**Practical Set - 1**

1. Given a two integer numbers return their product and if the product is greater than 1000, then return their sum.

a = int(input("Enter number 1: "))

b = int(input("Enter number 2: "))

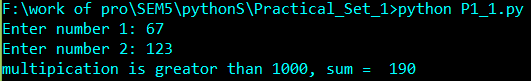
c = a\*b

if c > 1000:

print("number is greator than 1000, sum = ",a+b)

else:

print("numberr is less than 1000, multipication = ",c)



1. Given a range of first 10 numbers, Iterate from start number to the end number and print the sum of the current number and previous number

x = list()

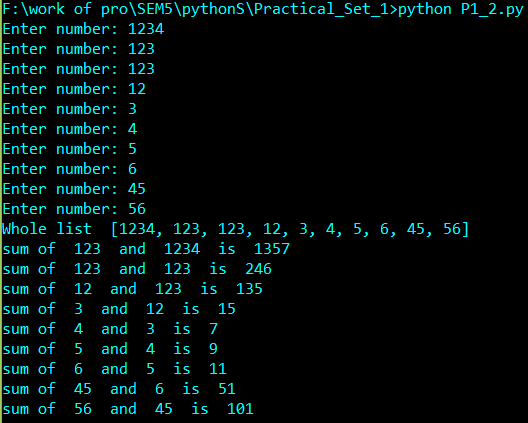
for i in range(10):

x.append(int(input("Enter number: ")))

print("Whole list ",x)

for i in range(9):

print("sum of ",x[i+1]," and ",x[i]," is ",x[i]+x[i+1])



1. Given a string, display only those characters which are present at an even index number.

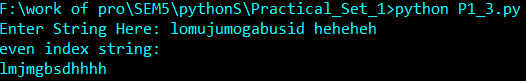
x = input("Enter String Here: ")

print("even index string: ")

for i in range(len(x)):

if i%2==0:

print(x[i],end="")



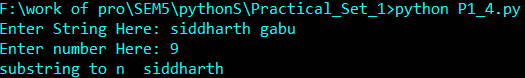
1. Given a string and an integer number n, remove characters from a string starting from zero up to n and return a new string.

x = input("Enter String Here: ")

n = int(input("Enter number Here: "))

y = x[:n]

print("substring to n ",y)



1. Given a list of numbers, return True if first and last number of a list is same

x = [int(x) for x in input("Enter multipule number: ").split()]

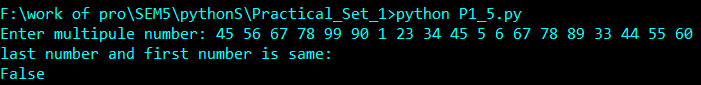
print("last number and first number is same: ")

if x[0]==x[len(x)-1]:

print("True")

else:

print("False")



1. Given a list of numbers, Iterate it and print only those numbers which are divisible of 5.

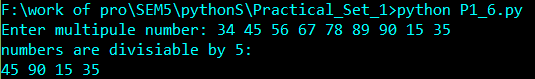
x = [int(x) for x in input("Enter multipule number: ").split()]

print("numbers are divisiable by 5:")

for i in range(len(x)):

if x[i]%5==0:

print(x[i],end=" ")



1. Return the total count of string “Emma” appears in the given string

x = input("Enter Your String Here: ")

print("In Your given string Emma appears ",x.count("Emma")," times")

P1_7.PNG

1. Reverse a given number and return true if it is the same as the original number.

x = input("Enter number : ")

y = x[::-1]

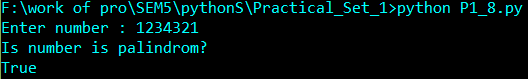
print("Is number is palindrom?")

if x == y:

print("True")

else:

print("False")



1. Given a two list of numbers create a new list such that new list should contain only odd numbers from the first list and even numbers from the second list.

x = [int(x) for x in input("Enter multipule value1: ").split()]

y = [int(y) for y in input("Enter multipule value2: ").split()]

z = list()

for i in range(len(x)):

if x[i]%2!=0:

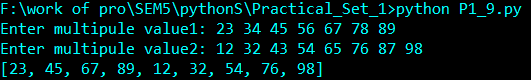
z.append(x[i])

for i in range(len(y)):

if y[i]%2==0:

z.append(y[i])

print(z)

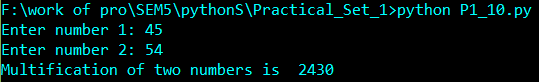


1. Accept two numbers from the user and calculate multiplication

num1 = int(input("Enter number 1: "))

num2 = int(input("Enter number 2: "))

print("Multification of two numbers is ",num1\*num2)



1. Display “My Name Is James” as “My\*\*Name\*\*Is\*\*James” using output formatting of a print() function.

print("formatting string using print:")

print("My","Name","Is","James",sep="\*\*")

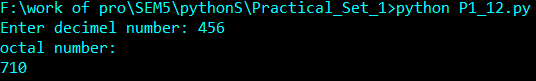
P1_11.PNG

1. Convert decimal number to octal using print() output formatting

x = int(input("Enter decimel number: "))

print("octal number: ")

print('%o' % (x))

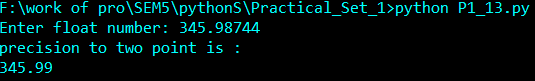


1. Display float number with 2 decimal places using print()

x = float(input("Enter float number: "))

print("precision to two point is :")

print('%.2f' % x)



1. Accept list of 5 float numbers as a input from user

numbersl = []

n = int(input("Enter size of inputs: "))

print("\n")

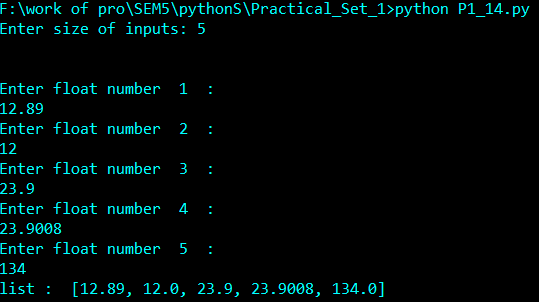
for i in range(0,n):

print("Enter float number ",i+1," :")

x = float(input())

numbersl.append(x)

print("list : ",numbersl)



1. Write all file content into new file by skiping line 5 from following file.

f1 = open("text1.txt","r")

f2 = open("text2.txt","w")

lines = f1.readlines()

for i in range(len(lines)):

if i == 4:

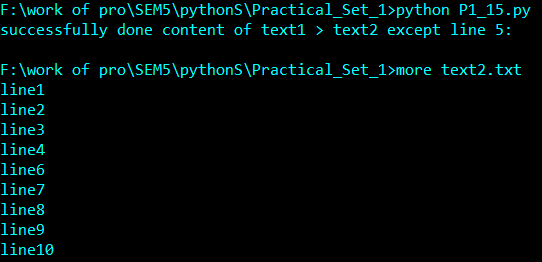
continue

f2.write(lines[i])

print("successfully done content of text1 > text2 except line 5:")

f1.close()

f2.close()



1. Accept any three string from one input() call.

str1,str2,str3 = input("Enter Three String: ").split()

print(str1," ",str2," ",str3)

P1_16.PNG

1. You have the following data. totalMoney = 1000, quantity = 3, price = 450 display above data using string.format() method

txt = "i have {totalMoney} Ruppes so i can't buy {quantity} laptop for {price} Ruppees."

print(txt.format(totalMoney = 1000,price = 450,quantity = 3))

P1_17.PNG

1. How to check file is empty or not

import os

print("file is empty?")

print(os.stat("text1.txt").st\_size == 0)

P1_18.PNG

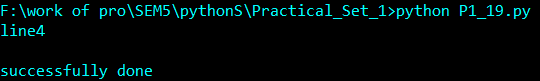
1. Read line number 4 from the following file

f = open("text2.txt","r")

line = f.readlines()

print(line[3])

print("successfully done")



**Practical set – 2**

1. Print the following pattern

1

1 2

1 2 3

1 2 3 4

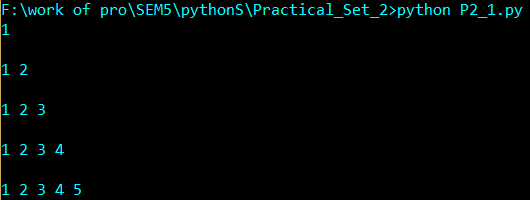
1 2 3 4 5

for i in range(1,6):

for j in range(1,i+1):

print(j,"",end="")

print("\n")



1. Given a list iterate it and display numbers which are divisible by 5 and if you find number greater than 150 stop the loop iteration

list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]

list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]

for i in range(len(list1)):

if list1[i]>150:

break

if list1[i]%5==0:

print(list1[i],end=" ")

P2_2.PNG

1. Write a function func1() such that it can accept a variable length of argument and print all arguments value

func1(20, 40, 60)

func1(80, 100)

def func1(\*vargas):

for i in vargas:

print(i,end=" ")

print()

func1(20, 40, 60)

func1(80, 100)

P2_3.PNG

1. Write a function calculation() such that it can accept two variables and calculate the addition and subtraction of it. And also it must return both addition and subtraction in a single return call.

def calculation(a,b):

return a+b,a-b

suma,sub = calculation(10,5)

print(suma,sub)

P2_4.PNG

1. Write a recursive function to calculate the sum of numbers from 0 to 10.

def sum\_10(n):

if(n > 0):

result = n + sum\_10(n - 1)

else:

result = 0

return result

print(sum\_10(10))

P2_5.PNG

**Practical set – 3**

1. Given a string of odd length greater 7, return a string made of the middle three chars of a given String

def getMiddleThreeChars(x):

y = int(len(x)/2)

z = x[y-1:y+2]

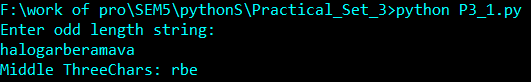
return z

print("Enter odd length string: ")

x = input()

z = getMiddleThreeChars(x)

print("Middle ThreeChars:",z)



1. Given 2 strings, s1 and s2, create a new string by appending s2 in the middle of s1Expected Outcome: appendMiddle("Chrisdem", IamNewString) → "ChrIamNewStringisdem"

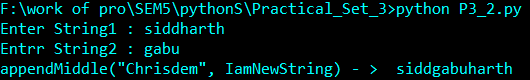
s1 = input("Enter String1 : ")

s2 = input("Entrr String2 : ")

l = int(len(s1)/2)

new = s1[:l]+s2+s1[l:]

print("appendMiddle(\"Chrisdem\", IamNewString) - > ",new)



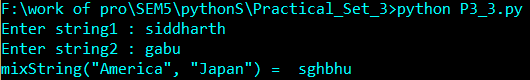
1. Given 2 strings, s1, and s2 return a new string made of the first, middle and last char each input string Expected Outcome: mixString("America", "Japan") = ""AJrpan"

s1,s2 = input("Enter string1 : "),input("Enter string2 : ")

l1,l2 = len(s1),len(s2)

new = s1[0]+s2[0]+s1[int(l1/2)]+s2[int(l2/2)]+s1[l1-1]+s2[l2-1]

print("mixString(\"America\", \"Japan\") = ",new)



1. Find all occurrences of “USA” in given string ignoring the case Expected Outcome: input\_str = "Welcome to USA. usa awesome, isn't it?" The USA count is: 2

s = "Welcome to USA. usa awesome, isn't it?"

ct = s.lower().count("USA".lower())

print("The USA count is: ",ct)

P3_4.PNG

1. Given a two list. Create a third list by picking an odd-index element from the first list and even index elements from second.

listOne = [3, 6, 9, 12, 15, 18, 21]

listTwo = [4, 8, 12, 16, 20, 24, 28]

newlist = list()

for i in range(len(listOne)):

if i%2!=0:

newlist.append(listOne[i])

for i in range(len(listTwo)):

if i%2==0:

newlist.append(listTwo[i])

print("Printing Final third list ",newlist)

P3_5.PNG

1. Given a list iterate it and count the occurrence of each element and create a dictionary to show the count of each element

thislist = [11, 45, 8, 11, 23, 45, 23, 45, 89]

thisset = set(thislist)

print(thisset)

thisdict = dict()

for i in range(len(thisset)):

count = 0

y = thisset.pop()

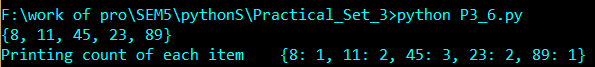
for j in range(len(thislist)):

if y == thislist[j]:

count = count + 1

thisdict.update({y:count})

print("Printing count of each item ",thisdict)



1. Given a two list of equal size create a set such that it shows the element from both lists in the pair.

firstList = [2, 3, 4, 5, 6, 7, 8]

secondList = [4, 9, 16, 25, 36, 49, 64]

result = zip(firstList, secondList)

resultSet = set(result)

print("Result is ",resultSet)

P3_7.PNG

1. Iterate a given list and Check if a given element already exists in a dictionary as a key’s value if not delete it from the list

rollNumber = [47, 64, 69, 37, 76, 83, 95, 97]

sampleDict ={'Jhon':47, 'Emma':69, 'Kelly':76, 'Jason':97}

checklist = list(sampleDict.values())

print("before list ",rollNumber)

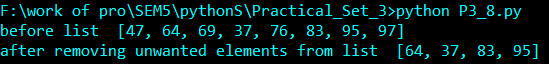
for i in checklist:

for j in rollNumber:

if i == j:

rollNumber.remove(j)

print("after removing unwanted elements from list ",rollNumber)



**Practical set – 4**

1. Reverse the following tuple aTuple = (10, 20, 30, 40, 50)

aTuple = (10, 20, 30, 40, 50)

aTuple = aTuple[::-1] #Stride while Slicing Strings

print("Reverse order tuples: ",aTuple)

P4_1.PNG

1. . Unpack the following tuple into 4 variables aTuple = (10, 20, 30, 40)

aTuple = (10, 20, 30, 40)

a,b,c,d = aTuple

print("before unpacking: ",aTuple)

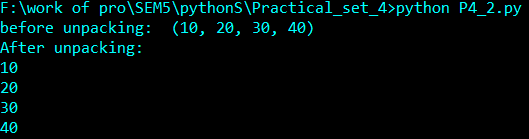
print("After unpacking: ")

print(a)

print(b)

print(c)

print(d)



1. Check if all items in the following tuple are the same tuple1 = (45, 45, 45, 45).

aTuple = (45,45,45,45)

aset = set(aTuple)

print("Tuple containes duplicate: ")

if len(aset) == 1:

print("True")

else:

print("False")

P4_3.PNG

1. Below are the two lists convert it into the dictionary: keys = ['Ten', 'Twenty', 'Thirty'], values = [10, 20, 30]

keys = ['Ten', 'Twenty', 'Thirty']

values = [10, 20, 30]

adict = dict()

for i in range(len(keys)):

adict.update({keys[i]:values[i]})

print("merging list into dict: ",adict)

P4_4.PNG

1. Merge following two Python dictionaries into one: dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}, dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}

dict1 = {'Ten': 10, 'Twenty': 20, 'Thirty': 30}

dict2 = {'Thirty': 30, 'Fourty': 40, 'Fifty': 50}

adict = dict1.copy()

adict.update(dict2)

print(adict)

P4_7.PNG

1. Delete set of keys from Python Dictionary.

sampleDict = {

"name": "Kelly",

"age":25,

"salary": 8000,

"city": "New york"

}

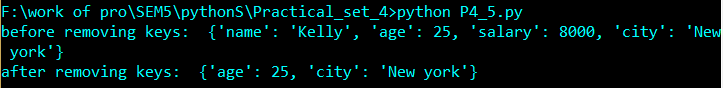
keysToRemove = ["name", "salary"]

print("before removing keys: ",sampleDict)

sampleDict.pop(keysToRemove[0])

sampleDict.pop(keysToRemove[1])

print("after removing keys: ",sampleDict)



1. Check if a value 200 exists in a dictionary sampleDict = {'a': 100, 'b': 200, 'c': 300}

sampleDict = {'a': 100, 'b': 200, 'c': 300}

print(sampleDict,"This dict contain 200 as value:")

print(200 in sampleDict.values())

P4_6.PNG

**Practical set – 5**

1. Write a NumPy program to compute the multiplication of two given matrixes.

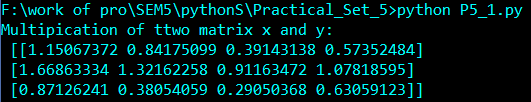
import numpy as np

x = np.random.rand(3,5)

y = np.random.rand(5,4)

z = np.dot(x,y)

print("Multipication of ttwo matrix x and y: \n",z)



1. Write a NumPy program to compute the cross product of two given vectors.

import numpy as np

x = np.random.rand(2,2)

y = np.random.rand(2,2)

z = np.cross(x,y)

print("Cross product of two vector: \n",z)

P5_2.PNG

1. Write a NumPy program to compute the determinant of a given square array.

import numpy as np

x = np.array([[1,0],[1,2]])

print("determinant of square array ",np.linalg.det(x))

P5_3.PNG

1. Write a NumPy program to compute the inverse of a given matrix.

import numpy as np

x = np.array([[1,2],[3,4]])

print("determinant of array :\n",np.linalg.det(x))

P5_4.PNG

1. Write a NumPy program to compute the eigenvalues and right eigenvectors of a given square array.

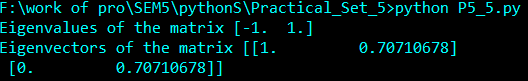
import numpy as np

x = np.array([[-1,2],[0,1]])

w, v = np.linalg.eig(x)

print( "Eigenvalues of the matrix",w)

print( "Eigenvectors of the matrix",v)



**Practical – 6**

1. Write a NumPy program to sort a given array of shape 2 along the first axis, last axis and on flattened array.

import numpy as np

x = np.array([[10,30,20],[20,20,10]])

print("Original array:")

print(x)

print("Sort the array along the first axis:")

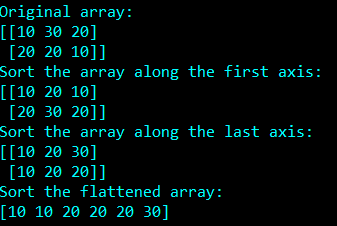
print(np.sort(x, axis=0))

print("Sort the array along the last axis:")

print(np.sort(x))

print("Sort the flattened array:")

print(np.sort(x, axis=None))



1. Write a NumPy program to create a structured array from given student name, height, class and their data types. Now sort by class, then height if class are equal.

import numpy as np

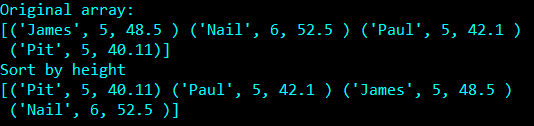
x = np.array([('James', 5, 48.5), ('Nail', 6, 52.5),('Paul', 5, 42.10), ('Pit', 5, 40.11)],dtype=[('name', 'U10'), ('class', int), ('height', float)])

print("Original array:")

print(x)

print("Sort by height")

print(np.sort(x, order=['class', 'height']))



1. Write a NumPy program to get the floor, ceiling and truncated values of the elements of a numpy array.

import numpy as np

x = np.array([1.7,40,34,23,12.09,-3,-34.56,21])

print("Original array:")

print(x)

print("Floor values of the above array elements:")

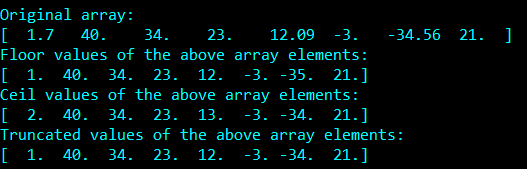
print(np.floor(x))

print("Ceil values of the above array elements:")

print(np.ceil(x))

print("Truncated values of the above array elements:")

print(np.trunc(x))



1. Write a NumPy program to multiply a matrix by another matrix of complex numbers and create a new matrix of complex numbers.

import numpy as np

x = np.array([[11+2j,34+4j],[9+6j,7+12j]])

print("First array:")

print(x)

y = np.array([[9+6j,7+12j],[11+2j,34+4j]])

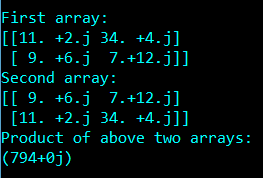
print("Second array:")

print(y)

z = np.vdot(x, y)

print("Product of above two arrays:")

print(z)



1. Write a NumPy program to find the roots of the following polynomials.    
   a) x2 - 4x + 7.

b) x4 - 11x3 + 9x2 + 11x – 10

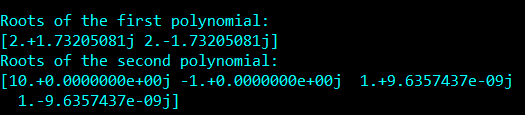
import numpy as np

print("Roots of the first polynomial:")

print(np.roots([1, -4, 7]))

print("Roots of the second polynomial:")

print(np.roots([1, -11, 9, 11, -10]))



**Practical – 7**

1. Write a Pandas program to write a DataFrame to CSV file using tab separator.

import numpy as np

import pandas as pd

Data = {'Name':['Siddharth','Gabu','SidPro','Ark'],'Age':[27,12,19,45]}

df = pd.DataFrame(data=Data)

print("Original DataFrame")

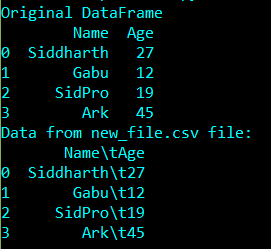
print(df)

print('Data from new\_file.csv file:')

df.to\_csv('new\_file.csv', sep='\t', index=False)

new\_df = pd.read\_csv('new\_file.csv')

print(new\_df)



1. Write a Pandas program to convert a given list of lists into a Dataframe.

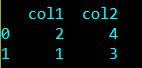
import numpy as np

import pandas as pd

lit = [[2, 4], [1, 3]]

df = pd.DataFrame(lit,columns=['col1','col2'])

print(df)



1. Write a Pandas program to get the first 3 rows of a given DataFrame..

import numpy as np

import pandas as pd

exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

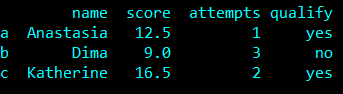
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data,labels)

print(df[0:3])



1. Write a Pandas program to display a summary of the basic information about a specified DataFrame and its data.

import numpy as np

import pandas as pd

exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

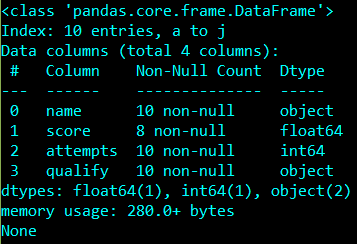
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data,labels)

print(df.info())



**Practical – 8**

1. Write a Pandas program to select the rows where the number of attempts in the examination is greater than 2.

import numpy as np

import pandas as pd

exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

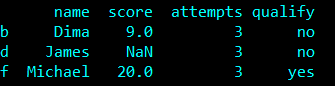
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data,labels)

print(df[df.attempts>2])



1. Write a Pandas program to select the rows where the score is missing, i.e. is NaN.

import numpy as np

import pandas as pd

exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

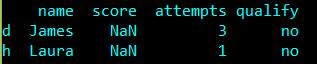
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data,labels)

print(df[df['score'].isnull()])



1. Write a Pandas program to sort the DataFrame first by 'name' in descending order, then by 'score' in ascending order.

import numpy as np

import pandas as pd

exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

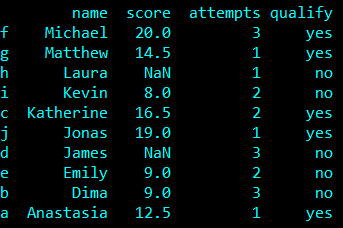
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data,labels)

df.sort\_values(["name", "score"], axis=0,

ascending=[False,True], inplace=True)

print(df)



1. Write a Pandas program to replace the 'qualify' column contains the values 'yes' and 'no' with True and False

import numpy as np

import pandas as pd

exam\_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

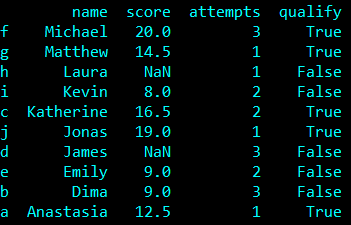
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam\_data,labels)

df['qualify']=df['qualify'].map({'yes':True,'no':False})

print(df)



**Practical – 9**

1. Create Python Programs using Pandas and Matplotlib to visualize Company Sales Data.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| month\_number | facecream | facewash | toothpaste | bathingsoap | shampoo | moisturizer | total\_units | total\_profit |
| 1 | 2500 | 1500 | 5200 | 9200 | 1200 | 1500 | 21100 | 211000 |
| 2 | 2630 | 1200 | 5100 | 6100 | 2100 | 1200 | 18330 | 183300 |
| 3 | 2140 | 1340 | 4550 | 9550 | 3550 | 1340 | 22470 | 224700 |
| 4 | 3400 | 1130 | 5870 | 8870 | 1870 | 1130 | 22270 | 222700 |
| 5 | 3600 | 1740 | 4560 | 7760 | 1560 | 1740 | 20960 | 209600 |
| 6 | 2760 | 1555 | 4890 | 7490 | 1890 | 1555 | 20140 | 201400 |
| 7 | 2980 | 1120 | 4780 | 8980 | 1780 | 1120 | 29550 | 295500 |
| 8 | 3700 | 1400 | 5860 | 9960 | 2860 | 1400 | 36140 | 361400 |
| 9 | 3540 | 1780 | 6100 | 8100 | 2100 | 1780 | 23400 | 234000 |
| 10 | 1990 | 1890 | 8300 | 10300 | 2300 | 1890 | 26670 | 266700 |
| 11 | 2340 | 2100 | 7300 | 13300 | 2400 | 2100 | 41280 | 412800 |
| 12 | 2900 | 1760 | 7400 | 14400 | 1800 | 1760 | 30020 | 300200 |

* + 1. Read Total profit of all months and show it using a line plot.

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('F:\work of pro\SEM5\pythonS\Practical\_Set\_9\data.csv')

plt.figure(figsize=(8,5))

plt.plot(df.month\_number,df.total\_profit,'g.-',label="Total Profit")

plt.title("Total profit By months")

plt.xlabel("Month number")

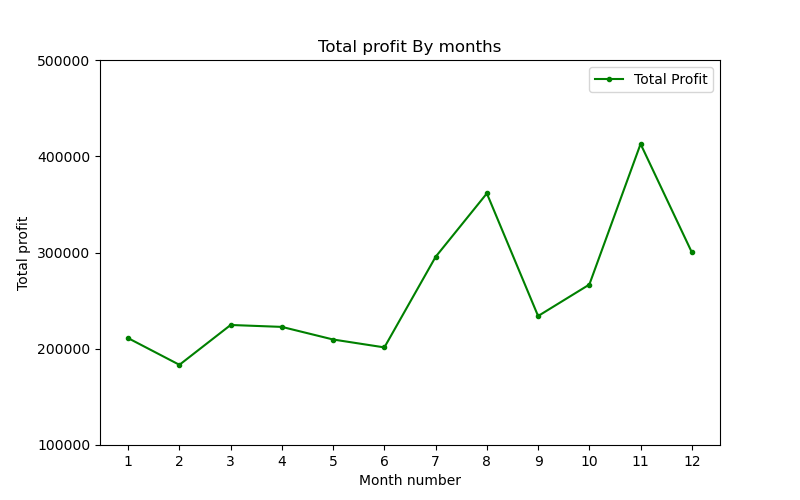
plt.ylabel("Total profit")

plt.yticks([100000,200000,300000,400000,500000])

plt.xticks(df.month\_number)

plt.legend()

plt.show()



* + 1. Read all product sales data and show it using a multiline plot Display the number of units sold per month for each product using multiline plots. (i.e., separate Plotline for each product).

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('F:\work of pro\SEM5\pythonS\Practical\_Set\_9\data.csv')

plt.figure(figsize=(8,5))

plt.plot(df.month\_number,df.facecream,'r.-',label="FaceCream Sales Units")

plt.plot(df.month\_number,df.facewash,'k.-',label="FaceWash Sales Units")

plt.plot(df.month\_number,df.toothpaste,'b.-',label="Toothpaste Sales Units")

plt.plot(df.month\_number,df.bathingsoap,'c.-',label="BathingSoap Sales Units")

plt.plot(df.month\_number,df.shampoo,'m.-',label="Shampoo Sales Units")

plt.plot(df.month\_number,df.moisturizer,'y',label="Moisturizer Sales Units")

plt.title("Total Sales Data of Each Product By Month")

plt.xlabel("Month number")

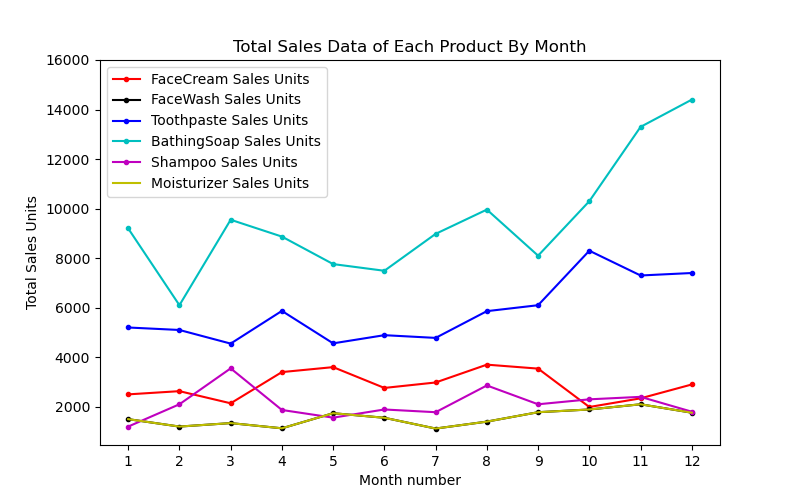
plt.ylabel("Total Sales Units")

plt.yticks([2000,4000,6000,8000,10000,12000,14000,16000])

plt.xticks(df.month\_number)

plt.legend()

plt.show()



1. Create Python Programs using Pandas and Matplotlib to visualize Company Sales Data.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| month\_number | facecream | facewash | toothpaste | bathingsoap | shampoo | moisturizer | total\_units | total\_profit |
| 1 | 2500 | 1500 | 5200 | 9200 | 1200 | 1500 | 21100 | 211000 |
| 2 | 2630 | 1200 | 5100 | 6100 | 2100 | 1200 | 18330 | 183300 |
| 3 | 2140 | 1340 | 4550 | 9550 | 3550 | 1340 | 22470 | 224700 |
| 4 | 3400 | 1130 | 5870 | 8870 | 1870 | 1130 | 22270 | 222700 |
| 5 | 3600 | 1740 | 4560 | 7760 | 1560 | 1740 | 20960 | 209600 |
| 6 | 2760 | 1555 | 4890 | 7490 | 1890 | 1555 | 20140 | 201400 |
| 7 | 2980 | 1120 | 4780 | 8980 | 1780 | 1120 | 29550 | 295500 |
| 8 | 3700 | 1400 | 5860 | 9960 | 2860 | 1400 | 36140 | 361400 |
| 9 | 3540 | 1780 | 6100 | 8100 | 2100 | 1780 | 23400 | 234000 |
| 10 | 1990 | 1890 | 8300 | 10300 | 2300 | 1890 | 26670 | 266700 |
| 11 | 2340 | 2100 | 7300 | 13300 | 2400 | 2100 | 41280 | 412800 |
| 12 | 2900 | 1760 | 7400 | 14400 | 1800 | 1760 | 30020 | 300200 |

* 1. Read toothpaste sales data of each month and show it using a scatter plot.

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('F:\work of pro\SEM5\pythonS\Practical\_Set\_9\data.csv')

plt.figure(figsize=(8,5))

plt.scatter(df.month\_number,df.toothpaste,label="Total Profit",color='b')

plt.title("Toothpaste sales data of each month")

plt.xlabel("Month number")

plt.ylabel("Toothpaste sales Units")

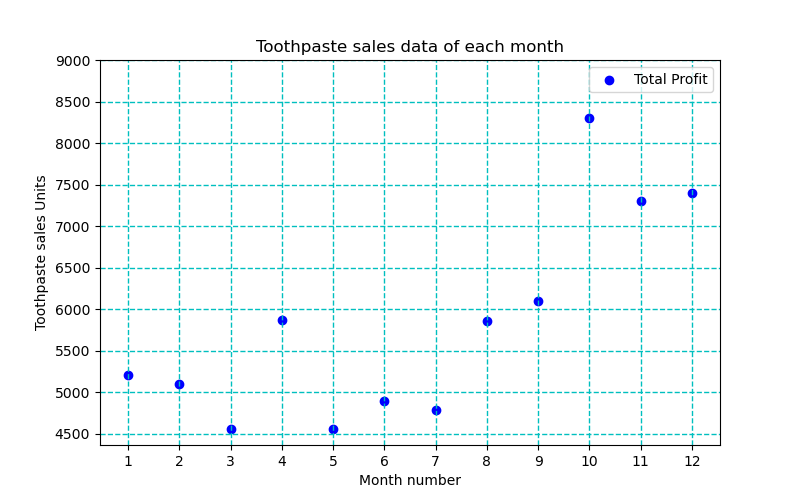
plt.yticks([4500,5000,5500,6000,6500,7000,7500,8000,8500,9000])

plt.xticks(df.month\_number)

plt.grid(color='c',linestyle='--',linewidth=1)

plt.legend()

plt.show()



* 1. Read face cream and facewash product sales data and show it using the bar chart.

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('F:\work of pro\SEM5\pythonS\Practical\_Set\_9\data.csv')

plt.figure(figsize=(8,5))

plt.bar([i - 0.25 for i in df.month\_number],df.facecream,width=0.25,label="FaceCream Sales Units",color='k',align='edge')

plt.bar([i for i in df.month\_number],df.facewash,width=0.25,label="FaceWash Sales Units",color='y',align='edge')

plt.title("Face Cream and FaceWash Product Sales Sata Of Each Month")

plt.xlabel("Month number")

plt.ylabel("Face Cream and FaceWash sales Units")

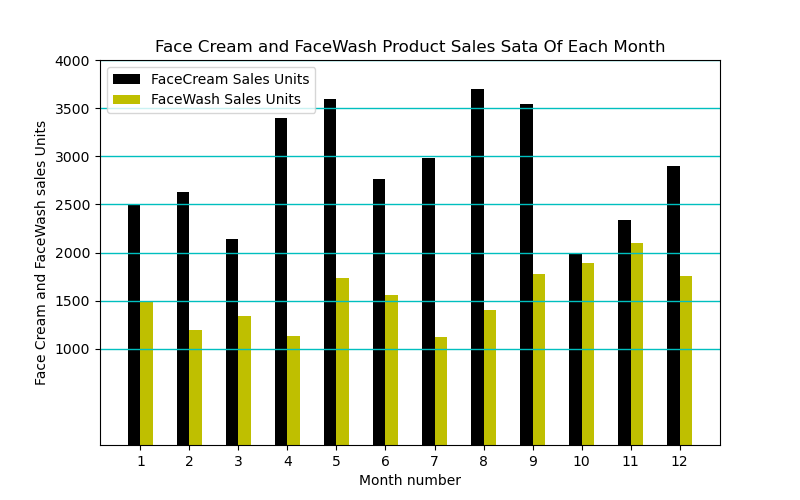
plt.yticks([1000,1500,2000,2500,3000,3500,4000])

plt.xticks(df.month\_number)

plt.grid(axis='y',color='c',linestyle='-',linewidth=1)

plt.legend()

plt.show()



* 1. Calculate total sale data for last year for each product and show it using a Pie chart Note: In Pie chart display Number of units sold per year for each product in percentage.

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('F:\work of pro\SEM5\pythonS\Practical\_Set\_9\data.csv')

slices = [df.facecream.sum(),df.facewash.sum(),df.toothpaste.sum(),df.bathingsoap.sum(),df.shampoo.sum(),df.moisturizer.sum()]

label = ["FaceCream","FaceWash","ToothPaste","BathingSoap","shampoo","Moisturizer"]

plt.pie(slices,

labels=label,

startangle=150,

shadow=False,

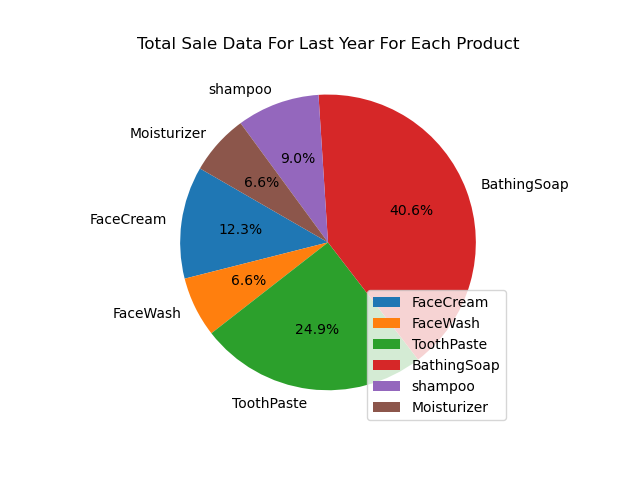
explode=(0,0,0,0,0,0),

autopct='%1.1f%%')

plt.title("Total Sale Data For Last Year For Each Product")

plt.legend(loc='lower right')

plt.show()



**Practical – 10**

Create a python program using Scikit-learn to implement the following model on dataset:

**Logistic Regression:**

If the desired output consists of one or more continuous variables, then the task is called *regression*.

Plot decision surface of multinomial and One-vs-Rest Logistic Regression. The hyperplanes corresponding to the three One-vs-Rest (OVR) classifiers are represented by the dashed lines.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.linear\_model import LogisticRegression

# make 3-class dataset for classification

centers = [[-5, 0], [0, 1.5], [5, -1]]

X, y = make\_blobs(n\_samples=1000, centers=centers, random\_state=40)

transformation = [[0.4, 0.2], [-0.4, 1.2]]

X = np.dot(X, transformation)

for multi\_class in ('multinomial', 'ovr'):

clf = LogisticRegression(solver='sag', max\_iter=100, random\_state=42,

multi\_class=multi\_class).fit(X, y)

# print the training scores

print("training score : %.3f (%s)" % (clf.score(X, y), multi\_class))

# create a mesh to plot in

h = .02 # step size in the mesh

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),

np.arange(y\_min, y\_max, h))

# Plot the decision boundary. For that, we will assign a color to each

# point in the mesh [x\_min, x\_max]x[y\_min, y\_max].

Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])

# Put the result into a color plot

Z = Z.reshape(xx.shape)

plt.figure()

plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)

plt.title("Decision surface of LogisticRegression (%s)" % multi\_class)

plt.axis('tight')

# Plot also the training points

colors = "bry"

for i, color in zip(clf.classes\_, colors):

idx = np.where(y == i)

plt.scatter(X[idx, 0], X[idx, 1], c=color, cmap=plt.cm.Paired,

edgecolor='black', s=20)

# Plot the three one-against-all classifiers

xmin, xmax = plt.xlim()

ymin, ymax = plt.ylim()

coef = clf.coef\_

intercept = clf.intercept\_

def plot\_hyperplane(c, color):

def line(x0):

return (-(x0 \* coef[c, 0]) - intercept[c]) / coef[c, 1]

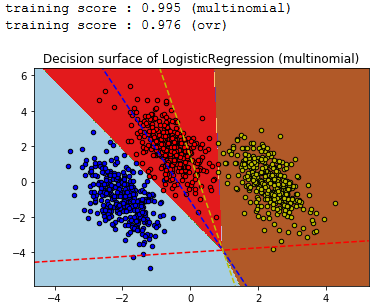
plt.plot([xmin, xmax], [line(xmin), line(xmax)],

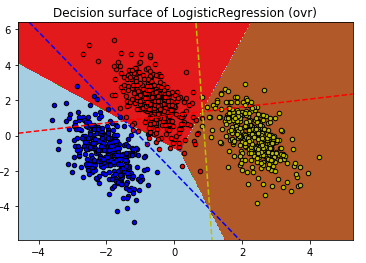
ls="--", color=color)

for i, color in zip(clf.classes\_, colors):

plot\_hyperplane(i, color)

plt.show()





**Support Vector Machines:**

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

Plot decision function of a weighted dataset, where the size of points is proportional to its weight. The sample weighting rescales the C parameter, which means that the classifier puts more emphasis on getting these points right. The effect might often be subtle. To emphasize the effect here, we particularly weight outliers, making the deformation of the decision boundary very visible.

import numpy as np

import matplotlib.pyplot as plt

from sklearn import svm

def plot\_decision\_function(classifier, sample\_weight, axis, title):

# plot the decision function

xx, yy = np.meshgrid(np.linspace(-4, 5, 500), np.linspace(-4, 5, 500))

Z = classifier.decision\_function(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

# plot the line, the points, and the nearest vectors to the plane

axis.contourf(xx, yy, Z, alpha=0.75, cmap=plt.cm.bone)

axis.scatter(X[:, 0], X[:, 1], c=y, s=100 \* sample\_weight, alpha=0.9,

cmap=plt.cm.bone, edgecolors='black')

axis.axis('off')

axis.set\_title(title)

# we create 20 points

np.random.seed(0)

X = np.r\_[np.random.randn(10, 2) + [1, 1], np.random.randn(10, 2)]

y = [1] \* 10 + [-1] \* 10

sample\_weight\_last\_ten = abs(np.random.randn(len(X)))

sample\_weight\_constant = np.ones(len(X))

# and bigger weights to some outliers

sample\_weight\_last\_ten[15:] \*= 5

sample\_weight\_last\_ten[9] \*= 15

# for reference, first fit without sample weights

# fit the model

clf\_weights = svm.SVC(gamma=1)

clf\_weights.fit(X, y, sample\_weight=sample\_weight\_last\_ten)

clf\_no\_weights = svm.SVC(gamma=1)

clf\_no\_weights.fit(X, y)

fig, axes = plt.subplots(1, 2, figsize=(14, 6))

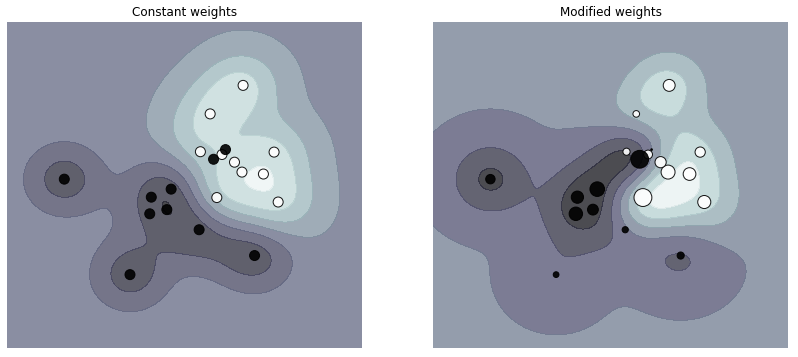
plot\_decision\_function(clf\_no\_weights, sample\_weight\_constant, axes[0],

"Constant weights")

plot\_decision\_function(clf\_weights, sample\_weight\_last\_ten, axes[1],

"Modified weights")

plt.show()



**Naive Bayes:**

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes’ Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Implementation of Gaussian Naive Bayes classifier using scikit-learn.

# load the iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# store the feature matrix (X) and response vector (y)

X = iris.data

y = iris.target

# splitting X and y into training and testing sets

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=1)

# training the model on training set

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X\_train, y\_train)

# making predictions on the testing set

y\_pred = gnb.predict(X\_test)

# comparing actual response values (y\_test) with predicted response values (y\_pred)

from sklearn import metrics

print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy\_score(y\_test, y\_pred)\*100)



**Random Forest:**

The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

IRIS flower datasets to train and test the model. We will build a model to classify the type of flower.

# importing required libraries

# importing Scikit-learn library and datasets package

from sklearn import datasets

# Loading the iris plants dataset (classification)

iris = datasets.load\_iris()

print(iris.target\_names)

print(iris.feature\_names)

# dividing the datasets into two parts i.e. training datasets and test datasets

X, y = datasets.load\_iris( return\_X\_y = True)

# Spliting arrays or matrices into random train and test subsets

from sklearn.model\_selection import train\_test\_split

# i.e. 80 % training dataset and 30 % test datasets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.70)

# importing random forest classifier from assemble module

from sklearn.ensemble import RandomForestClassifier

import pandas as pd

# creating dataframe of IRIS dataset

data = pd.DataFrame({'sepallength': iris.data[:, 0], 'sepalwidth': iris.data[:, 1],'petallength': iris.data[:, 2], 'petalwidth': iris.data[:, 3],'species': iris.target})

# printing the top 5 datasets in iris dataset

print(data.head())

# creating a RF classifier

clf = RandomForestClassifier(n\_estimators = 100)

# Training the model on the training dataset

# fit function is used to train the model using the training sets as parameters

clf.fit(X\_train, y\_train)

# performing predictions on the test dataset

y\_pred = clf.predict(X\_test)

# metrics are used to find accuracy or error

from sklearn import metrics

print()

# using metrics module for accuracy calculation

print("ACCURACY OF THE MODEL: ", metrics.accuracy\_score(y\_test, y\_pred))

# predicting which type of flower it is.

clf.predict([[3, 3, 2, 2]])

# importing random forest classifier from assemble module

from sklearn.ensemble import RandomForestClassifier

# Create a Random forest Classifier

clf = RandomForestClassifier(n\_estimators = 100)

# Train the model using the training sets

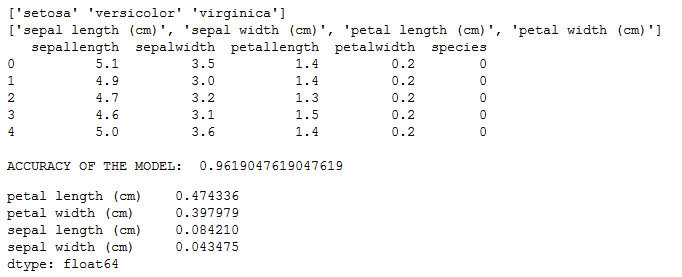
clf.fit(X\_train, y\_train)

# using the feature importance variable

import pandas as pd

feature\_imp = pd.Series(clf.feature\_importances\_, index = iris.feature\_names).sort\_values(ascending = False)

feature\_imp



**AdaBoost:**

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers.

We’ll be using a dataset that categories people as attractive or not based on certain features.

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.datasets import load\_breast\_cancer

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix

from sklearn.preprocessing import LabelEncoder

breast\_cancer = load\_breast\_cancer()

print(breast\_cancer.feature\_names)

print(breast\_cancer.data)

X = pd.DataFrame(breast\_cancer.data, columns=breast\_cancer.feature\_names)

y = pd.Categorical.from\_codes(breast\_cancer.target, breast\_cancer.target\_names)

encoder = LabelEncoder()

binary\_encoded\_y = pd.Series(encoder.fit\_transform(y))

train\_X, test\_X, train\_y, test\_y = train\_test\_split(X, binary\_encoded\_y, random\_state=1)

classifier = AdaBoostClassifier(DecisionTreeClassifier(max\_depth=1),n\_estimators=200)

classifier.fit(train\_X, train\_y)

predictions = classifier.predict(test\_X)

print(predictions)

confusion\_matrix(test\_y, predictions)

