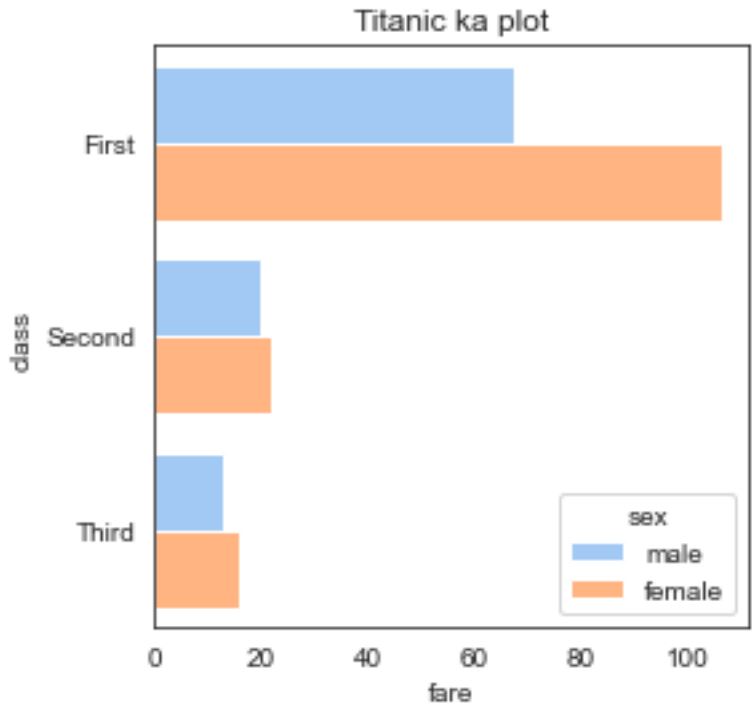


Horizontal plot

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

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

#figure size
plt.figure(figsize=(4,4))

#draw a line plot
# order of data
# ci graph se dande hatane k lye (Confidence Interval)
# yahan tm ne order hataya tha tabhe sahe plot hua yad karlena tmhe lazmi yad␣
 ↪aega
# color saturation
sns.
 ↪barplot(x="fare",y="class",hue="sex",data=kashti,color="brown",ci=None,palette="pastel",est
plt.title("Titanic ka plot")

#style
sns.set_style(style=None, rc=None)
sns.set_style("white")

#limits x and y
#plt.xlim(0)
#plt.ylim(0)




plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>

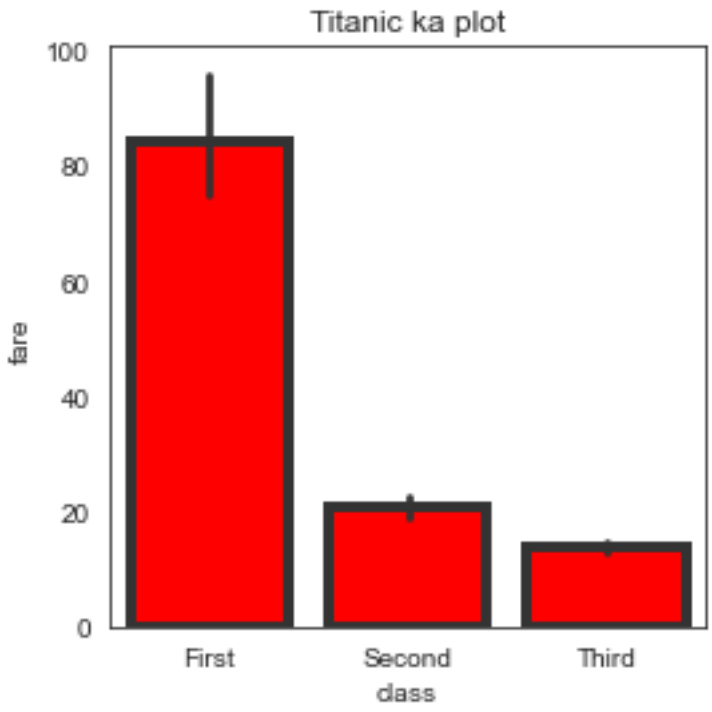


More functionalities related to Design

[ ]:
```
# import libraries
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from numpy import mean
#load data set
kashti = sns.load_dataset("titanic")
kashti

#figure size
plt.figure(figsize=(4,4))

#draw a line plot
# order of data
# ci graph se dande hatane k lye (Confidence Interval)
# yahan tm ne order hataya tha tabhe sahe plot hua yad karlena tmhe lazmi yad␣
 ↪aega
# color saturation
# line width= motai bar ki lines ki, #edgecolor= simple yar edge ka color,␣
 ↪errcolor=bech wale dande ka color,
#facecolor=rgba
```

```
#sns.
 ↪barplot(x="fare",y="class",hue="sex",data=kashti,color="brown",ci=None,palette="pastel",est
sns.barplot(x="class", y="fare",data=kashti,linewidth=4, facecolor=(1,0,0,1)
,errcolor=".2", edgecolor=".2")

plt.title("Titanic ka plot")

#style
sns.set_style(style=None, rc=None)
sns.set_style("white")

#limits x and y
#plt.xlim(0)
#plt.ylim(0)




plt.show
```

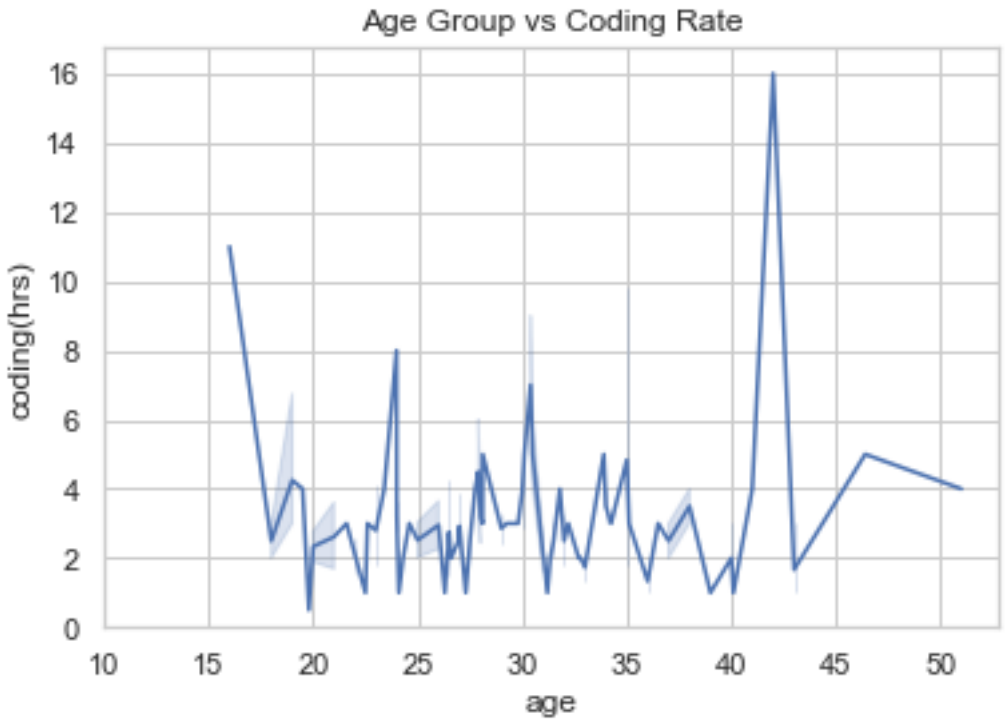
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



3.0.1 Assignment: Line plot between Age Group and Hours/Day coding

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

# import data from file
chilla = pd.read_csv("Chilla_data2_for_plots.csv")
#print(chilla)

#figure size
#plt.figure(figsize=(1,1))

#draw a line plot
sns.lineplot(x="age",y="coding(hrs)",data=chilla)
plt.title("Age Group vs Coding Rate")

#style
#sns.set_style(style=None, rc=None)
#sns.set_style("dark")

#limits x and y
plt.xlim(10)
plt.ylim(0)




plt.show
```

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

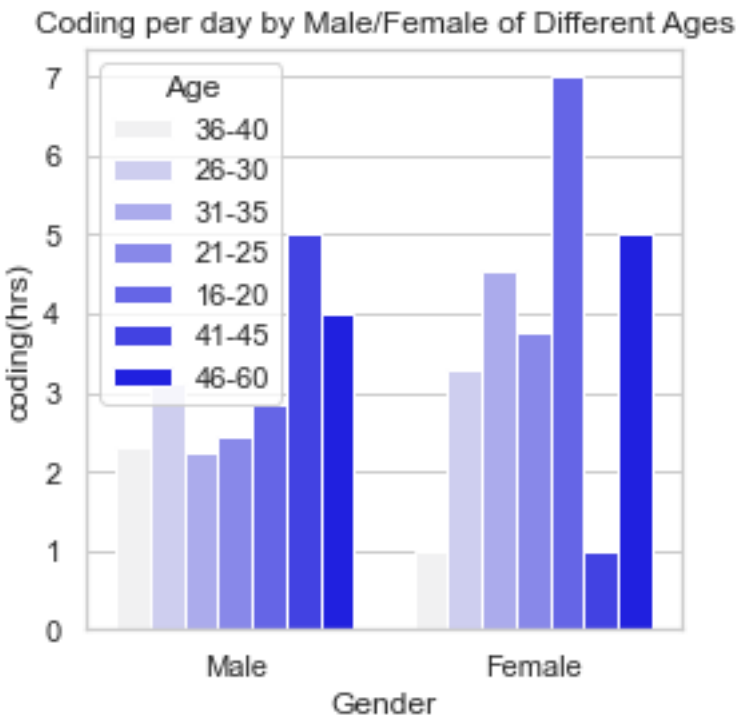


3.0.2 Assignment: Bar plot with the same data

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


# import data from file
chilla = pd.read_csv("Chilla_data2_for_plots.csv")
#print(chilla)

#figure size
plt.figure(figsize=(4,4))

#draw a bar plot
# order of data
# ci graph se dande hatane k lye
# line 256 and 289 wrong data replaced by NAN
```

```
sns.
  ↪barplot(x="Gender",y="coding(hrs)",hue="Age",data=chilla,color="blue",ci=None)
plt.title("Coding per day by Male/Female of Different Ages")

#style
sns.set_style(style=None, rc=None)
sns.set_style("white")

#limits x and y
#plt.xlim(0)
#plt.ylim(0)




plt.show
```

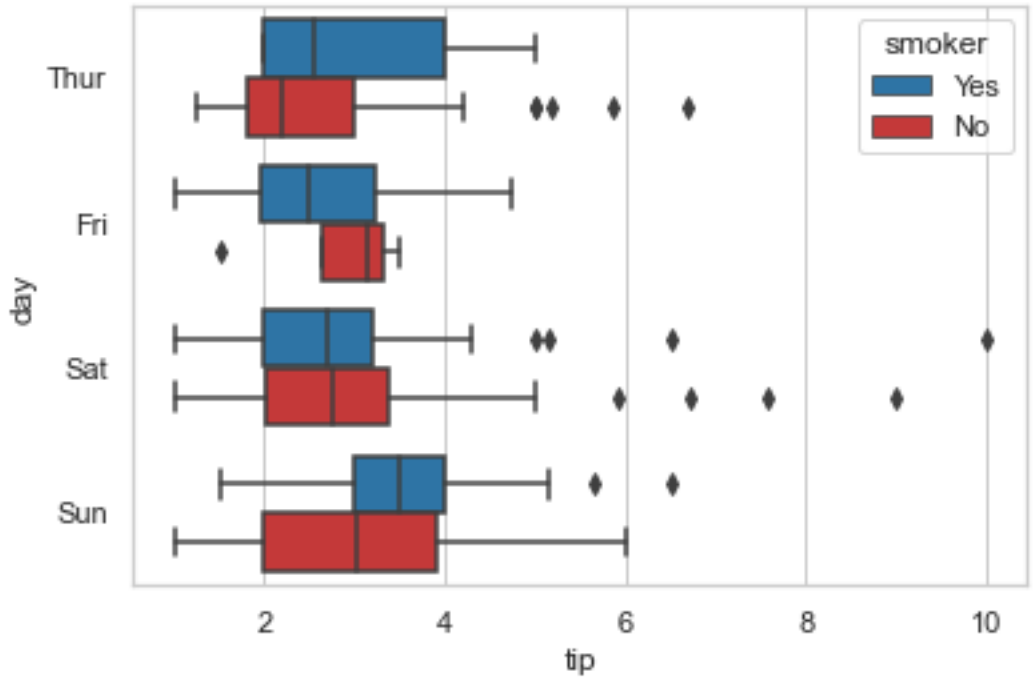
[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



3.0.3 Assignment: Task of assigning different HUE color to each HUE value (Customized)

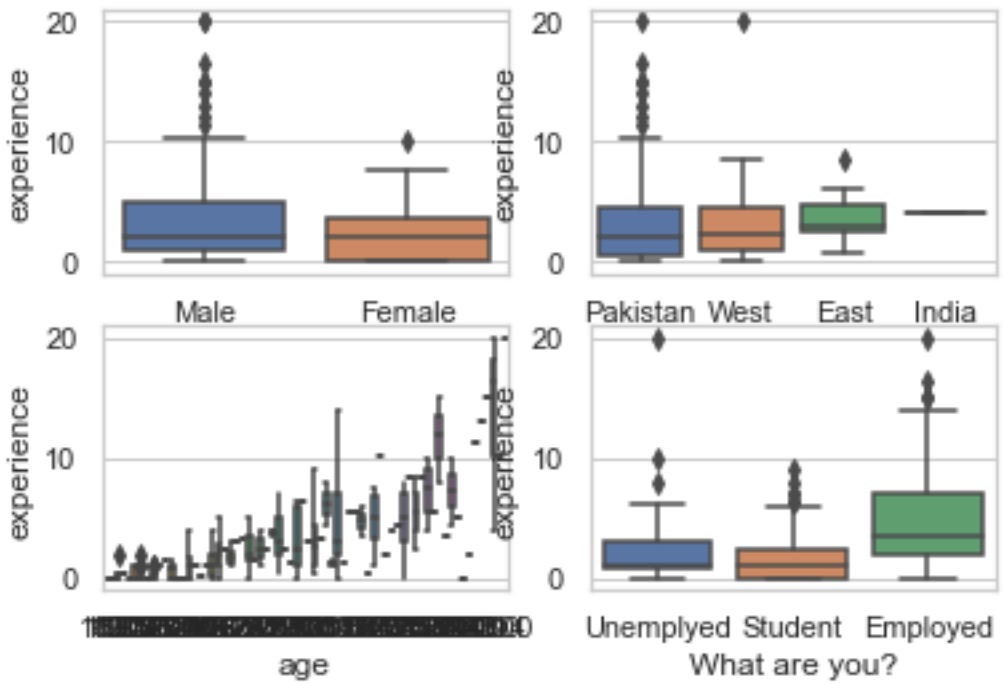
[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


#Canvas Ballonn board
sns.set(style='whitegrid')

tip = sns.load_dataset("tips")
# it is describing all numeric values
tip.describe()
palette = {
    'Yes': 'tab:blue',
    'No': 'tab:red',

}

sns.boxplot(x="tip",y="day",hue="smoker",data=tip,saturation=0.
 ↪8,dodge=True,palette=palette,color="#1515eb")
```

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



3.0.4 Assignment: Facet Wrap Using Stack Overflow Help On Chilla Data (Needs Refining)

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

sns.set(style="whitegrid")

# import data from file
chilla = pd.read_csv("Chilla_data2_for_plots.csv")
#print(chilla)
#plt.figure(figsize=(4,12))
fig, axes = plt.subplots(2, 2)
figsize=(4,12)
z= "experience"
ax = sns.boxplot(x="Gender", y=z,data=chilla,
    ax=axes[0,0])
ax = sns.boxplot(x="Location", y="experience",data=chilla,
                 ax=axes[0,1])
ax = sns.boxplot(x="age", y="experience",data=chilla,
                 ax=axes[1,0])
ax = sns.boxplot(x="What are you?", y='experience',data=chilla,
                 ax=axes[1,1])

#this is self created error for future learning
#ax = sns.boxplot(x="", y="petal_width",data=chilla, orient='v',
 #                 ax=axes[2,1])
```



3.0.5 Plotly Express

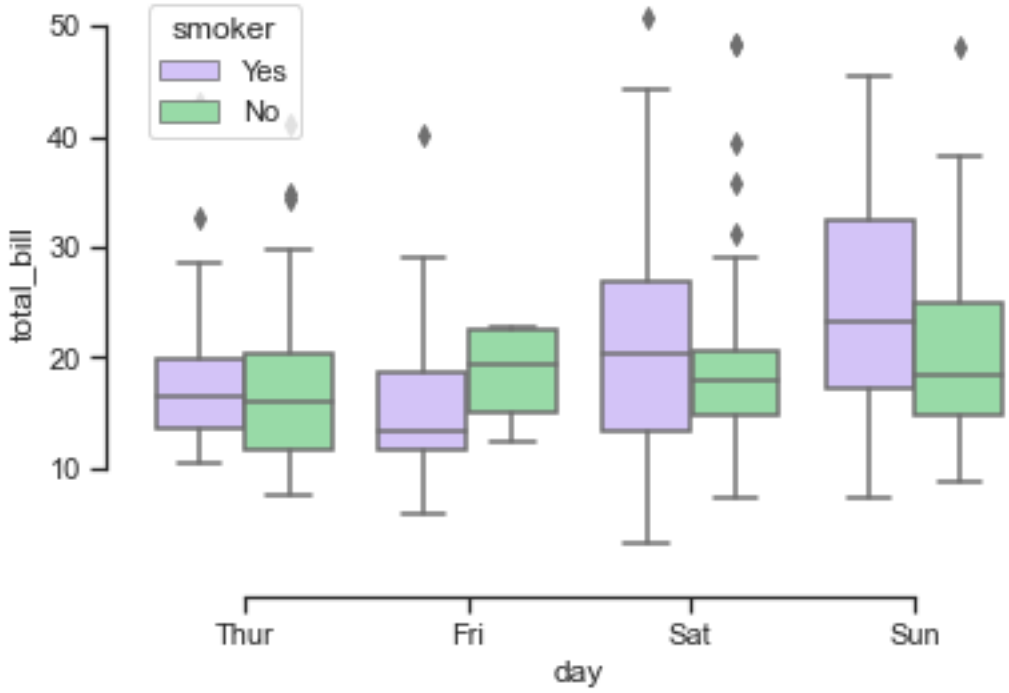
[ ]:
```
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
df = pd.read_csv("Chilla_data2_for_plots.csv")
fig = px.scatter_matrix(df, dimensions=["weight", "coding(hrs)", "age",␣
↪"experience"], color="Location")
fig.show()
```

Plotly Express (More Practise)

[ ]:
```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
fig.show()
```

[ ]:
```
import seaborn as sns
sns.set_theme(style="ticks", palette="pastel")

# Load the example tips dataset
```

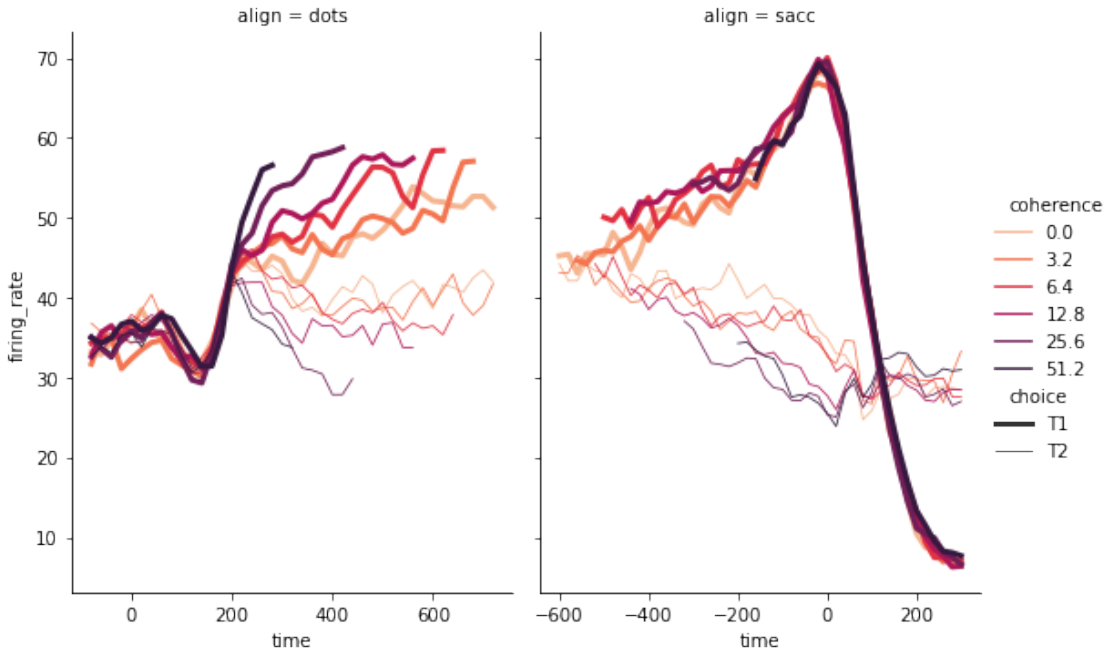
```
tips = sns.load_dataset("tips")

# Draw a nested boxplot to show bills by day and time
sns.boxplot(x="day", y="total_bill",
            hue="smoker", palette=["m", "g"],
            data=tips)
sns.despine(offset=10, trim=True)
```



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

nukta = sns.load_dataset("dots")

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

# plot line plot
sns.relplot(data=nukta,
            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)
           )
```

```
<seaborn.axisgrid.FacetGrid at 0x1f4d64160a0>
```



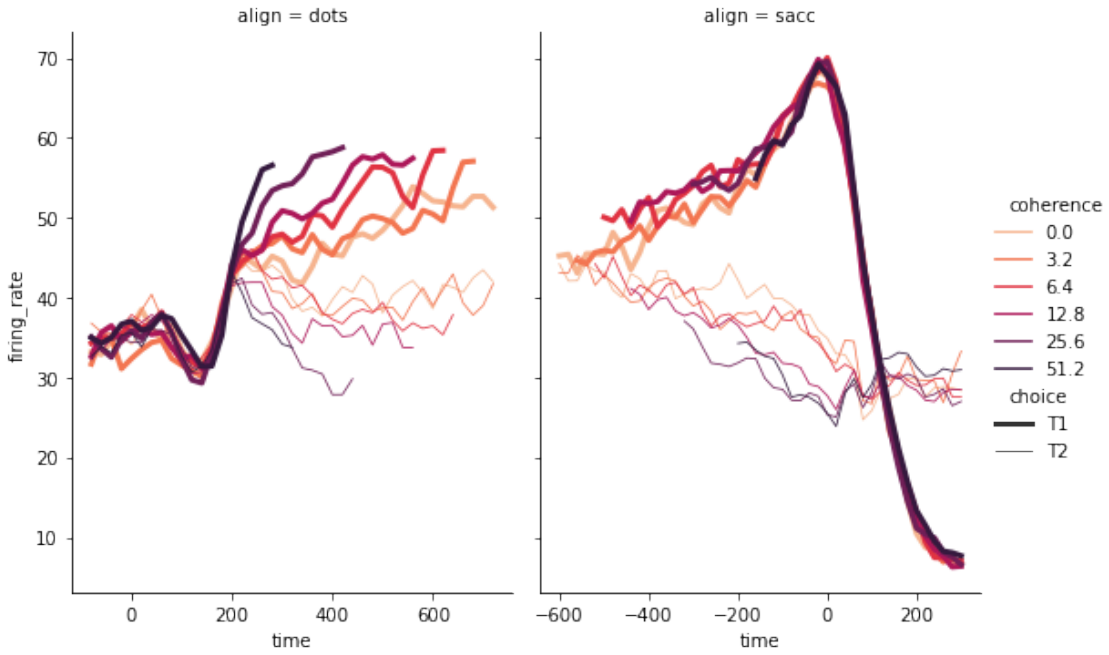
Task 2 : concatenate 2 array of different dimensions

3.1 1. Lineplot with Multifacets

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

nukta = sns.load_dataset("dots")

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

# plot line plot
```

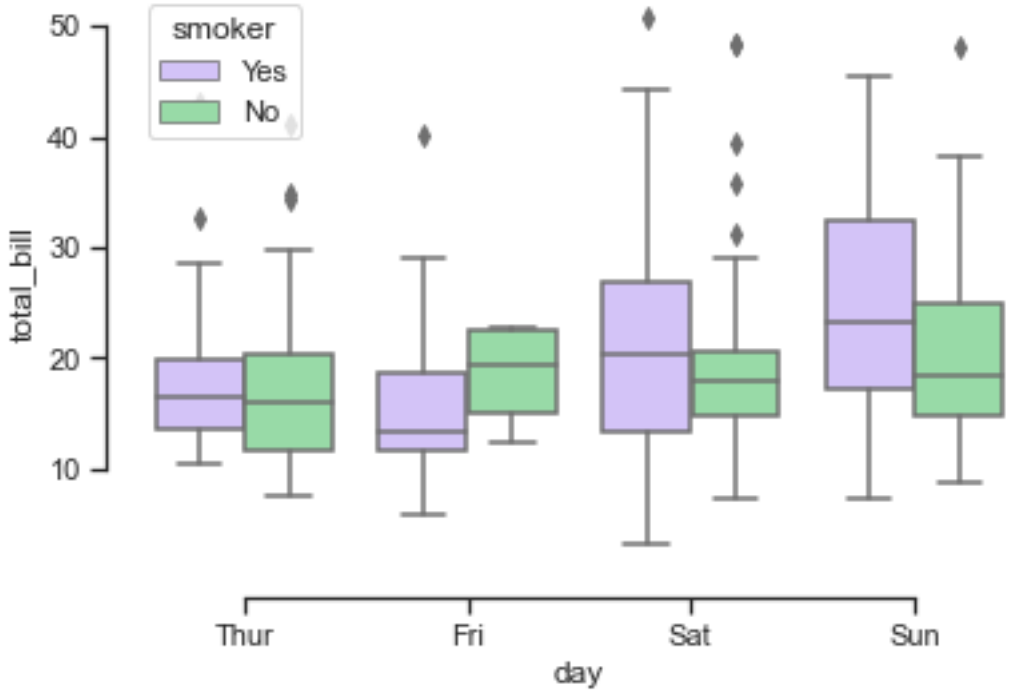
```
sns.relplot(data=nukta,
            x="time",y="firing_rate", hue="coherence", size="choice",␣
 ↪col="align", kind="line", size_order=
             ["T1", "T2"],
             palette=palette, height=5, aspect=.75, facet_kws=dict(sharex=False)
            )
```

[ ]: <seaborn.axisgrid.FacetGrid at 0x1f4d64160a0>



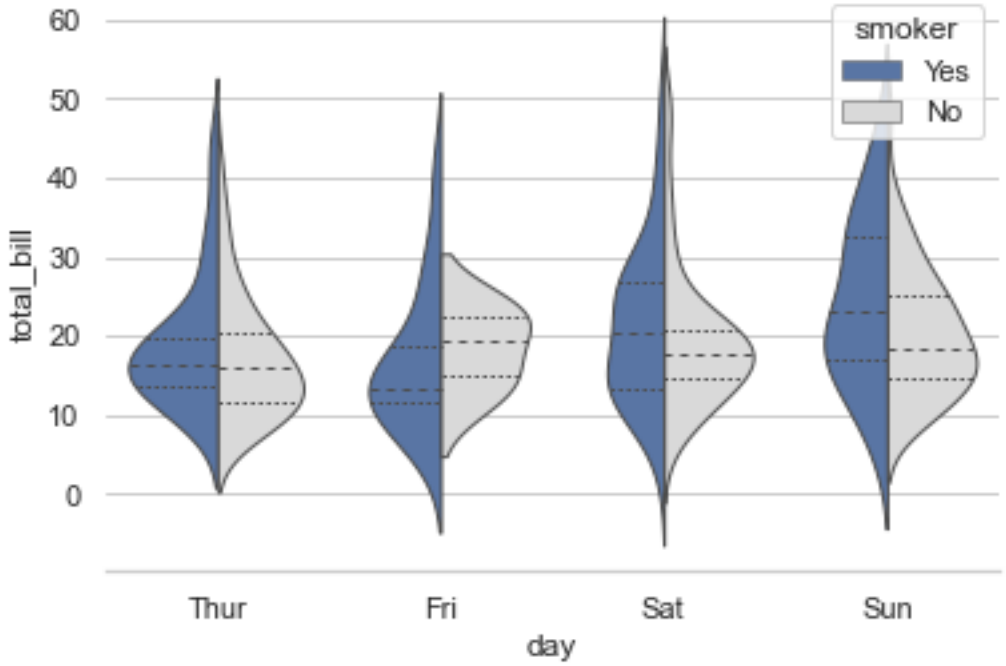
3.2 Nested Boxplot

[ ]:
```
import seaborn as sns
sns.set_theme(style="ticks", palette="pastel")

# Load the example tips dataset
tips = sns.load_dataset("tips")

# Draw a nested boxplot to show bills by day and time
sns.boxplot(x="day", y="total_bill",
            hue="smoker", palette=["m", "g"],
            data=tips)
sns.despine(offset=10, trim=True)
```



3.3 Violin plot

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

# Load the example tips dataset
tips = sns.load_dataset("tips")

# Draw a nested violinplot and split the violins for easier comparison
sns.violinplot(data=tips, x="day", y="total_bill", hue="smoker",
               split=True, inner="quart", linewidth=1,
               palette={"Yes": "b", "No": ".85"})
sns.despine(left=True)
```

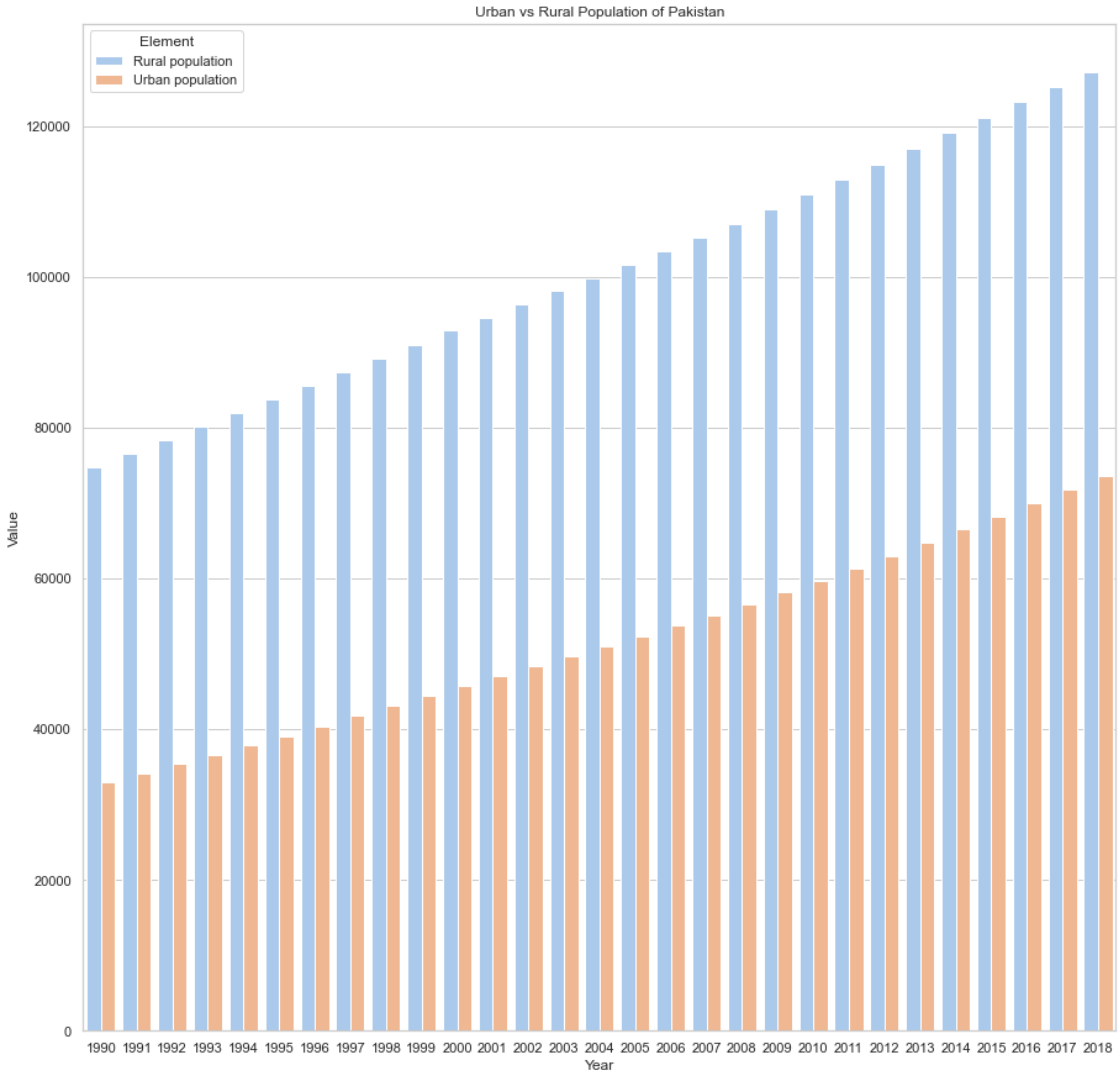


3.3.1 Assignment: Graph on FAO Data

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

#load data set
chilla = pd.read_csv("fao_plot/faostat.csv")
#print(chilla)

#figure size
plt.figure(figsize=(15,15))

#draw a bar plot
# order of data
# ci graph se dande hatane k lye
#
sns.barplot(x="Year",y="Value",hue="Element",data=chilla,ci=None)
plt.title("Urban vs Rural Population of Pakistan")

#style
sns.set_style(style=None, rc=None)
sns.set_style("whitegrid")
```

```
#limits x and y
#plt.xlim(0)
#plt.ylim(0)




plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>



3.4 Numpy

[ ]:
```
np.empty(7)
np.empty(5)
# basically it creates a vacant error in memory location. I verified with␣
 ↪different sizes.
```

```
array([8.40e-323, 8.89e-323, 9.39e-323, 9.88e-323, 1.04e-322])
```

Creating an Array using Numpy

[ ]:
```
# array creation 1D
import numpy as np
food = np.array(["Pakora","Samosa","Raita"])
food
```

[ ]:
```
array(['Pakora', 'Samosa', 'Raita'], dtype='<U6')
```

[ ]:
```
# array type
price = np.array([5,5,5])
price
type(price)
```

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

[ ]:
```
len(price)
len(food)
```

[ ]:
```
3
```

[ ]:
```
# price[3] Index error
price[2]
price[0:2]
z = price[0:]
z
```

[ ]:
```
array([5, 5, 5])
```

[ ]:
```
p= food[1]
p
```

[ ]:
```
'Samosa'
```

[ ]:
```
#Array k functions
price.mean()
```

[ ]:
```
5.0
```

[ ]:
```
#zeros array
np.zeros(6)
#1s array
np.ones(5)
```

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

[ ]:
```
np.empty(7)
```

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

[ ]:
```
# with a spacing of interval
np.arange(2,20,5)
```

[ ]:
```
array([ 2,  7, 12, 17])
```

[ ]:
```
np.arange(10)
# last element is excluded
```

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

[ ]:
```
# line space at specific interval (10 nums at fixed interval)
# isme akhri num ko include kar leta hay
np.linspace(3,20,10)
```

[ ]:
```
array([ 3.        ,  4.88888889,  6.77777778,  8.66666667, 10.55555556,
       12.44444444, 14.33333333, 16.22222222, 18.11111111, 20.        ])
```

[ ]:
```
# specify your data types
np.empty(50, dtype=np.int64)
```

[ ]:
```
array([3706497945030232697, 3328209646291068976, 4123389851770370361,
       3761694506697177401, 2340008602714185781, 4485090493615726966,
       3832057680150884384, 3832617357338806825, 4122818071313266484,
       2531084808905307188, 8241998674912177440, 8319675098974521977,
       4210425200352785012, 7883868074393078282, 8386103967300611952,
       3342349787993173104, 6998721842843253104, 4485033774059364467,
       7935409752961982526, 7021238737122897520, 3900165871320458595,
       4470430867062333484, 8079524940746866238, 2915077370344797292,
       3325662225219202168, 8079506593066003495, 3253604629844359208,
       8079506593066003495, 2318273471217938483, 4485033450774801703,
       7816329705848578110, 2968483698408189289, 2968470478583048202,
       8389203489669922314,   11584967480472366, 1009865545543123192,
                         0,     140723773312096,                3905,
                        -1, 7954884616238688484,                   0,
       8174913433131640140, 8029953815596917109, 7809639147579013484,
       3317475270149629472, 4195777553609138222, 8316292897441849354,
       8079584645411202592, 8367800735126286945], dtype=int64)
```

3.4.1 Array functions

[ ]:
```
a = np.array ([10,12,15,2,4,6,100,320,0,5,10,3])
a.sort()
a
```

[ ]:
```
array([  0,   2,   3,   4,   5,   6,  10,  10,  12,  15, 100, 320])
```

[ ]:
```
b = np.array([10.2,3.4,53.6,91.6,45.5])
c= np.concatenate((a,b))
c
c.sort()
c
```

[ ]:
```
array([  0. ,   2. ,   3. ,   3.4,   4. ,   5. ,   6. ,  10. ,  10. ,
        10.2,  12. ,  15. ,  45.5,  53.6,  91.6, 100. , 320. ])
```

3.4.2 2D arrays

[ ]:
```
import numpy as np
# You have to have same no dimensions to concatenate
a = np.array([[1,2],[5,4]])
b = np.array([[6,7,],[8,9]])

c = np.concatenate((a,b),axis=0)
c
```

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

[ ]:
```
c = np.concatenate((a,b),axis=1)
c
```

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

3.4.3 3D Arrays

[ ]:
```
a = np.array([
     [[0,1,2,3],[4,5,6,7]],              # 1st 2d dim of a 3d
               [[0,1,2,3],[4,5,6,7]],    # 2nd dimension of a 3d
               [[0,1,2,3],[4,5,6,7]]     # 3rd dimension of a 3d
 ])
print(a)
a.ndim
```

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

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

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

[ ]:
```
3
```

[ ]:
```
import numpy as np
b = np.array([
    [1,2,3,4],[1,2,3,4],[1,2,3,4]
             ])
print(b)
c=b.ndim
print("The dimension of array b is 2D",c)
d=b.size
print("The size (no of elements) of array b is",d)
e= a.shape
print("The shape  of array a is 3 dimension and 2 row 4 column as above example␣
 ↪that is ",e)

f= b.shape
print("The shape  of array b is 3 dimension and column first row last above␣
 ↪example that is ",f)
```

```
[[1 2 3 4]
 [1 2 3 4]
 [1 2 3 4]]
The dimension of array b is 2D 2
The size (no of elements) of array b is 12
The shape  of array a is 3 dimension and 2 row 4 column as above example that is
(3, 2, 4)
The shape  of array b is 3 dimension and column first row last above example
that is  (3, 4)
```

[ ]:
```
# reshaaping concept (like transpose) and dimension conversion
# a = np.arange(5)    this will generate error cux (3*2=6 me reshape is a␣
 ↪hassle) see line 4
a = np.arange(6)
print(a)
c=a.ndim
print("the dimension of a is",c)
b = a.reshape(3,2)
print(b)
d=b.ndim
```

```
print("The converted dimension of",c,"D into b is 2d=",d)
```

```
[0 1 2 3 4 5]
the dimension of a is 1
[[0 1]
 [2 3]
 [4 5]]
The converted dimension of 1 D into b is 2d= 2
```

[ ]:
```
# More Reshape
import numpy as np
f= np.reshape(b, newshape=(1,6), order='C')
f
```

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

Converting 1D array to 2D array by Axes Method

[ ]:
```
a = np.array([1,2,3,4,5,6,7,8,9])
print(a)
a.shape
print("the shape of 1D array is 9 elements",a.shape)

# row wise 2D conversion
b=a[np.newaxis,:]
print("a is converted to 2D. Nishani is braces",b)
print("the dimension of b is ",b.shape)

# column wise 2D conversion
c=a[:,np.newaxis]
print("a is converted to 2D. Nishani is braces",c)
print("the dimension of b is ",c.shape)
```

```
[1 2 3 4 5 6 7 8 9]
the shape of 1D array is 9 elements (9,)
a is converted to 2D. Nishani is braces [[1 2 3 4 5 6 7 8 9]]
the dimension of b is  (1, 9)
a is converted to 2D. Nishani is braces [[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
the dimension of b is  (9, 1)
```

Adjusting dimensions of converted 2D from above

[ ]:
```
d = c.reshape(3,3)
d[1][1]
```

[ ]: 5

3.5 Lecture on Markdown

4 1-Table of Contents

Headings
Block of Words/Citation
Line Breaks
Combine Two things
Face of Text
Bullet Points Lists
Breaks
Links and Hyperlink
Images and Figures
Adding Code Block
Adding table
Table of Content
Installation of Extension
Adding Comment Color Changing
Equation and Math Function

5 2- Headings

How to give heading in Markdown File # Heading 1 ## Heading 2 ### Heading 3 #### Heading 4 ##### Heading 6

6 3- Block of Words or Citation

This is a normal text in markdown

> This is a block of special text
>
> This is also special

7 4- Line Breaks

This is a 40 days long Course Data Science with Python. AKA Python_ka_Chilla_with_BabaAammar.
This is a second line. you can use enter or

8 5- Combine two things

Block of words and heading

8.1 Heading 2

9 6- Face of Text

Bold

Italic

Bold and Italic

Or you can use these symbols **Bold**
Italic

Write in comments about bold and italic

10 7- Bullet points/ Lists

- Day-1
- Day-2
- Day-3
- Day-4
- Day-5
  - Day-5a
    - sublist
    - sublist 2
  - Day-5b
- Day-6 -Day-7

Numbering of list 1. Day-1 2. Day-2 4. Day-3 1. Day-3a 3. Day-3b 4. Day-4 5. Day-5 6. Day-6

Using * or #

- Day-1
- Day-2

11 8- Line Breaks or Page Breaks

This is page1.

This is page 2.

12 9- Links and HyperLinks

https://www.youtube.com/watch?v=qJqAXjz-Rh4&list=PL9XvIvvVL5OHVsu-Ao8NBrOUJSO8O6lBI&index=21

The playlist of python ka chilla can be found here

The whole course is here.

13 10- Images and Figures with link

To join this course, please scan the following QR code and join telegram group:

QR QR

Codanics

14 11- Adding Code or Code Block

To print a string use `print("Codanics")`

```
print("hello babaji")
```

This code will show color of syntax

```
x=2=3;
y=3+3;
```

15 12- Adding Tables

Species	petal_length	sepal_length
virginica	18.2	19.2
setosa	12.2	12.5

16 13- Install Extensions

Sample text

Link

image

Column A	Column B	Column C
A1	B1	C1
A2	B2	C2
A3	B3	C3

17 14. Commenting

Baba Aammar ka Chilla

OK

Task accomplished search commenting and shortcut for it.

18 15. How to change Color

Example

Use span command
span style=“color:red”>
This text color is red

19 16. Adding equations in Markdown

In-line Math

$this_{is}^{inline}$

Maths Block

$$\int_0^\infty \frac{x^3}{e^x - 1}\,dx = \frac{\pi^4}{15}$$

You can watch the following link for more information: MathJax

19.1 End of Markdown File

19.2 1. Pandas

Intalling libraries libraries
Importing libraries
Define a Series (a column list with a Not A Number)

[ ]:
```
# pip install numpy
# pip install pandas
import numpy as np
import pandas as pd
s = pd.Series([1,3,np.nan,5,7,8,9,10])
s
```

[ ]:
```
0     1.0
1     3.0
2     NaN
3     5.0
4     7.0
5     8.0
6     9.0
7    10.0
dtype: float64
```

19.2.1 2. Generating Data Series/Frames

Printing Dates in a series

[ ]:
```
dates = pd.date_range("20220101",periods=20)
dates
```

[ ]:
```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05', '2022-01-06', '2022-01-07', '2022-01-08',
               '2022-01-09', '2022-01-10', '2022-01-11', '2022-01-12',
               '2022-01-13', '2022-01-14', '2022-01-15', '2022-01-16',
               '2022-01-17', '2022-01-18', '2022-01-19', '2022-01-20'],
              dtype='datetime64[ns]', freq='D')
```

Generating a Data Frame using the Dates as Index of that Data set

[ ]:
```
# np.random.randn(20,4) this indicates that keep index = 20 (row split) of␣
 ↪table, where as 4 is the column split
df = pd.DataFrame(data=np.random.randn(20,4), index=dates,␣
 ↪columns=list("ABCD"), dtype=float, copy=None)
df
```

[ ]:

	A	B	C	D
2022-01-01	-0.807823	-0.067819	-1.330167	0.074816
2022-01-02	0.470514	0.100841	1.017651	-1.245134
2022-01-03	-0.040778	1.192630	0.935529	-1.628175
2022-01-04	-0.493067	-0.660789	-0.601053	-1.287295
2022-01-05	0.689037	-0.448571	-0.712814	0.695055
2022-01-06	-1.302253	-1.658869	-1.811578	0.671019
2022-01-07	0.838272	0.111321	-0.809320	-0.475192
2022-01-08	-0.377271	0.510839	0.041828	-0.060369
2022-01-09	0.396088	1.338157	-0.759130	1.828155
2022-01-10	-0.557112	-0.273150	2.068674	0.741978
2022-01-11	0.550546	-0.565107	-0.085478	-0.423174
2022-01-12	-0.083672	-1.471417	-0.039479	0.111297
2022-01-13	0.074566	-0.125679	1.443004	0.034680
2022-01-14	1.265377	0.554316	1.188476	-2.156915
2022-01-15	1.275653	-0.101605	0.780956	0.675729
2022-01-16	-1.602749	0.251363	0.271529	2.246843
2022-01-17	-1.566494	0.545225	-0.388877	-0.317478
2022-01-18	1.251463	1.940221	-1.155982	-1.181810
2022-01-19	0.062309	-1.656761	0.531129	-1.674739
2022-01-20	1.019831	-0.298755	-0.044523	0.686477

Checking the data type of Data frame

[ ]:
```
df.dtypes
```

[ ]:
```
A    float64
B    float64
C    float64
D    float64
dtype: object
```

Generating a Data Frame using Dictionary Method (Key=Column names)

[ ]:
```
df2 = pd.DataFrame(
    {
        "A":2.5,
        "B":pd.Timestamp("20220114"),
        "C": pd.Series(1,index=list(range(4)),dtype="float32"),
        "D":np.array([3]*4, dtype="int32"),
        "E":pd.Categorical(["boy","baba","sakht londay","sigma male"]),
        "F": "Males",

    }

)
df2
```

[ ]:

	A	B	C	D	E	F
0	2.5	2022-01-14	1.0	3	boy	Males
1	2.5	2022-01-14	1.0	3	baba	Males
2	2.5	2022-01-14	1.0	3	sakht londay	Males
3	2.5	2022-01-14	1.0	3	sigma male	Males

[ ]:
```
df7 = pd.DataFrame(
    {
        "A":2.5,
        "B":pd.Timestamp("20220114"),
        "C": pd.Series(1,index=list(range(4)),dtype="float32"),
        "D":np.array([3]*4, dtype="int32"),
        "E":pd.Categorical(["boy","baba","sakht londay","sigma male"]),
        "F": "Males",

    }
    ,index=['first', 'second','third','four']

)
df7
```

[ ]:

	A	B	C	D	E	F
first	2.5	2022-01-14	NaN	3	boy	Males
second	2.5	2022-01-14	NaN	3	baba	Males
third	2.5	2022-01-14	NaN	3	sakht londay	Males

```
four    2.5 2022-01-14 NaN  3    sigma male  Males
```

[ ]:
```
import pandas as pd
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df8 = pd.DataFrame(data, index=['first', 'second'])
print(df8)
```

```
        a   b     c
first   1   2   NaN
second  5  10  20.0
```

Checking the Data type of data frame created with Dictionary

[ ]:
```
df2.describe()
```

[ ]:
```
          A    C    D
count  4.0  4.0  4.0
mean   2.5  1.0  3.0
std    0.0  0.0  0.0
min    2.5  1.0  3.0
25%    2.5  1.0  3.0
50%    2.5  1.0  3.0
75%    2.5  1.0  3.0
max    2.5  1.0  3.0
```

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

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

Another Data Frame generation (Mapping according to columns)

[ ]:
```
data = [['Alex',10],['Bob',12],['Clarke',13]]
df6 = pd.DataFrame(data,columns=['Name','Age'])
print(df6)
```

```
     Name  Age
0    Alex   10
1     Bob   12
2  Clarke   13
```

[ ]:
```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df15 = pd.DataFrame(data)
```

```
print(df15)
```

```
    Name  Age
0    Tom   28
1   Jack   34
2  Steve   29
3  Ricky   42
```

Converting data frame (df) to numpy (Array)

[ ]:
```
f= df.to_numpy()
f
```

[ ]:
```
array([[-0.80782255, -0.06781884, -1.33016708,  0.07481607],
       [ 0.47051419,  0.10084056,  1.01765052, -1.24513377],
       [-0.04077768,  1.19262986,  0.93552934, -1.62817466],
       [-0.4930668 , -0.6607893 , -0.60105347, -1.28729493],
       [ 0.68903713, -0.4485706 , -0.71281443,  0.69505513],
       [-1.30225331, -1.65886928, -1.81157792,  0.67101941],
       [ 0.83827152,  0.11132093, -0.80932006, -0.47519209],
       [-0.37727078,  0.51083937,  0.04182803, -0.0603689 ],
       [ 0.39608832,  1.33815701, -0.75913005,  1.82815546],
       [-0.5571121 , -0.27315032,  2.0686735 ,  0.7419781 ],
       [ 0.55054643, -0.56510706, -0.08547801, -0.42317445],
       [-0.08367169, -1.47141736, -0.03947904,  0.11129695],
       [ 0.07456605, -0.12567864,  1.44300411,  0.03468045],
       [ 1.26537728,  0.5543164 ,  1.18847556, -2.15691471],
       [ 1.2756534 , -0.10160487,  0.78095598,  0.67572942],
       [-1.60274865,  0.25136267,  0.27152883,  2.24684335],
       [-1.56649381,  0.54522529, -0.3888765 , -0.31747769],
       [ 1.25146326,  1.94022066, -1.15598215, -1.18181022],
       [ 0.0623093 , -1.65676135,  0.53112893, -1.67473858],
       [ 1.01983136, -0.29875472, -0.04452312,  0.68647655]])
```

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

[ ]:
```
array([[2.5, Timestamp('2022-01-14 00:00:00'), 1.0, 3, 'boy', 'Males'],
       [2.5, Timestamp('2022-01-14 00:00:00'), 1.0, 3, 'baba', 'Males'],
       [2.5, Timestamp('2022-01-14 00:00:00'), 1.0, 3, 'sakht londay',
        'Males'],
       [2.5, Timestamp('2022-01-14 00:00:00'), 1.0, 3, 'sigma male',
        'Males']], dtype=object)
```

Transpose

[ ]:
```
# to transpose
df2.T
```

[ ]:
```
                     0                   1                   2  \
A                  2.5                 2.5                 2.5
B  2022-01-14 00:00:00  2022-01-14 00:00:00  2022-01-14 00:00:00
C                  1.0                 1.0                 1.0
D                    3                   3                   3
E                  boy                baba        sakht londay
F                Males               Males               Males

                     3
A                  2.5
B  2022-01-14 00:00:00
C                  1.0
D                    3
E            sigma male
F                Males
```

3. Sorting (Index Based) Row / Column heads only

Sorting Ascending/Descending Row index (row head) Wise

[ ]:
```
df.sort_index(axis=0, ascending=False)
```

[ ]:
```
                   A         B         C         D
2022-01-20  1.019831 -0.298755 -0.044523  0.686477
2022-01-19  0.062309 -1.656761  0.531129 -1.674739
2022-01-18  1.251463  1.940221 -1.155982 -1.181810
2022-01-17 -1.566494  0.545225 -0.388877 -0.317478
2022-01-16 -1.602749  0.251363  0.271529  2.246843
2022-01-15  1.275653 -0.101605  0.780956  0.675729
2022-01-14  1.265377  0.554316  1.188476 -2.156915
2022-01-13  0.074566 -0.125679  1.443004  0.034680
2022-01-12 -0.083672 -1.471417 -0.039479  0.111297
2022-01-11  0.550546 -0.565107 -0.085478 -0.423174
2022-01-10 -0.557112 -0.273150  2.068674  0.741978
2022-01-09  0.396088  1.338157 -0.759130  1.828155
2022-01-08 -0.377271  0.510839  0.041828 -0.060369
2022-01-07  0.838272  0.111321 -0.809320 -0.475192
2022-01-06 -1.302253 -1.658869 -1.811578  0.671019
2022-01-05  0.689037 -0.448571 -0.712814  0.695055
2022-01-04 -0.493067 -0.660789 -0.601053 -1.287295
2022-01-03 -0.040778  1.192630  0.935529 -1.628175
2022-01-02  0.470514  0.100841  1.017651 -1.245134
2022-01-01 -0.807823 -0.067819 -1.330167  0.074816
```

[ ]:
```
df.sort_index(axis=0, ascending=True)
```

[ ]:
```
                   A         B         C         D
2022-01-01 -0.807823 -0.067819 -1.330167  0.074816
```

```
2022-01-02  0.470514  0.100841  1.017651 -1.245134
2022-01-03 -0.040778  1.192630  0.935529 -1.628175
2022-01-04 -0.493067 -0.660789 -0.601053 -1.287295
2022-01-05  0.689037 -0.448571 -0.712814  0.695055
2022-01-06 -1.302253 -1.658869 -1.811578  0.671019
2022-01-07  0.838272  0.111321 -0.809320 -0.475192
2022-01-08 -0.377271  0.510839  0.041828 -0.060369
2022-01-09  0.396088  1.338157 -0.759130  1.828155
2022-01-10 -0.557112 -0.273150  2.068674  0.741978
2022-01-11  0.550546 -0.565107 -0.085478 -0.423174
2022-01-12 -0.083672 -1.471417 -0.039479  0.111297
2022-01-13  0.074566 -0.125679  1.443004  0.034680
2022-01-14  1.265377  0.554316  1.188476 -2.156915
2022-01-15  1.275653 -0.101605  0.780956  0.675729
2022-01-16 -1.602749  0.251363  0.271529  2.246843
2022-01-17 -1.566494  0.545225 -0.388877 -0.317478
2022-01-18  1.251463  1.940221 -1.155982 -1.181810
2022-01-19  0.062309 -1.656761  0.531129 -1.674739
2022-01-20  1.019831 -0.298755 -0.044523  0.686477
```

Sorting Ascending/Descending (Only Column Heads) not Values

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

[ ]:
```
                   D         C         B         A
2022-01-01  0.074816 -1.330167 -0.067819 -0.807823
2022-01-02 -1.245134  1.017651  0.100841  0.470514
2022-01-03 -1.628175  0.935529  1.192630 -0.040778
2022-01-04 -1.287295 -0.601053 -0.660789 -0.493067
2022-01-05  0.695055 -0.712814 -0.448571  0.689037
2022-01-06  0.671019 -1.811578 -1.658869 -1.302253
2022-01-07 -0.475192 -0.809320  0.111321  0.838272
2022-01-08 -0.060369  0.041828  0.510839 -0.377271
2022-01-09  1.828155 -0.759130  1.338157  0.396088
2022-01-10  0.741978  2.068674 -0.273150 -0.557112
2022-01-11 -0.423174 -0.085478 -0.565107  0.550546
2022-01-12  0.111297 -0.039479 -1.471417 -0.083672
2022-01-13  0.034680  1.443004 -0.125679  0.074566
2022-01-14 -2.156915  1.188476  0.554316  1.265377
2022-01-15  0.675729  0.780956 -0.101605  1.275653
2022-01-16  2.246843  0.271529  0.251363 -1.602749
2022-01-17 -0.317478 -0.388877  0.545225 -1.566494
2022-01-18 -1.181810 -1.155982  1.940221  1.251463
2022-01-19 -1.674739  0.531129 -1.656761  0.062309
2022-01-20  0.686477 -0.044523 -0.298755  1.019831
```

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

[ ]:

	A	B	C	D
2022-01-01	-0.807823	-0.067819	-1.330167	0.074816
2022-01-02	0.470514	0.100841	1.017651	-1.245134
2022-01-03	-0.040778	1.192630	0.935529	-1.628175
2022-01-04	-0.493067	-0.660789	-0.601053	-1.287295
2022-01-05	0.689037	-0.448571	-0.712814	0.695055
2022-01-06	-1.302253	-1.658869	-1.811578	0.671019
2022-01-07	0.838272	0.111321	-0.809320	-0.475192
2022-01-08	-0.377271	0.510839	0.041828	-0.060369
2022-01-09	0.396088	1.338157	-0.759130	1.828155
2022-01-10	-0.557112	-0.273150	2.068674	0.741978
2022-01-11	0.550546	-0.565107	-0.085478	-0.423174
2022-01-12	-0.083672	-1.471417	-0.039479	0.111297
2022-01-13	0.074566	-0.125679	1.443004	0.034680
2022-01-14	1.265377	0.554316	1.188476	-2.156915
2022-01-15	1.275653	-0.101605	0.780956	0.675729
2022-01-16	-1.602749	0.251363	0.271529	2.246843
2022-01-17	-1.566494	0.545225	-0.388877	-0.317478
2022-01-18	1.251463	1.940221	-1.155982	-1.181810
2022-01-19	0.062309	-1.656761	0.531129	-1.674739
2022-01-20	1.019831	-0.298755	-0.044523	0.686477

Sorting a specified Column of Data Frame sorting its values

[ ]:
```
df.sort_values('B',axis=0, ascending=True )
```

[ ]:

	A	B	C	D
2022-01-06	-1.302253	-1.658869	-1.811578	0.671019
2022-01-19	0.062309	-1.656761	0.531129	-1.674739
2022-01-12	-0.083672	-1.471417	-0.039479	0.111297
2022-01-04	-0.493067	-0.660789	-0.601053	-1.287295
2022-01-11	0.550546	-0.565107	-0.085478	-0.423174
2022-01-05	0.689037	-0.448571	-0.712814	0.695055
2022-01-20	1.019831	-0.298755	-0.044523	0.686477
2022-01-10	-0.557112	-0.273150	2.068674	0.741978
2022-01-13	0.074566	-0.125679	1.443004	0.034680
2022-01-15	1.275653	-0.101605	0.780956	0.675729
2022-01-01	-0.807823	-0.067819	-1.330167	0.074816
2022-01-02	0.470514	0.100841	1.017651	-1.245134
2022-01-07	0.838272	0.111321	-0.809320	-0.475192
2022-01-16	-1.602749	0.251363	0.271529	2.246843
2022-01-08	-0.377271	0.510839	0.041828	-0.060369
2022-01-17	-1.566494	0.545225	-0.388877	-0.317478
2022-01-14	1.265377	0.554316	1.188476	-2.156915
2022-01-03	-0.040778	1.192630	0.935529	-1.628175
2022-01-09	0.396088	1.338157	-0.759130	1.828155
2022-01-18	1.251463	1.940221	-1.155982	-1.181810

[ ]:
```
df.sort_values(by=['B', 'A'])
```

[ ]:
```
                   A         B         C         D
2022-01-06 -1.302253 -1.658869 -1.811578  0.671019
2022-01-19  0.062309 -1.656761  0.531129 -1.674739
2022-01-12 -0.083672 -1.471417 -0.039479  0.111297
2022-01-04 -0.493067 -0.660789 -0.601053 -1.287295
2022-01-11  0.550546 -0.565107 -0.085478 -0.423174
2022-01-05  0.689037 -0.448571 -0.712814  0.695055
2022-01-20  1.019831 -0.298755 -0.044523  0.686477
2022-01-10 -0.557112 -0.273150  2.068674  0.741978
2022-01-13  0.074566 -0.125679  1.443004  0.034680
2022-01-15  1.275653 -0.101605  0.780956  0.675729
2022-01-01 -0.807823 -0.067819 -1.330167  0.074816
2022-01-02  0.470514  0.100841  1.017651 -1.245134
2022-01-07  0.838272  0.111321 -0.809320 -0.475192
2022-01-16 -1.602749  0.251363  0.271529  2.246843
2022-01-08 -0.377271  0.510839  0.041828 -0.060369
2022-01-17 -1.566494  0.545225 -0.388877 -0.317478
2022-01-14  1.265377  0.554316  1.188476 -2.156915
2022-01-03 -0.040778  1.192630  0.935529 -1.628175
2022-01-09  0.396088  1.338157 -0.759130  1.828155
2022-01-18  1.251463  1.940221 -1.155982 -1.181810
```

19.3 4. Displaying Data in a Data frames

To Display an entire column

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

[ ]:
```
2022-01-01   -0.807823
2022-01-02    0.470514
2022-01-03   -0.040778
2022-01-04   -0.493067
2022-01-05    0.689037
2022-01-06   -1.302253
2022-01-07    0.838272
2022-01-08   -0.377271
2022-01-09    0.396088
2022-01-10   -0.557112
2022-01-11    0.550546
2022-01-12   -0.083672
2022-01-13    0.074566
2022-01-14    1.265377
2022-01-15    1.275653
2022-01-16   -1.602749
2022-01-17   -1.566494
2022-01-18    1.251463
```

```
2022-01-19    0.062309
2022-01-20    1.019831
Freq: D, Name: A, dtype: float64
```

To display selected rows

[ ]:
```
df[0:2]
```

[ ]:
```
                   A         B         C         D
2022-01-01 -0.807823 -0.067819 -1.330167  0.074816
2022-01-02  0.470514  0.100841  1.017651 -1.245134
```

[ ]:
```
# 2 indicates starting row for frames and 10 will print 10 index values
df[2:10]
```

[ ]:
```
                   A         B         C         D
2022-01-03 -0.040778  1.192630  0.935529 -1.628175
2022-01-04 -0.493067 -0.660789 -0.601053 -1.287295
2022-01-05  0.689037 -0.448571 -0.712814  0.695055
2022-01-06 -1.302253 -1.658869 -1.811578  0.671019
2022-01-07  0.838272  0.111321 -0.809320 -0.475192
2022-01-08 -0.377271  0.510839  0.041828 -0.060369
2022-01-09  0.396088  1.338157 -0.759130  1.828155
2022-01-10 -0.557112 -0.273150  2.068674  0.741978
```

Reaching a specific value in Table (Interpret it as 2D array indexing)

[ ]:
```
df.at[dates[5],"C"]
```

[ ]:
```
-1.8115779218408021
```

19.4 5. Targeted Index:Column data Filtration using Loc and ILoc functions

The main distinction between loc and iloc is: loc is label-based, which means that you have to specify rows and columns based on their row and column labels. iloc is integer position-based, so you have to specify rows and columns by their integer position values (0-based integer position).

This displays row 5 column values in vertical order

[ ]:
```
# row 5 ka column A,B,C,D parameters have been generated
df.loc[dates[5]]
```

[ ]:
```
A   -1.302253
B   -1.658869
C   -1.811578
D    0.671019
Name: 2022-01-06 00:00:00, dtype: float64
```

Display chunk of Data using loc command
limited operation on Columns as range cannot be defined like iloc

```
[ ]: # row index (3 to 6) par only column A and B displayed
     df.loc[dates[3:6],["A","C"]]
```

```
[ ]:                    A         C
     2022-01-04 -0.493067 -0.601053
     2022-01-05  0.689037 -0.712814
     2022-01-06 -1.302253 -1.811578
```

```
[ ]: # specific row and specific column
     df.loc[["20220105","20220107"],["A","C"]]
```

```
[ ]:                    A         C
     2022-01-05  0.689037 -0.712814
     2022-01-07  0.838272 -0.809320
```

Display chunk of Data using iloc command
Independant operation on Columns as range can be defined

```
[ ]: # another way of targeted filtration (row x column filters)
     # row bhe limited and coolumn bhe limited
     df.iloc[3:10,1:4]
```

```
[ ]:                    B         C         D
     2022-01-04 -0.660789 -0.601053 -1.287295
     2022-01-05 -0.448571 -0.712814  0.695055
     2022-01-06 -1.658869 -1.811578  0.671019
     2022-01-07  0.111321 -0.809320 -0.475192
     2022-01-08  0.510839  0.041828 -0.060369
     2022-01-09  1.338157 -0.759130  1.828155
     2022-01-10 -0.273150  2.068674  0.741978
```

Reaching a specific value in Table (Interpret it as 2D array indexing)

```
[ ]: df.at[dates[5],"C"]
```

```
[ ]: -1.8115779218408021
```

19.5 6. Condition (<,>) Checking

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

```
[ ]: 2022-01-01    False
     2022-01-02    False
     2022-01-03    False
     2022-01-04    False
     2022-01-05    False
     2022-01-06    False
     2022-01-07    False
     2022-01-08    False
```

```
2022-01-09    False
2022-01-10    False
2022-01-11    False
2022-01-12    False
2022-01-13    False
2022-01-14    False
2022-01-15    False
2022-01-16    False
2022-01-17    False
2022-01-18    False
2022-01-19    False
2022-01-20    False
Freq: D, Name: A, dtype: bool
```

19.5.1 6a. This is most important

you were facing error becoz of column ki data type and tmhe is se related google par bhe kuch nh mila tu ye yaad rakho

To sort column on the basis of a condition applied on a specific Column

[ ]:
```
df[df["A"] >0.1 ]
```

[ ]:
```
                   A         B         C         D
2022-01-02  0.470514  0.100841  1.017651 -1.245134
2022-01-05  0.689037 -0.448571 -0.712814  0.695055
2022-01-07  0.838272  0.111321 -0.809320 -0.475192
2022-01-09  0.396088  1.338157 -0.759130  1.828155
2022-01-11  0.550546 -0.565107 -0.085478 -0.423174
2022-01-14  1.265377  0.554316  1.188476 -2.156915
2022-01-15  1.275653 -0.101605  0.780956  0.675729
2022-01-18  1.251463  1.940221 -1.155982 -1.181810
2022-01-20  1.019831 -0.298755 -0.044523  0.686477
```

To ***display certain columns based on condition applied on another column***

[ ]:
```
criterion = df['A'].map(lambda x: x>0)
df.loc[criterion & (df['B'] > 0.3), 'C':'D']
```

[ ]:
```
                   C         D
2022-01-09 -0.759130  1.828155
2022-01-14  1.188476 -2.156915
2022-01-18 -1.155982 -1.181810
```

Hamesha yaad rakhna Saad k jab bhe bool milen tu loc se khel k real value get karna hay

To check multiple condition and display multiple values

[ ]:
```
s = (df['A'] > 0) & (df['B'] > 0)
print(s)
print("\n \n The sorted value that satisifies condition in A is")
e=df.loc[s]

df.loc[s,'A']
```

```
2022-01-01    False
2022-01-02     True
2022-01-03    False
2022-01-04    False
2022-01-05    False
2022-01-06    False
2022-01-07     True
2022-01-08    False
2022-01-09     True
2022-01-10    False
2022-01-11    False
2022-01-12    False
2022-01-13    False
2022-01-14     True
2022-01-15    False
2022-01-16    False
2022-01-17    False
2022-01-18     True
2022-01-19    False
2022-01-20    False
Freq: D, dtype: bool


 The sorted value that satisifies condition in A is
```

[ ]:
```
2022-01-02    0.470514
2022-01-07    0.838272
2022-01-09    0.396088
2022-01-14    1.265377
2022-01-18    1.251463
Name: A, dtype: float64
```

[ ]:
```
e
```

[ ]:
```
                   A         B         C         D
2022-01-02  0.470514  0.100841  1.017651 -1.245134
2022-01-07  0.838272  0.111321 -0.809320 -0.475192
2022-01-09  0.396088  1.338157 -0.759130  1.828155
2022-01-14  1.265377  0.554316  1.188476 -2.156915
2022-01-18  1.251463  1.940221 -1.155982 -1.181810
```

To find values greater or less than a specific number

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

```
[ ]:                   A         B         C         D
     2022-01-01       NaN       NaN       NaN  0.074816
     2022-01-02  0.470514  0.100841  1.017651       NaN
     2022-01-03       NaN  1.192630  0.935529       NaN
     2022-01-04       NaN       NaN       NaN       NaN
     2022-01-05  0.689037       NaN       NaN  0.695055
     2022-01-06       NaN       NaN       NaN  0.671019
     2022-01-07  0.838272  0.111321       NaN       NaN
     2022-01-08       NaN  0.510839  0.041828       NaN
     2022-01-09  0.396088  1.338157       NaN  1.828155
     2022-01-10       NaN       NaN  2.068674  0.741978
     2022-01-11  0.550546       NaN       NaN       NaN
     2022-01-12       NaN       NaN       NaN  0.111297
     2022-01-13  0.074566       NaN  1.443004  0.034680
     2022-01-14  1.265377  0.554316  1.188476       NaN
     2022-01-15  1.275653       NaN  0.780956  0.675729
     2022-01-16       NaN  0.251363  0.271529  2.246843
     2022-01-17       NaN  0.545225       NaN       NaN
     2022-01-18  1.251463  1.940221       NaN       NaN
     2022-01-19  0.062309       NaN  0.531129       NaN
     2022-01-20  1.019831       NaN       NaN  0.686477
```

19.6 7. Adding/ Removing Data Columns and Recreating New Data Frame

Creating a new DF with old data frame and appending a new column

```
[ ]: df3 = df.copy()
     df3["E"]=["one","two","three","four","five",
     "one","two","three","four","five","one","two","three","four","five","one","two","three","four"
     df3
```

```
[ ]:                   A         B         C         D      E
     2022-01-01 -0.807823 -0.067819 -1.330167  0.074816    one
     2022-01-02  0.470514  0.100841  1.017651 -1.245134    two
     2022-01-03 -0.040778  1.192630  0.935529 -1.628175  three
     2022-01-04 -0.493067 -0.660789 -0.601053 -1.287295   four
     2022-01-05  0.689037 -0.448571 -0.712814  0.695055   five
     2022-01-06 -1.302253 -1.658869 -1.811578  0.671019    one
     2022-01-07  0.838272  0.111321 -0.809320 -0.475192    two
     2022-01-08 -0.377271  0.510839  0.041828 -0.060369  three
     2022-01-09  0.396088  1.338157 -0.759130  1.828155   four
     2022-01-10 -0.557112 -0.273150  2.068674  0.741978   five
     2022-01-11  0.550546 -0.565107 -0.085478 -0.423174    one
     2022-01-12 -0.083672 -1.471417 -0.039479  0.111297    two
```

```
2022-01-13  0.074566 -0.125679  1.443004  0.034680  three
2022-01-14  1.265377  0.554316  1.188476 -2.156915   four
2022-01-15  1.275653 -0.101605  0.780956  0.675729   five
2022-01-16 -1.602749  0.251363  0.271529  2.246843    one
2022-01-17 -1.566494  0.545225 -0.388877 -0.317478    two
2022-01-18  1.251463  1.940221 -1.155982 -1.181810  three
2022-01-19  0.062309 -1.656761  0.531129 -1.674739   four
2022-01-20  1.019831 -0.298755 -0.044523  0.686477   five
```

Creating a reduced DF from a existing long data frame (data set)

[ ]:
```
df4=df3.iloc[:,0:4]
df4
```

[ ]:
```
                   A         B         C         D
2022-01-01 -0.807823 -0.067819 -1.330167  0.074816
2022-01-02  0.470514  0.100841  1.017651 -1.245134
2022-01-03 -0.040778  1.192630  0.935529 -1.628175
2022-01-04 -0.493067 -0.660789 -0.601053 -1.287295
2022-01-05  0.689037 -0.448571 -0.712814  0.695055
2022-01-06 -1.302253 -1.658869 -1.811578  0.671019
2022-01-07  0.838272  0.111321 -0.809320 -0.475192
2022-01-08 -0.377271  0.510839  0.041828 -0.060369
2022-01-09  0.396088  1.338157 -0.759130  1.828155
2022-01-10 -0.557112 -0.273150  2.068674  0.741978
2022-01-11  0.550546 -0.565107 -0.085478 -0.423174
2022-01-12 -0.083672 -1.471417 -0.039479  0.111297
2022-01-13  0.074566 -0.125679  1.443004  0.034680
2022-01-14  1.265377  0.554316  1.188476 -2.156915
2022-01-15  1.275653 -0.101605  0.780956  0.675729
2022-01-16 -1.602749  0.251363  0.271529  2.246843
2022-01-17 -1.566494  0.545225 -0.388877 -0.317478
2022-01-18  1.251463  1.940221 -1.155982 -1.181810
2022-01-19  0.062309 -1.656761  0.531129 -1.674739
2022-01-20  1.019831 -0.298755 -0.044523  0.686477
```

Calculating Mean on selected columns and generating a new Column (Assignment Qs)

[ ]:
```
df3['average'] = df3.iloc[:, [0,1,2,3]].mean(axis=1)
df3
```

[ ]:
```
                   A         B         C         D      E   average
2022-01-01 -0.807823 -0.067819 -1.330167  0.074816    one -0.532748
2022-01-02  0.470514  0.100841  1.017651 -1.245134    two  0.085968
2022-01-03 -0.040778  1.192630  0.935529 -1.628175  three  0.114802
2022-01-04 -0.493067 -0.660789 -0.601053 -1.287295   four -0.760551
2022-01-05  0.689037 -0.448571 -0.712814  0.695055   five  0.055677
```

```
2022-01-06 -1.302253 -1.658869 -1.811578  0.671019    one -1.025420
2022-01-07  0.838272  0.111321 -0.809320 -0.475192    two -0.083730
2022-01-08 -0.377271  0.510839  0.041828 -0.060369  three  0.028757
2022-01-09  0.396088  1.338157 -0.759130  1.828155   four  0.700818
2022-01-10 -0.557112 -0.273150  2.068674  0.741978   five  0.495097
2022-01-11  0.550546 -0.565107 -0.085478 -0.423174    one -0.130803
2022-01-12 -0.083672 -1.471417 -0.039479  0.111297    two -0.370818
2022-01-13  0.074566 -0.125679  1.443004  0.034680  three  0.356643
2022-01-14  1.265377  0.554316  1.188476 -2.156915   four  0.212814
2022-01-15  1.275653 -0.101605  0.780956  0.675729   five  0.657683
2022-01-16 -1.602749  0.251363  0.271529  2.246843    one  0.291747
2022-01-17 -1.566494  0.545225 -0.388877 -0.317478    two -0.431906
2022-01-18  1.251463  1.940221 -1.155982 -1.181810  three  0.213473
2022-01-19  0.062309 -1.656761  0.531129 -1.674739   four -0.684515
2022-01-20  1.019831 -0.298755 -0.044523  0.686477   five  0.340758
```

Appending one Data frame into another DF1 into DF2

[ ]:
```
df10 = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df11 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])

df10 = df10.append(df11)
print(df10)
```

```
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
```

[ ]:
```
df10 = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df11 = pd.DataFrame([[5, 6], [7, 8]], columns = ['c','d'])

df10 = df10.append(df11)
print(df10)
```

```
     a    b    c    d
0  1.0  2.0  NaN  NaN
1  3.0  4.0  NaN  NaN
0  NaN  NaN  5.0  6.0
1  NaN  NaN  7.0  8.0
```

[ ]:
```
df10 = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df11 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','c'])

df10 = df10.append(df11)
print(df10)
```

```
   a    b    c
```

```
0  1  2.0  NaN
1  3  4.0  NaN
0  5  NaN  6.0
1  7  NaN  8.0
```

19.7 8. Other Functions (Delete/Pop/Drop)

Deleting a column using del and POP command

[ ]:
```
# Using the previous DataFrame, we will delete a column
# using del function
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd']),
   'three' : pd.Series([10,20,30], index=['a','b','c'])}

df9 = pd.DataFrame(d)
print ("Our dataframe is:")
print(df9)

# using del function
print ("Deleting the first column using DEL function:")
del df9['one']
print(df9)

# using pop function
print ("Deleting another column using POP function:")
df9.pop('two')
print(df9)
```

```
Our dataframe is:
   one  two  three
a  1.0    1   10.0
b  2.0    2   20.0
c  3.0    3   30.0
d  NaN    4    NaN
Deleting the first column using DEL function:
   two  three
a    1   10.0
b    2   20.0
c    3   30.0
d    4    NaN
Deleting another column using POP function:
   three
a   10.0
b   20.0
```

```
c   30.0
d    NaN
```

Drop or delete a specific row

```
[ ]: df10 = df10.drop(0)
     print(df10)
```

```
   a    b    c
1  3  4.0  NaN
1  7  NaN  8.0
```

20 Exploratory Data Analysis

20.1 Three important things to keep in mind are

1. Understand the data
2. Clean the data.
3. Find a relationship between data

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

Load dataset into Kashti and make a csv file

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

```
[ ]: kashti.to_csv("kashti.cv")
```

Finding Info of Data Frame (Null value, Data type)

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

```
[ ]: ks = kashti
     ks.head()
```

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

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

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

20.1.1 Finding Unique values in a Column

unique values in a single column

unique values in multiple column

[ ]:
```
#unique values
ks["survived"].unique()
```

Unique no of values in each column in a dataset without Unique values

[ ]:
```
ks.nunique()
```

Ravel Function (I think u dont need to use it)

sex who and class k unique categories again

[ ]:
```
col1=ks[['sex','who','class']].values.ravel()
col2=pd.unique(col1)
print(col2)
```

[ ]:
```
ks[["who","survived","age","fare"]].nunique()
```

[ ]:
```
for i in ks.columns:
    print(ks[i].unique())
```

[ ]:
```
ks.columns
```

20.2 Data Cleaning and Filtration

Finding null values in every column of a dataset

[ ]:
```
## Cleaning and Filtering the Data
ks.isnull().sum()
```

Removing Missing values

Dropped deck column here

[ ]:
```
# removing missing values
ks_clean = ks.drop(['deck'],axis=1)
```

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

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

Dropping all null values

[ ]:
```
ks_clean =ks_clean.dropna()
```

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

checking if null values exists

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

Checking age column and seeing how many numbers exists for unique values

[ ]:
```
ks_clean['age'].value_counts()
```

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

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

Boxplot

Removing Outliers

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

[ ]:
```
#age me masla hay
sns.boxplot(y='age',data=ks_clean)
```

Normality Check

[ ]:
```
#Normality check.
# remove outliers to obtain perfect bell curve
sns.distplot(ks_clean['age'])
```

20.2.1 Removing Outliers

[ ]:
```
# out liers removal
ks_clean['age'].mean()
```

Removing age values less than 68

[ ]:
```
ks_clean = ks_clean[ks_clean['age']< 68]
ks_clean.tail()
```

[ ]:
```
#ks_clean = ks_clean[ks_clean['age']< 68].mean()
```

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

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

[ ]:
```
#after age  me issue resolving
sns.boxplot(y='age',data=ks_clean)
```

[ ]:
```
#Normality check.
# remove outliers to obtain perfect bell curve
sns.distplot(ks_clean['age'])
```

Box plot for all Column heads

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

Cleaning Fare Column

[ ]:
```
ks_clean = ks_clean[ks_clean['fare']<200]
ks_clean.boxplot()
```

Dist plot (histogram on Fare data)

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

Hist plot on every column Instance

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

Count bar plot on Class

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

Group kar k Data categorize karna

[ ]:
```
# column hamesha square braces
 #. k bad wale chez me hamesha round braces
ks_clean.groupby(['sex','class','who']).mean()
```

20.3 Relationship (Corelation and Heatmap)

[ ]:
```
cor_ks_clean =ks_clean.corr()
```

[ ]:
```
sns.heatmap(cor_ks_clean)
```

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

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

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

Concept of log scale to Remove outliers

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

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

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

Removing last Column

[ ]:
```
vv=ks_clean.iloc[:,:-2]
```

[ ]:
```

```

21 Data Wrangling

[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
#load dataset
kashti = sns.load_dataset('titanic')
#saving data set into two variable
ks1 = sns.load_dataset('titanic')
#ks2 = kashti
kashti.head(2)
```

[ ]:
```
   survived  pclass     sex  age  sibsp  parch  fare embarked  class    who  \
0         0       3    male   22      1      0     7        S  Third    man
1         1       1  female   38      1      0    71        C  First  woman

   adult_male deck  embark_town alive  alone
0        True  NaN  Southampton    no  False
1       False    C    Cherbourg   yes  False
```

[ ]:
```
# simple math operation on a series
(kashti['age']+12).head(2)
```

[ ]:
```
0    34
1    50
Name: age, dtype: float64
```

21.1 Dealing with Missing Values

- In a dataset missing values are either ? or NA or NAN or 0 or a blank cell
- Jab data na ho kisi row me kisi bhi ek parameter ka

  Steps: 1. Try recollecting data and check for mistakes. 2. Try to remove missing entries column or remove that entire row 3. Replace the missing values * How ? * Take average value of dat entire data row (column) and substitute null values * Frequency or Mode replacement * Replace based on other functions (Data sampler knows that) * ML algorithms can also be used (like age se salary predict mising) * Leave it like that * Why we deal with the missing values * It is better because no data is lost * Less accurate

[ ]:
```
# where exactly missing values are
kashti.isnull().sum()
```

[ ]:
```
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
```

[ ]:
```
# use drop.na method
print(kashti.shape)
kashti.dropna(subset=["deck"],axis=0, inplace=True)
# this will remove specifically rows of deck with 0 values
#inpace = True modifies the frame
```

```
(891, 15)
```

[ ]:
```
kashti.isnull().sum()
```

[ ]:
```
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
```

[ ]:
```
kashti = kashti.dropna()
kashti.dropna().isnull().sum()
```

[ ]:
```
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
```

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

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

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

[ ]:
```
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
```

21.2 Replacing missing Values with the average and Mode of that Column

[ ]:
```
# finding mean
mean_age =ks1['age'].mean()
```

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

ks1['deck'].fillna(ks1['deck'].mode()[0], inplace=True)
ks1['embark_town'].fillna(ks1['embark_town'].mode()[0], inplace=True)
ks1['embarked'].fillna(ks1['embarked'].mode()[0], inplace=True)

#ks1[['deck','embark_town']] = ks1[['age','embark_town']].replace(np.nan,mean)
```

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

[ ]:
```
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
```

21.3 Data Formatting

- Data ko aik common standard par rakhna
- Ensure data is consistent and understandable
  - Easy to gather
  - Easy to work with
    - Faisalabad (FSD)
    - Karachi (KHI)
    - Convert gm to kg or same unit for all.
    - one standard unit

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

[ ]:
```
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
```

[ ]:
```
# Convert data type of fixed column(series)     Type Casting
kashti['survived'] = kashti['survived'].astype('int64')
kashti.dtypes
```

[ ]:
```
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
```

[ ]:
```
# convert age into years
ks1['age'] = ks1['age'] * 365
#ks1['age'] = pd.set_option('precision', 0)
ks1.head(3)
```

[ ]:
```
   survived  pclass     sex    age  sibsp  parch  fare embarked  class    who  \
0         0       3    male   8030      1      0     7        S  Third    man
1         1       1  female  13870      1      0    71        C  First  woman
2         1       3  female   9490      0      0     8        S  Third  woman

   adult_male deck  embark_town alive  alone
0        True    C  Southampton    no  False
1       False    C    Cherbourg   yes  False
2       False    C  Southampton   yes   True
```

[ ]:
```
# Renaming    Columns
ks1.rename(columns={"age":"age in Days"},inplace=True)
ks1.head(2)
```

[ ]:
```
   survived  pclass     sex  age in Days  sibsp  parch  fare embarked  class  \
0         0       3    male         8030      1      0     7        S  Third
1         1       1  female        13870      1      0    71        C  First

     who  adult_male deck  embark_town alive  alone
```

```
0    man        True    C  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
```

21.4 Data Normalization

- uniform data
- They have same impact
- sea fish vs jar fish
- Also for computational reasons

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

[ ]:
```
   age in Days  fare
0         8030     7
1        13870    71
2         9490     8
3        12775    53
4        12775     8
```

1. The above data between fare and age in days is really in wide range. We need to N o r m a l i z e
2. Normalization changes the value to the range of 0 to 1. (both variable will have same influence)

21.4.1 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

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

```
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/1908861037.py:2:
SettingWithCopyWarning:
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\dell7450\AppData\Local\Temp/ipykernel_5824/1908861037.py:3:
```

```
SettingWithCopyWarning:
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()
```

[ ]:

	age in Days	fare
0	3e-01	1e-02
1	5e-01	1e-01
2	3e-01	2e-02
3	4e-01	1e-01
4	4e-01	2e-02

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

```
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/887406347.py:2:
SettingWithCopyWarning:
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'])
```

[ ]:

	age in Days	fare
0	3e-01	1e-02
1	5e-01	2e-01
2	3e-01	2e-02
3	4e-01	1e-01
4	4e-01	2e-02

[ ]:
```
# z score Method  R A N G E (0   to +3)
ks4['age in Days'] = (ks4['age in Days']-ks4['age in Days'].mean()) /( ks4['age␣
 ↪in Days'].std() )
ks4.head()
```

```
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/4054113253.py:2:
SettingWithCopyWarning:
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'].mean()) /(
ks4['age in Days'].std() )
```

[ ]:
```
   age in Days   fare
0       -6e-01  1e-02
1        6e-01  2e-01
2       -3e-01  2e-02
3        4e-01  1e-01
4        4e-01  2e-02
```

[ ]:
```
# 4. log transformation
ks4['fare'] = np.log(ks4['fare'])
ks4.head()
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\arraylike.py:364: RuntimeWarning: divide by zero
encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
C:\Users\dell7450\AppData\Local\Temp/ipykernel_5824/2813506387.py:2:
SettingWithCopyWarning:
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'] = np.log(ks4['fare'])
```

[ ]:
```
   age in Days  fare
0       -6e-01    -4
1        6e-01    -2
2       -3e-01    -4
3        4e-01    -2
4        4e-01    -4
```

21.5 Binning

1. Grouping of values into small set of values (groups)
2. convert numeric into categories
    1. for example: age (0-10) = bachay 2. age (10-20) = jawan 3. age (30-40) borhay
3. To have better understanding of groups
    1. low vs mid vs high prices

[ ]:
```
ks1.sort_values("age in Days")
```

[ ]:
```
     survived  pclass     sex  age in Days  sibsp  parch  fare embarked  \
803         1       3    male          153      0      1     9        C
755         1       2    male          245      1      1    14        S
644         1       3  female          274      2      1    19        C
469         1       3  female          274      2      1    19        C
```

```
831         1       2    male          303      1      1    19        S
..        …     …      …           …    …    …    …       …
116         0       3    male        25732      0      0     8        Q
96          0       1    male        25915      0      0    35        C
493         0       1    male        25915      0      0    50        C
851         0       3    male        27010      0      0     8        S
630         1       1    male        29200      0      0    30        S

      class    who  adult_male deck  embark_town alive  alone
803   Third  child       False    C    Cherbourg   yes  False
755  Second  child       False    C  Southampton   yes  False
644   Third  child       False    C    Cherbourg   yes  False
469   Third  child       False    C    Cherbourg   yes  False
831  Second  child       False    C  Southampton   yes  False
..      …    …         …  …           …   …    …
116   Third    man        True    C   Queenstown    no   True
96    First    man        True    A    Cherbourg    no   True
493   First    man        True    C    Cherbourg    no   True
851   Third    man        True    C  Southampton    no   True
630   First    man        True    A  Southampton   yes   True

[891 rows x 15 columns]
```

[ ]:
```
# bins = np.linspace(min(ks1['age in Days']), max(ks1['age in Days']) , 29200)
# age_groups = ["Bachay","Jawaan","Boorhay"]
# ks1['age in Days']=pd.cut(ks1['age in Days'],bins, labels=age_groups,␣
 ↪include_lowest=True)
# ks1['age in Days']
```

[ ]:
```
kashti["age_bin"] = pd.cut(kashti["age"],bins=[0,2,17,65,99],
   labels=['Toddler/baby','Child','Adult','Elderly'])
```

[ ]:
```
kashti
```

[ ]:
```
     survived  pclass     sex  age  sibsp  parch  fare embarked  class    who  \
1           1       1  female   38      1      0    71        C  First  woman
3           1       1  female   35      1      0    53        S  First  woman
6           0       1    male   54      0      0    52        S  First    man
10          1       3  female    4      1      1    17        S  Third  child
11          1       1  female   58      0      0    27        S  First  woman
..        …     …      …  …    …    …    …       …    …    …
871         1       1  female   47      1      1    53        S  First  woman
872         0       1    male   33      0      0     5        S  First    man
879         1       1  female   56      0      1    83        C  First  woman
887         1       1  female   19      0      0    30        S  First  woman
889         1       1    male   26      0      0    30        C  First    man
```

```
     adult_male deck  embark_town alive  alone age_bin
1         False    C    Cherbourg   yes  False   Adult
3         False    C  Southampton   yes  False   Adult
6          True    E  Southampton    no   True   Adult
10        False    G  Southampton   yes  False   Child
11        False    C  Southampton   yes   True   Adult
..          …  …          …    …    …      …
871       False    D  Southampton   yes  False   Adult
872        True    B  Southampton    no   True   Adult
879       False    C    Cherbourg   yes  False   Adult
887       False    B  Southampton   yes   True   Adult
889        True    C    Cherbourg   yes   True   Adult

[182 rows x 16 columns]
```

21.6 Dummies

[ ]:
```
ks1
```

[ ]:
```
     survived  pclass     sex  age in Days  sibsp  parch  fare embarked  \
0           0       3    male         8030      1      0     7        S
1           1       1  female        13870      1      0    71        C
2           1       3  female         9490      0      0     8        S
3           1       1  female        12775      1      0    53        S
4           0       3    male        12775      0      0     8        S
..        …     …      …           …    …    …      …
886         0       2    male         9855      0      0    13        S
887         1       1  female         6935      0      0    30        S
888         0       3  female        10840      1      2    23        S
889         1       1    male         9490      0      0    30        C
890         0       3    male        11680      0      0     8        Q

      class    who  adult_male deck  embark_town alive  alone
0     Third    man        True    C  Southampton    no  False
1     First  woman       False    C    Cherbourg   yes  False
2     Third  woman       False    C  Southampton   yes   True
3     First  woman       False    C  Southampton   yes  False
4     Third    man        True    C  Southampton    no   True
..      …    …          …  …          …    …    …
886  Second    man        True    C  Southampton    no   True
887   First  woman       False    B  Southampton   yes   True
888   Third  woman       False    C  Southampton    no  False
889   First    man        True    C    Cherbourg   yes   True
890   Third    man        True    C   Queenstown    no   True

[891 rows x 15 columns]
```

```
[ ]: # converting categories to dummy values
     pd.get_dummies(ks1['sex'])
```

```
[ ]:      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 x 2 columns]
```

```
[ ]: ks1 =pd.concat([ks1, pd.get_dummies(ks1['sex'])], axis=1)
     ks1 =ks1.drop("sex", axis=1)      #####
```

```
[ ]: ks1
```

```
[ ]:      survived  pclass  age in Days  sibsp  parch  fare embarked   class  \
     0           0       3         8030      1      0     7        S   Third
     1           1       1        13870      1      0    71        C   First
     2           1       3         9490      0      0     8        S   Third
     3           1       1        12775      1      0    53        S   First
     4           0       3        12775      0      0     8        S   Third
     ..        …     …           …    …    …   …       …      …
     886         0       2         9855      0      0    13        S  Second
     887         1       1         6935      0      0    30        S   First
     888         0       3        10840      1      2    23        S   Third
     889         1       1         9490      0      0    30        C   First
     890         0       3        11680      0      0     8        Q   Third

            who  adult_male deck  embark_town alive  alone  female  male
     0      man        True    C  Southampton    no  False       0     1
     1    woman       False    C    Cherbourg   yes  False       1     0
     2    woman       False    C  Southampton   yes   True       1     0
     3    woman       False    C  Southampton   yes  False       1     0
     4      man        True    C  Southampton    no   True       0     1
     ..     …          …  …          …    …    …      …    …
     886    man        True    C  Southampton    no   True       0     1
     887  woman       False    B  Southampton   yes   True       1     0
     888  woman       False    C  Southampton    no  False       1     0
     889    man        True    C    Cherbourg   yes   True       0     1
```

```
890    man        True    C    Queenstown    no    True       0      1

[891 rows x 16 columns]
```

22 Machine Learning Algorithms

22.0.1 Install libraries

- Use pip if you are uisng windows
- Use pip3 if you are using macOS

[ ]:
```
#pip install numpy
#pip install pandas
#pip install scikit-learn
```

22.0.2 Import Libraries

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

22.0.3 Load Dataset

- It is better to keep the dataset in the same folder in which you have your notebook, otherwise you have to enter the complete path

[ ]:
```
# load dataset
df = pd.read_csv("mldata.csv")
df.head(2)
```

[ ]:
```
   age  weight  height
0   27      76     171
1   41      70     165
```

[ ]:
```
# Take relevant data
workshop_data = df[["age","weight","height"]]
workshop_data.head(2)
```

[ ]:
```
   age  weight  height
0   27      76     171
1   41      70     165
```

[ ]:
```
#X = workshop_data.iloc[:, :-1].values.reshape(-1,2) #get a copy of dataset␣
 ↪exclude last column
#y = workshop_data.iloc[:, 2:3].values #get array of dataset in column 1st
X=workshop_data[["age","weight"]]
y=workshop_data["height"]
```

[ ]:
```
X
#y.reshape(-1, 1)
y
```

[ ]:
```
0      171
1      165
2      171
3      164
4      174
      …
190    165
191    160
192    172
193    178
194    157
Name: height, Length: 195, dtype: int64
```

[ ]:
```
X
```

[ ]:
```
     age  weight
0     27      76
1     41      70
2     29      80
3     29      67
4     28      46
..   …      …
190   27      63
191   31      60
192   26      70
193   40      80
194   33      56

[195 rows x 2 columns]
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2,␣
↪random_state=0)
```

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

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

[ ]:
```
# Predicting the Test set results
y_pred = regressor.predict(X_test)
y_pred
```

[ ]:

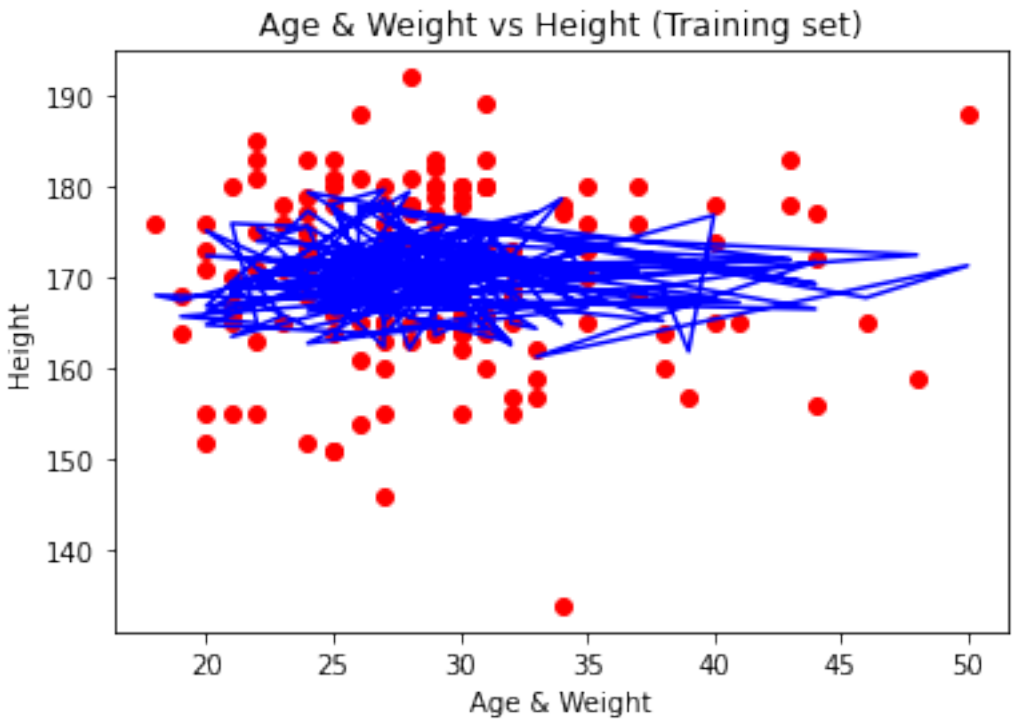
[ ]:
```
array([166.5142163 , 171.83170977, 165.76254402, 165.74394818,
       175.87065761, 172.42015482, 178.65254779, 166.0059388 ,
       172.1581642 , 166.5142163 , 175.68883455, 168.62460175,
       173.00859988, 172.09659248, 173.15033916, 171.18169299,
       168.90808031, 165.51914924, 172.70652548, 171.24326471,
       170.43002071, 171.83170977, 174.75533922, 169.43784575,
       172.21684382, 169.96471909, 169.69983637, 171.68997049,
       170.61473587, 172.58338204, 165.1955869 , 171.34492021,
       165.7009723 , 175.87065761, 162.17030195, 176.94589223,
       167.48779542, 166.55430008, 173.13174332])
```

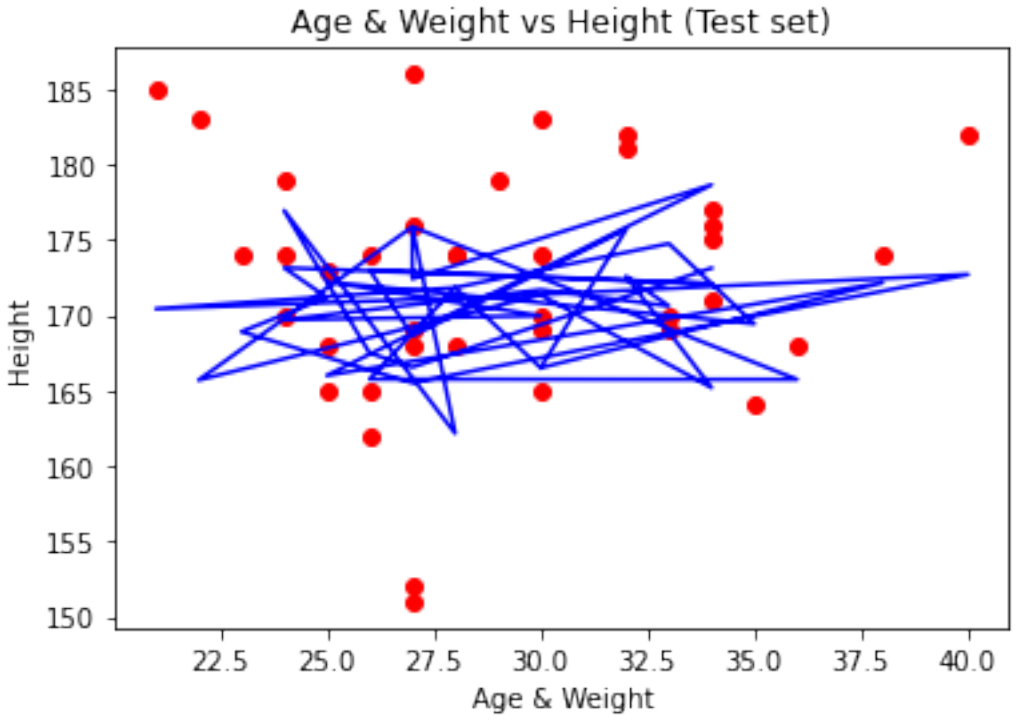
[ ]:
```
X_testin = [[28,85],[22,34]]
y_pred = regressor.predict(X_testin)
y_pred
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:450: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

[ ]:
```
array([175.62726283, 159.49006728])
```

[ ]:
```
# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train.iloc[:,0], y_train, color='red')
viz_train.plot(X_train.iloc[:,0]
, regressor.predict(X_train), color='blue')
viz_train.title('Age & Weight vs Height (Training set)')
viz_train.xlabel('Age & Weight')
viz_train.ylabel('Height')
viz_train.show()

# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test.iloc[:,0], y_test, color='red')
viz_test.plot(X_test.iloc[:,0], regressor.predict(X_test), color='blue')
viz_test.title('Age & Weight vs Height (Test set)')
viz_test.xlabel('Age & Weight')
viz_test.ylabel('Height')
viz_test.show()
```


Age & Weight vs Height (Training set)
Height
Age & Weight
190
180
170
160
150
140
20
25
30
35
40
45
50


Age & Weight vs Height (Test set)
Height
Age & Weight
185
180
175
170
165
160
155
150
22.5
25.0
27.5
30.0
32.5
35.0
37.5
40.0

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

23 Simple Linear Regression

[ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
df = pd.read_csv("salary_data.csv")
df.head(2)
```

[ ]:
```
   YearsExperience  Salary
0              1.1   39343
1              1.3   46205
```

[ ]:
```
X=df[["YearsExperience"]]
y=df['Salary']
```

[ ]:
```
# Import Library
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,␣
 ↪random_state=0)
```

23.0.1 Fit linear Regression Model

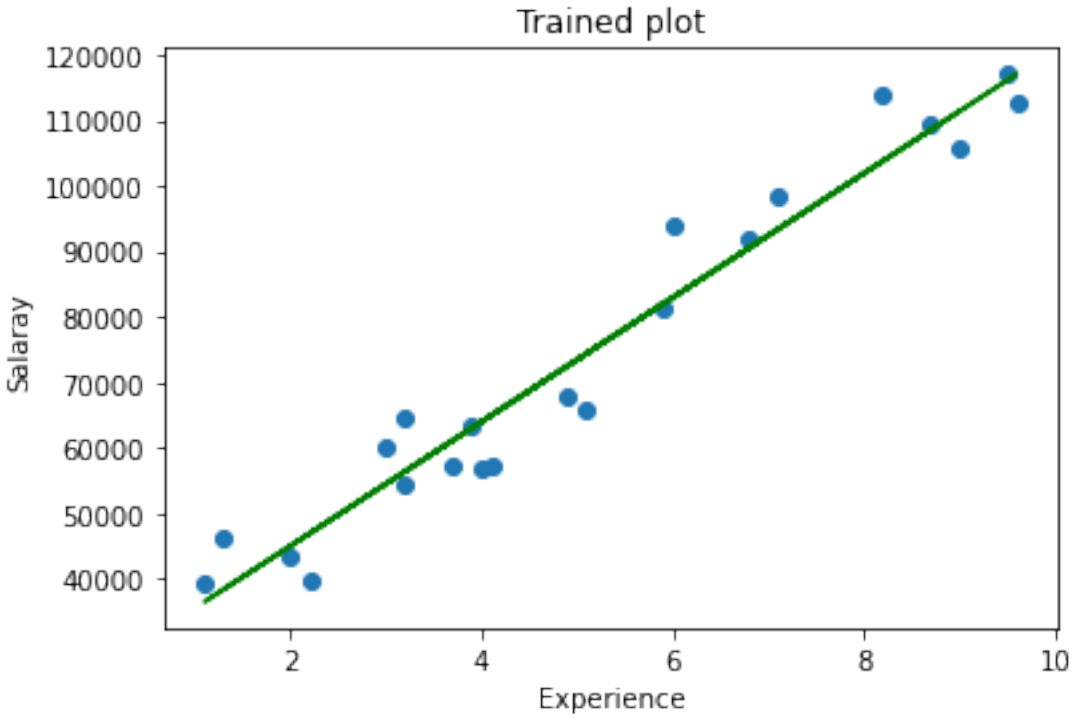
[ ]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(X_train, y_train)
model
```

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

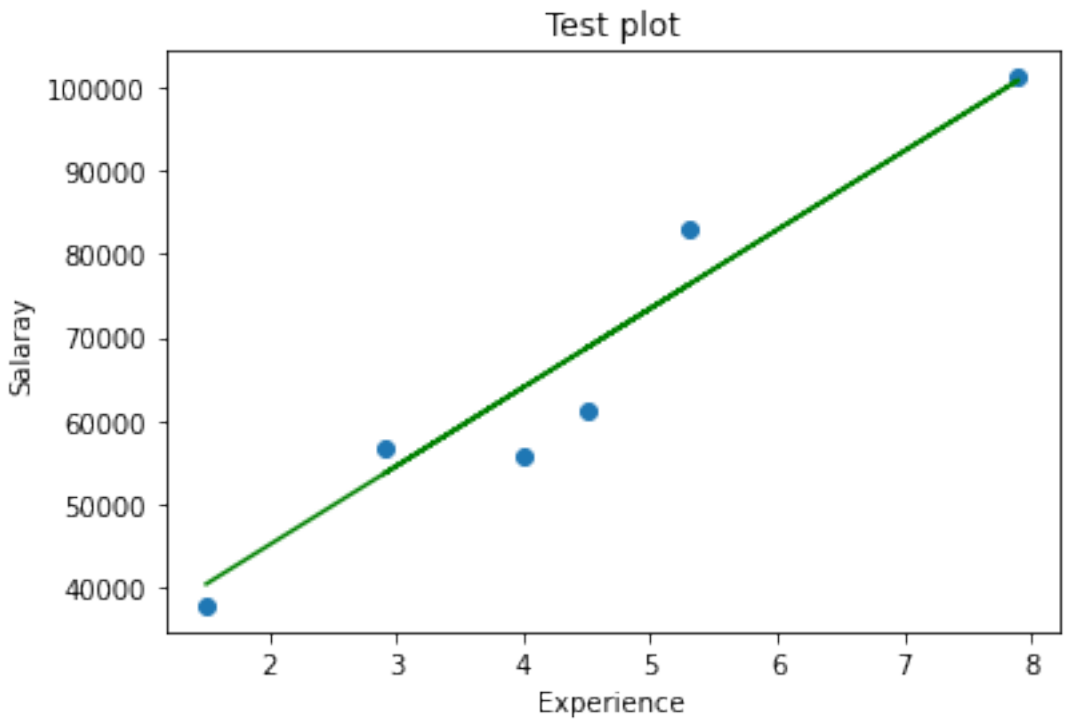
23.0.2 Plotting

[ ]:
```
import matplotlib.pyplot as plt
plt.scatter(X_train, y_train)
plt.plot(X_train, model.predict(X_train), color='green')
plt.xlabel("Experience")
plt.ylabel("Salaray")
```

```
plt.title("Trained plot")
plt.show()
plt.scatter(X_test, y_test)
plt.plot(X_test, model.predict(X_test), color='green')
plt.xlabel("Experience")
plt.ylabel("Salaray")
plt.title("Test plot")


#plt.scatter(X_test,y_test)
```



[ ]: Text(0.5, 1.0, 'Test plot')



23.0.3 Model Testing/Fitness

[ ]:
```
print("Score for testing data",model.score(X_test,y_test)) # corelation
print("Score for training data",model.score(X_train,y_train))
```

```
Score for testing data 0.9265115445546935
Score for training data 0.9482946812971009
```

23.0.4 Prediction of Unknown Values

[ ]:
```
model.predict([[5]])
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:450: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

[ ]:
```
array([73476.22072173])
```

[ ]:
```
y_pred = model.predict(X_test)
y_pred
```

[ ]:
```
array([ 40321.21895116, 100947.50790307,  68739.79189737,  76318.07801636,
        53583.21965939,  64003.363073  ])
```

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
#Import Data
df = pd.read_csv("ml_data_salary.csv")
df.head(2)
```

[ ]:

	age	distance	YearsExperience	Salary
0	31.1	77.75	1.1	39343
1	31.3	78.25	1.3	46205

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

23.1 Creating a model and Data fitting

[ ]:
```
model= LinearRegression().fit(X,y)
model
```

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

23.1.1 Checking coefficients of Input and Slope

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

[ ]:
```
array([-3.00216193e+15,  1.18788781e+15,  3.24424072e+13])
```

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

[ ]:
```
973272214586587.5
```

[ ]:
```
model.predict([[31.1,80,1.1]])
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:450: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

[ ]:
```
array([2.67274757e+15])
```

23.1.2 Splitting and Training (80-20 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)

model = LinearRegression().fit(X_train, y_train)
```

```
model
```

[ ]: LinearRegression()

[ ]:
```
# Assignment is how to plot multiple linear regression model
# How to test efficacy of the model? ( split train / test)
```

23.1.3 Regression Score (Accuracy Measurement)

https://scikit-learn.org/stable/modules/model_evaluation.html#mean-absolute-percentage-error

[ ]:
```
#reg= LinearRegression().fit(X_test, y_test)
print("Regression score without splitting =",model.score(X,y))
print("After splitting my train score =",model.score(X_train,y_train))
# this test will not work on numeric It will only work on classification data␣
 ↪like in decision tree algorithm
print(" After splitting test score =",model.score(X_test,y_test))
```

```
Regression score without splitting = 0.9565684395539251
After splitting my train score = 0.9409532368371482
 After splitting test score = 0.988401541985491
```

23.1.4 Score Checking

[ ]:
```
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
y_pred
# Compare with side p rakhi we test vs predicted test
score = accuracy_score(y_test,y_pred,normalize=False)
print("The accuracy score of model when compared with two test values is",score)
```

```
The accuracy score of model when compared with two test values is 0
```

Explained Variance Score

[ ]:
```
from sklearn.metrics import explained_variance_score
explained_variance_score(y_test, y_pred)
```

[ ]: 0.9896930311538696

Max Error

[ ]:
```
from sklearn.metrics import max_error
max_error(y_test, y_pred)
```

[ ]: 7751.0

Mean Absolute Error

[ ]:
```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, y_pred)
```

[ ]: 2469.1666666666665

Mean Squared Error

[ ]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
```

[ ]: 12571912.166666666

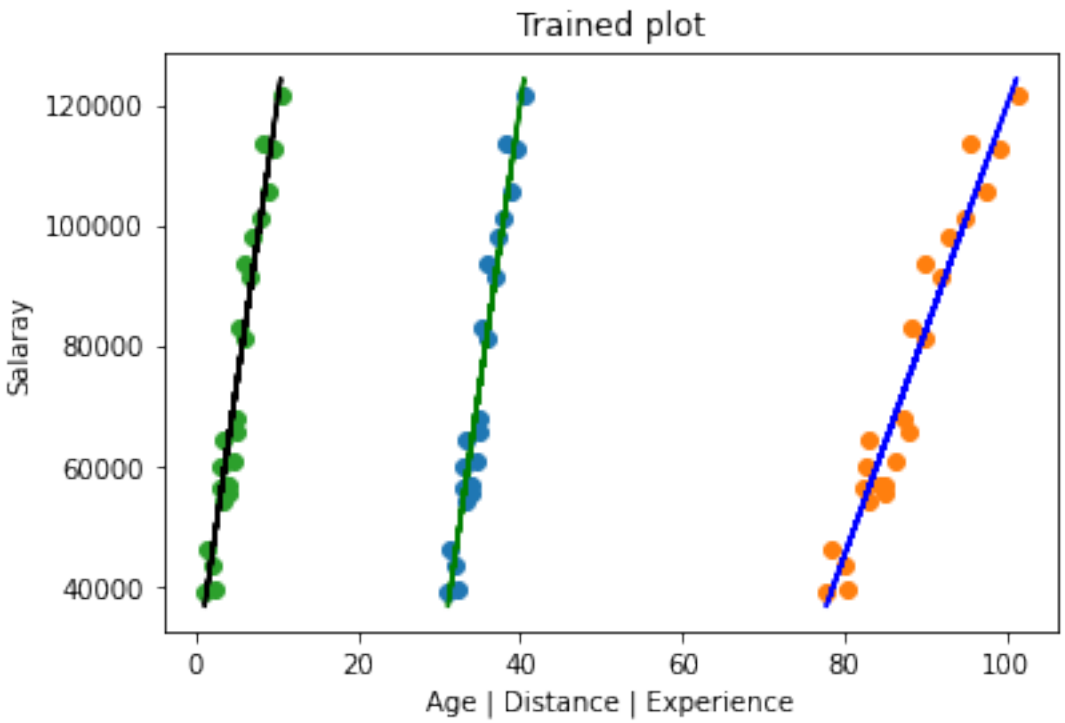
Mean Absolute Percentage Error

[ ]:
```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_test, y_pred)
```

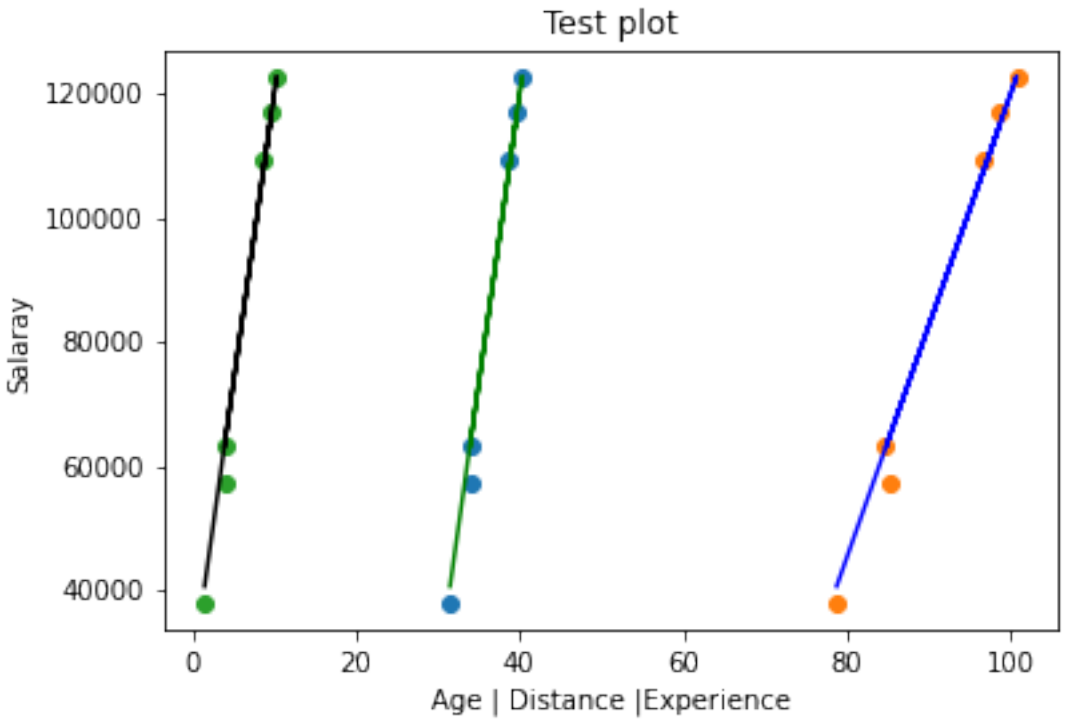
[ ]: 0.041779872719803136

23.1.5 Plotting multiple Linear Regression Model

[ ]:
```
import matplotlib.pyplot as plt
plt.scatter(X_train.age , y_train)
plt.plot(X_train.age, model.predict(X_train), color='green')

plt.scatter(X_train.distance , y_train)
plt.plot(X_train.distance, model.predict(X_train), color='blue')

plt.scatter(X_train.YearsExperience , y_train)
plt.plot(X_train.YearsExperience, model.predict(X_train), color='black')

plt.xlabel("Age | Distance | Experience")
plt.ylabel("Salaray")
plt.title("Trained plot")
plt.show()


plt.scatter(X_test.age, y_test)
plt.plot(X_test.age, model.predict(X_test), color='green')

plt.scatter(X_test.distance, y_test)
plt.plot(X_test.distance, model.predict(X_test), color='blue')

plt.scatter(X_test.YearsExperience , y_test)
plt.plot(X_test.YearsExperience, model.predict(X_test), color='black')

plt.xlabel("Age | Distance |Experience")
plt.ylabel("Salaray")
plt.title("Test plot")
```



[ ]: Text(0.5, 1.0, 'Test plot')



23.1.6 Prediction of future and Test Values

[ ]:
```
# Predicting the Test set results
y_pred = model.predict(X_test)
y_pred
```

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

[ ]:
```
# Predicitng fixed values
X_testin = [[28,45,1.1],[22,23,3,]]
y_pred = model.predict(X_testin)
y_pred
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:450: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

[ ]:
```
array([4.70711563e+16, 1.52251782e+17])
```

24 Doctor Sahab recommended in plots to use 3D plots with multiple lines

Must explore this because you used only 2d plot for this

Out of sample accuracy increase by splitting

[ ]:
```

```

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

# Importing the dataset
#dataset = pd.read_csv('181105_missing-data.csv')
dataset = pd.read_csv('salary_data.csv')
dataset =dataset.dropna()
X = dataset.iloc[:, :-1].values #get a copy of dataset exclude last column
y = dataset.iloc[:, 1].values #get array of dataset in column 1st

#print(X)
# Splitting the dataset into the Training set and Test set
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)
```

```
"""
# Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
"""

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

# Predicting the Test set results
y_pred = regressor.predict(X_test)

# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train[:,0], y_train, color='red')
viz_train.plot(X_train[:,0], regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()

# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test[:,0], y_test, color='red')
viz_test.plot(X_train[:,0], regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()
```


Salary VS Experience (Training set)
120000
110000
100000
90000
80000
70000
60000
50000
40000
Salary
2
4
6
8
10
Year of Experience


Salary VS Experience (Test set)
120000
110000
100000
90000
80000
70000
60000
50000
40000
Salary
2
4
6
8
10
Year of Experience

[ ]:

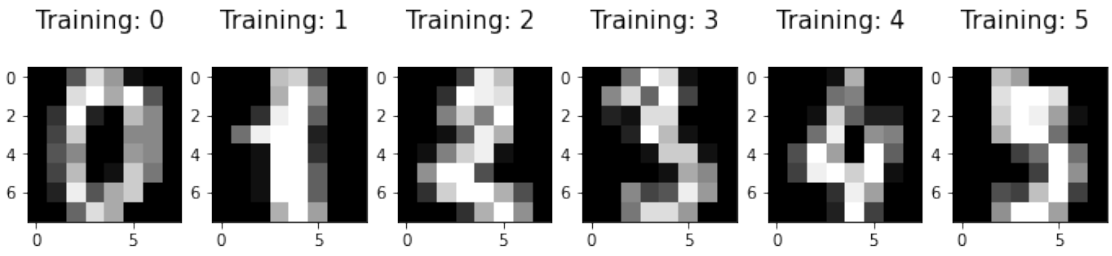
https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a

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

from sklearn.datasets import load_digits
digits= load_digits()
```

[ ]:
```
#features and outputs
X=digits.data # input (means rows (pics,dataset) )
y=digits.target # output (labels only )

print(digits.data.shape) # input (means rows (pics,dataset) )
print(digits.target.shape) # output (labels only )
```

```
(1797, 64)
(1797,)
```

[ ]:
```
plt.figure(figsize=(20,4))
for index, (image, label) in enumerate (zip(digits.data[0:6],digits.target[0:
↪6])):
    plt.subplot(1,10,index+1)
    plt.imshow(np.reshape(image, (8,8)),cmap=plt.cm.gray)
    plt.title('Training: %i \n' %label, fontsize=15)
```



24.0.1 Split Data into Test/Train

[ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# split syntax
```

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

[ ]:
```
print("Train Input data",X_train.shape)
print("Train Output data",y_train.shape)
print("Test output data",X_test.shape)
print("Test output data",y_test.shape)
```

```
Train Input data (1437, 64)
Train Output data (1437,)
Test output data (360, 64)
Test output data (360,)
```

24.0.2 Model Training

[ ]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression().fit(X,y)
#model
model.predict(X_test[0:20])
```

```
C:\Users\dell7450\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(
```

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

24.0.3 Accuracy Test

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

```
The accuracy score is  1.0
```

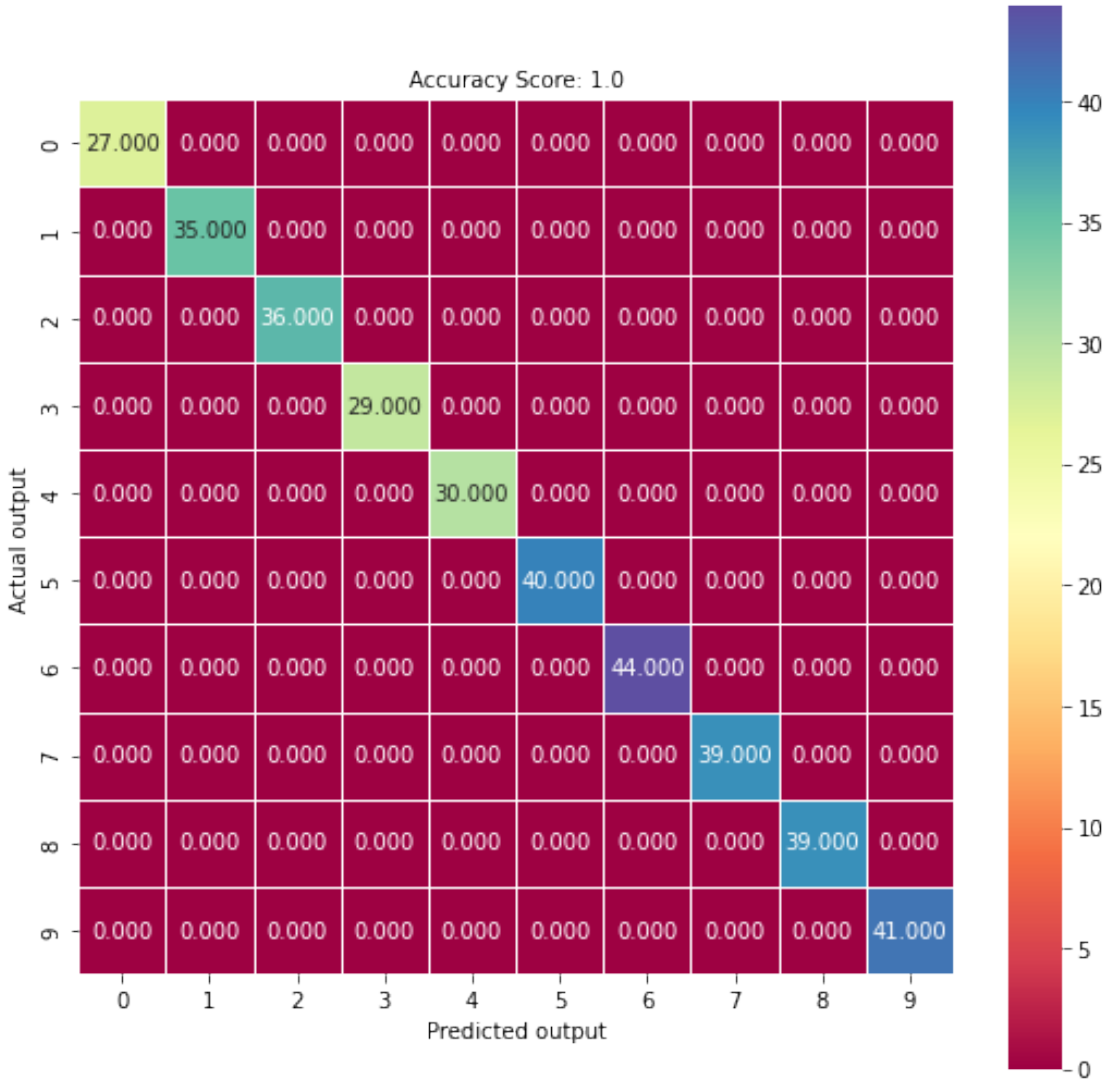
24.0.4 Confusion Matrix

[ ]:
```
from sklearn.metrics import confusion_matrix
predictions = model.predict(X_test)
cm= confusion_matrix(y_test, predictions)
cm
```

[ ]:
```
array([[27,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 35,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 36,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 29,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 30,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 40,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 44,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 39,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0, 39,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 41]], dtype=int64)
```

[ ]:
```
# Heatmap to visualize COnfusion Matrix
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_sampletitle= 'Accuracy Score: {0}'.format(score)
plt.title(all_sampletitle,size =10)
```

[ ]:
```
Text(0.5, 1.0, 'Accuracy Score: 1.0')
```



Mis classified Labels

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


misclassifiedIndexes
```

[ ]: []

[ ]:
```
# plots of misclassified 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]), fontsize = 10)
```

```
<Figure size 1440x288 with 0 Axes>
```

[ ]:

[ ]:
```
import pandas as pd
df = pd.read_csv("mldata_dtc.csv")
df.head(1)
```

[ ]:

	age	height	weight	gender	likeness
0	27	170.688	76.0	Male	Biryani

24.0.5 Convert gender (M/F) to 1 and 0

[ ]:
```
df['gender'] = df['gender'].replace("Male",1)
df['gender'] = df['gender'].replace("Female",0)
df.tail(2)
```

[ ]:

	age	height	weight	gender	likeness
243	25	5.7	65.0	1	Biryani
244	33	157.0	56.0	0	Samosa

[ ]:
```
X=df[['weight','gender','age','height']]
#print("the value in X feature is ",X.head(3))
y=df['likeness']
#print("the value in y output  is ",y.head(3))
```

[ ]:
```
#machine learning algorithm
from sklearn.tree import DecisionTreeClassifier
# create and fit model
model = DecisionTreeClassifier().fit(X,y)
#Prediction
model.predict([[23,0,23,171]])
```

```
C:\Users\dell7450\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(['Pakora'], dtype=object)
```

24.0.6 How to measure accuracy (SPlit 80-20)

[ ]:
```
# accuracy by splitting
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# split syntax
X_train,X_test,y_train,y_test =train_test_split(X, y, test_size=0.2)
#Creating and model fitting
model = DecisionTreeClassifier().fit(X_train,y_train)
# checking predicted values with input test data
predicted_values = model.predict(X_test)
print("The predicted values from 20% of test input is",predicted_values,"\n")
```

```
The predicted values from 20% of test input is ['Biryani' 'Biryani' 'Biryani'
'Pakora' 'Biryani' 'Samosa' 'Biryani'
 'Samosa' 'Biryani' 'Samosa' 'Pakora' 'Biryani' 'Biryani' 'Biryani'
 'Samosa' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Samosa' 'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Pakora' 'Biryani' 'Biryani' 'Samosa' 'Pakora' 'Samosa' 'Biryani'
 'Samosa' 'Pakora' 'Biryani' 'Samosa' 'Biryani' 'Pakora' 'Pakora'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani']
```

24.0.7 Score Checking

[ ]:
```
#Now compare y_test values with the values of y_test(predicted)
score = accuracy_score(y_test,predicted_values)
print("The accuracy score of model when compared with two test values is",score)
```

```
The accuracy score of model when compared with two test values is
0.4897959183673469
```

[ ]:
```
#graph
from sklearn import tree
model = DecisionTreeClassifier().fit(X,y)

#graphic
tree.export_graphviz(model,
out_file="foodie.dot",
feature_names=["age","gender","weight","height"],
class_names=sorted(y.unique()),
label="all",
rounded=True,
filled=True)
```

24.0.8 How to train and save our Model

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

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

# How to run save stored model (Assignment)
saved_model=joblib.load('foodie.joblib')

Final_predictions=saved_model.predict(X_test)
Final_predictions
```

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

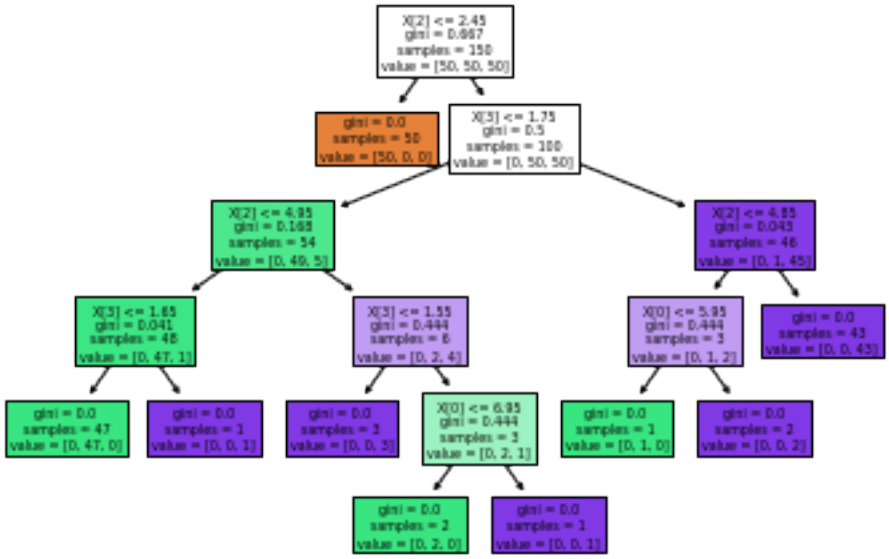
[ ]:
```

```

[ ]:
```
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

df= sns.load_dataset("iris")
df.head(1)
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
X= df.iloc[:,:-1]
y=df.iloc[:, -1:] # conisder only last column
```

24.0.9 Saving High Resolution plots

[ ]:
```
model = DecisionTreeClassifier()
model.fit(X,y)
plt.title("Decision Model trained model of Iris")

model.predict(X)
```

```
plot_tree(model, filled=True)
# save in tiff and jpeg
plt.savefig('tiff_compressed.tiff',dpi=600, format='tiff',
facecolor='white', edgecolor='none',
pil_kwargs={"compression": "tiff_lzw"})

plt.show()
```



24.0.10 Running a Saved Model

[ ]:
```
import joblib

joblib.dump(model,"iris.joblib")

# How to run save stored model (Assignment)
saved_model=joblib.load('iris.joblib')

Final_predictions=saved_model.predict(X)
#Final_predictions
```

24.0.11 80-20

[ ]:
```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=0)
#model.fit(X,y)
#Accuracy test
score = model.score(X_test,y_test)
```

```
print("The accuracy score with 80-20 (X_test) and (y_test) is ",score)

# checking predicted values with input test data
predicted_values = model.predict(X_test)
print("The predicted values from 20% of test input is",predicted_values,"\n")

superscore = accuracy_score(y_test,predicted_values)
print("The accuracy score of model when compared with twenty percent original␣
 ↪test values is",superscore)

#Checking unknown 5 Values
unknownvalues=[[5.2,3.6,1.8,0.1],[5.2,3.5,1.3,1.1],[5.2,3.6,1.4,0.1],[5.5,3.6,1.
 ↪66,0.1],[5.4,3.6,1.8,0.1]]
unknownvalues = model.predict(unknownvalues)
print("\n The prediction of 5 unknown values is ",unknownvalues)
```

```
The accuracy score with 80-20 (X_test) and (y_test) is  1.0
The predicted values from 20% of test input is ['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']

The accuracy score of model when compared with twenty percent original test
values is 1.0

 The prediction of 5 unknown values is  ['setosa' 'setosa' 'setosa' 'setosa'
'setosa']

C:\Users\dell7450\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(
```

24.0.12 90-10

[ ]:
```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.1,␣
 ↪random_state=0)
#model.fit(X,y)
#Accuracy test
score = model.score(X_test,y_test)
print("The accuracy score with 90-10 (X_test) and (y_test) is ",score)

# checking predicted values with input test data
predicted_values = model.predict(X_test)
print("The predicted values from 10% of test input is",predicted_values,"\n")
```

```
superscore = accuracy_score(y_test,predicted_values)
print("The accuracy score of model when compared with ten percent original test␣
 ↪values is",superscore)

#Checking unknown 5 Values
unknownvalues=[[5.2,3.6,1.8,0.1],[5.2,3.5,1.3,1.1],[5.2,3.6,1.4,0.1],[5.5,3.6,1.
 ↪66,0.1],[5.4,3.6,1.8,0.1]]
unknownvalues = model.predict(unknownvalues)
print("\n The prediction of 5 unknown values is ",unknownvalues)
```

```
The accuracy score with 90-10 (X_test) and (y_test) is  1.0
The predicted values from 10% of test input is ['virginica' 'versicolor'
'setosa' 'virginica' 'setosa' 'virginica'
 'setosa' 'versicolor' 'versicolor' 'versicolor' 'virginica' 'versicolor'
 'versicolor' 'versicolor' 'versicolor']

The accuracy score of model when compared with ten percent original test values
is 1.0

 The prediction of 5 unknown values is  ['setosa' 'setosa' 'setosa' 'setosa'
'setosa']

C:\Users\dell7450\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(
```

24.0.13 70-30

[ ]:
```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=0)
#model.fit(X,y)
#Accuracy test
score = model.score(X_test,y_test)
print("The accuracy score with 70-30 (X_test) and (y_test) is ",score)

# checking predicted values with input test data
predicted_values = model.predict(X_test)
print("The predicted values from 30% of test input is",predicted_values,"\n")

superscore = accuracy_score(y_test,predicted_values)
print("The accuracy score of model when compared with thirty percent original␣
 ↪test values is",superscore)

#Checking unknown 5 Values
unknownvalues=[[5.2,3.6,1.8,0.1],[5.2,3.5,1.3,1.1],[5.2,3.6,1.4,0.1],[5.5,3.6,1.
 ↪66,0.1],[5.4,3.6,1.8,0.1]]
unknownvalues = model.predict(unknownvalues)
```

```
print("\n The prediction of 5 unknown values is ",unknownvalues)
```

```
The accuracy score with 70-30 (X_test) and (y_test) is  1.0
The predicted values from 30% of test input is ['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'
 'versicolor' 'versicolor' 'versicolor' 'virginica' 'setosa' 'virginica'
 'setosa' 'setosa']

The accuracy score of model when compared with thirty percent original test
values is 1.0

 The prediction of 5 unknown values is  ['setosa' 'setosa' 'setosa' 'setosa'
'setosa']

C:\Users\dell7450\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(
```

[ ]:
```
import pandas as pd
df = pd.read_csv("mldata_dtc.csv")
df.head(1)
```

[ ]:
```
   age   height  weight gender likeness
0   27  170.688    76.0   Male  Biryani
```

24.0.14 Convert gender (M/F) to 1 and 0

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

[ ]:
```
     age  height  weight  gender likeness
244   33   157.0    56.0       0   Samosa
```

[ ]:
```
X=df[['weight','gender','age']]
#print("the value in X feature is ",X.head(3))
y=df['likeness']
#print("the value in y output  is ",y.head(3))
```

[ ]:
```
#machine learning algorithm
from sklearn.neighbors import KNeighborsClassifier
# create and fit model
```

```
model = KNeighborsClassifier(n_neighbors=5).fit(X,y)
#Prediction
model.predict([[23,0,23]])
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:450: UserWarning: X does not have valid feature names,
but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

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

24.0.15 accuracy evaluation (SPlit 80-20)

Metrics for Evaluation of Classification Data

[ ]:
```
# accuracy by splitting
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# split syntax
X_train,X_test,y_train,y_test =train_test_split(X, y, test_size=0.2)
#Creating and model fitting
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train,y_train)
# checking predicted values with input test data
predicted_values = model.predict(X_test)
print("The predicted values from 20% of test input is",predicted_values,"\n")
```

```
The predicted values from 20% of test input is ['Biryani' 'Biryani' 'Biryani'
'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani' 'Biryani'
 'Biryani' 'Biryani' 'Biryani' 'Pakora' 'Biryani' 'Biryani' 'Biryani']
```

24.0.16 Accuracy Score Checking

[ ]:
```
#Now compare y_test values with the values of y_test(predicted)
score = accuracy_score(y_test,predicted_values)
print("The accuracy score of model when compared with twenty percent test␣
 ↪values is",score)
```

```
The accuracy score of model when compared with twenty percent test values is
0.6122448979591837
```

24.0.17 Top k-Accuracy Score Checking

[ ]:
```
# from sklearn.metrics import top_k_accuracy_score
# top_k_accuracy_score(y_test, predicted_values, k=2, normalize=False)

# ## this will generate error because it work on Numeric Data. I need to␣
 ↪convert Biryani Samosa Pakora into  1,2,3
# # This is my future task
```

24.0.18 Confusion Matrix

It is really a big confusion

[ ]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, predicted_values)
```

[ ]:
```
array([[29,  0,  0],
       [ 6,  1,  0],
       [13,  0,  0]], dtype=int64)
```

[ ]:
```
tn, fp, fn, tp ,tn1, fp1, fn1, tp1, tp2 = confusion_matrix(y_test,␣
 ↪predicted_values).ravel()
tn, fp, fn, tp ,tn1, fp1, fn1, tp1, tp2
```

[ ]:
```
(29, 0, 0, 6, 1, 0, 13, 0, 0)
```

[ ]:
```

```

[ ]:
```
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

df= sns.load_dataset("iris")
X=df.iloc[:,:-1]
y=df.iloc[:,-1:]
```

[ ]:
```
model = RandomForestClassifier(n_estimators=100)
model.fit(X,y)
model.predict([[2,3,4,5]])
```

```
C:\Users\dell7450\AppData\Local\Temp/ipykernel_11344/1312009047.py:2:
DataConversionWarning: 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\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\base.py:450: UserWarning: X does not have valid feature names,
but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

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

[ ]:
```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
#model.fit(X,y)
```

24.0.19 Accuracy (X-test,y_test)

[ ]:
```
score = model.score(X_test,y_test)
print("The accuracy score with 80-20 (X_test) and (y_test) is ",score)
```

```
The accuracy score with 80-20 (X_test) and (y_test) is  1.0
```

24.0.20 Metric Accuracy (y_test, predictions)

[ ]:
```
predicted_values = model.predict(X_test)
print("The predicted values from 20% of test input is",predicted_values,"\n")

superscore = accuracy_score(y_test,predicted_values)
print("The accuracy score of model when compared with twenty percent original␣
 ↪test values is",superscore)
```

```
The predicted values from 20% of test input is ['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']

The accuracy score of model when compared with twenty percent original test
values is 1.0
```

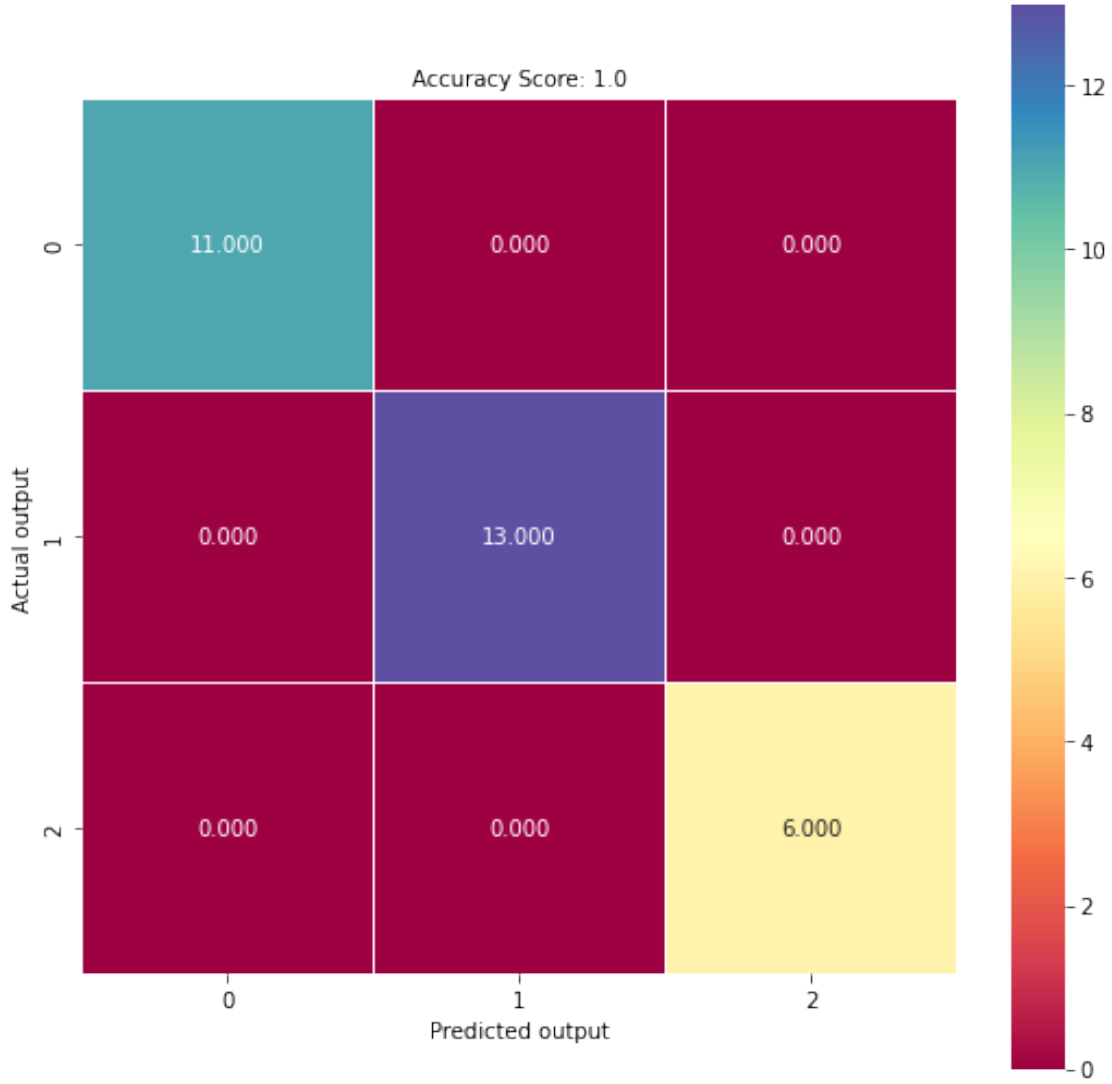
24.0.21 Confusion Matrix (y_test,predictions)

[ ]:
```
from sklearn.metrics import confusion_matrix
predictions = model.predict(X_test)
cm= confusion_matrix(y_test, predictions)
cm
```

[ ]:
```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]], dtype=int64)
```

[ ]:
```
# Heatmap to visualize COnfusion Matrix
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_sampletitle= 'Accuracy Score: {0}'.format(score)
plt.title(all_sampletitle,size =10)
```

[ ]:
```
Text(0.5, 1.0, 'Accuracy Score: 1.0')
```



[ ]:

https://towardsdatascience.com/random-forest-in-python-24d0893d51c0

24.0.22 Data Import, Refining and Classification according to Algorithm

[ ]:
```
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor

df= pd.read_csv("temps.csv")
df=df.drop(['forecast_noaa','forecast_acc','forecast_under'],axis=1)

df =pd.get_dummies(df)



# I was getting a outlier value in temp_1 in final plotting, so I find and␣
 ↪delete the entire row
s = df.temp_1 > 100
e=df.loc[s,'temp_1']
print("This index with value is creating problem in final graph",e)
df = df.drop(286)

X= df.drop("actual",axis=1)
#y=df.pop('actual')
y=df.iloc[:, 6:7]
```

```
This index with value is creating problem in final graph 286    117
Name: temp_1, dtype: int64
```

24.0.23 Splitting and Data Training

[ ]:
```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)

model = RandomForestRegressor(n_estimators=50)
model.fit(X,y)

print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)
```

```
C:\Users\dell7450\AppData\Local\Temp/ipykernel_11020/137745024.py:4:
DataConversionWarning: 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)
```

```
Training Features Shape: (277, 14)
Training Labels Shape: (277, 1)
Testing Features Shape: (70, 14)
Testing Labels Shape: (70, 1)
```

Accuracy (X-test,y_test)

[ ]:
```
score = model.score(X_test,y_test)
print("The accuracy score with 80-20 (X_test) and (y_test) is ",score)
```

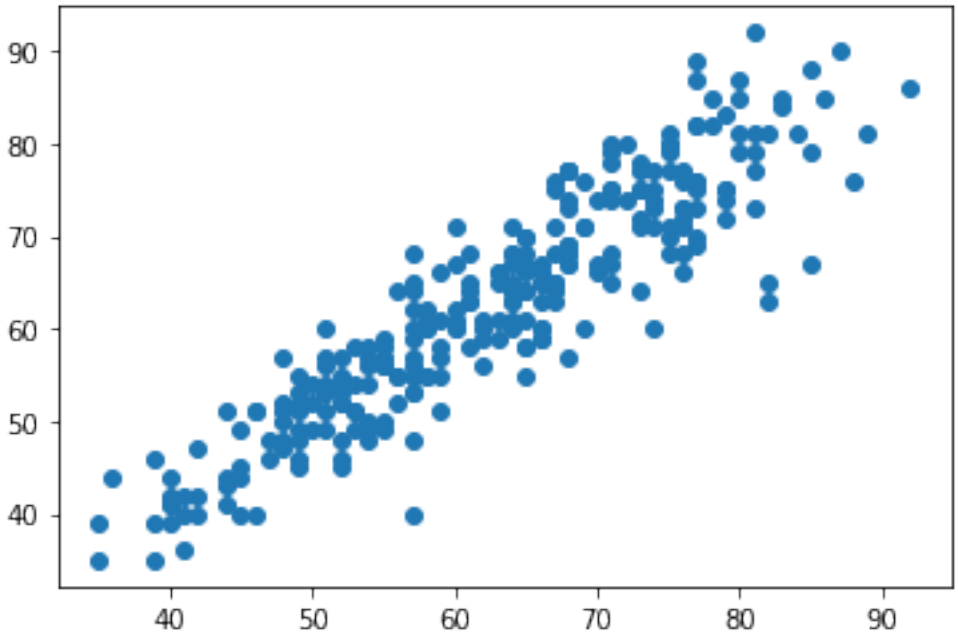
```
The accuracy score with 80-20 (X_test) and (y_test) is  0.9652858761430959
```

24.0.24 Metric Accuracy (y_test, predictions)

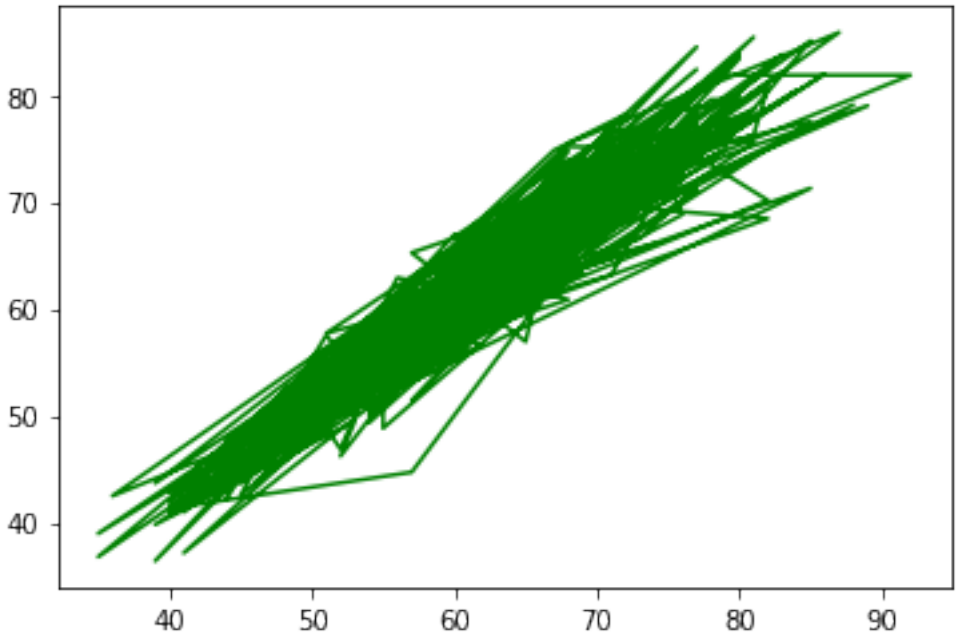
[ ]:
```
# not possible in regressive data
```

24.0.25 Plot (not so accurate)

[ ]:
```
import matplotlib.pyplot as plt
plt.scatter(X_train.temp_1 , y_train)
plt.show()
plt.plot(X_train.temp_1, model.predict(X_train), color='green')
```



[ ]:
```
[<matplotlib.lines.Line2D at 0x1e47eea3e20>]
```



24.1 Baba g mera qs hay

Confusion matrix sirf classification data par hota hay ? Agar me ghalat hon tu please correct karen Peer sab

[ ]:
```
# from sklearn.metrics import confusion_matrix
# predictions = model.predict(X_test)
# cm= confusion_matrix(y_test, predictions)
# cm
```

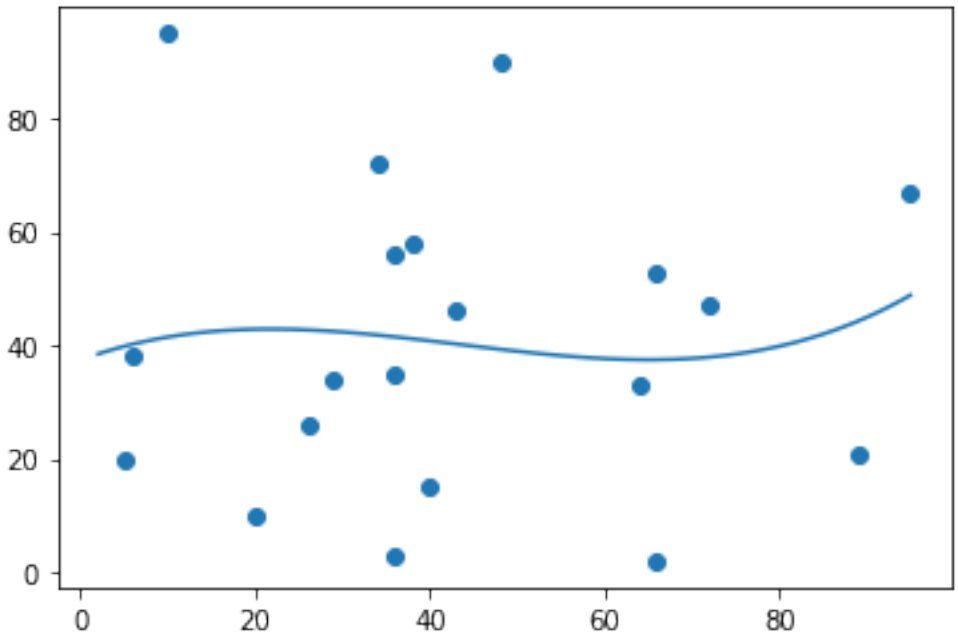
[ ]:
```

```

24.1.1 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=[21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]
print(np.polyfit(x,y,3))
mymodel= np.poly1d(np.polyfit(x,y,3))
#polyfit coefficients dedecting
#poly1d is making a eqn
print(mymodel)
myline = np.linspace(2,95,100)
plt.scatter(x,y)
plt.plot(myline,mymodel(myline))
```

```
plt.show()
#print(mymodel(myline))
```

```
[ 1.33138691e-04 -1.73190094e-02  5.63719354e-01  3.74427648e+01]
          3           2
0.0001331 x - 0.01732 x + 0.5637 x + 37.44
```



[ ]:
```
# R- squared for bad fit
from sklearn.metrics import r2_score

x=[89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y=[21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

model = np.poly1d(np.polyfit(x,y,3))
print(r2_score(y,model(x)))
```

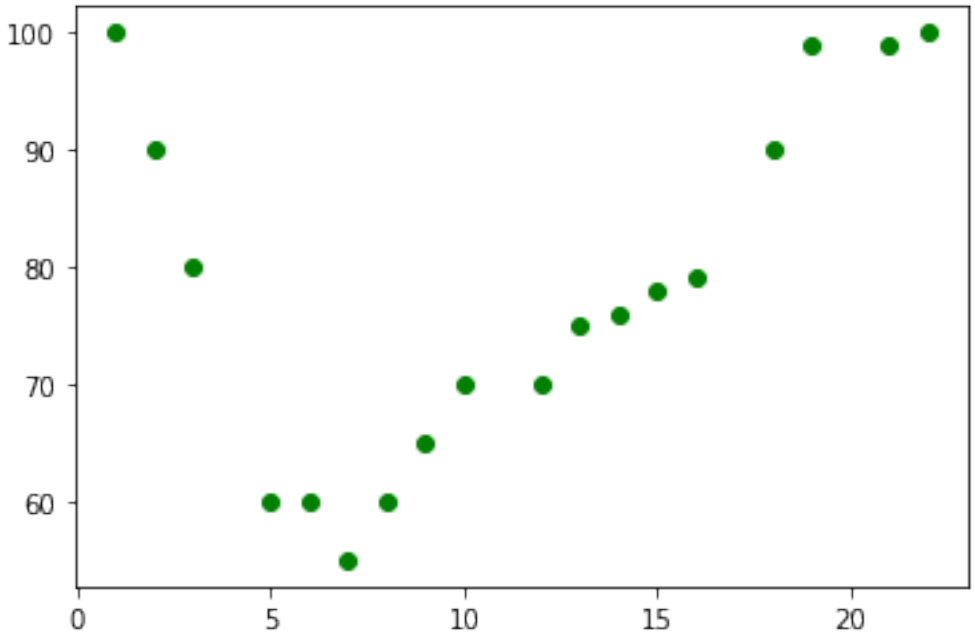
```
0.009952707566680652
```

24.2 Polynomial Regression

Data

[ ]:
```
# Step1 : 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]
```

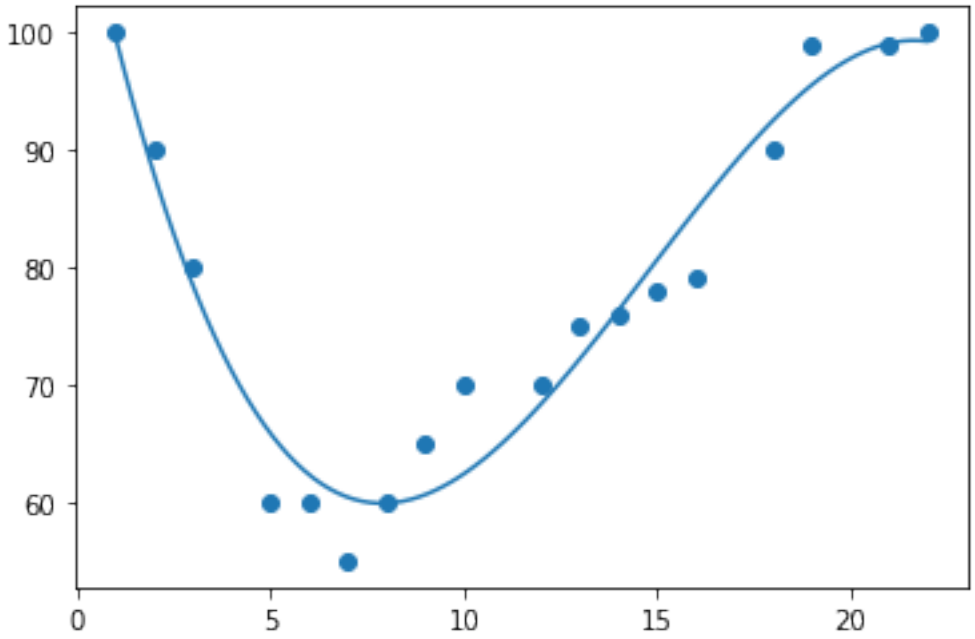
```
plt.scatter(x,y, color='green')
plt.show()
```



Line Plotting and Curve Checking

[ ]:
```
# Step2: Draw the line
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]
print(np.polyfit(x,y,3))
mymodel= np.poly1d(np.polyfit(x,y,3))
#polyfit coefficients dedecting
#poly1d is making a eqn
print(mymodel)
myline = np.linspace(1,22,100)
plt.scatter(x,y)
plt.plot(myline,mymodel(myline))
plt.show()
```

```
[-3.03208795e-02  1.34333191e+00 -1.55383039e+01  1.13768037e+02]
          3         2
-0.03032 x + 1.343 x - 15.54 x + 113.8
```



Measure accuracy through R2

[ ]:
```
# R- squared
from sklearn.metrics import r2_score
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))
print(r2_score(y,model(x)))
```

```
0.9432150416451026
```

Checking Prediction on an unknown Value

[ ]:
```
# Predictions
mymodel = np.poly1d(np.polyfit(x,y,3))
print(mymodel)
speed = mymodel(18)
speed
```

```
          3         2
-0.03032 x + 1.343 x - 15.54 x + 113.8
```

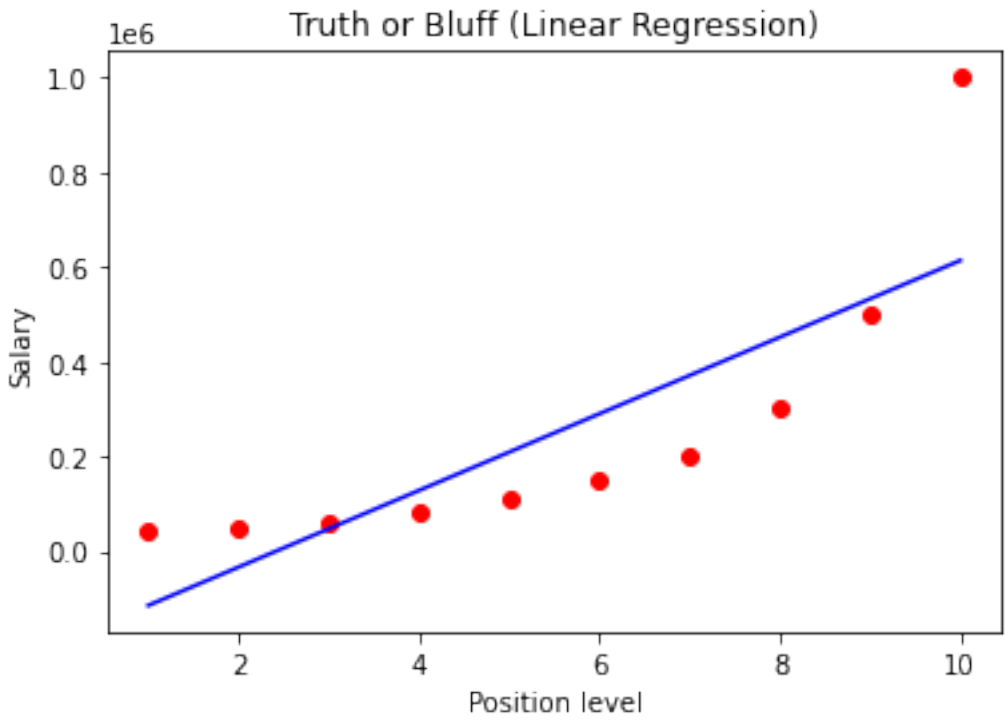
[ ]:
```
92.48673749579999
```

24.2.1 Practical Example using Sklearn

[ ]:
```
# Another Important Example
import pandas as pd
df= pd.read_csv("https://s3.us-west-2.amazonaws.com/public.gamelab.fun/dataset/
 ↪position_salaries.csv")
X = df.iloc[:, 1:2].values
y = df.iloc[:, 2].values
#SPlit Test train
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)
```

Linear Regression Model

[ ]:
```
# Fitting linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_reg= LinearRegression()
lin_reg.fit(X,y)
# Visualizing the Linear Regression result
def viz_linear():
    plt.scatter(X,y, color='red')
    plt.plot(X, lin_reg.predict(X), color='blue')
    plt.title('Truth or Bluff (Linear Regression)')
    plt.xlabel('Position level')
    plt.ylabel('Salary')
    return
viz_linear()
```



Polynomial Regression

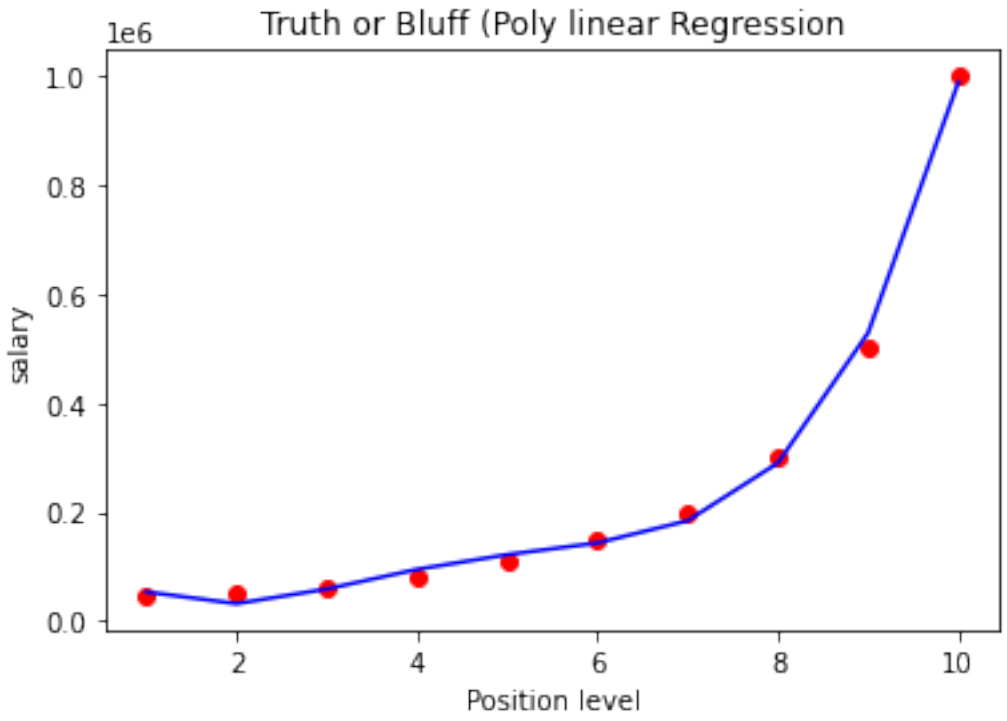
[ ]:
```
# Fitting plynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg_degrees = PolynomialFeatures(degree=4)
X_poly =  poly_reg_degrees.fit_transform(X)
# First equation from an array
t = np.squeeze(X_poly)
print('X data after polynomial eqn is',t)
# checking equation generated from 2nd Data set
khel= np.poly1d(X_poly[1])
print(khel)
pol_reg = LinearRegression()
pol_reg.fit(X_poly, y)
```

```
X data after polynomial eqn is [[1.000e+00 1.000e+00 1.000e+00 1.000e+00
1.000e+00]
 [1.000e+00 2.000e+00 4.000e+00 8.000e+00 1.600e+01]
 [1.000e+00 3.000e+00 9.000e+00 2.700e+01 8.100e+01]
 [1.000e+00 4.000e+00 1.600e+01 6.400e+01 2.560e+02]
 [1.000e+00 5.000e+00 2.500e+01 1.250e+02 6.250e+02]
 [1.000e+00 6.000e+00 3.600e+01 2.160e+02 1.296e+03]
 [1.000e+00 7.000e+00 4.900e+01 3.430e+02 2.401e+03]
 [1.000e+00 8.000e+00 6.400e+01 5.120e+02 4.096e+03]
```

```
 [1.000e+00 9.000e+00 8.100e+01 7.290e+02 6.561e+03]
 [1.000e+00 1.000e+01 1.000e+02 1.000e+03 1.000e+04]]
   4     3     2
1 x + 2 x + 4 x + 8 x + 16
```

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

[ ]:
```
# Visualizzing the polynomial regression results
def viz_polynomial():
    plt.scatter(X,y, color='red')
    plt.plot(X, pol_reg.predict(X_poly),color='blue')
    plt.title('Truth or Bluff (Poly linear Regression')
    plt.xlabel('Position level')
    plt.ylabel('salary')
    return
viz_polynomial()
```



24.2.2 Comparison between Linear and Polynomial Regression

Linear Regression Prediction

[ ]:
```
pred_linear= lin_reg.predict([[11]])
```

[ ]:
```
pred_polynomial= pol_reg.predict(poly_reg_degrees.fit_transform([[11]]))
```

[ ]:
```
print("Linear Regression Result  =",pred_linear)
print("Polynomial Regression Result",pred_polynomial)

print('The difference between two regression results is ',pred_linear -␣
 ↪pred_polynomial)
```

```
Linear Regression Result  = [694333.33333333]
Polynomial Regression Result [1780833.33333359]
The difference between two regression results is  [-1086500.00000025]
```

Polynomial k graph dekh k (like sine cosines) you have to decide kaunsa degree ka ap k points ko best fit karega

[ ]:
```

```

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
#Load the dataset of iris
flower = sns.load_dataset("iris")
#input and output
X = flower.iloc[:, :-1]
y=flower.iloc[:, -1:]
# Training and fitting the model
model= GaussianNB().fit(X,y)
model
#train test split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,␣
 ↪random_state=0)
```

```
C:\Users\dell7450\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\utils\validation.py:993: DataConversionWarning: 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().
  y = column_or_1d(y, warn=True)
```

[ ]:
```

```

[ ]:
```
# making predictions on the test set
y_pred = model.predict(X_test)
```

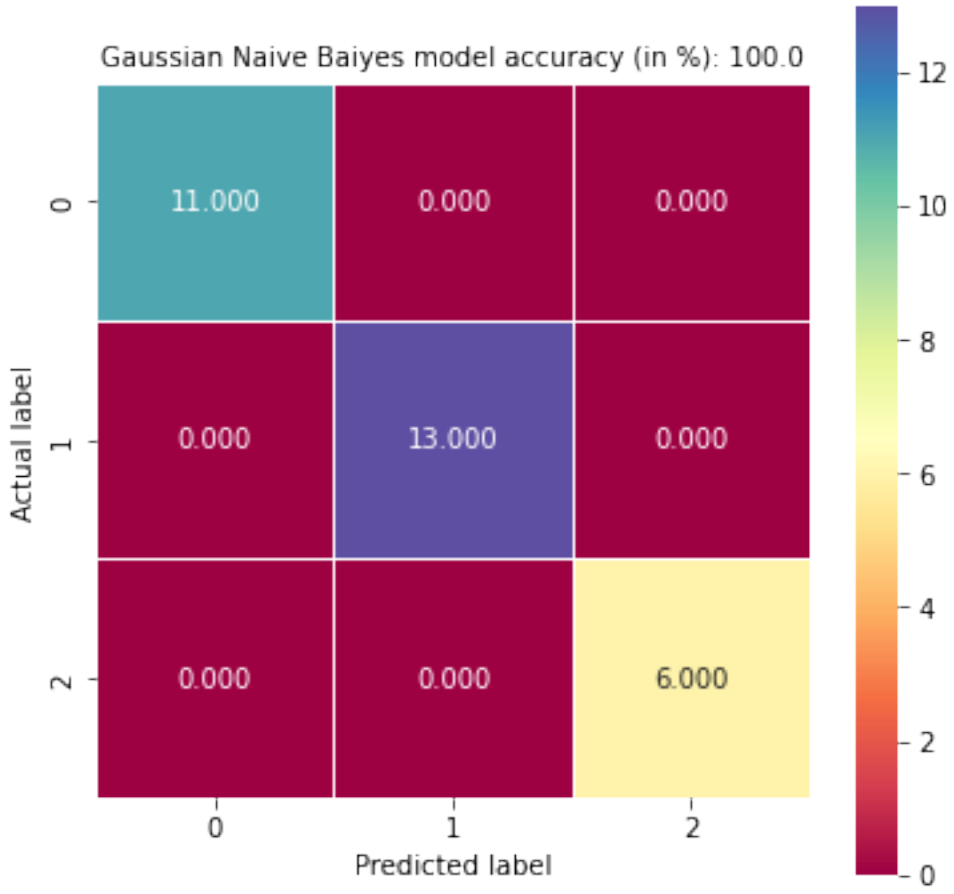
[ ]:
```
# Accuracy score between y_test and y_pred
score= metrics.accuracy_score(y_test,y_pred)
print("Gaussian Naives Bayes model accuracy in % when compared to predicted␣
 ↪test and actual values is",score,"%")
```

Gaussian Naives Bayes model accuracy in % when compared to predicted test and actual values is 1.0 %

[ ]:
```
#Confusion Matrix
cm = metrics.confusion_matrix(y_test,y_pred)
print(cm)
# plotting a confusion matrix
plt.figure(figsize=(6,6)) # height width in inches
sns.heatmap(cm,annot=True, fmt=".3f", linewidths=.5, square=True,␣
 ↪cmap='Spectral')
plt.ylabel("Actual label")
plt.xlabel("Predicted label")
all_sample_title= "Gaussian Naive Baiyes model accuracy (in %): {0}".
 ↪format(score*100)
plt.title(all_sample_title,size=10)
```

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

[ ]: Text(0.5, 1.0, 'Gaussian Naive Baiyes model accuracy (in %): 100.0')



[ ]:

24.2.3 Multinomial Naive Bayes with Basic NLP

https://iq.opengenus.org/text-classification-naive-bayes/

You need to refer to Svm lecture exercise for doing it other way around

[ ]:
```
import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import fetch_20newsgroups

# We defined the categories which we want to classify
categories = ['rec.motorcycles', 'sci.electronics','sci.med','comp.graphics']

# sklearn provides us with subset data for training and testing
X_train = fetch_20newsgroups(subset='train',
                                categories=categories, shuffle=True,␣
 ↪random_state=42)




# Let's look at categories of our first ten training data
for t in X_train.target[:8]:
    print(X_train.target_names[t])
```

```
comp.graphics
comp.graphics
rec.motorcycles
comp.graphics
sci.med
sci.electronics
sci.electronics
comp.graphics
```

[ ]:
```
#printing target
print(X_train.target_names)
```

```
['comp.graphics', 'rec.motorcycles', 'sci.electronics', 'sci.med']
```

[ ]:
```
#getting first data set
#print("\n".join(X_train.data[0:1]))
```

[ ]:
```
#getting first target label
print(X_train.target_names[X_train.target[4]])

#this will give target value index as accordance with line 11.
print(X_train.target[X_train.target[4]])

# fifth dataset has this label in the form of 0 to 3 .
#Target 5 answer =2 means sci.electronics
print(X_train.target[4])
# This only contains name array of total labels
print(X_train.target_names)
```

```
sci.med
0
3
['comp.graphics', 'rec.motorcycles', 'sci.electronics', 'sci.med']
```

[ ]:
```
# Builds a dictionary of features and transforms documents to feature vectors␣
 ↪and convert our text documents to a
# matrix of token counts (CountVectorizer)
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train.data)

# transform a count matrix to a normalized tf-idf representation (tf-idf␣
 ↪transformer)
#this is a theory topic and I have studied and build concept around it
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
```

[ ]:
```
# training our classifier ; train_data.target will be having numbers assigned␣
 ↪for each category in train data
model = MultinomialNB().fit(X_train_tfidf, X_train.target)

# Input Data to predict their classes of the given categories
docs_new = ['I have a Honda 125.', 'I have a GTX 1050 GPU card']
# building up feature vector of our input
X_new_counts = count_vect.transform(docs_new)
# We call transform instead of fit_transform because it's already been fit
X_new_tfidf = tfidf_transformer.transform(X_new_counts)
```

[ ]:
```
# predicting the category of our input text: Will give out number for category
predicted = model.predict(X_new_tfidf)
```

```
for doc, category in zip(docs_new, predicted):
    print('%r => %s' % (doc, X_train.target_names[category]))
```

```
'I have a Honda 125.' => rec.motorcycles
'I have a GTX 1050 GPU card' => comp.graphics
```

[ ]:
```
# We can use Pipeline to add vectorizer -> transformer -> classifier all in a␣
 ↪one compound classifier
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
])
# Fitting our train data to the pipeline
z=pipeline.fit(X_train.data, X_train.target)

# Test data
X_test = fetch_20newsgroups(subset='test',
                            categories=categories, shuffle=True,␣
 ↪random_state=42)

# Predicting our test data
prediction = pipeline.predict(X_test.data)
print('We got an accuracy of',np.mean(prediction == X_test.target)*100, '% over␣
 ↪the test data.')
```

```
We got an accuracy of 91.49746192893402 % over the test data.
```

[ ]:
```
from sklearn.metrics import accuracy_score

# Compare with side p rakhi we test vs predicted test
score = accuracy_score(X_test.target,prediction,normalize=False)
print("The accuracy score of model when compared with predicted test and␣
 ↪original test values is",score)
```

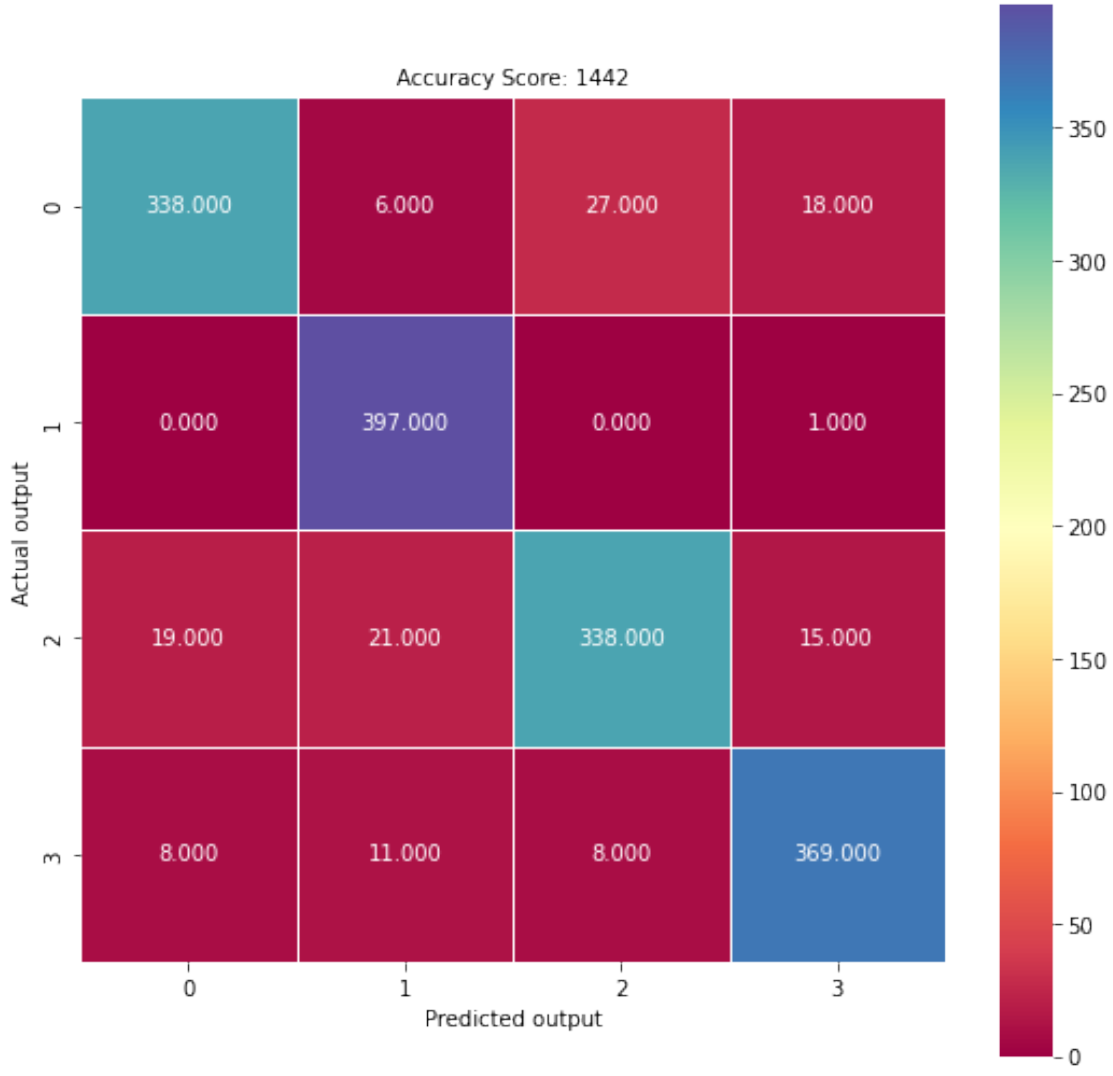
```
The accuracy score of model when compared with predicted test and original test
values is 1442
```

[ ]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(X_test.target, prediction)
cm
```

[ ]:
```
array([[338,   6,  27,  18],
       [  0, 397,   0,   1],
       [ 19,  21, 338,  15],
       [  8,  11,   8, 369]], dtype=int64)
```

[ ]:
```
# Heatmap to visualize COnfusion Matrix
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_sampletitle= 'Accuracy Score: {0}'.format(score)
plt.title(all_sampletitle,size =10)
```

[ ]:
```
Text(0.5, 1.0, 'Accuracy Score: 1442')
```



[ ]:
```

```

[ ]:
```
from sklearn import datasets
#load datasets
cancer = datasets.load_breast_cancer()
# print the names of 13 features
#print("Feature values in data set ",cancer.data)
print("\n \n Feature names",cancer.feature_names)
# jo output hyan unki names
print("\n \n Labels names",cancer.target_names)
# jo label benign and malignnat hyan unki values
# 0 malignant 1 benign
#print("Labels Values",cancer.target)

# getting to know the array size of dataset
print("\n The shape of the data matrix is ",cancer.data.shape)
# print top 2 records
#print(cancer.data[0:2])
```

```
Feature names ['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 names ['malignant' 'benign']

The shape of the data matrix is  (569, 30)
```

[ ]:
```
# Test train split in our dataset
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=0.2, random_state=42)
# import svm model
from sklearn import svm
# Create a svm classifier
model = svm.SVC(kernel='linear') # Linear Kernel
# train the model using the training sets
model.fit(X_train,y_train)
# prediction on test data
y_pred = model.predict(X_test)
```

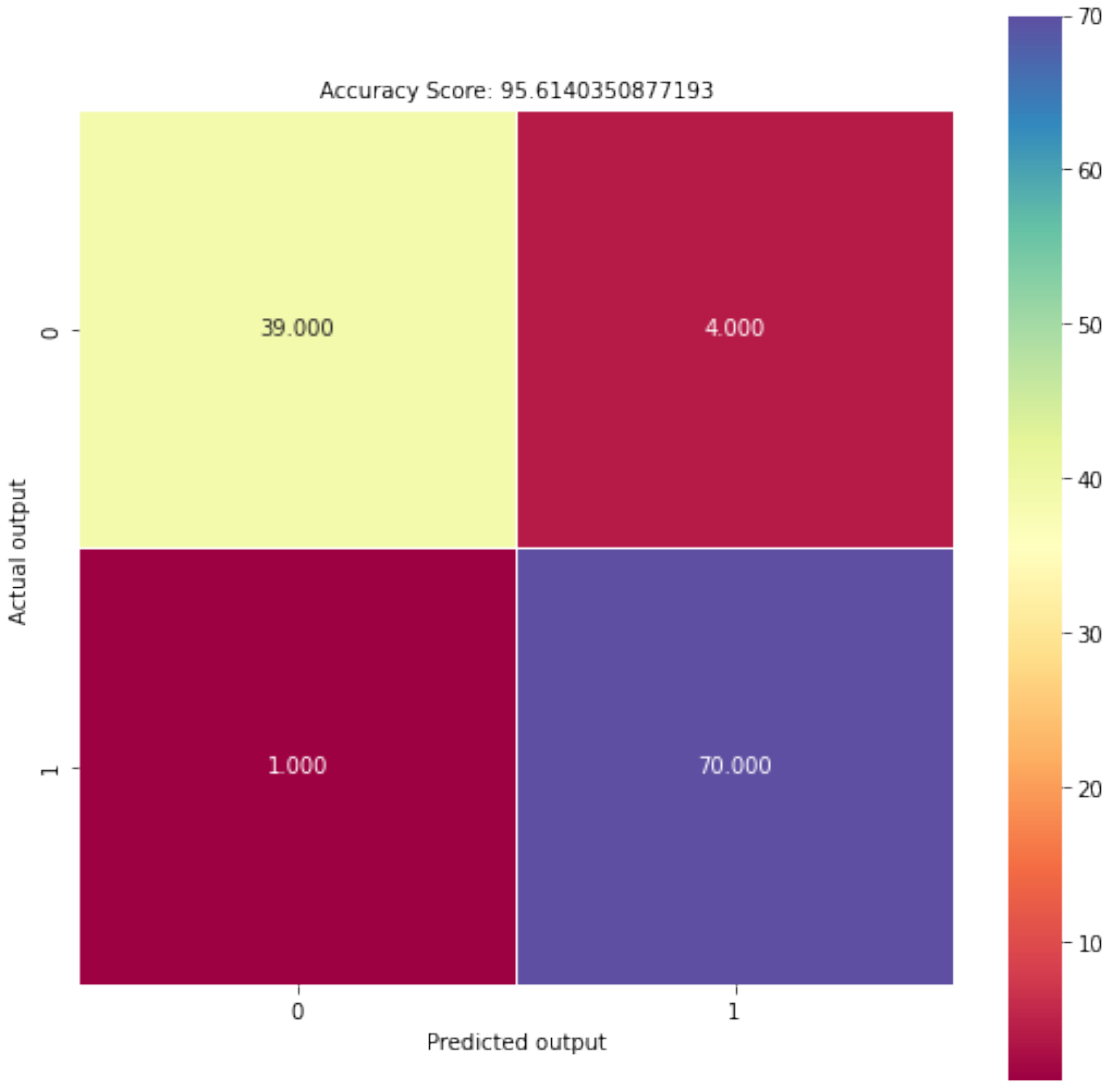
[ ]:
```
from sklearn import metrics
#accuracy score
score = metrics.accuracy_score(y_test,y_pred)
print("The accuracy score of above is ",score)
# Model Precision
# what percentage of positve tuples are labelled as such
print("Precision of above data set is :",metrics.precision_score(y_test,y_pred))
# Mode Recall
# What percentage of positive tuples are labelled as such
print("Recall score is ",metrics.recall_score(y_test,y_pred))
```

```
The accuracy score of above is  0.956140350877193
Precision of above data set is : 0.9459459459459459
Recall score is  0.9859154929577465
```

[ ]:
```
# confusion matrix
cm = metrics.confusion_matrix(y_test,y_pred)
print(cm)
import seaborn as sns
import matplotlib.pyplot as plt
# Heatmap to visualize COnfusion Matrix
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_sampletitle= 'Accuracy Score: {0}'.format(score*100)
plt.title(all_sampletitle,size =10)
```

```
[[39  4]
 [ 1 70]]
```

[ ]:
```
Text(0.5, 1.0, 'Accuracy Score: 95.6140350877193')
```



[ ]:

25 Unsupervised Learning

25.1 Importing Libraries

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

25.2 Importing data set

[ ]:
```
flower = sns.load_dataset('iris')
print(flower.head())
```

```
   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
```

25.3 Remove the Target Feature, we will deal with Descriptive features only.

[ ]:
```
descriptive_features = flower.iloc[:,:-1].values
descriptive_features
```

[ ]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
```

```
[5.4, 3.4, 1.5, 0.4],
[5.2, 4.1, 1.5, 0.1],
[5.5, 4.2, 1.4, 0.2],
[4.9, 3.1, 1.5, 0.2],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.6, 1.4, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
```

```
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
```

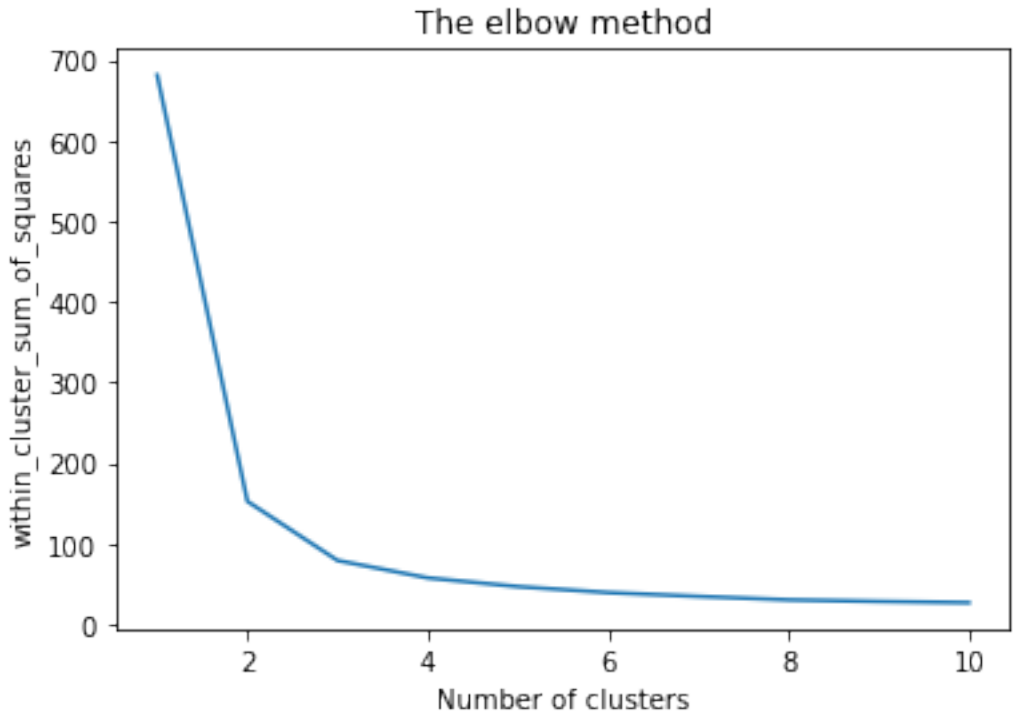
```
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

25.4 Finding the optimum number of clusters for k-means classification

The K-means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum-of-squares criterion. Inertia can be recognized as a measure of how internally coherent clusters are. This is what the KMeans tries to minimize with each iteration.

[ ]:
```
#  Elbow Method
from sklearn.cluster import KMeans
within_cluster_sum_of_squares = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init␣
 ↪= 10, random_state = 0)
    kmeans.fit(descriptive_features)
    within_cluster_sum_of_squares.append(kmeans.inertia_)

#Plotting the results onto a line graph, allowing us to observe 'The elbow'
plt.plot(range(1, 11), within_cluster_sum_of_squares)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('within_cluster_sum_of_squares') #within cluster sum of squares
```

```
plt.show()
```



25.5 Apply K-means Clustering

[ ]:
```
#Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init =␣
 ↪10, random_state = 0)
y_prediction = kmeans.fit_predict(descriptive_features)
```

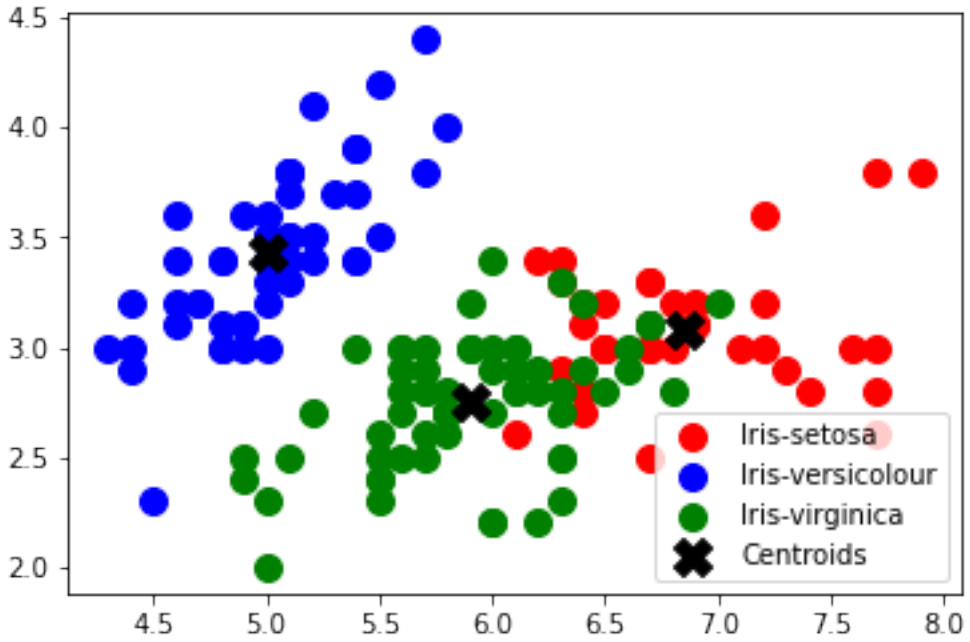
25.6 Visualizing the Clusters

25.6.1 Plotting on the basis of Sepal Features

[ ]:
```
#Visualising the clusters
plt.scatter(descriptive_features[y_prediction == 0, 0],␣
 ↪descriptive_features[y_prediction == 0, 1], s = 100, c = 'red', label =␣
 ↪'Iris-setosa')
plt.scatter(descriptive_features[y_prediction == 1, 0],␣
 ↪descriptive_features[y_prediction == 1, 1], s = 100, c = 'blue', label =␣
 ↪'Iris-versicolour')
plt.scatter(descriptive_features[y_prediction == 2, 0],␣
 ↪descriptive_features[y_prediction == 2, 1], s = 100, c = 'green', label =␣
 ↪'Iris-virginica')
```

```
#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],␣
 ↪marker='X', s = 200, c = 'black', label = 'Centroids')

plt.legend()
```

[ ]: <matplotlib.legend.Legend at 0x1da1d41e640>

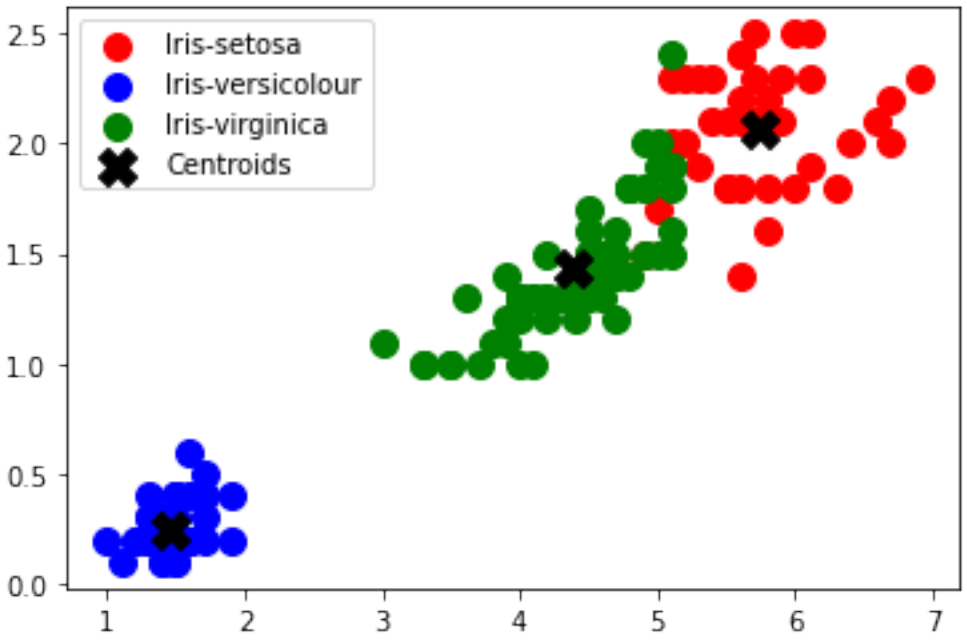


25.6.2 Plotting on the basis of Petal Features

[ ]:
```
#Visualising the clusters
plt.scatter(descriptive_features[y_prediction == 0, 2],␣
 ↪descriptive_features[y_prediction == 0, 3], s = 100, c = 'red', label =␣
 ↪'Iris-setosa')
plt.scatter(descriptive_features[y_prediction == 1, 2],␣
 ↪descriptive_features[y_prediction == 1, 3], s = 100, c = 'blue', label =␣
 ↪'Iris-versicolour')
plt.scatter(descriptive_features[y_prediction == 2, 2],␣
 ↪descriptive_features[y_prediction == 2, 3], s = 100, c = 'green', label =␣
 ↪'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 2], kmeans.cluster_centers_[:,3],␣
 ↪marker='X', s = 200, c = 'black', label = 'Centroids')
```

```
plt.legend()
```

[ ]: <matplotlib.legend.Legend at 0x1da1d321910>



25.7 Match Accuracy with Known Labels i.e Species in Iris Dataset

[ ]:
```
target_feature = flower.iloc[:,-1:]
target_feature
```

[ ]:
```
        species
0        setosa
1        setosa
2        setosa
3        setosa
4        setosa
..          ...
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 1 columns]
```

25.8 Changing categorical data into Numeric Data

[ ]:
```
target_feature = target_feature.replace('setosa',1)
target_feature = target_feature.replace('versicolor',0)
target_feature = target_feature.replace('virginica',2)

target_feature
```

[ ]:
```
     species
0          1
1          1
2          1
3          1
4          1
..       …
145        2
146        2
147        2
148        2
149        2

[150 rows x 1 columns]
```

25.9 Measuring Accuracy

[ ]:
```
# Measuring score
from sklearn.metrics import accuracy_score
score = accuracy_score(target_feature,y_prediction)
print('Accuracy Score of K-means Classification is:', score)
```

```
Accuracy Score of K-means Classification is: 0.44
```

25.10 Confusion Matrix

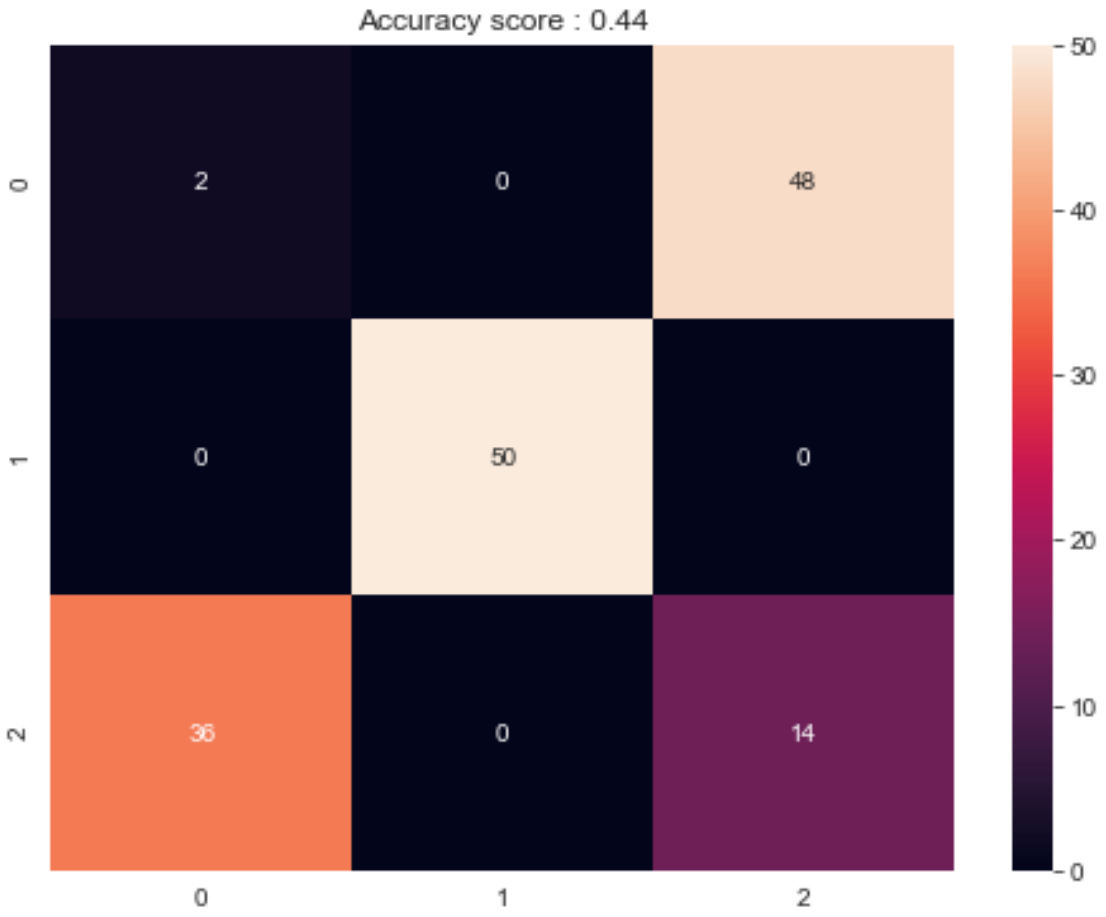
[ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(target_feature,y_prediction)
cm
```

[ ]:
```
array([[ 2,  0, 48],
       [ 0, 50,  0],
       [36,  0, 14]], dtype=int64)
```

[ ]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot = True,)
plt.ylabel = 'Actual Output'
plt.xlabel = 'Predicted Output'
cm_title = 'Accuracy score : {0}'.format(score)
```

```
plt.title(cm_title)
```

[ ]: Text(0.5, 1.0, 'Accuracy score : 0.44')



25.11 K mediods

[ ]:
```
#pip install scikit-learn-extra
```

[ ]:
```
#Applying kmeans to the dataset / Creating the kmeans classifier
from sklearn_extra.cluster import KMedoids
k_medions = KMedoids(n_clusters=3, metric='euclidean', method='alternate',␣
 ↪init='heuristic', max_iter=600, random_state=False)
y_prediction = k_medions.fit_predict(descriptive_features)
```

[ ]:
```
y_prediction
```

[ ]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
```

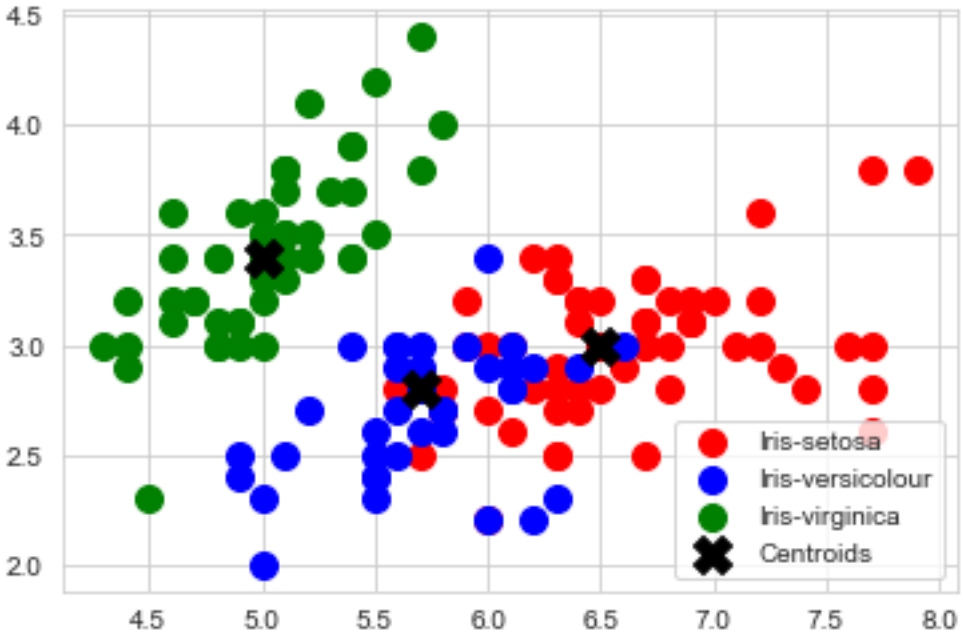
```
2, 2, 2, 2, 2, 2, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

25.12 Visualizing the Clusters

25.12.1 Sepal Features

[ ]:
```
#Visualising the clusters
plt.scatter(descriptive_features[y_prediction == 0, 0],␣
 ↪descriptive_features[y_prediction == 0, 1], s = 100, c = 'red', label =␣
 ↪'Iris-setosa')
plt.scatter(descriptive_features[y_prediction == 1, 0],␣
 ↪descriptive_features[y_prediction == 1, 1], s = 100, c = 'blue', label =␣
 ↪'Iris-versicolour')
plt.scatter(descriptive_features[y_prediction == 2, 0],␣
 ↪descriptive_features[y_prediction == 2, 1], s = 100, c = 'green', label =␣
 ↪'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(k_medions.cluster_centers_[:, 0], k_medions.cluster_centers_[:,1],␣
 ↪marker='X', s = 200, c = 'black', label = 'Centroids')

plt.legend()
```

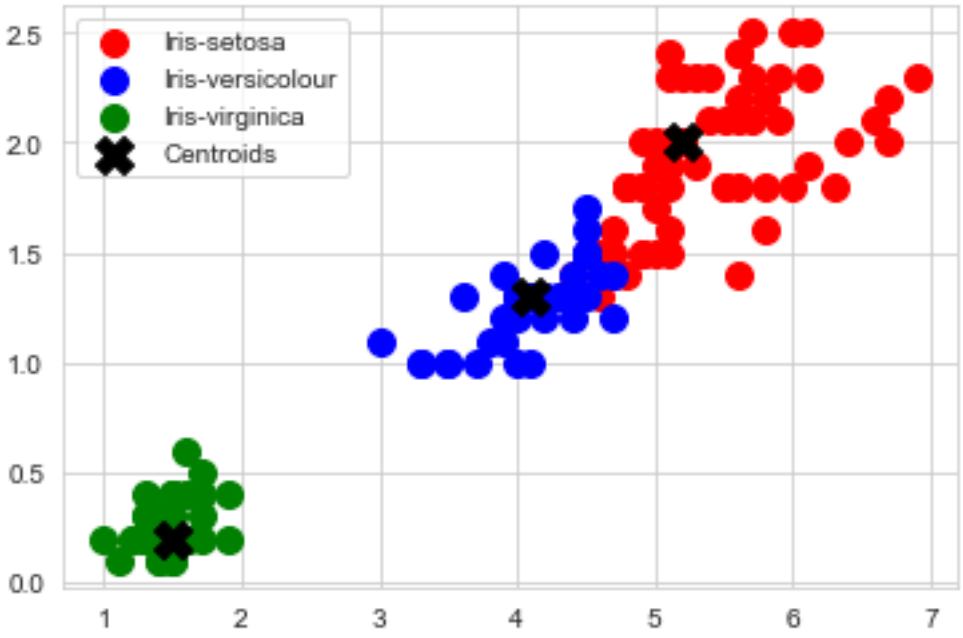
[ ]:
```
<matplotlib.legend.Legend at 0x1da1de45c10>
```



25.12.2 Plotting on the basis of Petal Features

[ ]:
```
#Visualising the clusters
plt.scatter(descriptive_features[y_prediction == 0, 2],␣
 ↪descriptive_features[y_prediction == 0, 3], s = 100, c = 'red', label =␣
 ↪'Iris-setosa')
plt.scatter(descriptive_features[y_prediction == 1, 2],␣
 ↪descriptive_features[y_prediction == 1, 3], s = 100, c = 'blue', label =␣
 ↪'Iris-versicolour')
plt.scatter(descriptive_features[y_prediction == 2, 2],␣
 ↪descriptive_features[y_prediction == 2, 3], s = 100, c = 'green', label =␣
 ↪'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(k_medions.cluster_centers_[:, 2], k_medions.cluster_centers_[:,3],␣
 ↪marker='X', s = 200, c = 'black', label = 'Centroids')


plt.legend()
```

[ ]:
```
<matplotlib.legend.Legend at 0x1da1deb2d60>
```



25.13 Match Accuracy with Known Labels i.e Species in Iris Dataset

[ ]:
```
target_feature = flower.iloc[:,-1:]
target_feature
```

[ ]:
```
        species
0        setosa
1        setosa
2        setosa
3        setosa
4        setosa
..          …
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 1 columns]
```

25.14 Changing categorical data into Numeric Data

[ ]:
```
target_feature = target_feature.replace('setosa',2)
target_feature = target_feature.replace('virginica',0)
target_feature = target_feature.replace('versicolor',1)
target_feature
```

[ ]:
```
     species
0          2
1          2
2          2
3          2
4          2
..       …
145        0
146        0
147        0
148        0
149        0

[150 rows x 1 columns]
```

[ ]:
```
# Measuring score
from sklearn.metrics import accuracy_score
score = accuracy_score(target_feature,y_prediction)
print('Accuracy Score of K-medoids Clustering is:', score)
```

```
Accuracy Score of K-medoids Clustering is: 0.9066666666666666
```

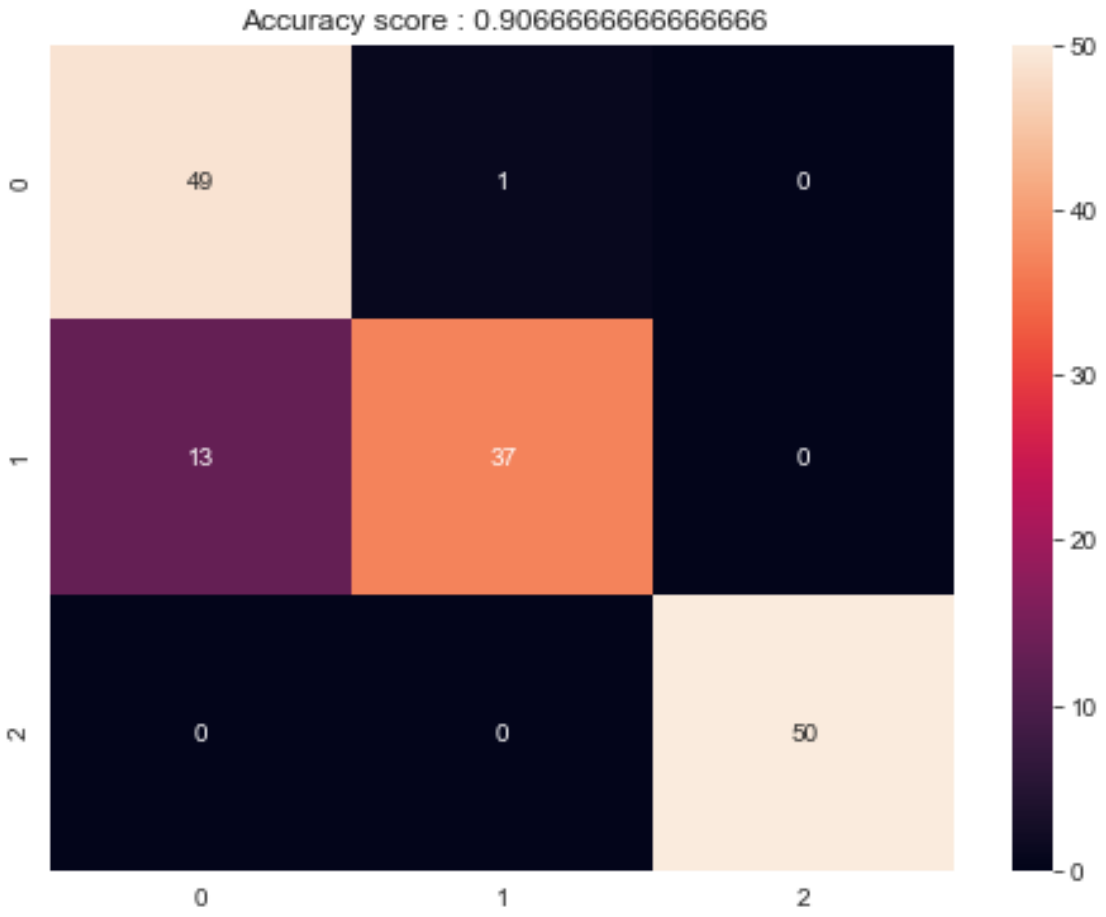
25.15 Confusion Matrix

[ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(target_feature,y_prediction)
cm
```

[ ]:
```
array([[49,  1,  0],
       [13, 37,  0],
       [ 0,  0, 50]], dtype=int64)
```

[ ]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot = True,)
plt.ylabel = 'Actual Output'
plt.xlabel = 'Predicted Output'
cm_title = 'Accuracy score : {0}'.format(score)
plt.title(cm_title)
```

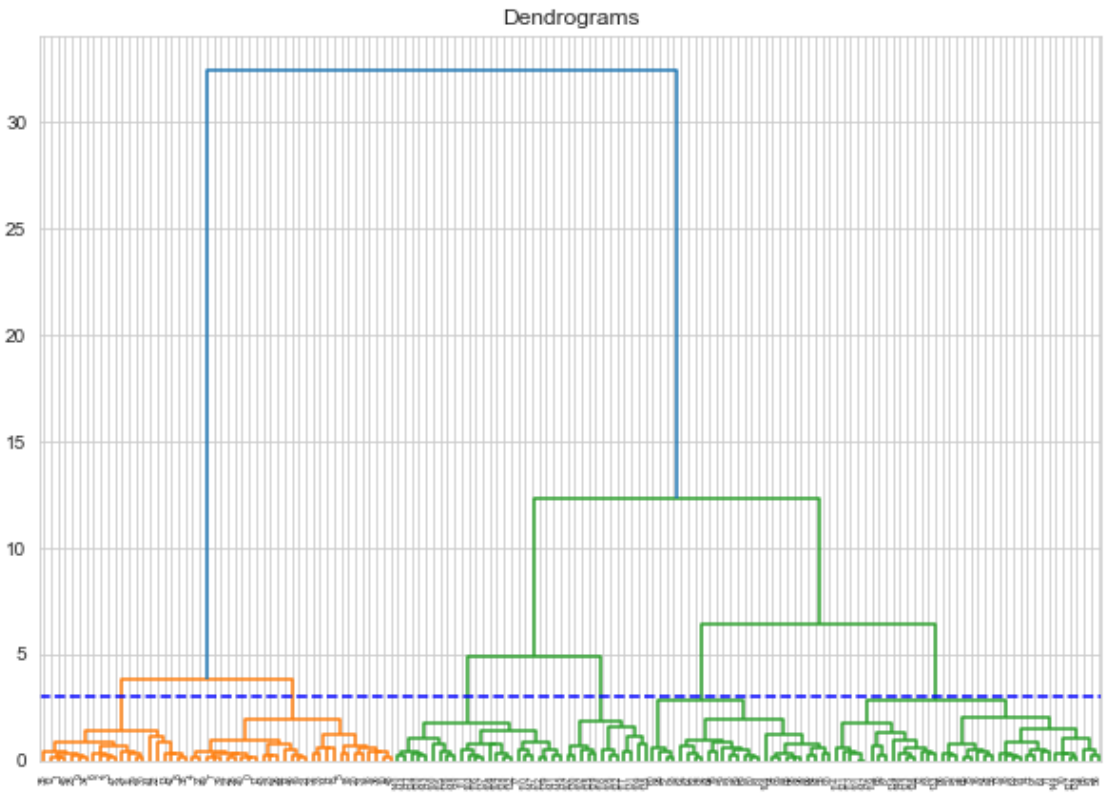
[ ]:
```
Text(0.5, 1.0, 'Accuracy score : 0.9066666666666666')
```



25.16 Dendograms

[ ]:
```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(descriptive_features, method='ward'))
plt.axhline(y=3, color='b', linestyle='--')
```

[ ]:
```
<matplotlib.lines.Line2D at 0x1da1e0cd2b0>
```



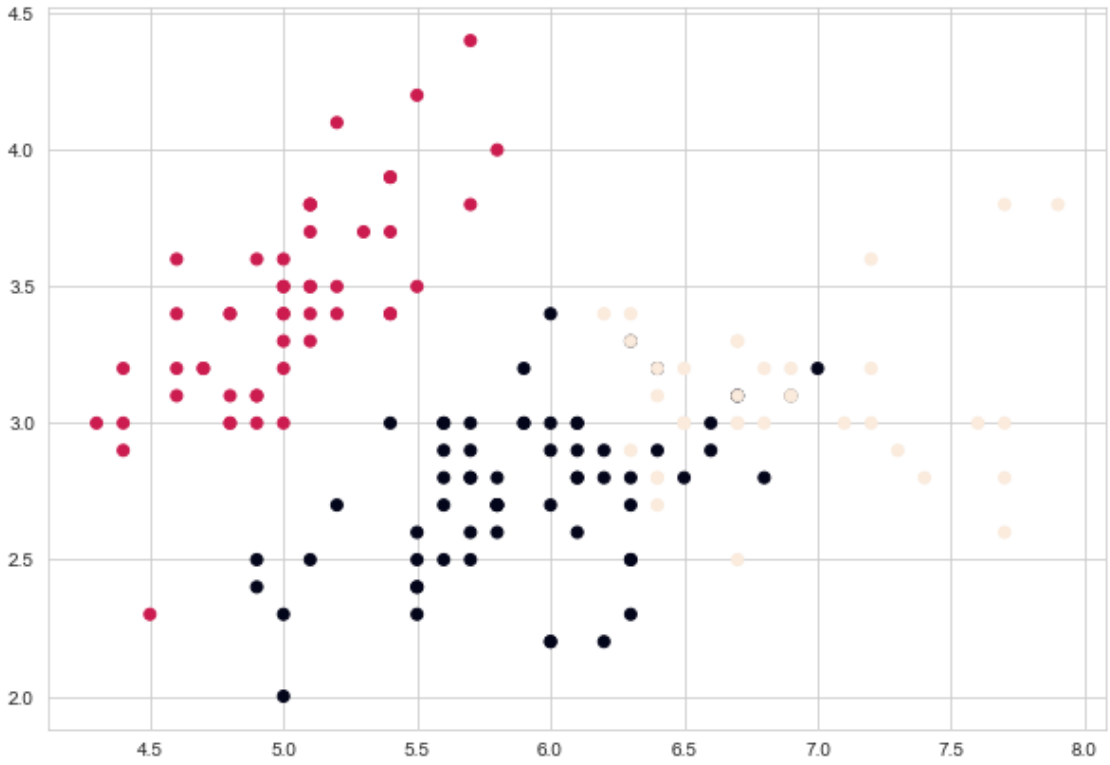
25.17 Aglomerative Clustering

[ ]:
```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean',␣
 ↪linkage='ward')
y_prediction=cluster.fit_predict(descriptive_features)
y_prediction
```

[ ]:
```
array([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, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2,
2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0], dtype=int64)
```

[ ]:
```
plt.figure(figsize=(10, 7))
plt.scatter(descriptive_features[:,:-3], descriptive_features[:,1:-2],␣
↪c=cluster.labels_)

plt.show()
```



25.18 Getting Target Feature

[ ]:
```
target_feature = flower.iloc[:,-1:]
target_feature
```

[ ]:
```
      species
0      setosa
1      setosa
2      setosa
3      setosa
4      setosa
..        …
```

```
145  virginica
146  virginica
147  virginica
148  virginica
149  virginica

[150 rows x 1 columns]
```

25.19 Changing categorical data into Numeric Data

[ ]:
```
target_feature = target_feature.replace('setosa',1)
target_feature = target_feature.replace('virginica',2)
target_feature = target_feature.replace('versicolor',0)
target_feature
```

[ ]:
```
     species
0          1
1          1
2          1
3          1
4          1
..       …
145        2
146        2
147        2
148        2
149        2

[150 rows x 1 columns]
```

[ ]:
```
# Measuring score
from sklearn.metrics import accuracy_score
score = accuracy_score(target_feature,y_prediction)
print('Accuracy Score of Aglomerative Clustering is:', score)
```

```
Accuracy Score of Aglomerative Clustering is: 0.8933333333333333
```

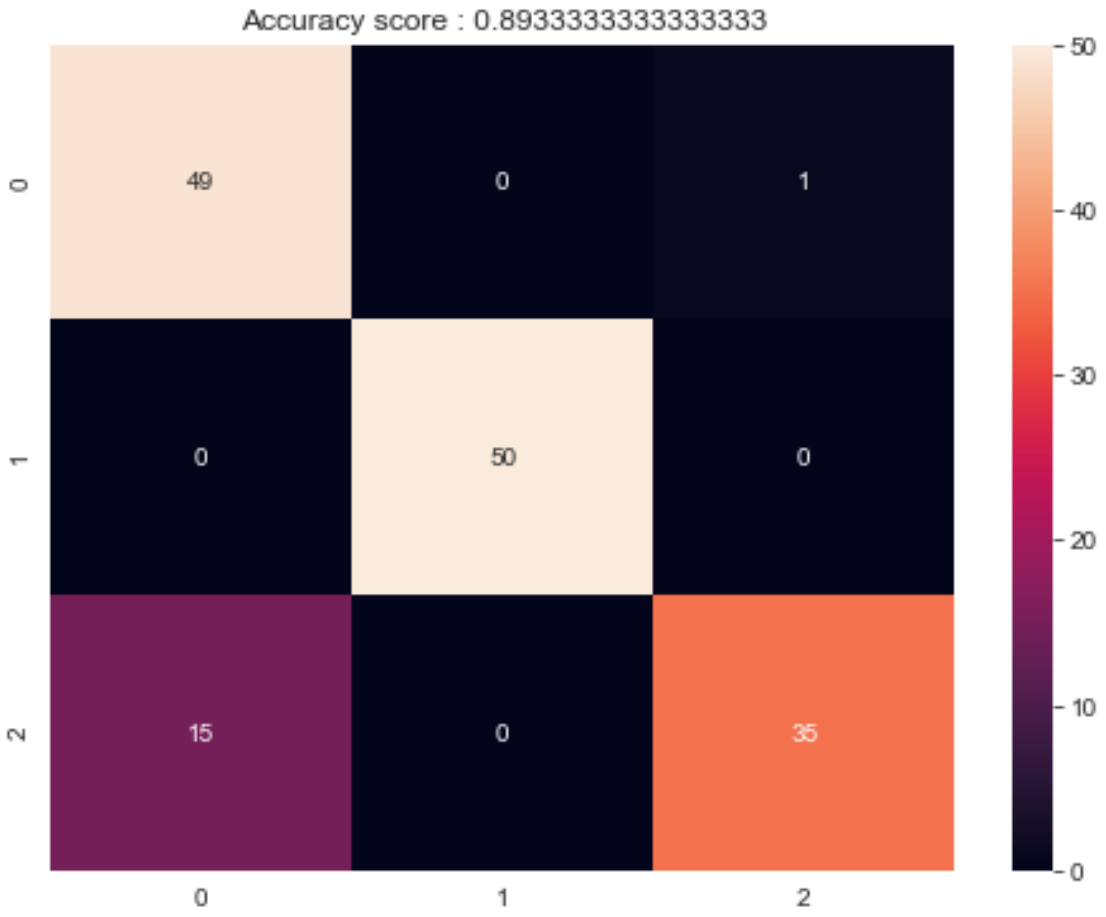
25.20 Confusion Matrix

[ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(target_feature,y_prediction)
cm
```

[ ]:
```
array([[49,  0,  1],
       [ 0, 50,  0],
       [15,  0, 35]], dtype=int64)
```

[ ]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot = True,)
plt.ylabel = 'Actual Output'
plt.xlabel = 'Predicted Output'
cm_title = 'Accuracy score : {0}'.format(score)
plt.title(cm_title)
```

[ ]:
```
Text(0.5, 1.0, 'Accuracy score : 0.8933333333333333')
```



25.21 DB SCAN with Data Points in Circular pattern

[ ]:
```
import math
import matplotlib.pyplot as plt
import matplotlib
```

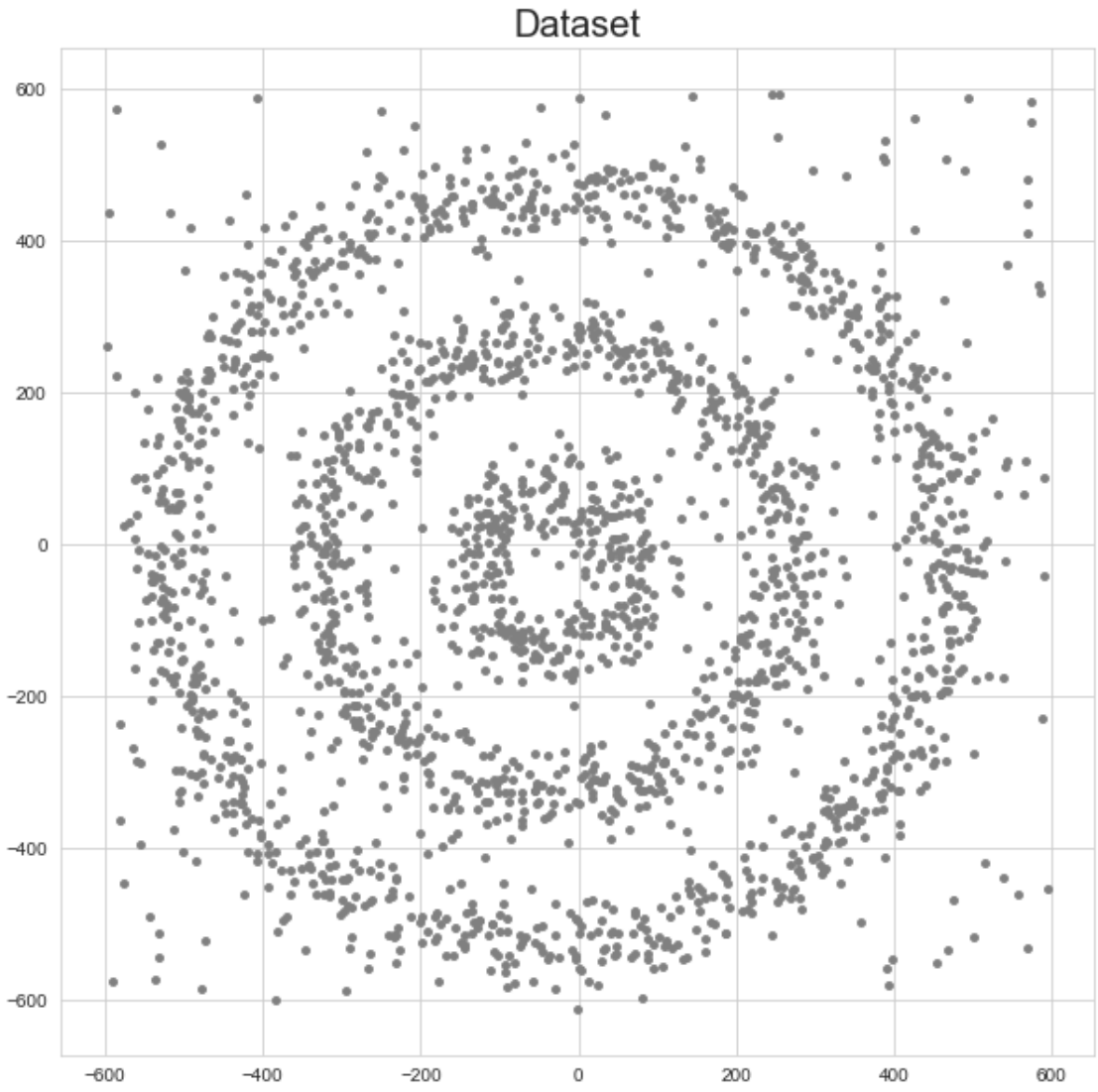
25.21.1 Create Datapoints in a form of Circle

[ ]:
```
np.random.seed(42)

# Function for creating datapoints in the form of a circle
def PointsInCircum(r,n=100):
    return [(math.cos(2*math.pi/n*x)*r+np.random.normal(-30,30),math.sin(2*math.
 ↪pi/n*x)*r+np.random.normal(-30,30)) for x in range(1,n+1)]
```

[ ]:
```
# Creating data points in the form of a circle
df=pd.DataFrame(PointsInCircum(500,1000))
df=df.append(PointsInCircum(300,700))
df=df.append(PointsInCircum(100,300))

# Adding noise to the dataset
df=df.append([(np.random.randint(-600,600),np.random.randint(-600,600)) for i␣
 ↪in range(300)])
```

25.21.2 Plotting datapoints

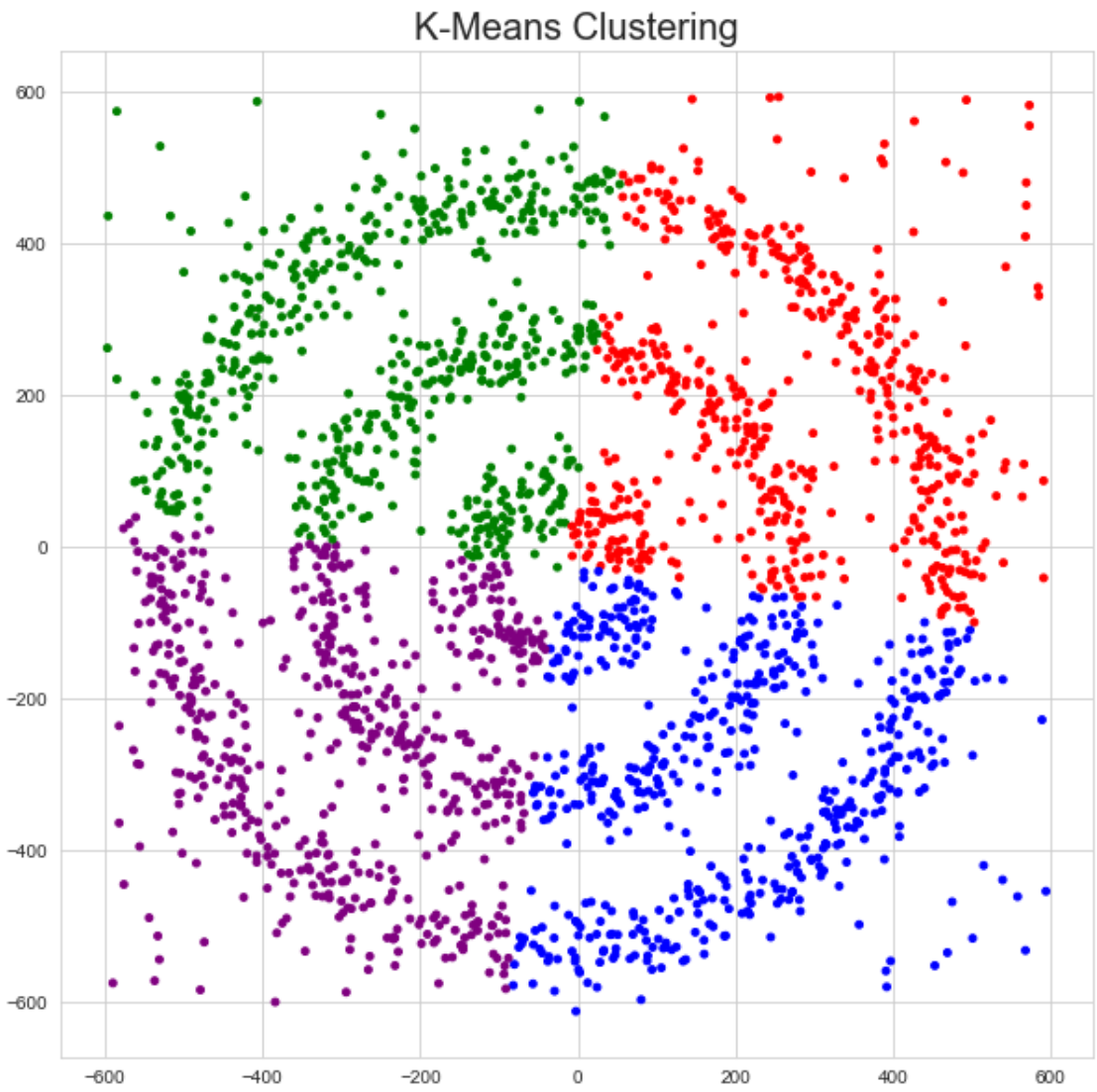
[ ]:
```
plt.figure(figsize=(10,10))
plt.scatter(df[0],df[1],s=15,color='grey')
plt.title('Dataset',fontsize=20)
plt.ylabel = 'Actual Output'
plt.xlabel = 'Predicted Output'
plt.show()
```



25.21.3 Clustering on the basis of K-Means

[ ]:
```
from sklearn.cluster import KMeans
k_means=KMeans(n_clusters=4,random_state=42)
k_means.fit(df[[0,1]])
df['KMeans_labels']=k_means.labels_

# Plotting resulting clusters
colors=['purple','red','blue','green']
plt.figure(figsize=(10,10))
plt.scatter(df[0],df[1],c=df['KMeans_labels'],cmap=matplotlib.colors.
  ↪ListedColormap(colors),s=15)
plt.title('K-Means Clustering',fontsize=20)
```

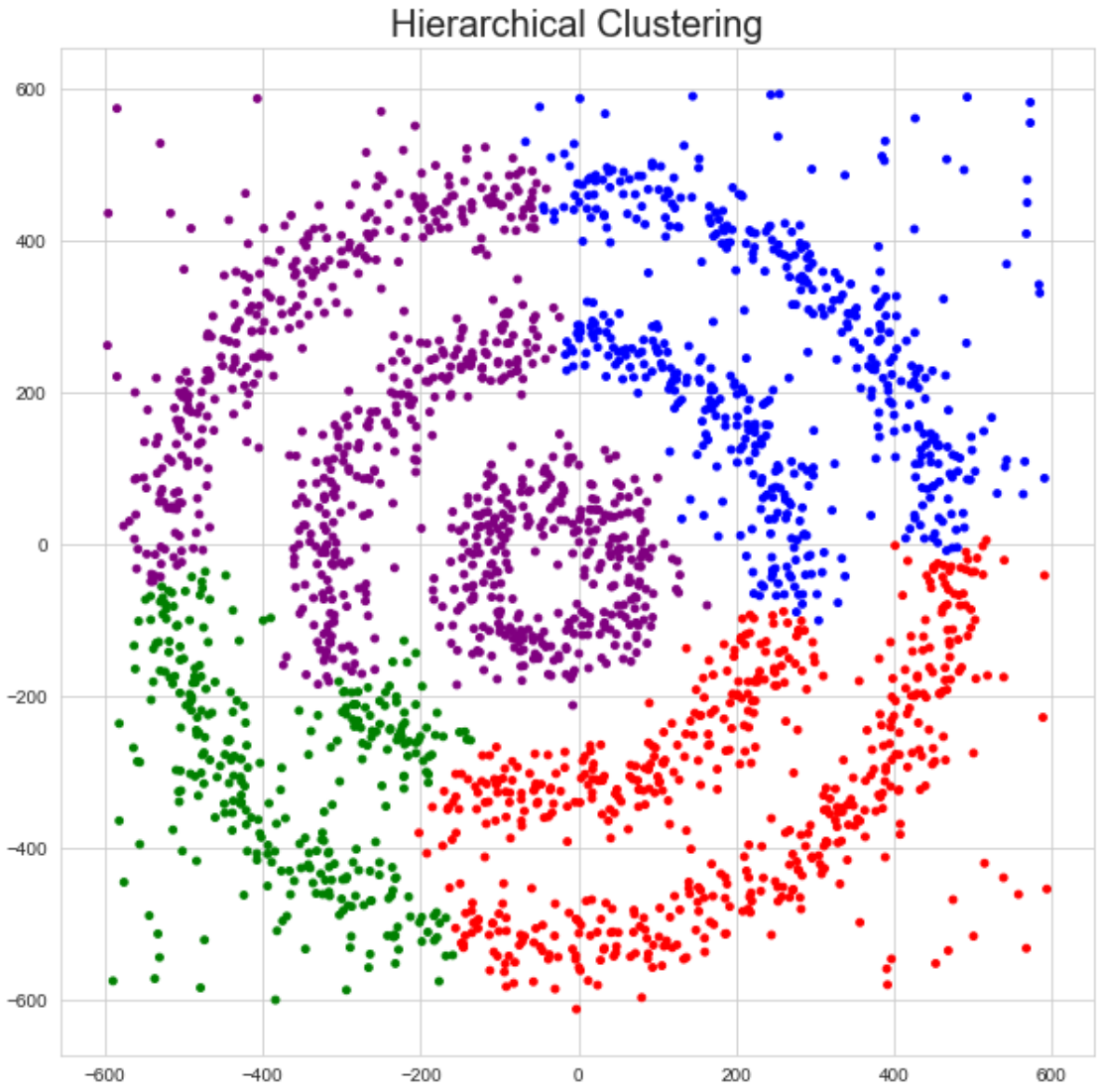
```
plt.ylabel = 'Actual Output'
plt.xlabel = 'Predicted Output'
plt.show()
```



25.21.4 Clustering Based on Aglomerative Clustering

[ ]:
```
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=4, affinity='euclidean')
model.fit(df[[0,1]])
df['HR_labels']=model.labels_

# Plotting resulting clusters
plt.figure(figsize=(10,10))
```

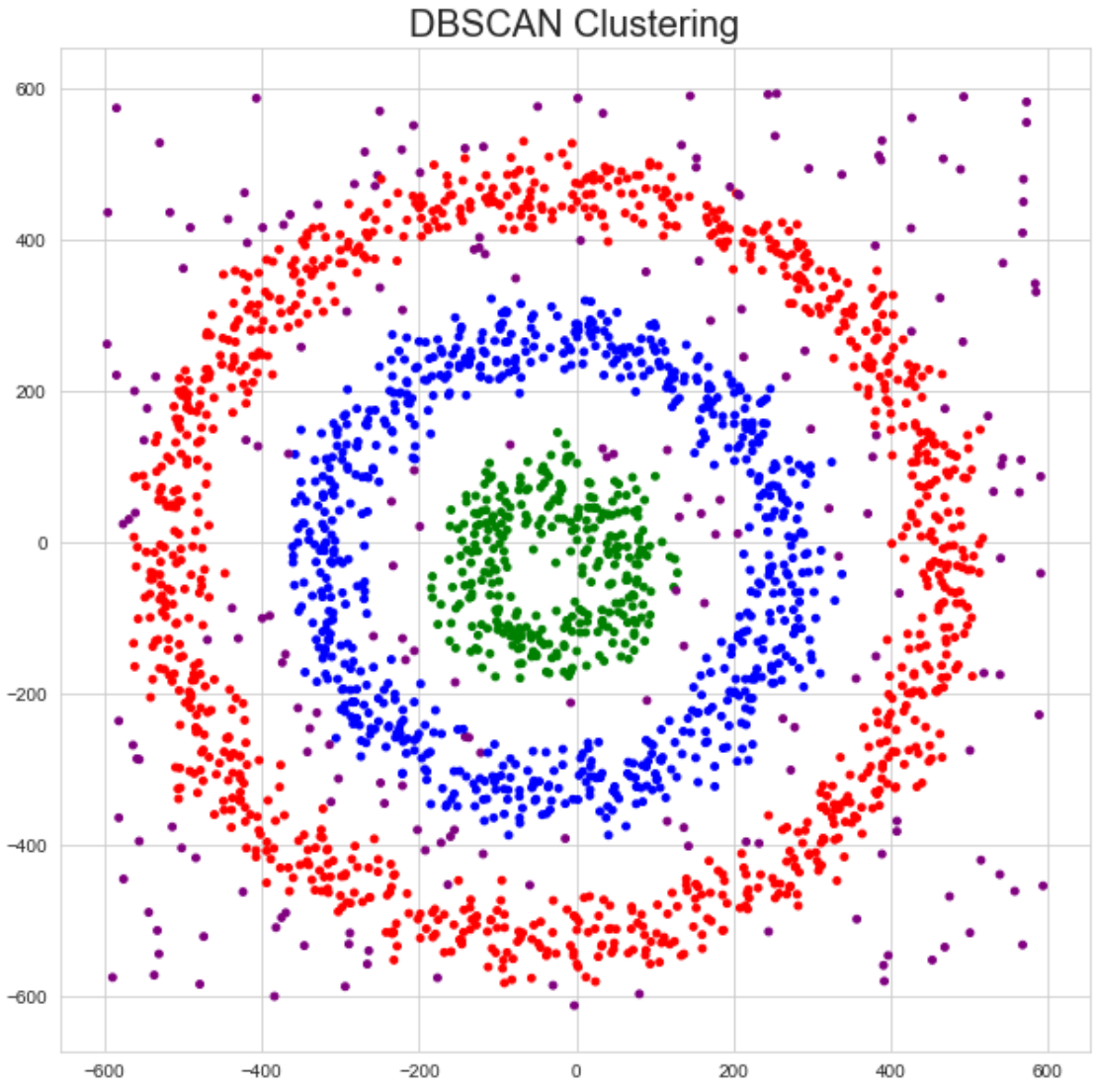
```
plt.scatter(df[0],df[1],c=df['HR_labels'],cmap=matplotlib.colors.
  ↪ListedColormap(colors),s=15)
plt.title('Hierarchical Clustering',fontsize=20)
plt.show()
```



25.21.5 DB Scan method

[ ]:
```
from sklearn.neighbors import NearestNeighbors
neigh = NearestNeighbors(n_neighbors=2)
nbrs = neigh.fit(df[[0,1]])
distances, indices = nbrs.kneighbors(df[[0,1]])

from sklearn.cluster import DBSCAN
```

```
dbscan_opt=DBSCAN(eps=30,min_samples=6)
dbscan_opt.fit(df[[0,1]])

df['DBSCAN_opt_labels']=dbscan_opt.labels_
df['DBSCAN_opt_labels'].value_counts()
# Plotting the resulting clusters
plt.figure(figsize=(10,10))
plt.scatter(df[0],df[1],c=df['DBSCAN_opt_labels'],cmap=matplotlib.colors.
  ↪ListedColormap(colors),s=15)
plt.title('DBSCAN Clustering',fontsize=20)
plt.show()
```



26 Neural Networks

[ ]:

27 Neural Networks

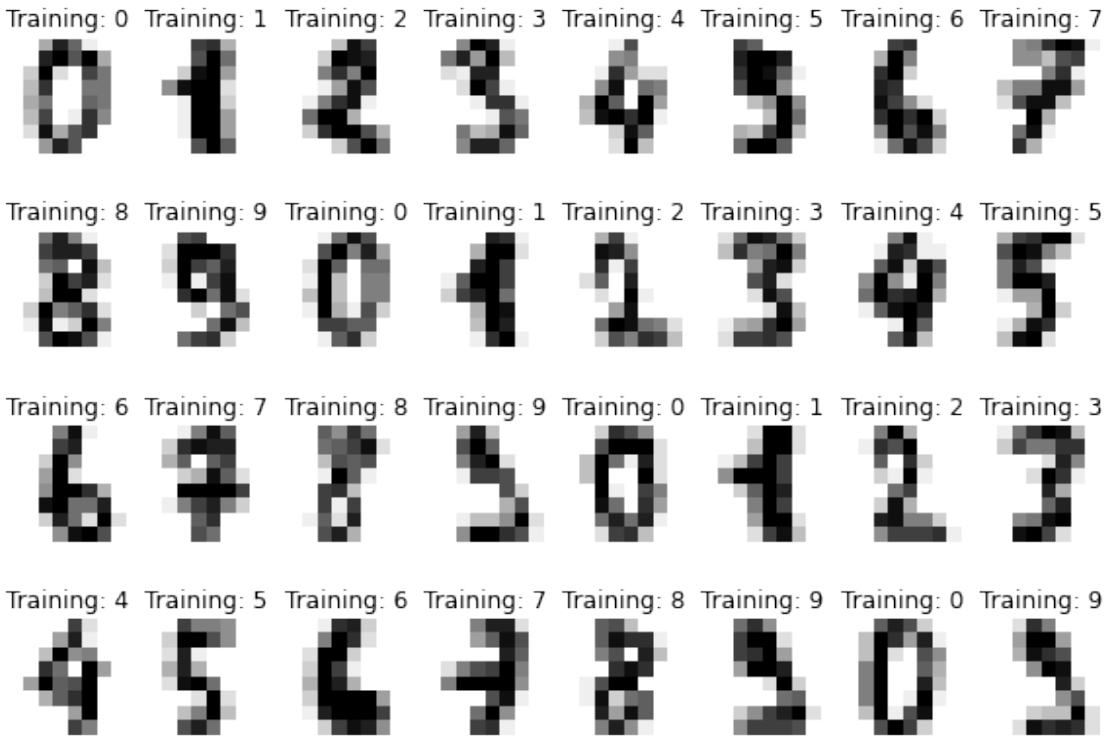
27.1 Data set for digit number recognition

The data set is made of 8x8 images of digits. We start with loading the dataset.

[ ]:
```
import matplotlib.pyplot as plt

# Import datasets, classifiers and performance metrics
from sklearn import datasets, metrics
from sklearn.neural_network import MLPClassifier
from sklearn import tree

# The digits dataset
digits = datasets.load_digits()
```

Now, let's have a look at some of the first images, stored in the images attribute of the dataset. - If we were working from image files, we could load them using matplotlib.pyplot.imread. Note that each image must have the same size. - For these images, we know which digit they represent: it is given in the 'target' of the dataset.

[ ]:
```
images_and_labels = list(zip(digits.images, digits.target))
plt.figure(figsize=(10,7))
for index, (image, label) in enumerate(images_and_labels[:32]):
    plt.subplot(4, 8, index + 1)
    plt.axis('off')
    plt.imshow(image,cmap=plt.cm.gray_r)
    plt.title('Training: %i' % label)
plt.show()
```



[ ]:
```
len(images_and_labels)
```

[ ]: 1797

We need to do some preprocessing!

[ ]:
```
# To apply a classifier on this data, we need to flatten the image, to
# turn the data in a (samples, feature) matrix:
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
```

Next, let's see what our classification algorithms does:

[ ]:
```
# Create a classifier:

classifier = MLPClassifier(hidden_layer_sizes=(100,))

# We learn the digits on the first half of the digits
classifier.fit(data, digits.target)

# Now predict the value of the digit on the second half:
expected = digits.target
predicted = classifier.predict(data)
```

Lets show the predictions:

[ ]:
```
classifier
```

[ ]:
```
MLPClassifier()
```

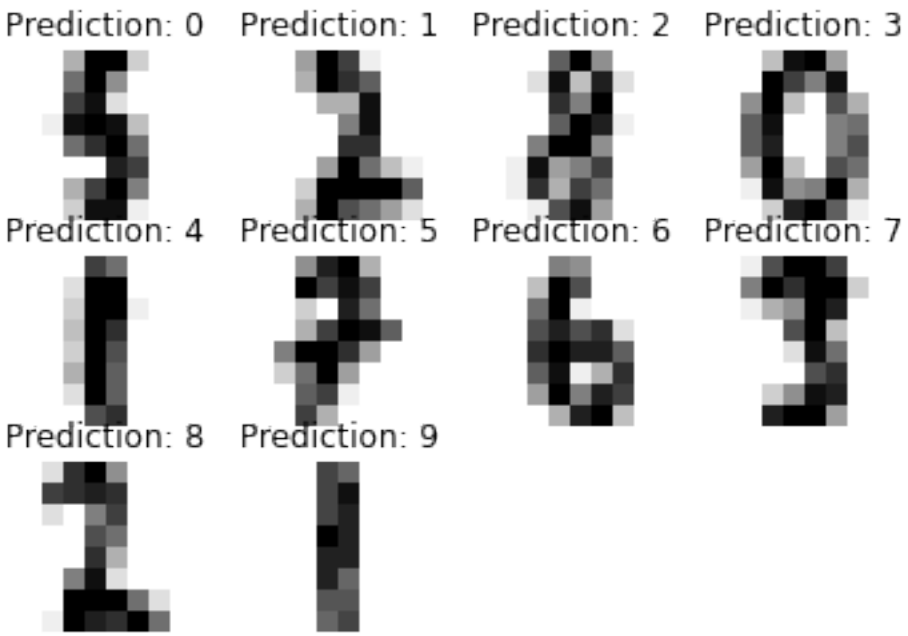
[ ]:
```
n_samples,predicted.size
```

```
(1797, 1797)
```

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
index = 0
misclassified_indicies = []
for label, predict in zip(expected, predicted):
 if label != predict:
  misclassified_indicies.append(index)
  index +=1
number_of_missclassified_data =len(misclassified_indicies)
print('indicies of missclssified data', misclassified_indicies)
print('number of missclassified pictures :',number_of_missclassified_data )
```

```
indicies of missclssified data []
number of missclassified pictures : 0
```

[ ]:
```
images_and_predictions = list(zip(digits.images[9*(n_samples) // 10:],␣
 ↪predicted))
for index, (image, prediction) in enumerate(images_and_predictions[:10]):
    plt.subplot(3, 4, index+1)
    plt.axis('off')
    plt.imshow(image, cmap=plt.cm.gray_r)
    plt.title('Prediction: %i' % prediction)

plt.show()
```



Evaluation metrics can be used to compare your classifications:

[ ]:
```
print("Classification report for classifier %s:\n%s\n"
      % (classifier, metrics.classification_report(expected, predicted)))
print("Confusion matrix:\n%s" % metrics.confusion_matrix(expected, predicted))
```

```
Classification report for classifier MLPClassifier():
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       178
           1       1.00      1.00      1.00       182
           2       1.00      1.00      1.00       177
           3       1.00      1.00      1.00       183
           4       1.00      1.00      1.00       181
           5       1.00      1.00      1.00       182
           6       1.00      1.00      1.00       181
           7       1.00      1.00      1.00       179
           8       1.00      1.00      1.00       174
           9       1.00      1.00      1.00       180

    accuracy                           1.00      1797
   macro avg       1.00      1.00      1.00      1797
weighted avg       1.00      1.00      1.00      1797


Confusion matrix:
[[178   0   0   0   0   0   0   0   0   0]
```

```
[  0 182   0   0   0   0   0   0   0   0]
[  0   0 177   0   0   0   0   0   0   0]
[  0   0   0 183   0   0   0   0   0   0]
[  0   0   0   0 181   0   0   0   0   0]
[  0   0   0   0   0 182   0   0   0   0]
[  0   0   0   0   0   0 181   0   0   0]
[  0   0   0   0   0   0   0 179   0   0]
[  0   0   0   0   0   0   0   0 174   0]
[  0   0   0   0   0   0   0   0   0 180]]
```

27.2 Splitting into Training and Testing Data

27.3 Importing Digits data set from SKLearn

[ ]:
```
from sklearn.datasets import load_digits

new_digit = load_digits()
type(new_digit)
```

[ ]:
```
sklearn.utils.Bunch
```

[ ]:
```
# View Digits data
new_digit.data.shape
```

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

Data Contains 1797 pictures of size 8x8 i.e 64

27.4 Setting Descriptive and Target features (data = descriptive , target = target in this data set)

[ ]:
```
descriptive_features = new_digit.data
target_feature = new_digit.target

descriptive_features.shape, target_feature.shape
```

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

27.5 Splitting data

[ ]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(descriptive_features,␣
 ↪target_feature, test_size=1/5, random_state=0)
```

[ ]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

[ ]:
```
((1437, 64), (360, 64), (1437,), (360,))
```

27.6 Training Model - Neural Network

27.6.1 10 Hidden Layers

[ ]:
```
# Create a classifier:

classifier_10 = MLPClassifier(hidden_layer_sizes=(10,))

# We learn the digits on the first half of the digits
classifier_10.fit(x_train, y_train)
```

```
/Users/asadtariq/opt/anaconda3/lib/python3.9/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:614:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

[ ]:
```
MLPClassifier(hidden_layer_sizes=(10,))
```

27.7 Predicting test data

[ ]:
```
# Predicting the Test set results
y_pred = classifier_10.predict(x_test)
```

27.8 Accuracy test

[ ]:
```
score_10 = classifier_10.score(x_test,y_test)
print('acuracy of model is :' , score_10)
```

```
acuracy of model is : 0.925
```

27.9 Training Model - Neural Network - Hiidden Layers = 100

[ ]:
```
# Create a classifier:

classifier_100 = MLPClassifier(hidden_layer_sizes=(100,))

# We learn the digits on the first half of the digits
classifier_100.fit(x_train, y_train)
```

[ ]:
```
MLPClassifier()
```

27.10 Predicting test data

[ ]:
```
# Predicting the Test set results
y_pred = classifier_100.predict(x_test)
```

27.11 Accuracy test

[ ]:
```
score_100 = classifier.score(x_test,y_test)
print('acuracy of model is :' , score_100)
```

```
acuracy of model is : 0.9472222222222222
```

27.12 Training Model - Neural Network - Hiidden Layers = 1000

[ ]:
```
# Create a classifier:

classifier_1000 = MLPClassifier(hidden_layer_sizes=(1000,))

# We learn the digits on the first half of the digits
classifier_1000.fit(x_train, y_train)
```

[ ]:
```
MLPClassifier(hidden_layer_sizes=(1000,))
```

27.13 Predicting test data

[ ]:
```
# Predicting the Test set results
y_pred = classifier_1000.predict(x_test)
```

27.14 Accuracy test

[ ]:
```
score_1000 = classifier_1000.score(x_test,y_test)
print('acuracy of model is :' , score_1000)
```

```
acuracy of model is : 0.9861111111111112
```

27.15 Training Model - Neural Network - Hiidden Layers = 10000

[ ]:
```
# Create a classifier:

classifier_10000 = MLPClassifier(hidden_layer_sizes=(10000,))

# We learn the digits on the first half of the digits
classifier_10000.fit(x_train, y_train)
```

[ ]:
```
MLPClassifier(hidden_layer_sizes=(10000,))
```

27.16 Predicting test data

[ ]:
```
# Predicting the Test set results
y_pred = classifier_10000.predict(x_test)
```

27.17 Accuracy test

[ ]:
```
score_10000 = classifier_10000.score(x_test,y_test)
print('acuracy of model is :' , score_10000)
```

```
acuracy of model is : 0.9833333333333333
```

27.18 Training Model - Neural Network - Hiidden Layers = 100000

[ ]:
```
# Create a classifier:

classifier_100000 = MLPClassifier(hidden_layer_sizes=(100000,))

# We learn the digits on the first half of the digits
classifier_100000.fit(x_train, y_train)
```

[ ]:
```
MLPClassifier(hidden_layer_sizes=(100000,))
```

27.19 Predicting test data

[ ]:
```
# Predicting the Test set results
y_pred = classifier_100000.predict(x_test)
```

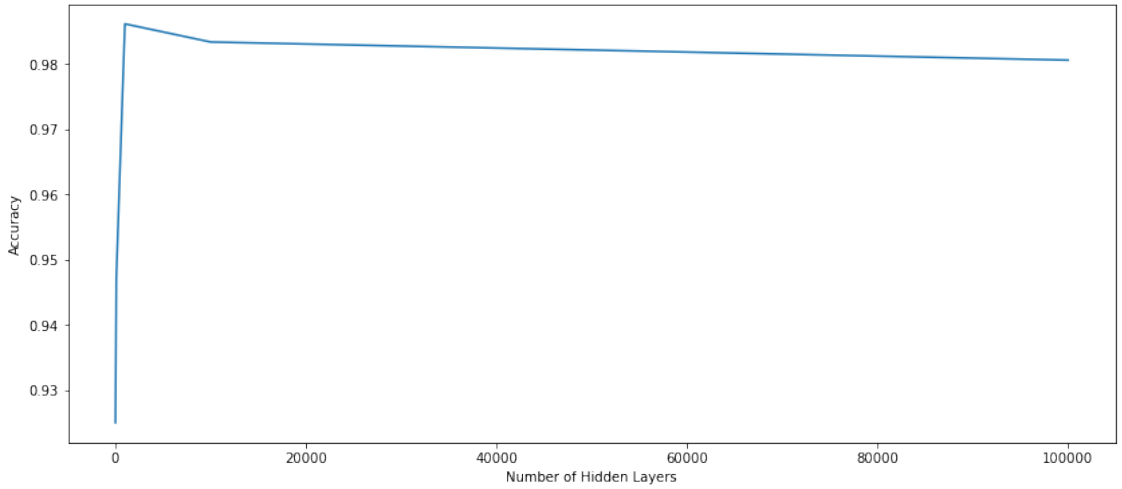
27.20 Accuracy test

[ ]:
```
score_100000 = classifier_100000.score(x_test,y_test)
print('acuracy of model is :' , score_100000)
```

```
acuracy of model is : 0.9805555555555555
```

[ ]:
```
x =[10,100,1000,10000,100000]
y =[score_10,score_100,score_1000,score_10000,score_100000]
```

[ ]:
```
import seaborn as sns
plt.figure(figsize=(14,6))
a =sns.lineplot(x=x, y=y, ci=None)
a.set(xlabel='Number of Hidden Layers', ylabel='Accuracy')
plt.show()
```



27.21 Confusion Matrix

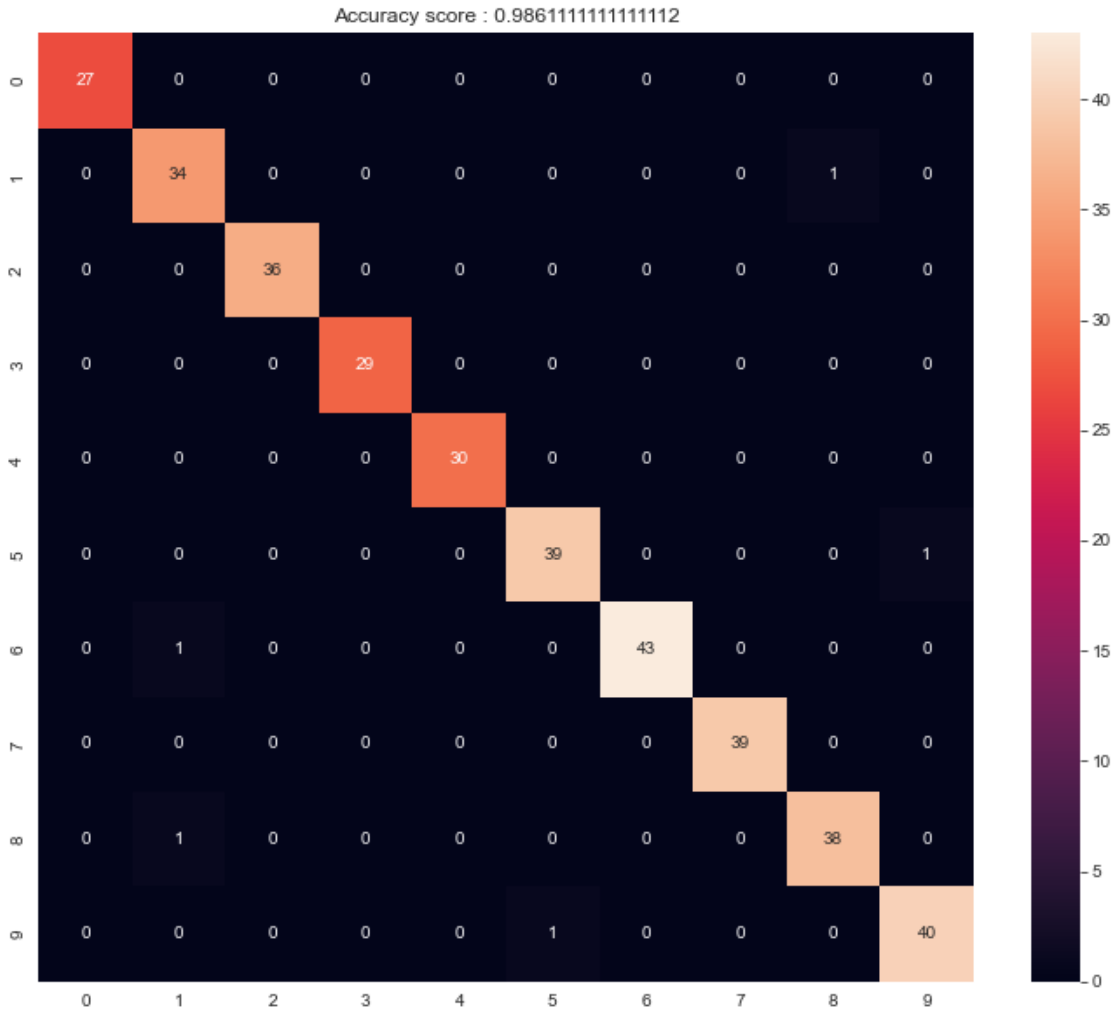
[ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

[ ]:
```
array([[27,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 34,  0,  0,  0,  0,  0,  0,  1,  0],
       [ 0,  0, 36,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 29,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 30,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0, 39,  0,  0,  0,  1],
       [ 0,  1,  0,  0,  0,  0, 43,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 39,  0,  0],
       [ 0,  1,  0,  0,  0,  0,  0,  0, 38,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  0,  0, 40]])
```

27.22 Creating HEATMAP to understand Confusion matrix

[ ]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(12,10))
sns.heatmap(cm, annot = True,)
plt.ylabel = 'Actual Output'
plt.xlabel = 'Predicted Output'
cm_title = 'Accuracy score : {0}'.format(score_1000)
plt.title(cm_title)
```

[ ]:
```
Text(0.5, 1.0, 'Accuracy score : 0.9861111111111112')
```



27.23 Getting Misclassified Labels

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
index = 0
misclassifiedIndexes = []
for label, predict in zip(y_test, y_pred):
 if label != predict:
  misclassifiedIndexes.append(index)
  index +=1
```

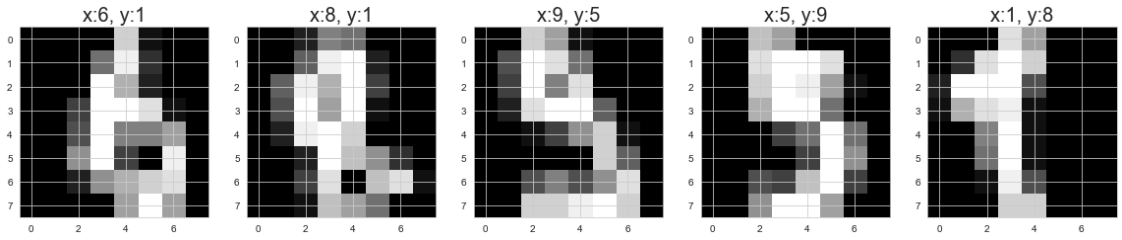
[ ]:
```
index = 0
misclassified_indicies = []
for label, predict in zip(y_test,y_pred):
    if label != predict:
```

```
        #print(index,label,predict) #Debugging
        misclassified_indicies.append(index)
    index = index +1
number_of_missclassified_data =len(misclassified_indicies)
print('indicies of missclssified data', misclassified_indicies)
print('number of missclassified pictures :',number_of_missclassified_data )
```

```
indicies of missclssified data [118, 124, 130, 181, 331]
number of missclassified pictures : 5
```

27.24 Plotting Missclassified Lables with Real Lables

[ ]:
```
sns.set_style(style='whitegrid')
plt.figure(figsize=(20,4))
for plotindex, badindex in enumerate(misclassified_indicies):
    plt.subplot(1,number_of_missclassified_data, plotindex+1)
    plt.imshow(np.reshape(x_test[badindex],(8,8)), cmap=plt.cm.gray)
    plt.title('x:{}, y:{}'.format(y_test[badindex],y_pred[badindex]), fontsize
 ↪=20)# predicted =y ; actual =x
```



28 Open CV

[ ]:
```
import cv2 as cv
img = cv.imread("resources/image.jpg")
cv.imshow("Pehli picture",img)
cv.waitKey(0)
```

[ ]:
```
# Resize image
img1= cv.resize(img,(700,600))
cv.imshow("Pehli picture",img)
cv.imshow("doosri picture",img1)
cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# converting to grayscale
from cv2 import cvtColor

img= cv.imread("resources/image.jpg")
img= cv.resize(img,(600,600))
#conversion part
gray_img= cvtColor(img,cv.COLOR_BGR2GRAY)
# show image
cv.imshow("Pehli image",img)
cv.imshow("Gray scale",gray_img)

cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# Image into black and white
import cv2 as cv
img = cv.imread("resources/image.jpg")
img= cv.resize(img,(700,600))
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
(thresh,binary)= cv.threshold(gray,127,255,cv.THRESH_BINARY)
cv.imshow('original',img)
cv.imshow('gray scale',gray)
cv.imshow('Black and white',binary)
print(binary)

cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# saving an image to external
from cv2 import imwrite
img = cv.imread("resources/image.jpg")
img = cv.resize(img,(700,500))
gray= cv.cvtColor(img, cv.COLOR_BGR2GRAY)
(thresh, binary)= cv.threshold(gray,127,255,cv.THRESH_BINARY)

imwrite('resources/image_gray.png',gray)
imwrite('resources/image_bnw.png',binary)
```

[ ]:
```
# video capture
import cv2 as cv
cap = cv.VideoCapture("resources/video.mp4")
#indication that video is present
if (cap.isOpened() == False):
    print("Error in reading video")
#reading and playing video file
while (cap.isOpened()):
    ret, frame = cap.read()
```

```
    if ret == True:
        cv.imshow("Video",frame)
        if cv.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

    # release after play back
cap.release()
cv.destroyAllWindows()
```

[ ]:
```
# converting video to black and white and gray
import cv2 as cv
cap = cv.VideoCapture("resources/video.mp4")

while(True):
    (ret, frame)= cap.read()
    # convert to gray
    grayframe= cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    # convert to binary
    (thresh, binary)= cv.threshold(grayframe,127,255,cv.THRESH_BINARY)
    if ret == True:
        # display gray video
        cv.imshow("Video Gray",grayframe)
        # display binary
        cv.imshow("Video  Binary",binary)
        if cv.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
cv.destroyAllWindows()
```

[ ]:
```
# saving a converted or a raw video
import cv2 as cv
cap = cv.VideoCapture('resources/video.mp4')

# writing format, codec,  video writer object , file output
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
output= cv.VideoWriter("resources/out_video.avi",cv.
 ↪VideoWriter_fourcc("M","J","P","G"),10,(frame_width,frame_height),isColor=False)

while(True):
    (ret,frame) = cap.read()
    grayframe= cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
    # toshow in video player
```

```
    if ret == True:
        output.write(grayframe)
        cv.imshow("Video Gray",grayframe)
        #to quit with q key
        if cv.waitKey(1) & 0XFF == ord('q'):
            break
    else:
        break

cap.release()
cv.destroyAllWindows()
```

[ ]:
```
# taking webcam input
import cv2 as cv
# read the frames from camera
cap = cv.VideoCapture(0) # 0 is webcam id
# to display video frame by frame till the end of video
while (cap.isOpened()):
    #start collecting frame and return True
    ret, frame = cap.read()
    if ret == True:
        cv.imshow("Webcam output",frame)
        if cv.waitKey(1) & 0xFF == ord('q'):
            break

    else:
        break
 # smooth close or release after completion
cv.release()
cv.destroyAllWindows()
```

[ ]:
```
# 10 chapter
# converting webcam video to gray and monotonic (binary)
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)

    cv.imshow("Original Cam",frame)
    cv.imshow("Gray cam",gray_frame)
    cv.imshow("Monotonic",binary)
```

```
    if cv.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

[ ]:
```
# chapter 11
# saving the video played from webcam in all three color ranges

# import libraries
import cv2 as cv
import numpy as np

# capture frames
cap = cv.VideoCapture(0)

# setting ratios of frame (default is 480 p)
frame_width = int(cap.get(3))
frame_height= int(cap.get(4))

# writing format, codec, video writer object, and file output
# writing format for simple video
output=cv.VideoWriter("resources/webcamvid.avi",cv.
 ↪VideoWriter_fourcc("M","J","P","G"),30,(frame_width,frame_height),isColor=True)
# writing format for gray video
output_gray=cv.VideoWriter("resources/webcam_gray_vid.avi",cv.
 ↪VideoWriter_fourcc("M","J","P","G"),30,(frame_width,frame_height),isColor=False)
# writing format for monotonic video
output_binary=cv.VideoWriter("resources/webcam_monotonic_vid.avi",cv.
 ↪VideoWriter_fourcc("M","J","P","G"),30,(frame_width,frame_height),isColor=False)

# loops runs until true
while (True):
# capture frames and boolean value inside ret variable
    (ret,frame) = cap.read()
# convert the frame into grayscale
    grayframe= cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
# convert each frame from gray scale to monotonic
    (thresh, binaryframe)= cv.threshold(grayframe,127,255,cv.THRESH_BINARY)
# check if frames are being captured
    if ret == True:
# saving the file by writing frames in a loop
        output.write(frame)
        output_gray.write(grayframe)
        output_binary.write(binaryframe)
```

```
# to have real time display of every color format
        cv.imshow("Video",frame)
        cv.imshow("Grayscale",grayframe)
        cv.imshow("monotonic",binaryframe)

# to quit and kill windows
        if cv.waitKey(1) & 0XFF == ord('q'):
            break
    else:
        break

# easy close and capture
cap.release()
output.release()
cv.destroyAllWindows()
```

[ ]:
```
# Assignment: Increase the resolution and control fps and check compression␣
 ↪four cc

# chapter 11
# saving the video played from webcam
# https://www.fatalerrors.org/a/
 ↪python-opencv-save-camera-video-as-well-as-the-introduction-of-fourc-coding.
 ↪html
import cv2 as cv
import numpy as np
import time
cap = cv.VideoCapture(0)


# Get the frame rate of the camera
fps = cap.get(cv.CAP_PROP_FPS)
# Or CV2. Cap_ PROP_ The array id corresponding to the FPS attribute is also OK
# cap = cap.get(5)
print(f"camera fps: {fps}")  # camera fps: 30.0

# Get the resolution of the camera
width = cap.get(cv.CAP_PROP_FRAME_WIDTH)
height = cap.get(cv.CAP_PROP_FRAME_HEIGHT)
# width = cap.get(3)
# width = cap.get(4)
print(f"camera resolution: ({width}x{height})")


# Set camera resolution
cap.set(3, 1280)
cap.set(4, 720)
```

```
# At this time, the reset resolution is obtained
width2 = cap.get(cv.CAP_PROP_FRAME_WIDTH)
height2 = cap.get(cv.CAP_PROP_FRAME_HEIGHT)
print(f"camera resolution: ({width2}x{height2})")  # camera resolution: (1280.
 ↪0x720.0)

# writing format, codec, video writer object, and file output

output=cv.VideoWriter("resources/webcamvid_enhancedresolution.avi",cv.
 ↪VideoWriter_fourcc("M","J","P","G"),24,(1280,720),isColor=True)
output_gray=cv.VideoWriter("resources/webcam_gray_vid_enhanced_resolution.
 ↪avi",cv.VideoWriter_fourcc("M","J","P","G"),24,(1280,720),isColor=False)
output_binary=cv.VideoWriter("resources/
 ↪webcam_monotonic_vid_enhanced_resolution.avi",cv.
 ↪VideoWriter_fourcc("M","J","P","G"),24,(1280,720),isColor=False)


while (True):
    start = time.time()
    (ret,frame) = cap.read()
    grayframe= cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
    (thresh, binaryframe)= cv.threshold(grayframe,127,255,cv.THRESH_BINARY)
    # to show in player
    if ret == True:
        output.write(frame)
        output_gray.write(grayframe)
        output_binary.write(binaryframe)


        cv.imshow("Video",frame)
        cv.imshow("Grayscale",grayframe)
        cv.imshow("monotonic",binaryframe)

        # to quit
        if cv.waitKey(1) & 0XFF == ord('q'):
            break
    else:
        break
    end = time.time()
    print(f"Read a frame time: {(end-start)*1000:.3f}ms")
    #qtime.sleep(0.1)

cap.release()
output.release()
cv.destroyAllWindows()
```

[ ]:
```
# chapter 12
# setting of camera or video as per requirements

import cv2 as cv
import numpy as np
cap = cv.VideoCapture(0)

# to control brightness
cap.set(10, 100)
# to control size
cap.set(3,200) # width
cap.set(4, 50) # height
while(True):
    ret, frame = cap.read()
    if ret == True:
        cv.imshow("frame",frame)
        if cv.waitKey(1) & 0xFF == ord('q'):
            break




cap.release()
cv.destroyAllWindows()
```

[ ]:
```
# chapter 13
# Manipulation of an Image
import cv2 as cv
img = cv.read("resources/image.jpg")



cv.imshow("Original",img)

#resize
resized_img = cv.resize(img, (450,250))
#gray
gray_img= cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#monotonic
(thresh, binary)= cv.threshold(gray_img,127,255,cv.THRESH_BINARY)



#display
cv.imshow("Original",img)
```

```
cv.imshow("Resized",resized_img)
cv.imshow("Gray",gray_img)
cv.imshow("Monotonic",binary)



cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# chapter 13
# Manipulation of an Image
import cv2 as cv
from cv2 import dilate
import numpy as np
img = cv.imread("resources/image.jpg")



#cv.imshow("Original",img)

#resize
resized_img = cv.resize(img, (450,250))
#gray
gray_img= cv.cvtColor(resized_img, cv.COLOR_BGR2GRAY)
#monotonic
(thresh, binary)= cv.threshold(gray_img,127,255,cv.THRESH_BINARY)
# blur img
blur = cv.GaussianBlur(resized_img,(7,7),0)
# edge detection
edge = cv.Canny(resized_img, 48,48)
# thickness of line
dilated_img = cv.dilate(edge,(1,1),iterations=1)
# another method to control thickness
kernel_mat= np.ones((3,3),np.uint8)
dilated_img2 = cv.dilate(edge,(kernel_mat),iterations=1)
# Make thinner outline
erode = cv.erode(dilated_img2,kernel_mat,iterations=1)
# cropping an image
print("The size of our image is ",resized_img.shape)
cropped = resized_img[0:200,200:300]



#display
#cv.imshow("Original",img)
# cv.imshow("Resized",resized_img)
# cv.imshow("Gray",gray_img)
```

```
# cv.imshow("Monotonic",binary)
# cv.imshow("Blurred Image",blur)
cv.imshow("Canny edge",edge)
cv.imshow("Dilation",dilated_img)
cv.imshow("Dilation using Numpy Array",dilated_img)
cv.imshow("Eroded Image ",erode)
cv.imshow("cropped image",cropped)




cv.waitKey(0)
cv.destroyAllWindows()
```

How to equalize image by using Appropriate Library (Car detection problem)

Video 71 a

Histogram Equalization (Search in OpenCV)

[ ]:
```
# chapter 14
# how to draw lines and shapes in python
import cv2 as cv
import numpy as np

# draw a canvas
img = np.zeros((600,600)) # for black color
img1 = np.ones((600,600))
colored_img = np.zeros((600,600,3),np.uint8)
#color complete image
colored_img[:] = 255,177,201
colored_img[150:300,400:600] = 255,144,232
print("The size of our matrix is ",img.shape)
#coloring a specific part
print("The size of colored matrix is ",colored_img.shape)
# adding line to an image canvas
cv.line(colored_img,(0,0),(400,200),(255,0,0),3)
cv.line(colored_img,(0,0),(colored_img.shape[0],colored_img.
 ↪shape[1]),(255,0,255),3)
# adding rectangles
cv.rectangle(colored_img, (50,100),(300,400),(255,240,0),5)
cv.rectangle(colored_img, (500,500),(550,550),(255,240,0),cv.FILLED)
# adding circle
cv.circle(colored_img,(250,250),20,(255,100,100),5)
# add text
```

```
cv.putText(colored_img,"Python ka Chilla",(400,400),cv.FONT_ITALIC,0.
 ↪5,(255,234,234),2)


# cv.imshow("Black Canvas",img)
# cv.imshow("White Canvas",img1)
cv.imshow("Colored image",colored_img)


cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# Assignment Chapter 14



# Adapted from
#https://www.codegrepper.com/search.php?q=break%20line%20text%20opencv

import cv2
import numpy as np


colored_img = np.zeros((600,600,3),np.uint8)
colored_img[:] = 255,177,201
position = (30, 30)
text = "You can start from new line Dr sab \n This is new line  \n ok ."
font_scale = 0.75
color = (255, 0, 0)
thickness = 3
font = cv2.FONT_HERSHEY_SIMPLEX
line_type = cv2.LINE_AA

text_size, _ = cv2.getTextSize(text, font, font_scale, thickness)
line_height = text_size[1] + 5
x, y0 = position
for i, line in enumerate(text.split("\n")):
    y = y0 + i * line_height
    cv2.putText(colored_img,
                line,
                (x, y),
                font,
                font_scale,
                color,
                thickness,
                line_type)
```

```
cv2.imshow("Colored image",colored_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

[ ]:
```
# chapter 15
# HD resolution of webcam

import cv2 as cv
import numpy as np

cap = cv.VideoCapture(0)
#Resolution
# cap.set(3,1280)
# cap.set(4,720)

def hd_resolution():
    cap.set(3,1280)
    cap.set(4,720)

hd_resolution()

while(True):
    ret , frame = cap.read()
    if ret == True:
        cv.imshow("Camera",frame)

    if cv.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv.destroyAllWindows()
```

[ ]:
```
# Chapter 16
#  saving hd recording of Cam streaming
import cv2 as cv
import numpy as np

cap = cv.VideoCapture(0)
#Resolution
# cap.set(3,1280)
# cap.set(4,720)

def hd_resolution():
    cap.set(3,1280)
    cap.set(4,720)
```

```
def sd_resolution():
    cap.set(3,1280)
    cap.set(4,720)

def fhd_resolution():
    cap.set(3,1280)
    cap.set(4,720)

#fhd_resolution()
hd_resolution()
#sd_resolution



# setting ratios of frame (default is 480 p)
frame_width = int(cap.get(3))
frame_height= int(cap.get(4))

# writing format, codec, video writer object, and file output
# writing format for simple video
output=cv.VideoWriter("resources/chapter16.avi",cv.
 ↪VideoWriter_fourcc("M","J","P","G"),30,(frame_width,frame_height),isColor=True)

# loops runs until true
while (True):
# capture frames and boolean value inside ret variable
    (ret,frame) = cap.read()
# convert the frame into grayscale
    grayframe= cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
# convert each frame from gray scale to monotonic
    (thresh, binaryframe)= cv.threshold(grayframe,127,255,cv.THRESH_BINARY)
# check if frames are being captured
    if ret == True:
# saving the file by writing frames in a loop
        output.write(frame)

# to have real time display of every color format
        cv.imshow("Video",frame)


# to quit and kill windows
        if cv.waitKey(1) & 0XFF == ord('q'):
            break
    else:
        break
```

```
# easy close and capture
cap.release()
output.release()
cv.destroyAllWindows()
```

[ ]:
```
# chapter 16 Assignment
# FPs function
import cv2 as cv
import numpy as np

cap = cv.VideoCapture(0)
#Resolution
# cap.set(3,1280)
# cap.set(4,720)

def hd_resolution():
    cap.set(3,1280)
    cap.set(4,720)

def sd_resolution():
    cap.set(3,1280)
    cap.set(4,720)

def fhd_resolution():
    cap.set(3,1280)
    cap.set(4,720)

#fhd_resolution()
hd_resolution()
#sd_resolution


def fps10():
   cap.set(cv.CAP_PROP_FPS, 10)


def fps20():
   cap.set(cv.CAP_PROP_FPS, 20)


def fps30():
   cap.set(cv.CAP_PROP_FPS, 30)



fps10()
#fps20()
```

```
#fps30()

# setting ratios of frame (default is 480 p)
frame_width = int(cap.get(3))
frame_height= int(cap.get(4))

# writing format, codec, video writer object, and file output
# writing format for simple video
output=cv.VideoWriter("resources/chapter16_assign.avi",cv.
 ↪VideoWriter_fourcc("M","J","P","G"),10,(frame_width,frame_height),isColor=True)

# loops runs until true
while (True):
# capture frames and boolean value inside ret variable
    (ret,frame) = cap.read()
# convert the frame into grayscale
    grayframe= cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
# convert each frame from gray scale to monotonic
    (thresh, binaryframe)= cv.threshold(grayframe,127,255,cv.THRESH_BINARY)
# check if frames are being captured
    if ret == True:
# saving the file by writing frames in a loop
        output.write(frame)

# to have real time display of every color format
        cv.imshow("Video",frame)


# to quit and kill windows
        if cv.waitKey(1) & 0XFF == ord('q'):
            break
    else:
        break

# easy close and capture
cap.release()
output.release()
cv.destroyAllWindows()
```

[ ]:
```
# chapter 17

import cv2 as cv
import numpy as np

img = cv.imread("resources/image.jpg")
img = cv.resize(img,(400,200))
#stacking the same image
```

```
# 1- Horizontal Stack

horizontal = np.hstack((img,img))
cv.imshow("Horizontal stacking",horizontal)

#2- Vertical stack
vertical = np.vstack((img,img))
cv.imshow("Vertical",vertical)




cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# Chapter 17 Assignment
# Cascading Images
# Useful links
# https://note.nkmk.me/en/python-opencv-hconcat-vconcat-np-tile/
# https://www.geeksforgeeks.org/concatenate-images-using-opencv-in-python/
# https://note.nkmk.me/en/python-opencv-hconcat-vconcat-np-tile/

# This code shows functionality of cascading Images with different dimensions␣
 ↪and channel into one.


import cv2 as cv
import numpy as np

img = cv.imread("resources/image.jpg")
img1 = cv.resize(img,(500,400))
img2 = cv.resize(img,(500,275))
img3 = cv.resize(img,(260,400))
gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
img4 = cv.cvtColor(gray,cv.COLOR_GRAY2BGR)

cv.imshow("Image1",img1)
cv.imshow("Image2",img2)
cv.imshow("Image3",img3)

# define a function for vertically
# concatenating images of different sizes

def vconcat_resize_min(img_list, interpolation=cv.INTER_CUBIC):
    # To concatenate vertically all width (.shape[1]) should be equal and we␣
 ↪scale them to a min value
      w_min = min(img.shape[1] for img in img_list)
```

```
    im_list_resize = [cv.resize(img, (w_min, int(img.shape[0] * w_min / img.
↪shape[1])), interpolation=interpolation) for img in im_list]
    return cv.vconcat(im_list_resize)

im_v_resize = vconcat_resize_min([img1, img2, img3, img4])
cv.imwrite('resources/concatenated_img.jpg', im_v_resize)
cv.imshow("Concatenated Image",im_v_resize)



cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# chapter 18
# 47  210 # 249  98 # 64  338 # 304  219
# chapter 18
# how to change perspective of an image
#import libraries
import cv2 as cv
import numpy as np

img = cv.imread("resources/warp.png")
print(img.shape)
#defining points that needs to be transformed
point1 = np.float32([[47,210],[64,338],[249,98],[304,219]])

width, height = 399, 429
# defining points where point1 needs to be transformed
point2 = np.float32([[0,0],[399,0],[0,height],[width,height]])
# defining that point1 will be transformed to point2
matrix = cv.getPerspectiveTransform(point1,point2)
# takes image, transformation matrix and dimension as input
out_img = cv.warpPerspective(img, matrix, (width,height))


cv.imshow("Original",img)
cv.imshow("Transformed",out_img)


cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# chapter 19
# chapter 19
# how to show coordinates of a image or video
#import libraries
from threading import local
```

```
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:
        print(x,'',y)
        # how to define or print on the same image or window
        font = cv.FONT_HERSHEY_COMPLEX
        cv.putText(img,str(x)+'\n'+str(y),(x,y),font,1,(255,0,255),thickness=2)
        # show the image (image with text)
        cv.imshow("image",img)

        # for color finding
    if event == cv.EVENT_RBUTTONDOWN:


        font = cv.FONT_HERSHEY_DUPLEX
        # image (w*h*colorchannel)
        b = img[y,x,0]
        g = img[y,x,1]
        r = img [y,x,2]
        # printing on console
        print(b,'',g,'',r)
        # on image printing
        cv.
 ↪putText(img,str(b)+'\n'+str(g)+'\n'+str(r),(x,y),font,1,(255,0,127),thickness=2)
        cv.imshow("image",img)


if  __name__ == "__main__":
    #reading an image
    img = cv.imread("resources/warp.png")
    # display the image
    cv.imshow("image",img)
    #Setting call back function
    cv.setMouseCallback("image",find_coord)

cv.waitKey(0)
cv.destroyAllWindows()
```

[ ]:
```
# Chapter 20
# split video into frames

import cv2 as cv

cap = cv.VideoCapture("resources/video.mp4")
```

```
frameNr=0

while(True):
    ret, frames = cap.read()
    if ret:
        cv.imwrite(f"resources/frames/frame_{frameNr}.jpg",frames)
    else:
        break

    frameNr += 1

cap.release()
```

[ ]:
```
# Chapter 21
# Control HSV values
import cv2 as cv
import numpy as np

img = cv.imread("resources/image.jpg")

# convert to HSV
# hsv_img = cv.cvtColor(img, cv.COLOR_BGR2HSV)

# creating sliders
def slider():
    pass

# storing the path of target image in path variable just for easiness
path = "resources/image.jpg"
# this command create a control window with name bar and nothing more than that
cv.namedWindow("Bars")
cv.resizeWindow("Bars",400,300)

cv.createTrackbar("Hue min", "Bars",0,179,slider)
cv.createTrackbar("Hue max", "Bars",179,179,slider)
cv.createTrackbar("Sat min", "Bars",0,179,slider)
cv.createTrackbar("Sat max", "Bars",179,179,slider)
cv.createTrackbar("Value min", "Bars",0,255,slider)
cv.createTrackbar("Value max", "Bars",255,255,slider)

img = cv.imread(path)
img = cv.resize(img,(600,800))
hsv_img = cv.cvtColor(img, cv.COLOR_BGR2HSV)

# it can only run inside a conditonal loop
# hue_min = cv.getTrackbarPos("Hue min","Bars")
```

```
# print(hue_min)

while True:
    img = cv.imread(path)
    img = cv.resize(img,(600,800))
    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("Value min","Bars")
    val_max = cv.getTrackbarPos("Value max","Bars")

    # Track bar positions(value asigned) created and now be easily controlled␣
 ↪and displayed
    print(hue_min,hue_max,sat_min,sat_max,val_min,val_max)

    # to see effect of slider values on a real image
    # simply mapping values of HSV to their meaning
    lower = np.array([hue_min,sat_min,val_min])
    upper = np.array([hue_max,sat_max,val_max])
    # All HSV adjustment here
    # study bitwise operators from Geeksforgeeks
    mask_img = cv.inRange(hsv_img,lower,upper)
    out_img = cv.bitwise_and(img,img, mask= mask_img)

    # dsiplay
    # 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.waitKey(0)
cv.destroyAllWindows()
```

[ ]: