PYTHON & MACHINE LEARNING BASIC APPLICATIONS

A CERTIFICATE COURSE CONDUCTED

BY

THE SURE Trust

Skill Upgradation for Rural-youth Empowerment – Trust

https://suretrustforruralyouth.com/

Course Training Attended

By

CHELIMI NANDINI

MARCH 2023 – JULY 2023

DECLARATION

This is to certify that **Ms. CHELIMI NANDINI** has successfully completed the four months training given in "**PYTHON & MACHINE LEARNING BASIC APPLICATIONS**" by SURE Trust during the period from March 2023 to July 2023.

Mr. Bhargavesh Dakka

Trainer

Python with Machine Learning

SURE Trust

Prof.Ch. Radha Kumari

Executive Director & Founder,

SURE Trust

Mrs.Vandana Nagesh

Director & Co-Founder,

SURE Trust

TABLE OF CONTENTS

About the SURE Trust

Introduction to the SURE Trust

The SURE Trust is born to enhance the employability of educated unemployed rural youth. It is observed that there is a wide gap between the skills acquired by students from the academic institutions and the skills required by the industry to employ them. Employability enhancement is done through giving one on one training in emerging technologies, completely through online mode . The mission of the SURE Trust is to bridge the gap between the skills acquired and the skills required by training them in the most emerging technologies such as Autocad & Solidworks, Python Program, Machine Learning (ML), Deep Learning (DL), Data Science & Data Analytics, Block chain Technology, Robotic Process Automation (RPA), and Project Management and twenty other essential and high in demand Courses ,that will enhance their employability. After completion of four months training in the course, the trainees will get live projects from industries as internship activity to get experience in applying to real time situation what they have learnt during the course. These projects will give them hands on experience which is much sought after by the prospective industry employing them. Currently students from all over India are enrolling for various courses offered by the SURE Trust. The SURE Trust offers every course free of cost with no financial burden of any kind to students. This initiative is purely a service-oriented one aiming to guide the rural youth who are educated but unemployed due to lack of upgradation in their skill sets. The birth of SURE Trust is a God given boon to rural youth who could reach great heights either in employment or in entrepreneurship once they receive the training offered followed by the company internship. Many companies are coming forward to join their hands with us by offering internship projects to hand hold and lead the rural youth in their career settlement.

Vision of the SURE Trust

The vision of the SURE Trust is to enhance the employability of educated unemployed youth, particularly living in rural areas, through skill upgradation, with no cost to the students.

Mission of the SURE Trust

The mission is to bridge the gap between the skills acquired in the academic institutions and the skills required in industries as a pre-condition for employment.

Functioning of the SURE Trust

There are three dedicated, committed, and hard-working women on the board of management of the SURE Trust who will look into the various administrative and other matters relating to the enrolment of students, organizing trainers, entering into agreements with companies for getting live projects to students as internship programs, and so on.

All the three women on the board are all the alumni from Sri Sathya Sai Institute of Higher Learning, Anantapur Campus, deemed to be a University. The women board is supported by

five eminent advisories who are from different walks of life and have made outstanding mark in career in their respective fields. For more details about SURE Trust please visit the website www.suretrustforruralyouth.com.

2. Course Content

The SURE Trust conducts a five months training for every course on a uniform basis. A session spanning across one to one & half hour is taken by the trainers for every major course. Sessions are conducted to complete the predesigned course structure within the fixed time period. Course content is designed to suit the current requirement of the Industry and validated by the industry experts. The course content of all these courses is so dynamic that any changed condition noticed in the industry will automatically get reflected in the content of the respective course . As the course content is dynamic, the Following is the course content of the current course in Data Science and Data Analytics:

PYTHON & MACHINE LEARNING BASIC APPLICATIONS

Objective:

The objective of this course is to train the learners to program in Python, analyzing and visualizing data, developing & evaluating models; and get introduced to complex computations.students in the basic to design python software.

Course content:

Module 1. Python for data science and machine learning:

- Python basics
- Python OOPs

Module 2. Data analysis:

- Pandas library
- Numpy library

Module 3. Data visualization:

- Matplotlib
- Seaborn

Module 4. Exploratory data analysis:

- Univariate
- Bivariate
- Derived metrics
- Introduction to databases (MySQL for Data science students)

Module 5. Math for machine learning:

- Statistics
- Linear algebra
- Calculus
- Probability theory

Module 6. Supervised learning:

- Feature selection
- Regression
- Linear regression
- Polynomial regression
- Advanced regression introduction
- Classification
- Logistic regression
- Naive Bayes
- Support vector machine
- Tree models

Module 7. Advanced regression:

- Regularized regression
- Ridge and Lasso regression
- Model selection and Grid Search methods

Module 8. Unsupervised learning:

- K means clustering
- Hierarchical clustering
- Principal component analysis

Module 9. Theory introduction to deep learning and reinforcement learning :

- Theory about ANN, CNN, RNN, LSTM
- Theory about Markov Decision process

3.Conduct of the course

a) Modalities for the conduct of all the courses are fixed by the SURE Trust which are uniformly followed across the courses.

- Mode of Training --- Online
- Period of Training --- Four months
- Sessions per week --- 5 to 6
- Length of the session --- 1 to 2 hours
- Tests to be taken --- 2 per month
- Assignments --- 1 per day
- Last 15 days --- Final practice and preparing the course report

b) Student byelaws:

- Students enrolling for the courses under SURE Trust are strictly required to follow the following byelaws set for them.

Byelaws for students to become eligible for certificate at the end of the course

I. Minimum Attendance:

- Every student must put in a minimum of 85% attendance in attending the classes for getting the eligibility to receive the certificates.

II. Two written tests are to be taken in each month:

Since the objective of the certification program is to turn out well qualified students from the respective courses,minimum two written tests are to be taken in each month for each course to ensure that the students are pulled along the expected line of standard.

III. Assignment submissions:

Ten exercises constituting one assignment for every two to three new functions/topics taught, resulting in minimum seven such assignments are to be submitted during the four months period.

IV. Preparing the final course report in the prescribed format:

During the last fifteen days in the fourth month, students many be asked to consolidate and compile all the assignments submitted in a word document along with the other chapters which will constitute a course report for each student. This report will be the unique contribution a student carries from the trust to show case the rigorous training he/she received during the four months period. Besides the report will stand as a testimony for the detailed learning a student has

acquired in the chosen area. This will facilitate the industry in handpicking the required student for the job.

V. External Viva-vioce:

Every student has to successfully clear the external viva-voce arranged in their respective course.

VI. KYC norms:

Each student wishing to enroll for the course must submit a written letter saying that he/she will not drop from the course until its completion, which will also be signed by father / mother besides the student himself / herself.

VII. Attend the full class:

All the students are expected to attend each class for full duration. Some students are observed moving out of classes after logging in which does not go well with the learning objective of students.

VIII. Ensure discipline in the group:

All the students are advised strictly to follow group etiquette and restrain from posting in the group any unethical messages or teasing messages or personal interactive messages. This group is purely created for academic purpose and hence only academic interactions should go on.

Python & Machine Learning Basic Applications

- The course is taught to the students as per the syllabus prescribed and as per the course modalities keeping in mind the bylaws set for them.
- Periodical tests are conducted to assess the understanding of the students in the course.
- Assignments are given to ensure that students gain versality in the designing.
- The assignments of the student are given below. These assignments which are done with high creativity and out of box thinking not only reflect the student's innovativeness but also constitute solved exercises in the Various designs for the fresh learners to practice and gain confidence.
- The assignments are listed below in the order of their conducting during the course.

Assingment 1 :

1.Write a program to accept percentage from the User and display according

```
#Assignment 08-03-23
# Write a program to accept percentage from the User and display according
a = int(input("enter the percentage:"))
if a >= 90:
    print('Excellent')
elif a >= 60 and a < 90:
    print('good')
elif a >= 40 and a < 60:
    print('average')
else:
    print('fail')
```

```
enter the percentage:60
good
```

Assingment 2 :

Calculate Principle, Tenure and Rate of Interest
Output should be calculated of Simple interest by using user input format

```
Principle_Amount = int(input("Enter the Amount : "))
Tenure = int(input("Enter time in Years : "))
Rate_of_interest = int(input("Enter the Rate of Interest : "))

simple_interest = (Principle_Amount * Tenure * Rate_of_interest) / 100
print("simple interest",simple_interest)

#Find the Even number by using input User method

a = int(input("Enter the number : "))
if a % 2 ==0:
    print("Given number is even")
else:
    print("Given number is odd")

```

```
Enter the Amount : 25000
Enter time in Years : 3
Enter the Rate of Interest : 5
simple interest 3750.0
Enter the number : 6
Given number is even
```

Assingment 3 :

print the Message

Calculater :

1. Add
2. Subtract
3. Multiply
4. Divide

#input

Enter Choice (1-4) : 3 Enter A:10 Enter B:20

output

product = 200 by using input Format

```python
choice = int(input("enter the choice in between 1-4 : "))
b = int(input("Enter the number1 :"))
c = int(input("Enter the number2 :"))
if choice == 1:
    d = b + c
    print(d)
elif choice == 2:
    d = b - c
    print(d)
elif choice == 3:
    d = b * c
    print(d)
elif choice == 4:
    d = b / c
    print(d)
else:
    print("not in choice")


```

```
enter the choice in between 1-4 : 3
Enter the number1 :10
Enter the number2 :20
200
```

Assingment 4 :

Create a Vault by using input user method

Input

Enter a String = xyz Enter a String – xyz123

Output

Either it could be 1.Successfully logged in!!! 2.Please Check your userid Or Password Use by if and Else Statement

```python
A = input("Enter a String :")
B = input("Enter a String")
if A == 'xyz' and B == 'xyz123':
        print("Successfully Logged")
else:
    print("Please Check your userid Or Password")

```

```
Enter a String :xyz
Enter a Stringxyz123
Successfully Logged
```

<u>Assingment 5</u> :

Assignment 13-03-22
WAP Currency Converter should be able to convert a Specific Currency
Inputs :
USD (U.S.Dollars)(1 USD = 82.16 INR)
YEN (japanese YEN)(1 YEN = 0.62 INR)
EURO (1 EURO = 88.04 INR
U.K. POUND(1 U.K.Pound = 99.86 INR)

Enter the currency which you want to convert- eg USD-EURO
Enter the value of the currency you want to convert 5000
the converted value-

```python
currency = int(input("Enter the indian rupees you want to convert :"))
choice = int(input("enter the choice 1-4:"))
if choice == 1:
    USD = currency * 82.16
    print("USD currency : ",USD)
elif choice == 2:
    YEN = currency * 0.62
    print("YEN currency : ",YEN)
elif choice == 3:
    EURO = currency * 99.86
    print("EURO currency : ",EURO)
elif choice == 4:
    UK = currency * 99.86
    print("UK currency : ",UK)
else:
    print("correct choice")
```

```
Enter the indian rupees you want to convert :5000
enter the choice 1-4:4
UK currency :  499300.0
```

Assignment 6 :

Accept the three numbers form the user and display the Second largest number

```python
number1 = int(input("Enter the Number1 :"))
number2 = int(input("Enter the Number2 :"))
number3 = int(input("Enter the Number3 :"))
if (number1 <= number2) and (number2 <= number3):
    print(number2)
elif (number1 <= number3 and number3 <= number2:
    print(number3)
else:
    print(number1)

```

```
Enter the Number1 :1
Enter the Number2 :3
Enter the Number3 :2
2
```

Assignment 7 :

Write a Python program to check if a triangle is a equilateral,isosceles or Scalene
An equilateral triangle is a triangle in which all three sides are equal An scalenev triangle is a triangle in which all three sides are unequal An isosceles triangle is a triangle in which any two sides are equal

```python
a = int(input("enter the first side of triangle : "))
b = int(input("enter the second side of triangle : "))
c = int(input("enter the third side of triangle : "))
if a == b == c:
    print("equilateral triangle")
elif a != b != c:
    print("scalene triangle")
else:
    print("isosceles triangle")
```

```
enter the first side of triangle : 2
enter the second side of triangle : 2
enter the third side of triangle : 4
isosceles triangle
```

Assignment 8 :

Enter the Selling Price and caluculate discount based on the range of Selling Price The discount rate is: a)5% if Selling Price is atmost 5000. b)12% if Selling Price is atmost 15000. c)20% if

```
a = int(input("Enter the Selling_Price"))
if a > 0 and a <= 5000:
    discount  = 5
    discountAmount = (a*discount)/100
    print(discountAmount)
elif a > 0 and a <= 15000:
    discount  = 12
    discountAmount = (a*discount)/100
    print(discountAmount)
elif a > 0 and a <= 25000:
    discount  = 20
    discountAmount = (a*discount)/100
    print(discountAmount)
else:
    discount = 30
    discountAmount = (a*discount)/100
    print(discountAmount)


```

```
Enter the Selling_Price13000
1560.0
```

Selling Price is atmost 25000. d)30% if Selling Price is atmost 25000.

Assignment 9 :

Write a Python programe to convert a month name to a number of days.Expected Output: List of months:January,February,March,April,May,June,August,September,October,November,December Input the name of Month:February no of days:28/29 days.

```
month = input('enter the month')
a = ['January','March','May','July','September','November']
b = ['April','june','August','October','December']
if month in a:
    print("No of days : 31")
elif month in b:
    print("No of days :30")
else:
    print("No of days :28/29")
```

```
enter the monthFebrauary
No of days :28/29
```

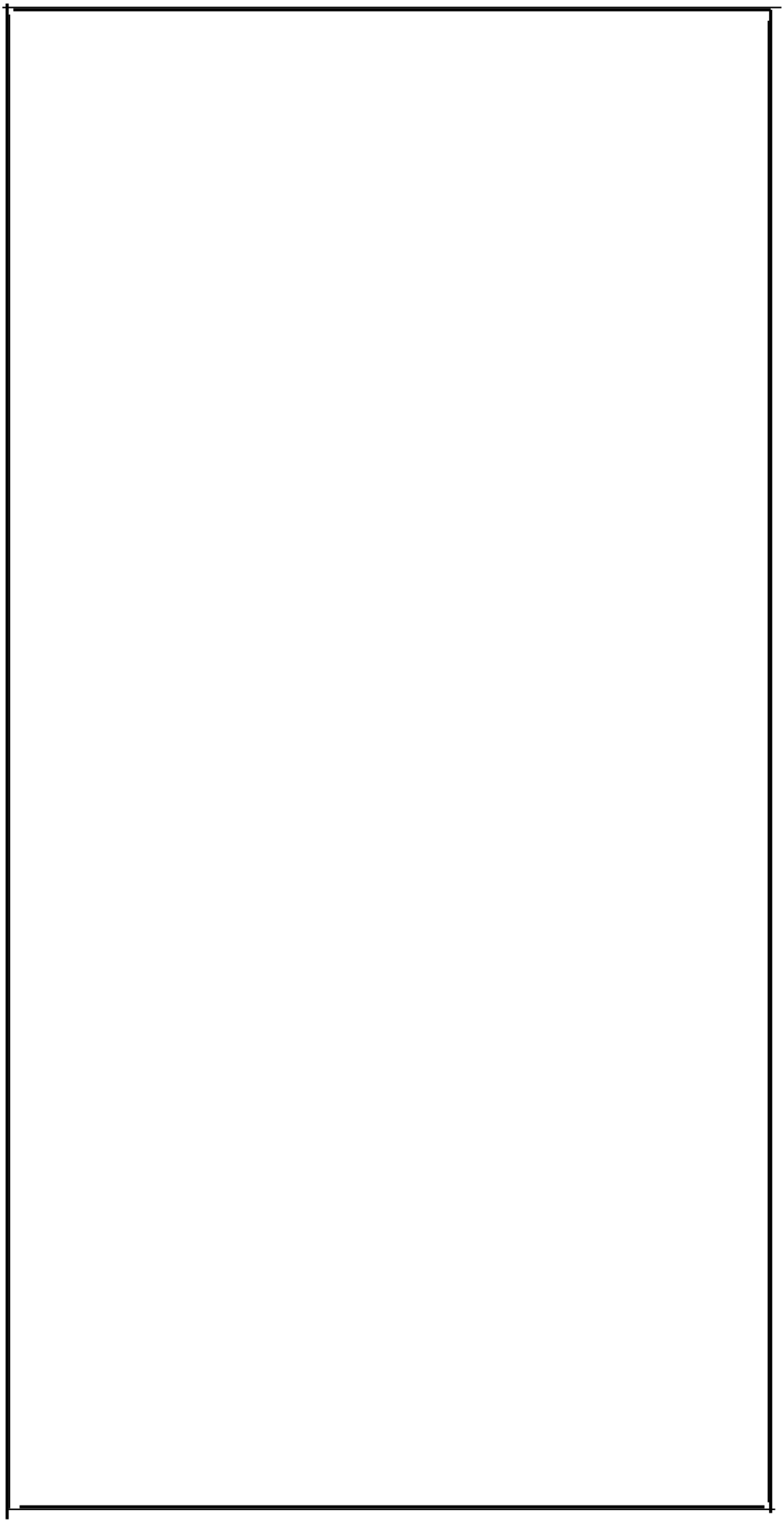
Assignment 10 :

Print a number pattern using a for loop and range function:

```
for numbers in range(15):
    for i in range(numbers):
        print(numbers,end = ' ')
    print('\n')


for i in range(0,5):
    for j in range(0,i+2):
        print('*',end = ' ')
    print()
for i in range(0,5):
    for j in range(0,10):
        print('*',end = ' ')
    print()
```

```
Pune
Mumbai
Delhi
23
45
12
9
29
5
```

```
12
9
29
5
4
3


1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

6 6 6 6 6 6

7 7 7 7 7 7 7

8 8 8 8 8 8 8 8

9 9 9 9 9 9 9 9 9
```



```
8 8 8 8 8 8 8 8

9 9 9 9 9 9 9 9 9

10 10 10 10 10 10 10 10 10 10

11 11 11 11 11 11 11 11 11 11 11

12 12 12 12 12 12 12 12 12 12 12 12

13 13 13 13 13 13 13 13 13 13 13 13 13

14 14 14 14 14 14 14 14 14 14 14 14 14 14
```

```
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
```

Assignment 11 :

if is used for comparison and for is used for you are analyzing house prices.The qiven code declare a list with house price in the heighborhood.you need to calculate and output the number of houses thst have a price that is above the average.

```
house_price = [500000,300000,40000,10000]
total = 0
number = 0
for i in house_price:
    total = total + i
    number = number +1
print(total)
print(number)
average = total/number
print(average)
```

```
850000
4
212500.0
```

Assignment 12 :

if we have a ticket number and age
write a program to bulid if age is greater than 18 then ticket cost is 20 otherwise 5

```
# Assignment 21-03-23
data = {"100-90":25,"42-01":48,"55-09":12,"128-64":71,"002-22":18,"321-54":19}
x = data.values()
print(x)
for i in x:
    if( i >= 18):
        ticket = 20
        print("t20",i)
    else:
        ticket = 5
        print("5",i)
```

```
dict_values([25, 48, 12, 71, 18, 19])
20 25
20 48
5 12
20 71
20 18
20 19
```

Assignment 13 :

write the combination of if and else statement The statement search for prime numbers
from 10 through Break condition

```python
for i in range(10,20):
    for j in range(2, i // 2 + 1):
        if (i % j == 0):
            break
        print(i)
```

```
11
11
11
11
13
13
13
13
13
15
17
17
17
17
17
17
```

17
17
19
19
19
19
19
19
19
19
19

Assignment 14 :

print the table of number for 13 to 17 by list comprehension

```python
# Assignment
# print the table of number for 13 to 17 by list comprehension
table = [[i,'x',j,'=',i*j] for i in range(13,18) for j in range(1,11)]
print(table)
```

```
[[13, 'x', 1, '=', 13], [13, 'x', 2, '=', 26], [13, 'x', 3, '=', 39], [13, 'x', 4, '=', 52], [13, 'x', 5, '=', 65],
[13, 'x', 6, '=', 78], [13, 'x', 7, '=', 91], [13, 'x', 8, '=', 104], [13, 'x', 9, '=', 117], [13, 'x', 10, '=', 13
0], [14, 'x', 1, '=', 14], [14, 'x', 2, '=', 28], [14, 'x', 3, '=', 42], [14, 'x', 4, '=', 56], [14, 'x', 5, '=', 7
0], [14, 'x', 6, '=', 84], [14, 'x', 7, '=', 98], [14, 'x', 8, '=', 112], [14, 'x', 9, '=', 126], [14, 'x', 10, '=',
140], [15, 'x', 1, '=', 15], [15, 'x', 2, '=', 30], [15, 'x', 3, '=', 45], [15, 'x', 4, '=', 60], [15, 'x', 5, '=',
75], [15, 'x', 6, '=', 90], [15, 'x', 7, '=', 105], [15, 'x', 8, '=', 120], [15, 'x', 9, '=', 135], [15, 'x', 10, '=
', 150], [16, 'x', 1, '=', 16], [16, 'x', 2, '=', 32], [16, 'x', 3, '=', 48], [16, 'x', 4, '=', 64], [16, 'x', 5, '=
', 80], [16, 'x', 6, '=', 96], [16, 'x', 7, '=', 112], [16, 'x', 8, '=', 128], [16, 'x', 9, '=', 144], [16, 'x', 10,
'=', 160], [17, 'x', 1, '=', 17], [17, 'x', 2, '=', 34], [17, 'x', 3, '=', 51], [17, 'x', 4, '=', 68], [17, 'x', 5,
'=', 85], [17, 'x', 6, '=', 102], [17, 'x', 7, '=', 119], [17, 'x', 8, '=', 136], [17, 'x', 9, '=', 153], [17, 'x',
10, '=', 170]]
```

Project 1 :

1.find all of the numbers from 1-1000 that are divisible by 7

2.find all the numbers from 1-1000 that have a 3 in them

3.count the number of spaces in a string

4.Create a list of all consonants in the string "Yellow Yaks like Yelling and yawning and yesturday they yodled while eating yuky yams"

5.Get the index and the values as a tuple for items in the list "hi",4,8.99,'apple',(t,b,n).result would look like(index,value)

6. Get only the numbers in a sentence like 'in 1984 there were 13 instances of a protest with over 1000 people attending'

7.Given numbers = range(20),products a list containing the word 'even' if a number in the number is even,and the word 'odd' if the number is odd.Result would look like 'odd','odd','even'

```python
#31/03/22
# 1.find all of the numbers from 1-1000 that are divisible by 7
numbers = [i for i in range(1,1000) if i%7 == 0]
print(numbers)
```

```
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112, 119, 126, 133, 140, 147, 154, 161, 168, 175, 182,
189, 196, 203, 210, 217, 224, 231, 238, 245, 252, 259, 266, 273, 280, 287, 294, 301, 308, 315, 322, 329, 336, 343, 3
50, 357, 364, 371, 378, 385, 392, 399, 406, 413, 420, 427, 434, 441, 448, 455, 462, 469, 476, 483, 490, 497, 504, 51
1, 518, 525, 532, 539, 546, 553, 560, 567, 574, 581, 588, 595, 602, 609, 616, 623, 630, 637, 644, 651, 658, 665, 67
2, 679, 686, 693, 700, 707, 714, 721, 728, 735, 742, 749, 756, 763, 770, 777, 784, 791, 798, 805, 812, 819, 826, 83
3, 840, 847, 854, 861, 868, 875, 882, 889, 896, 903, 910, 917, 924, 931, 938, 945, 952, 959, 966, 973, 980, 987, 99
4]
```

```python
# 2.find all the numbers from 1-1000 that have a 3 in them
numbers = [i for i in range(1,1000) if '3' in str(i)]
print(numbers)
```

```
[3, 13, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 43, 53, 63, 73, 83, 93, 103, 113, 123, 130, 131, 132, 133, 134,
135, 136, 137, 138, 139, 143, 153, 163, 173, 183, 193, 203, 213, 223, 230, 231, 232, 233, 234, 235, 236, 237, 238, 2
39, 243, 253, 263, 273, 283, 293, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 31
6, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 33
9, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 36
2, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 38
5, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 403, 413, 423, 430, 431, 432, 433, 434, 43
5, 436, 437, 438, 439, 443, 453, 463, 473, 483, 493, 503, 513, 523, 530, 531, 532, 533, 534, 535, 536, 537, 538, 53
9, 543, 553, 563, 573, 583, 593, 603, 613, 623, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 643, 653, 663, 67
3, 683, 693, 703, 713, 723, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 743, 753, 763, 773, 783, 793, 803, 81
3, 823, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 843, 853, 863, 873, 883, 893, 903, 913, 923, 930, 931, 93
2, 933, 934, 935, 936, 937, 938, 939, 943, 953, 963, 973, 983, 993]
```

```python
#3.count the number of spaces in a string
string = 'iidf jfkddk dfnk'
spaces = [i for i in string if i == ' ']
print(len(spaces))
```

```
2
```

```python
# 4.Create a list of all consonants in the string "Yellow Yaks like Yelling and yawning and yesturday
# they yodled  while eating yuky yams"
string = "Yellow Yaks like Yelling and yawning and yesturday they yodled  while eating yuky yams"
consonants = [i for i in string if i not in 'aeiou']
print(consonants)
```

```
['Y', 'l', 'l', 'w', ' ', 'Y', 'k', ' ', 'l', 'k', ' ', 'Y', 'l', 'l', 'n', 'g', ' ', 'n', 'd', ' ', 'y', 'w', 'n',
'n', 'g', ' ', 'n', 'd', ' ', 'y', 't', 'r', 'd', 'y', ' ', 't', 'h', 'y', ' ', 'y', 'd', 'l', 'd', ' ', ' ', 'w', '
h', 'l', ' ', 't', 'n', 'g', ' ', 'y', 'k', 'y', ' ', 'y', 'm']
```

```python
# 5.Get the index and the values as a tuple for items in the list "hi",4,8.99,'apple',(t,b,n).
#result would look like(index,value)
items = ("hi",4,8.99,'apple',('t','b','n'))
a = [(items.index(i),i) for i in items]
print(a)
```

```
[(0, 'hi'), (1, 4), (2, 8.99), (3, 'apple'), (4, ('t', 'b', 'n'))]
```

```python
#6. Get only the numbers in a sentence like 'in 1984 there were 13 instances of a protest with over 1000 people a
text = 'in 1984 there were 13 instances of a protest with over 1000 people attending'
number = [int(i) for i in text.split() if i.isdigit() == True]
print(number)
```

```
[1984, 13, 1000]
```

```python
# 9. Find all of the words in a string that are less then 4 letters
strings = 'dsfsd sdf sdfds sfgds yh d'
word = strings.split()
b = [i for i in word if len(i) < 4]
print(b)
```

```
['sdf', 'yh', 'd']
```

```
['even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd']
```

```python
# 8.Produce a list of tuples consisting of only the matching numbers in these lists list_a = 1,2,3,4,5,6,7,8,9
# list_b = 2,7,1,12.Result would look like(4,4),(12,12)
list_a = [1, 2, 3,4,5,6,7,8,9]
list_b = [2, 7, 1, 12]

matching_numbers = [(a, b) for a in list_a for b in list_b if a == b]
print(matching_numbers)
```

```
[(1, 1), (2, 2), (7, 7)]
```

Assignment 15 :

1.Have to take only even numbers from the key_value pairs

2.Have to take the values which are not even and lesser than 40

```python
# Assignment
original_dict = {'jack' : 38,'Michal' : 48,'Sara' : 57,'John' : 33}
print(original_dict)
# 1.Have to take only  even numbers from the key_value pairs
#2.Have to take the values which are not even and lesser than 40
new = {i : value for (i,value) in original_dict.items() if value%2 == 0}
print(new)
new = {i : value for (i,value) in original_dict.items() if value%2 != 0 if value < 40}
print(new)

```

```
{'jack': 38, 'Michal': 48, 'Sara': 57, 'John': 33}
{'jack': 38, 'Michal': 48}
{'John': 33}
```

Assignment 16:

Find the largest number inside the list without using sort function

```python
def largest(list1):
    lar = list1[0]
    for j in list1:
        if j > lar:
            lar = j
    return lar
list1 = [2,3,4,5,6,7]
print(largest(list1))
```

```
7
```

Assignment 17:

find life excectancy calculater
def new_life(name,age): ["jane",'Zack','Melissa']
smoker age 40 non smoker age 70
life remaining = life_exp - age
output: hii Jane! your life expectancy:-----years

```
list = ["jane",'Zack','Melissa']
def new_life(name,age):
    life_exp = int(input("enter the life_exp:"))
    life_remaining = life_exp - age
    if life_remaining >= 70:
        print('non smoker is:',life_remaining)
    elif life_remaining <= 40:
        print('smoker is:',life_remaining)
    return('hii Jane! your life expectancy:',life_remaining,'years')
print(new_life('jane',40))

```

```
enter the life_exp:90
('hii Jane! your life expectancy:', 50, 'years')
```

Assignment 18:

1.#Write a python program to sort a list of tuple using lambda

```
list1 = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
list1.sort(key = lambda x: x[1])
print(list1)
```

```
[('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]
```

2.write a python programe to find whether a given string starts with a given character #using lambda

```
##2.write a python programe to find whether a given string starts with a given character #using lambda
starts_with = lambda x: True if x.startswith('P') else False
print(starts_with('Python'))
starts_with = lambda x: True if x.startswith('P') else False
print(starts_with('Java'))
```

```
True
False
```

NUMPY :

Assignment 1:

write a numpy program to create a 2D array with 1 on the border and 0 inside

```
import numpy as np
#Assignments
#write a numpy program to create a 2D array with 1 on the border and 0 inside
a = np.ones([4,4])
print(a)
a[1:-1,1:-1] = 0
print(a)
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
[[1. 1. 1. 1.]
 [1. 0. 0. 1.]
 [1. 0. 0. 1.]
 [1. 1. 1. 1.]]
```

Assignment 2:

1.Write a numpy program to create a vector with value ranging from 15 to 55 and print all values except the first and last.

2.Write a numpy program to create a vector of length 5 filled with arbitrary integers from 0 to 10

3.Write a numpy program to create a 3 x 4 matrix filled with values 10 to 21

4.write a numpy program to create a 3 x 3 x 3 array filled with arbitrary values
5.write a numpy program to compute the inner product of two given vectors

```
#Write a numpy program to create a vector with value ranging from 15 to 55 and print al
import numpy as np
my_list = np.arange(15,55)
print(my_list[1:-1])
```

```
[16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
 40 41 42 43 44 45 46 47 48 49 50 51 52 53]
```

```
#Write a numpy program to create a vector of length 5 filled with arb
import numpy as np
x = np.random.randint(0,11,5)
print(x)
```

```
[1 3 2 2 2]
```

```
#Write a numpy program to create a 3 x 4 matrix filled with values 10 to 21
import numpy as np
matrix_empty = np.arange(10,22).reshape(3,4)
print(matrix_empty)
```

```
[[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]]
```

```
#write a numpy program to create a 3 x 3 x 3 array filled with arbitrary values
matrix = np.random.random((3,3,3))
print(matrix)
```

```
[[[0.68427598 0.59081529 0.70775988]
  [0.38529567 0.63227453 0.6659464 ]
  [0.77183711 0.22519084 0.46450498]]

 [[0.9351328  0.17641686 0.05244596]
  [0.89093222 0.19007541 0.79634274]
  [0.23541035 0.42078006 0.0684801 ]]

 [[0.28777127 0.49679965 0.34560908]
  [0.96190448 0.04514149 0.78453197]
  [0.45780534 0.73399965 0.16104794]]]
```

```
#write a numpy program to compute the inner product of two given vectors
a = np.array([1,2])
b = np.array([3,4])
print(np.dot(a,b))

```

```
11
```

Assignment 3:

Given a 2D binary matrix A of dimensions NXM,determine the row that contains a minimum number of 1's note-The matrix contains only 1s and 0s.Also,if two or more rows contain the minimum number of 1's, the answer is the lowest of those indices input: n=4,m=5 A=[[1,1,1,1],[1,1,0,0],[0,0,1,1],[1,1,1,1]] output: 2 Explanation: Rows 2 and 3 contain the minimum numberof 1's(2 each).Since,2 is less than 3. thus the answer is 2. your Task: you donot need to read input or print anything.your task is to complete the function minrow() which takes the two integers N,M as well as the 2D matrix as input parameter and returns the minimum index of the row which contains the least number of 1's

```
matrix = [[1,1,1,1],
          [1,1,0,0],
          [0,0,1,1],
          [1,1,1,1]]
column = []
for i in matrix:
    column.append(i.count(1))
    matrix_one = column[0]
for j in column:
    if(matrix_one > j):
        matrix_one = j
min_row_count = 0
for i in matrix:
    if(i.count(1) == matrix_one):
        print(i)
        min_row_count = min_row_count + 1
        print('\n')
    print("minimum Row of 1's : ",min_row_count)
```

```
minimum Row of 1's :  0
[1, 1, 0, 0]


minimum Row of 1's :  1
[0, 0, 1, 1]
```

```
minimum Row of 1's :  2
minimum Row of 1's :  2
```

Assignment

4:

Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N
find the missing elements
input:
n=5
a[]={1,2,3,5}
output:4
your task:
you donot need to read the input or print anything.

complete the function MissingNumber() that takes array and N as input parameters and returns the value of the missing number

```
arr = {1,2,3,5}
m = [element for element in arr]
print(m)
```

```
[1, 2, 3, 5]
```

```
def missingNo(a,n):
    total = (n + 1)*(n + 2)/2
    a_sum = sum(a)
    return total - a_sum
arr = {1,2,3,5}
a = [element for element in arr]
n = len(a)
miss = missingNo(a,n)
print(miss)
```

```
4.0
```

Assignment 5 :

Given an array arr[] of N+2 elements. all elements of the array are in the range of 1 to N.

And all elements occur once except two numbers occur twice.find the two repeating numbers.
input: arr = [4,2,4,5,2,3,1],N=5
output:4 2
Explanation: The above array has n + 2 =7 elements with all elements occuring once except 2 and 4 which occur twice.So the output should be 4 2

input arr =[2,1,2,1,3],N=3
output:1 2
Explanation:The above array has n + 2 = 5 elements with all elements occuring once except1 and 2 which occur twise.so output should be 1 2.

```
arr = [4, 2, 4, 5, 2, 3, 1]
n = len(arr)
N=5
if(N+2 == n):
    for i in range(0, n-1):
        for j in range(i + 1, n):
            if arr[i] == arr[j]:
                print(arr[i])
```

```
4
2
```

Assignment 6 :

Given two sorted arrays may have some commom elements.Find the sum of the maximum sum path to reach from the beginning

of any array to the end of any of the two arrays.we can switch from one array to another array only at common elements note: The common elements do not have to be at the same indexes. input: ar1 = {2,3,7,10,12}, ar2 = {1,5,7,8} output: 35.

```
def maxPathSum(ar1, ar2, m, n):
    i, j = 0, 0
    result, sum1, sum2 = 0, 0, 0
    while (i < m and j < n):
        if ar1[i] < ar2[j]:
            sum1 += ar1[i]
            i += 1
        elif ar1[i] > ar2[j]:
            sum2 += ar2[j]
            j += 1
        else:
            result += max(sum1, sum2) + ar1[i]
            sum1 = 0
            sum2 = 0
            i += 1
            j += 1
    print('result',result)
    print(sum1)
    print(sum2)
    while i < m:
        sum1 += ar1[i]
        i += 1
    while j < n:
        sum2 += ar2[j]
```

```
        j += 1
    print(sum1)
    print(sum2)
    result += max(sum1, sum2)

    return result
ar1 = [2, 3, 7, 10, 12]
ar2 = [1, 5, 7, 8]
m = len(ar1)
n = len(ar2)
print("Maximum sum path is", maxPathSum(ar1, ar2, m, n))
```

```
result 13
0
8
22
8
Maximum sum path is 35
```

Pandas:

Assignment 1:

Given an array of non-negative integers and an integers and an integer sum,find a subarray that adds to a given sum

```python
def subarray(arr, n, givensum):
    sum1 = arr[0]
    start = 0
    i = 1
    while i <= n:
        while sum1 > givensum and start < i-1:
            sum1 = sum1 - arr[start]
            start += 1
        if sum1 == givensum:
            print ("Subarray with given sum is between indexes % d and % d"%(start, i-1))
            return 1
        if i < n:
            sum1 = sum1 + arr[i]
        i += 1
    print ("Subarray with given sum is NOT Found")
    return 0
arr = [2, 6, 5, 31, 11, 8]
n = len(arr)
givensum = 53
subarray(arr, n, givensum)
```

```
Subarray with given sum is between indexes  1 and  4
1
```

Assignment 2:

1.create a series by using dictionary and filter out number which is greater than 3

```python
# 1.create a series by using dictionary and filter out number which is greater than 3
alphabet_dict = {'a' : 1,
                 'b' : 2,
                 'c' : 3,
                 'd' : 4}
series_dict = pd.Series(alphabet_dict)
print(series_dict[series_dict > 3])

```

```
d    4
dtype: int64
```

2.Create a series by using numpy array from the range 1 to 100 filter out all the number from 95 to 100

```python
import numpy as np
import pandas as pd
##2.Create a series by using numpy array from the range 1 to 100 filter out
series_array = pd.Series(np.arange(100))
print(series_array[series_array > 95].reset_index(drop = True))
```

```
0    96
1    97
2    98
3    99
dtype: int64
```

Assignment 3:

Write a python program to find palindromes in a given list of strings using lambda. Original list of strings: ['php','w3r','python','abcd','java','aaa'] List of pallindromes: ['php','aaa']

```python
String = ['php','w3r','python','abcd','java','aaa']
a = list(filter(lambda x : (x == ''.join(reversed(x))),String))
print(a)
```

```
['php', 'aaa']
```

Assignment 4:

Create a DataFrame with Index

```
data = {'Name' : ['Akshal','James','Mia','Emily','Robern','john','Jacob'],
        'Score' : [12,19,15,10,17,8,17],
        'Attempts' : [3,2,1,4,5,2,1],
        'Qualify' : ['Yes','Yes','Yes','NO','Yes','No','Yes']}
student = pd.DataFrame(data)
student
```

	Name	Score	Attempts	Qualify
0	Akshal	12	3	Yes
1	James	19	2	Yes
2	Mia	15	1	Yes
3	Emily	10	4	NO
4	Robern	17	5	Yes
5	john	8	2	No
6	Jacob	17	1	Yes

Assignment 5:

Customer id,Age,Gender,Salary,City_Residence

custermer_data_1

Create an another sheet with same custermer_data_2 and concate them

```
import pandas as pd
df_customer_1 = pd.read_excel('Untitled 2.xlsx',engine='openpyxl')
df_customer_1
```

	Customer_id	Age	Gender	Salary	City_Residence
0	Nandini	21	Female	50000	chennai
1	Ammu	19	Female	30000	Anantapur
2	Banny	18	Male	40000	kadapa
3	Siree	23	Female	45000	kurnool
4	Sunny	18	Male	40000	Dharmavaram

```
import pandas as pd
df_customer_2 = pd.read_excel('customer_data.xlsx',engine='openpyxl')
df_customer_2
```

	Customer_id	Age	Gender	Salary	City_Residence
0	Nandini	21	Female	50000	chennai
1	Ammu	19	Female	30000	Anantapur
2	Banny	18	Male	40000	kadapa
3	Siree	23	Female	45000	kurnool
4	Sunny	18	Male	40000	Dharmavaram

```
pd.concat([df_customer_1,df_customer_2],ignore_index = True)
```

	Customer_id	Age	Gender	Salary	City_Residence
0	Nandini	21	Female	50000	chennai
1	Ammu	19	Female	30000	Anantapur
2	Banny	18	Male	40000	kadapa
3	Siree	23	Female	45000	kurnool
4	Sunny	18	Male	40000	Dharmavaram
5	Nandini	21	Female	50000	chennai
6	Ammu	19	Female	30000	Anantapur
7	Banny	18	Male	40000	kadapa
8	Siree	23	Female	45000	kurnool
9	Sunny	18	Male	40000	Dharmavaram

Assignment 7:

Find city-wise distibution of salary for different genders in em-polye dataset

```
df_employee = pd.read_excel('employees.xlsx',engine = 'openpyxl')
df_employee
```

	Age	Gender	Ciry_residence	Annual	Year of Experience	Designation
0	45	Male	Mumbai	21	5	Cloud engineer
1	67	Male	Mumbai	1	5	Daat analyst intern
2	33	Male	Mumbai	2	5	Sr. Data Analyst
3	56	Male	Mumbai	8	5	Big data Engineer
4	32	Male	Mumbai	4	5	Tutor
5	21	Male	Mumbai	9	7	Tutor
6	56	Male	Bangalore	3	7	Staff
7	43	Female	Bangalore	5	7	Staff
8	43	Female	Bangalore	6	4	Staff
9	42	Female	Pune	7	4	Java Developer
10	55	Female	Pune	8	4	Java Developer
11	44	Female	Pune	10	3	Java Developer

12	27	Male	Pune	4	3	Human Resource
13	27	Female	Pune	3	3	Human Resource
14	27	Female	Delhi	8	2	Sales Manager
15	45	Male	Delhi	9	2	Sales Manager
16	89	Female	Delhi	10	21	Marketing Manager
17	56	Male	Delhi	12	1	Marketing Manager
18	78	Female	Delhi	5	6	Cloud engineer

```
# Find the City-Wise Gender count using the crossTab() method
pd.crosstab(df_employee.Gender,df_employee.Ciry_residence,
            rownames = ['Sex'],colnames = ['HomeTown'])
```

HomeTown	Bangalore	Delhi	Mumbai	Pune
Sex				
Female	2	3	0	4
Male	1	2	6	1

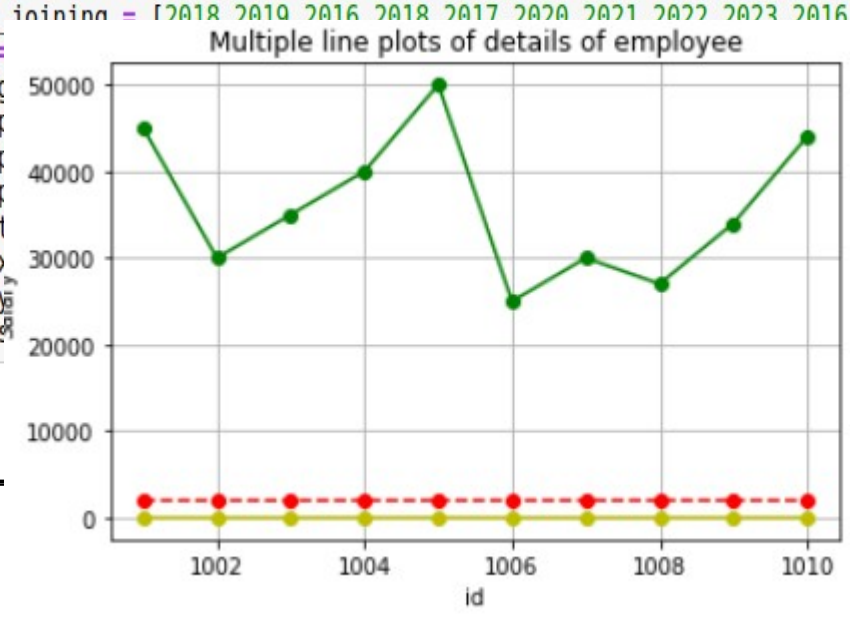
```
# Assignment
##Find city-wise distibution of salary for different genders
pd.crosstab(df_employee.Gender,df_employee.Annual,
            rownames = ['Sex'],colnames = ['Salary'])

```

Salary	1	2	3	4	5	6	7	8	9	10	12	21
Sex												
Female	0	0	1	0	2	1	1	2	0	2	0	0
Male	1	1	1	2	0	0	0	1	2	0	1	1

Assignment 8:

Give employee id 10 and then salary and year of joining and age do multiple line plots

```
id = [1001,1002,1003,1004,1005,1006,1007,1008,1009,1010]
salary = [45000,30000,35000,40000,50000,25000,30000,27000,34000,44000]
year_joining = [2018,2019,2016,2018,2017,2020,2021,2022,2023,2016]
age =
plt.
plt.
plt.
plt.
plt.
plt.
plt.
```




```
'o')
```

Project 2:

```
# importing Libraries
# importing Pandas Library as pd
import pandas as pd

# importing Numpy Library as np
import numpy as np

# importing matplotlib.pyplot as pl
import matplotlib.pyplot as plt

# imporing seaborn as sns
import seaborn as sns
```

```
# Loading the dataset using pandas
df = pd.read_csv('Expense.csv')

# Printing the dataset
df
```

1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	40	United-States	<=50K
2	50	Private	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	1902	65	United-States	>50K
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	40	United-States	>50K
4	23	Local-gov	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Female	0	0	60	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	38	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	<=50K
4996	26	Private	Some-college	10	Never-married	Tech-support	Own-child	White	Female	0	0	40	United-States	<=50K
4997	20	Private	11th	7	Never-married	Transport-moving	Own-child	White	Male	0	0	60	United-States	<=50K
4998	24	Private	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	60	Mexico	>50K
4999	40	Private	HS-grad	9	Divorced	Craft-repair	Not-in-family	White	Male	0	0	45	United-States	<=50K

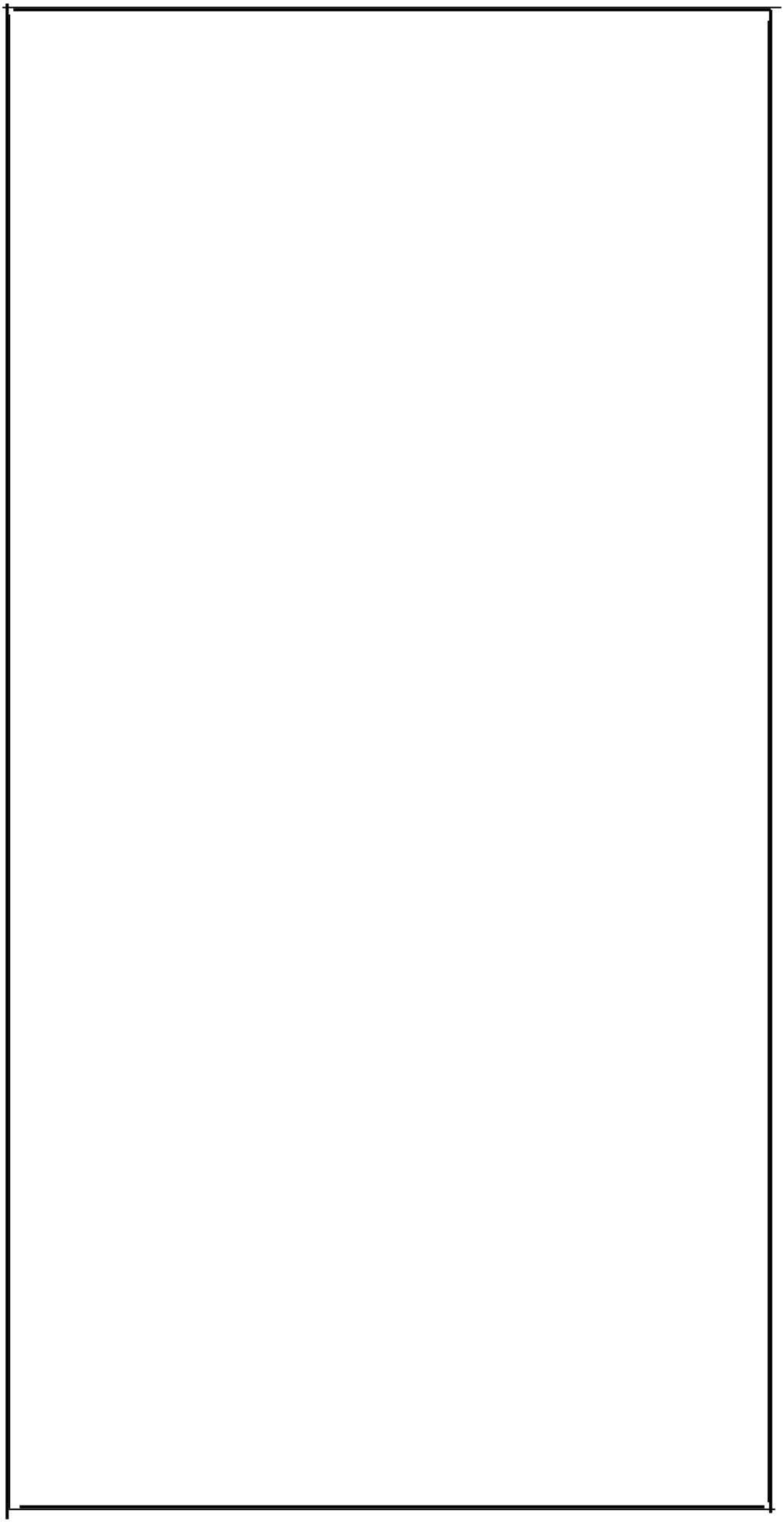
5000 rows × 14 columns

```
# By using head we are getting first 5 values
df.head(10)
```

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense
0	39	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	15024	0	50	United-States	>50K
1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	40	United-States	<=50K
2	50	Private	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	1902	65	United-States	>50K
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	40	United-States	>50K
4	23	Local-gov	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Female	0	0	60	United-States	<=50K
5	31	Private	HS-grad	9	Never-married	Other-service	Own-child	White	Male	0	0	16	United-States	<=50K
6	58	Private	Bachelors	13	Married-civ-spouse	Adm-clerical	Husband	White	Male	0	0	40	United-States	<=50K
7	66	Private	HS-grad	9	Separated	Machine-op-inspct	Not-in-family	Black	Male	0	0	40	United-States	<=50K
8	39	Self-emp-inc	HS-grad	9	Married-civ-	Exec-	Husband	White	Male	0	0	40	United-	>50K

```
# Information about the Dataset:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             5000 non-null   int64 
 1   workclass       5000 non-null   object
 2   education       5000 non-null   object
 3   education-num   5000 non-null   int64 
 4   marital-status  5000 non-null   object
 5   occupation      5000 non-null   object
 6   relationship    5000 non-null   object
 7   race            5000 non-null   object
 8   sex             5000 non-null   object
 9   capital-gain    5000 non-null   int64 
 10  capital-loss    5000 non-null   int64 
 11  hours-per-week  5000 non-null   int64 
 12  native-country  5000 non-null   object
 13  Expense         5000 non-null   object
dtypes: int64(5), object(9)
```



```
# By using tails we are getting last 5 values
df.tail(10)
```

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense
4990	32	Private	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States	<=50K
4991	34	Private	Bachelors	13	Never-married	Sales	Own-child	Black	Female	0	0	40	United-States	<=50K
4992	44	Private	9th	5	Divorced	Other-service	Own-child	White	Female	0	0	50	United-States	<=50K
4993	28	Private	HS-grad	9	Never-married	Handlers-cleaners	Unmarried	White	Male	0	0	40	United-States	<=50K
4994	35	Private	HS-grad	9	Divorced	Other-service	Not-in-family	White	Female	0	0	35	United-States	<=50K
4995	38	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	<=50K
4996	26	Private	Some-college	10	Never-married	Tech-support	Own-child	White	Female	0	0	40	United-States	<=50K

```
# isnull() method is used to check whether the dataset contain null values or not
# sum() is used to find count of null values
print(df.isnull().sum())

#getting size of the dataset(rows multiplied by column 1000x17)
print(df.size)

# getting rows and columnes in a dataset
print(df.shape)
```

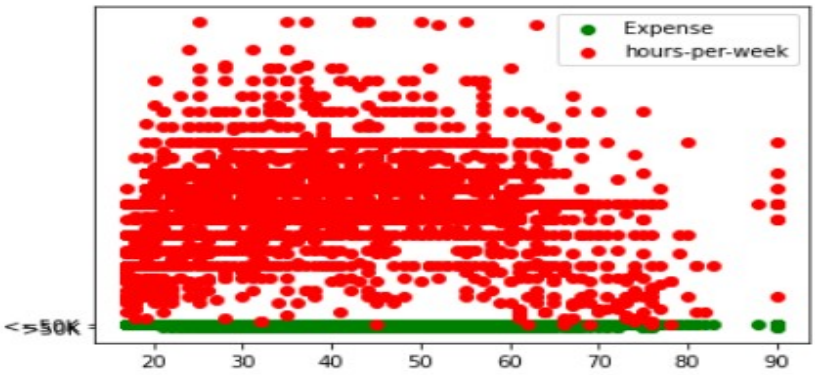
```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
```

```
# getting all statctical values like mean,median,count,min,max,standard deviation
# for numerical values
df.describe()
```

	age	education-num	capital-gain	capital-loss	hours-per-week
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	38.656000	10.065000	1104.080000	90.032800	40.566200
std	13.698292	2.558141	7579.674371	404.168991	12.154191
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	48.000000	12.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	3004.000000	99.000000

DATA VISUALISATION

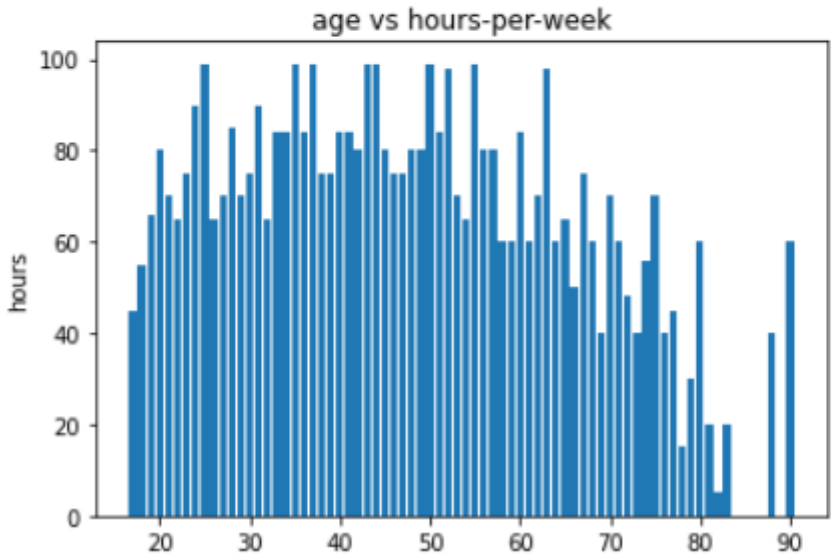
```
#Two column should be numerical
plt.scatter(x = 'age',y = 'Expense',data = df,color = 'g')
plt.scatter(x = 'age',y = 'hours-per-week',data = df,color = 'r')

#Describe all the elements of a graph
plt.legend()

#show the graph
plt.show()
```




```
# Summary stats for Categorical Column:
from warnings import filterwarnings
filterwarnings('ignore')
df.describe(include = [np.object])
```

	workclass	education	marital-status	occupation	relationship	race	sex	native-country	Expense
count	5000	5000	5000	5000	5000	5000	5000	5000	5000
unique	9	16	7	15	6	5	2	40	2
top	Private	HS-grad	Married-civ-spouse	Craft-repair	Husband	White	Male	United-States	<=50K
freq	3444	1602	2294	630	2026	4271	3374	4459	3776

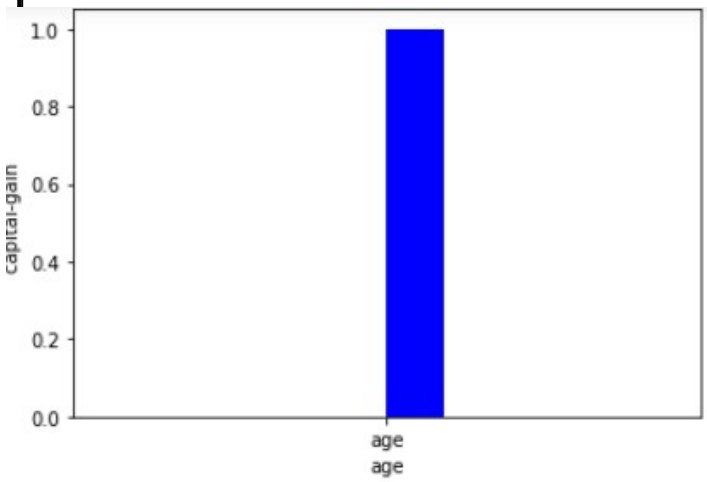
```python
# Bar chart with purpose against amount
plt.bar(df['age'], df['hours-per-week'])
plt.title("age vs hours-per-week")
# Setting the X and Y labels
plt.xlabel('age')
plt.ylabel('hours')

# Adding the legends
plt.show()
```



```python
# plot the histogram with multiple bins add grid :
plt.hist(x = df["age"],color ='r',bins=5)
plt.title("Credit dataset")
plt.xlabel("age")
plt.ylabel("hours-per-week")
# plot the grid:
plt.grid()
plt.show()
```

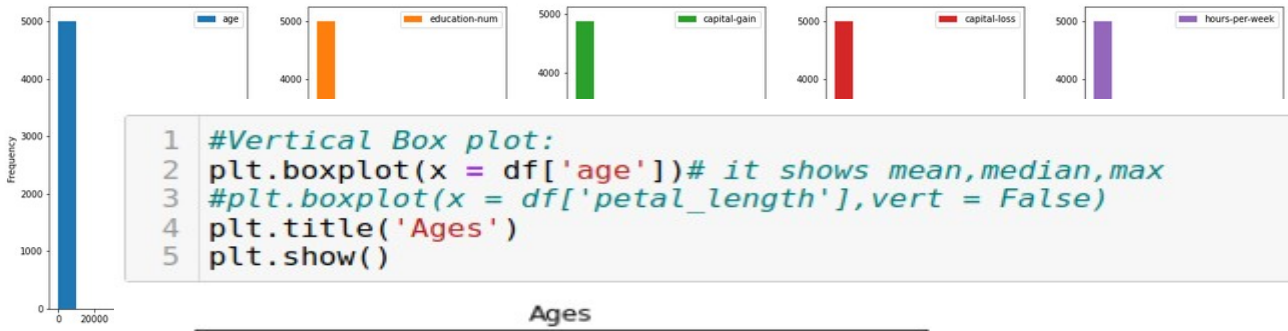
```
# pie chart :
plt.figure(figsize =(2,3))
plt.pie( x=df['age'],labels=df['hours-per-week'] ,autopct = '%1.2f%%')
plt.show()
```

```
plt.hist(x =['age'],color ='b')
# add title:
plt.title("expensive data")
# add laebels:
plt.xlabel("age")
plt.ylabel("capital-gain ")
# display the plot
plt.show()
```

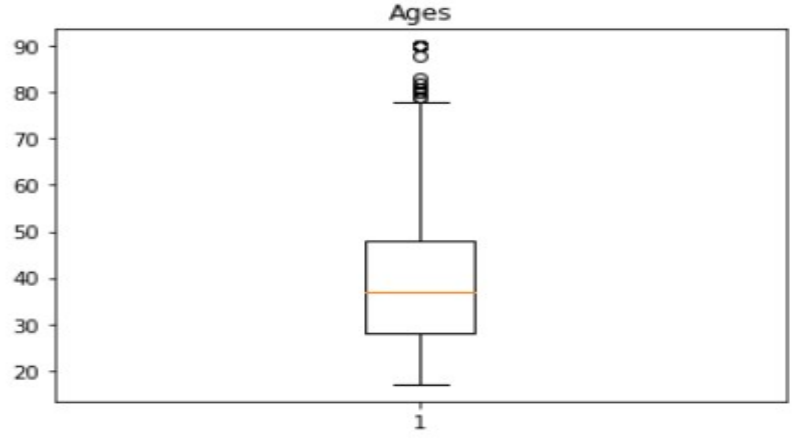



```
# plot multiple histogram for all the numerical column in my data set:
df.plot.hist(subplots = True , layout = (4,5),figsize=(20,20))
# rearrange the plot:
plt.tight_layout()
plt.show()
```


```
# plot multiple histogram for all the numerical column in my data set:
df.plot.hist(subplots = True , layout = (4,5),figsize=(20,20))
# rearrange the plot:
plt.tight_layout()
plt.show()
```


```
#Vertical Box plot:
plt.boxplot(x = df['age'])# it shows mean,median,max
#plt.boxplot(x = df['petal_length'],vert = False)
plt.title('Ages')
plt.show()
```



```python
import scipy
from scipy.stats import variation
# mean:
# trimmed mean
scipy.stats.trim_mean(df['age'] ,proportiontocut = 0.20)
```

```
37.443666666666665
```

```python
# median:
df['age'].median()
```

```
37.0
```

```python
# the first quartile:
Q1=df['age'].quantile(0.5)
Q1
```

```
37.0
```

```python
# IQR:
v1=df['age'].quantile(0.15)
v2=df['age'].quantile(0.65)
IQR = v2-v1
IQR
```

```
19.0
```

```python
# counting the values
df['hours-per-week'].value_counts()
```

```
40    2328
50     457
45     277
60     246
35     202
      ...
57       1
7        1
31       1
51       1
63       1
Name: hours-per-week, Length: 74, dtype: int64
```

```
## (N-1) Dummy Encoding
# Drop_first = 'we are just fropping row to get Dummy variable'
pd.get_dummies(df,columns = ['capital-gain'],drop_first = True)
```

ıtionship	race	sex	capital-loss	...	capital-gain_14084	capital-gain_14344	capital-gain_15020	capital-gain_15024	capital-gain_20051	capital-gain_22040	capital-gain_25124	capital-gain_27828	capital-gain_34095	capital-gain_99999
Husband	White	Male	0	...	0	0	0	1	0	0	0	0	0	0
)wn-child	White	Male	0	...	0	0	0	0	0	0	0	0	0	0
Husband	White	Male	1902	...	0	0	0	0	0	0	0	0	0	0
Wife	White	Female	0	...	0	0	0	0	0	0	0	0	0	0
)wn-child	White	Female	0	...	0	0	0	0	0	0	0	0	0	0

```
# Label Encoding
# The label Encoding consider a level in a Categorical variable by 'Alphabetical Order'
from sklearn.preprocessing import LabelEncoder
# Create an Instance
labelencoder = LabelEncoder()
#Fit the Encoder
df['Encoded_performance_of_a_worker'] = labelencoder.fit_transform(df['age'])
# Display the data
df
```

rkclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense	Encoded_performance_of_a_worker
elf-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	15024	0	50	United-States	>50K	22
Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	40	United-States	<=50K	3
Private	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	1902	65	United-States	>50K	33
ate-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	40	United-States	>50K	21

```
82      1
83      1
Name: age, Length: 69, dtype: int64
```

```
14    151
8     147
18    145
13    135
3     133
      ...
61      3
62      2
67      1
66      1
65      1
Name: Encoded_performance_of_a_worker, Length: 69, dtype: int64
```

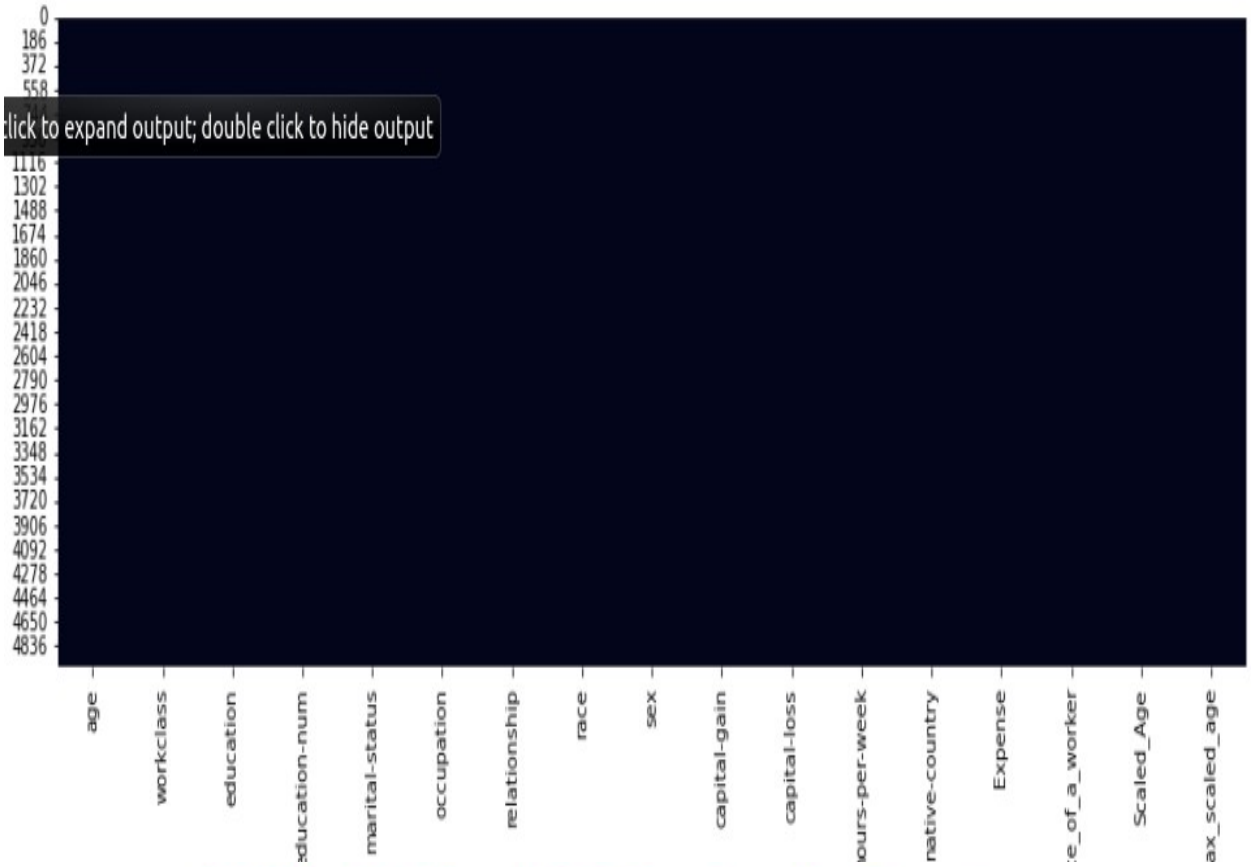
```
## Feature Scaling
# Standardization
# importing StandardScaler for Standardization
from sklearn.preprocessing import StandardScaler
print('Minimum   value Before Transformation :',df.age.min(),'\n'
      'maximium value Before Transformation:',df.age.max())
# Create an instance
standard_scale = StandardScaler()

# Fit the StandardScaler
# Transform the data

df['Scaled_Age'] = standard_scale.fit_transform(df[['age']])

print('Minimum   value After Transformation :',df['Scaled_Age'].min(),'\n',
      'maximium value After Transformation:',df['Scaled_Age'].max())
```

```
inimum   value Before Transformation : 17
aximium value Before Transformation: 90
inimum   value After Transformation : -1.5810851866976907
maximium value After Transformation: 3.7485795080257773
```

```
# Min-Max Normalization
# Importing MinMaxNormalization from sklearn
from sklearn.preprocessing import MinMaxScaler

# Create An Instance

min_max = MinMaxScaler()

# Fit and transform of weight column:
df['min_max_scaled_age'] = min_max.fit_transform(df[['age']])

#minimum and maximum of Normalization of weight:
df['min_max_scaled_age'].min(),df['min_max_scaled_age'].max()
```

```
(0.0, 1.0)
```

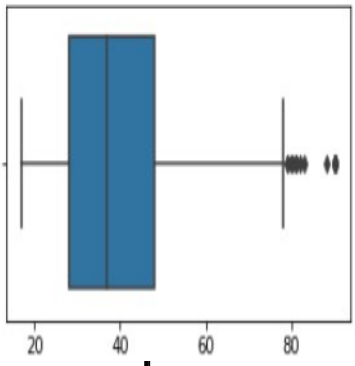
```
#set the figure size:
plt.rcParams['figure.figsize']  = [15,5]
# plot the heat map:
sns.heatmap(df.isnull(),cbar = False)

# display the heatmap:
plt.show()
```

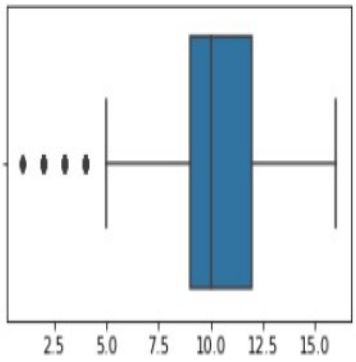


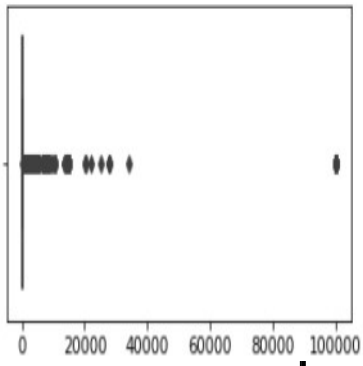

```
#Filling Out Numerical Columns from the Dataset

df_num = df.select_dtypes(include = [np.number])
print(df_num)
# Print heading names of columns contain only numbers
df_num.columns
```

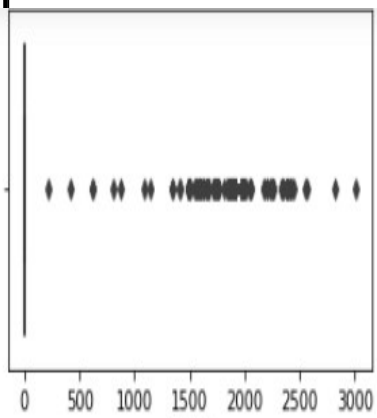
```
      age  education-num  capital-gain  capital-loss  hours-per-week  \
0      39             13         15024             0              50
1      20             10             0             0              40
2      50             16             0          1902              65
3      38              9             0             0              40
4      23             13             0             0              60
...   ...            ...           ...           ...             ...
4995   38              9             0             0              40
4996   26             10             0             0              40
4997   20              7             0             0              60
4998   24              9             0             0              60
4999   40              9             0             0              45

      Encoded_performance_of_a_worker  Scaled_Age  min_max_scaled_age
0                                  22    0.025115            0.301370
1                                   3   -1.362058            0.041096
2                                  33    0.828215            0.452055
```

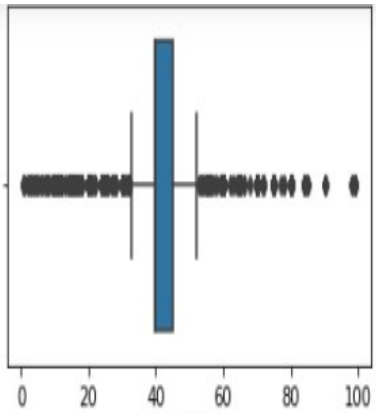
```python
#To identify the Outliers in Numerical Columns:
#Subplots()
fig, ax = plt.subplots(3,3,figsize =(15,9))


for variable,subplot in zip(df_num.columns,ax.flatten()):
    z = sns.boxplot(x = df_num[variable], orient = 'h',whis = 1.5,ax = subplot)

    z.set_xlabel(variable,fontsize = 5)
```

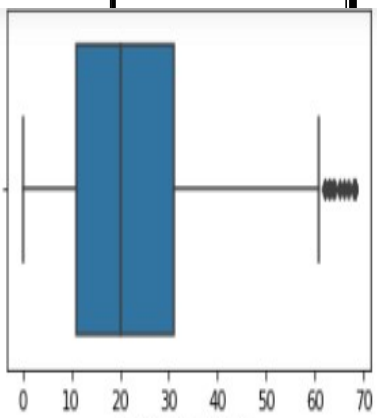


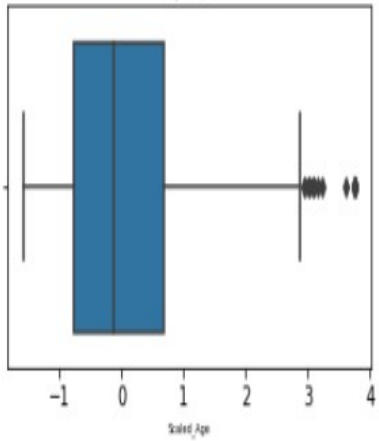


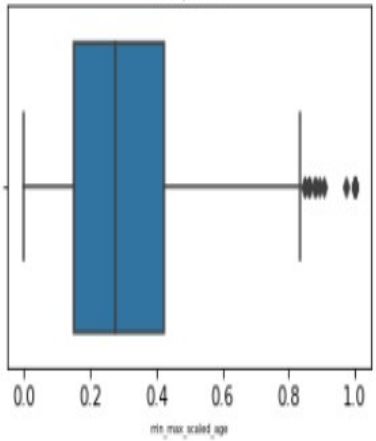


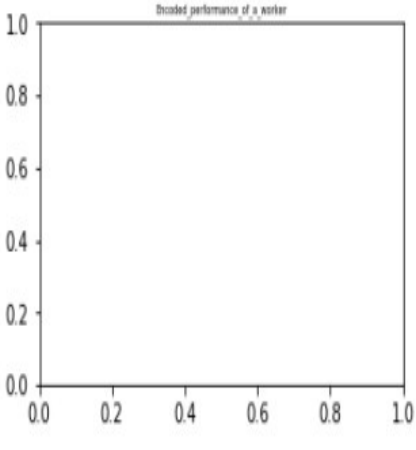












```python
# 1. based on IQR method:
Q1 = df_num.quantile(0.25)
Q3 = df_num.quantile(0.75)

# obtain the type;
IQR =Q3-Q1
print(IQR)
print('\n')
##removed the outliers using the formula and
#storing the cleaned data in df_cleaned variable
df_iqr = df[~((df<(Q1 - 1.5* IQR))|(df>(Q3+1.5*IQR))).any(axis = 1)]
print(df_iqr.shape)
print('\n')
print(df.shape)
```

```
age                                19.000000
education-num                       3.000000
capital-gain                        0.000000
capital-loss                        0.000000
hours-per-week                      1.000000
Encoded_performance_of_a_worker    19.000000
Scaled_Age                          1.387173
min_max_scaled_age                  0.260274
dtype: float64
```

```
df_iqr
```

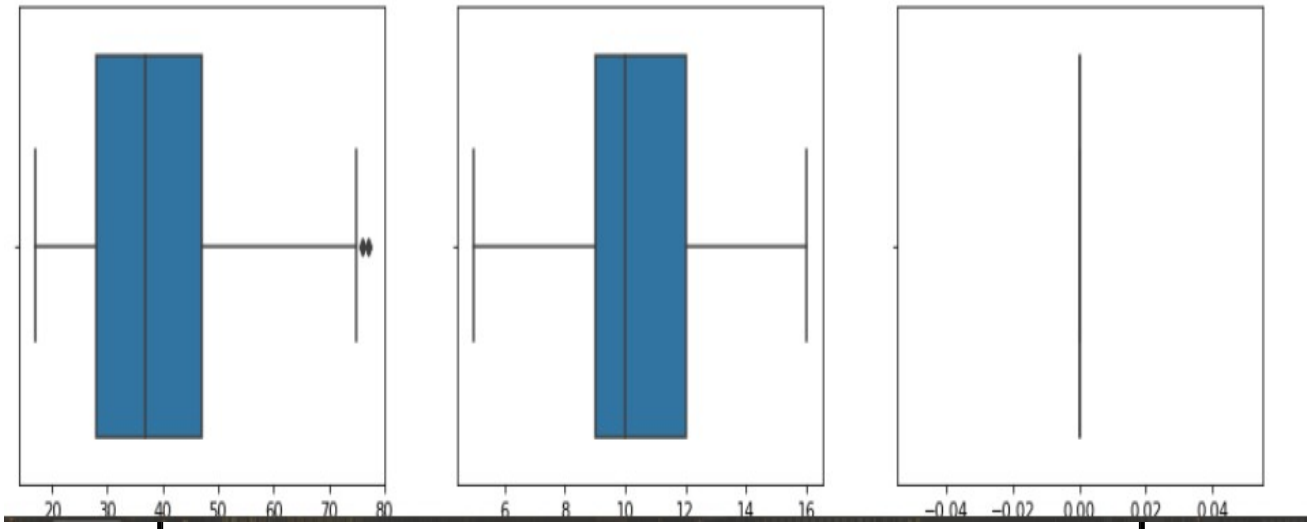
	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense	Encoded_performa
1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	40	United-States	<=50K	
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	40	United-States	>50K	
6	58	Private	Bachelors	13	Married-civ-spouse	Adm-clerical	Husband	White	Male	0	0	40	United-States	<=50K	
7	66	Private	HS-grad	9	Separated	Machine-op-inspct	Not-in-family	Black	Male	0	0	40	United-States	<=50K	
8	39	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States	>50K	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
993	28	Private	HS-grad	9	Never-married	Handlers-cleaners	Unmarried	White	Male	0	0	40	United-States	<=50K	
						Other-							United-		

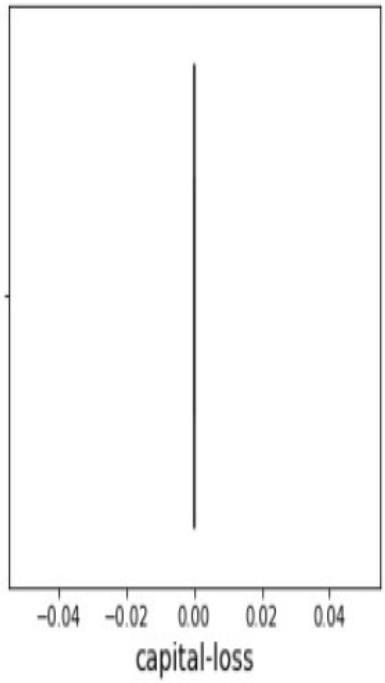
```
#lets divide the numerical columns into another dataset
df_num = df_iqr.select_dtypes(include = [np.number])
df_num
```

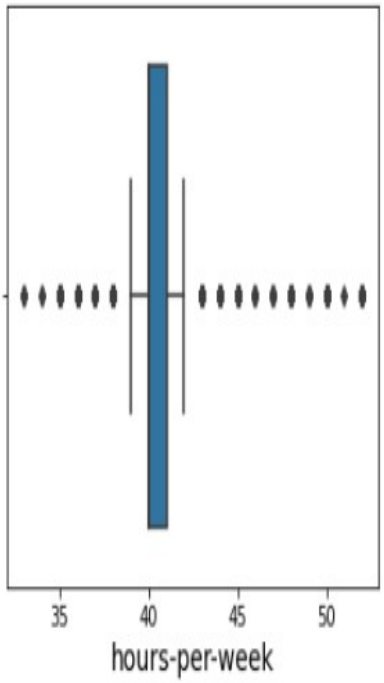
	age	education-num	capital-gain	capital-loss	hours-per-week	Encoded_performance_of_a_worker	Scaled_Age	min_max_scaled_age
1	20	10	0	0	40	3	-1.362058	0.041096
3	38	9	0	0	40	21	-0.047894	0.287671
6	58	13	0	0	40	41	1.412288	0.561644
7	66	9	0	0	40	49	1.996361	0.671233
8	39	9	0	0	40	22	0.025115	0.301370
...	...	...	...	...	...	...	...	...
4993	28	9	0	0	40	11	-0.777985	0.150685
4994	35	9	0	0	35	18	-0.266921	0.246575
4995	38	9	0	0	40	21	-0.047894	0.287671
4996	26	10	0	0	40	9	-0.924003	0.123288
4999	40	9	0	0	45	23	0.098124	0.315068

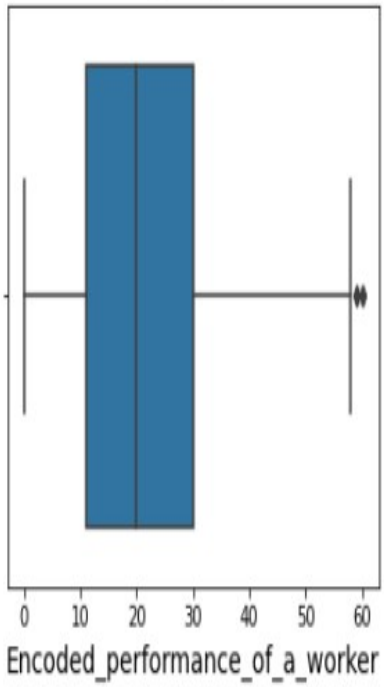
3032 rows x 8 columns

```
#Now lets find the Outliers for the Numerical columns
#for this we need to use subplots() function
import matplotlib.pyplot as plt
import seaborn as sns
#import matplotlib and seaborn as plt and sns respectively
fig,ax = plt.subplots(2,3,figsize = (15,9))
for variable,subplot in zip(df_num.columns,ax.flatten()):
    z = sns.boxplot(x = df_num[variable],orient = 'h',whis = 1.5,ax = subplot)
    z.set_xlabel(variable,fontsize = 14)
```




−0.04
−0.02
0.00
0.02
0.04
capital-loss


35
40
45
50
hours-per-week


0
10
20
30
40
50
60
Encoded_performance_of_a_worker

```
Q1 = df_num.quantile(0.25)
Q3 = df_num.quantile(0.75)
#we know that to find IQR
#The formula is
IQR = Q3 - Q1
IQR
#We got the IQR for respective columns
```

```
age                                19.000000
education-num                       3.000000
capital-gain                        0.000000
capital-loss                        0.000000
hours-per-week                      1.000000
Encoded_performance_of_a_worker    19.000000
Scaled_Age                          1.387173
min_max_scaled_age                  0.260274
dtype: float64
```

```
df_ultra_cleaned = df_iqr[~((df_iqr<(Q1 - 1.5* IQR))|(df_iqr>(Q3+1.5*IQR))).any(axis = 1)]
```

```
df_ultra_cleaned
```

```
df_ultra_cleaned
```

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex
1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female
6	58	Private	Bachelors	13	Married-civ-spouse	Adm-clerical	Husband	White	Male
7	66	Private	HS-grad	9	Separated	Machine-op-inspct	Not-in-family	Black	Male
8	39	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male
...	...	...	...	...	...	...	...	...	...
4990	32	Private	HS-grad	9	Married-civ-	Craft-repair	Husband	White	Male

```
df_ultra_cleaned.shape
```

```
(2000, 17)
```

```
#lets divide the numerical columns into another dataset
df_num = df_ultra_cleaned.select_dtypes(include = [np.number])
df_num
```

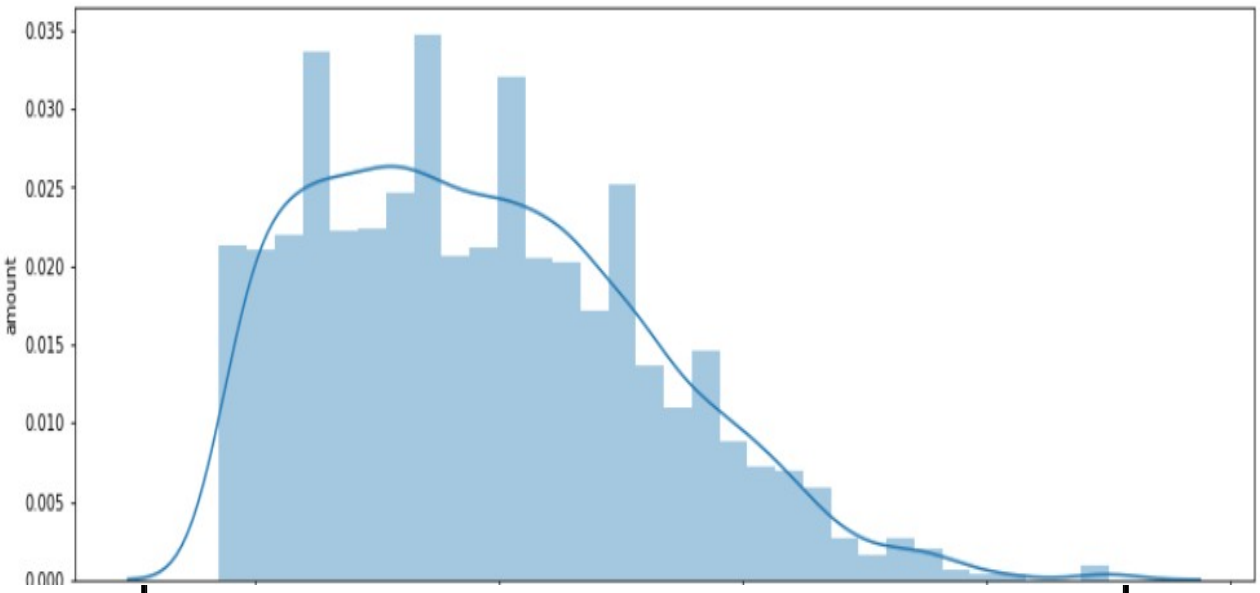
	age	education-num	capital-gain	capital-loss	hours-per-week	Encoded_performance_of_a_worker	Scaled_Age	min_max_scaled_age
1	20	10	0	0	40	3	-1.362058	0.041096
3	38	9	0	0	40	21	-0.047894	0.287671
6	58	13	0	0	40	41	1.412288	0.561644
7	66	9	0	0	40	49	1.996361	0.671233
8	39	9	0	0	40	22	0.025115	0.301370
...	...	...	...	...	...	...	...	...
4990	32	9	0	0	40	15	-0.485949	0.205479

```
# Filling Out only Categorical column from your Dataset
from warnings import filterwarnings
filterwarnings('ignore')
df_cat = df.select_dtypes(include = [np.object])
print(df_cat)
# Print heading names of columns contain only numbers
df_num.columns
```

```
          workclass      education       marital-status
0      Self-emp-inc      Bachelors   Married-civ-spouse
1           Private   Some-college        Never-married
2           Private      Doctorate   Married-civ-spouse
3         State-gov        HS-grad   Married-civ-spouse
4         Local-gov      Bachelors        Never-married
...             ...            ...                  ...
4995        Private        HS-grad   Married-civ-spouse   M
4996        Private   Some-college        Never-married
4997        Private           11th        Never-married
4998        Private        HS-grad   Married-civ-spouse
4999        Private        HS-grad             Divorced
```

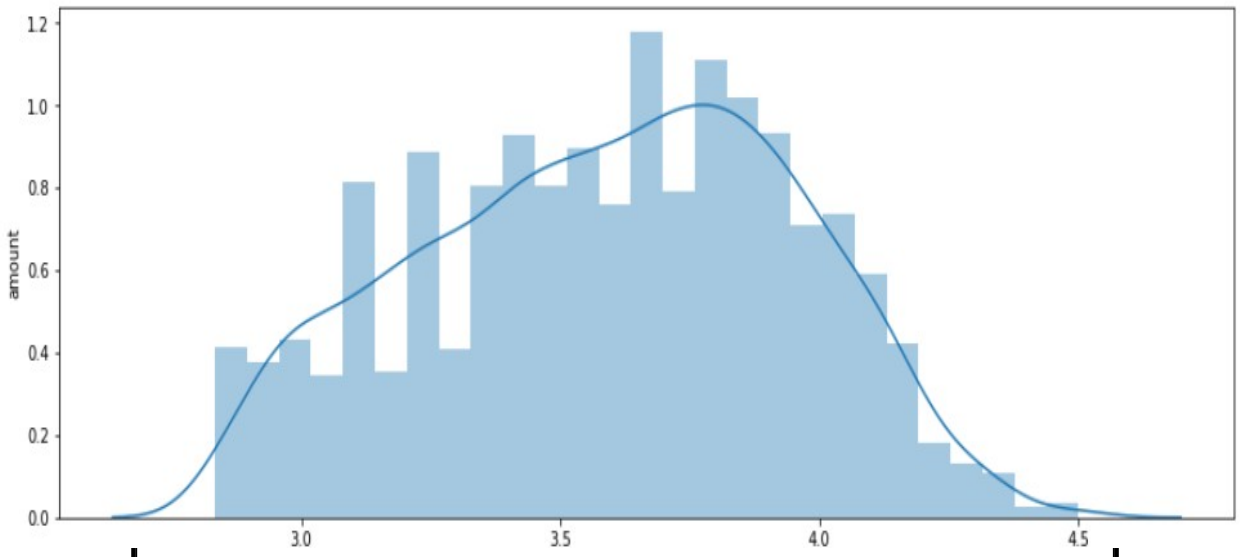
```
# Distribution of the Age column
sns.distplot(df['age'])
plt.ylabel('amount')
print('Skewness : ',df['age'].skew())
plt.show()
```

```
Skewness :  0.5639005150156318
```

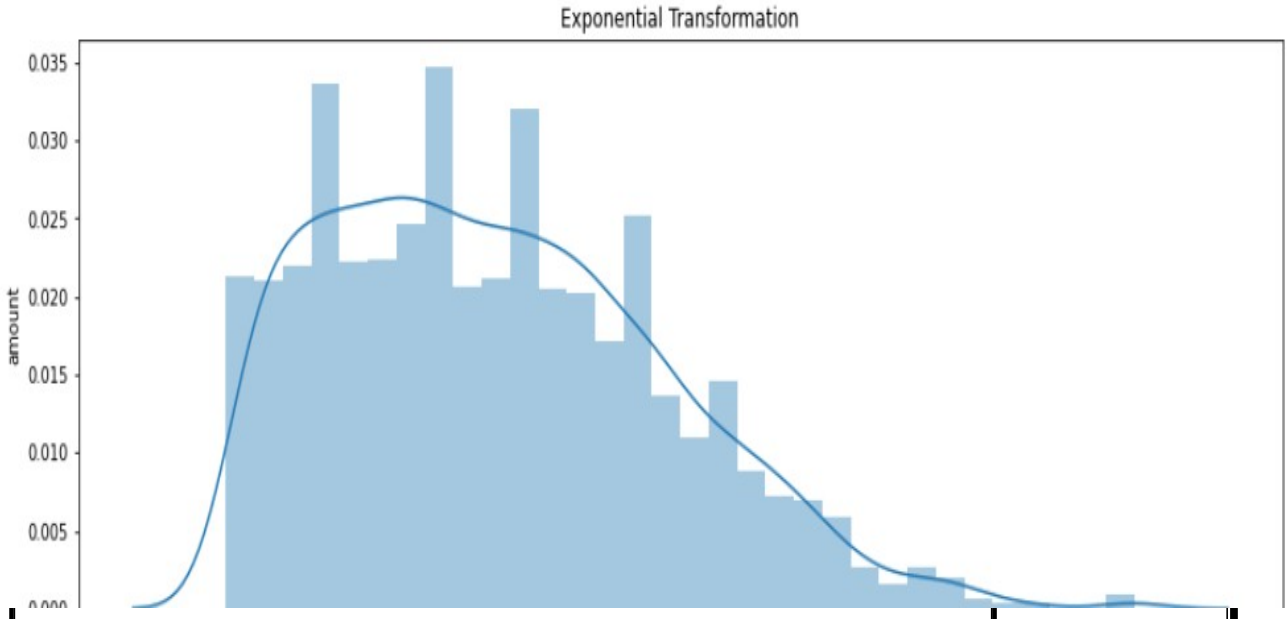


```
plt.ylabel('amount')
plt.show()
```

```
Skewness after log Transformtion :  -0.12574343649471817
```




```
age = np.exp(log_age)
# plot the Distribution
sns.distplot(age)
plt.ylabel('amount')
plt.title('Exponential Transformation')
plt.show()
```



Project 3:

```
# 1.Write a program that reads a text file and counts the number of words in it.
#The program should then display the total count of words.
file = open("File.txt","w")
print("Name of the file: ",file.name)
file.write("Hello All,hope you are enjoying learning Python")
```

```
Name of the file:  File.txt

47
```

```
file = open("File.txt","rt")
data = file.read()
words = data.split()

print('Number of words in text file :', len(words))
```

```
Number of words in text file : 7
```

```
# 2.Write a program that copies the contents of one text file to another.
#The program should take the names of both files as input from the user.
file = open("File.txt","rb")
file1 = open("File.txt","r")
print(file1.read())

```

```
Hello All,hope you are enjoying learning Python
```

3.Write a program that encrypts a text file. The program should read the contents

of the file, apply a simple encryption algorithm (such as shifting the ASCII values

of each character by a certain number), and write the encrypted

```
source_file_name = input("Enter the name of the source file: ")
destination_file_name = input("Enter the name of the destination file: ")
shift_value = int(input("Enter the shifting value: "))
source_file = open(source_file_name, 'r')
destination_file = open(destination_file_name, 'w')
contents = source_file.read()
encrypted_contents = ''
for char in contents:
    if char.isalpha():
        ascii_value = ord(char)
        shifted_value = (ascii_value - ord('a') + shift_value) % 26 + ord('a')
        encrypted_contents += chr(shifted_value)
    else:
        encrypted_contents += char
destination_file.write(encrypted_contents)
source_file.close()
destination_file.close()
print("File encrypted successfully!")
```

```
nter the name of the source file: File.txt
nter the name of the destination file: File1.txt
nter the shifting value: 2
ile encrypted successfully!
```

text to a new file.

4.Write a program that searches for a specific word in a text file. The program should prompt the user to enter the word to search for, read the file, and display all the lines that contain the specified word.

```python
def search_word_in_file(filename, word):
    with open(filename, 'r') as file:
        lines_with_word = []
        for line in file:
            if word.lower() in line.lower():
                lines_with_word.append(line.rstrip('\n'))

        if lines_with_word:
            print(f"Lines containing the word '{word}':")
            for line in lines_with_word:
                print(line)
        else:
            print(f"No lines found containing the word '{word}'.")
# Example usage:
file_path = input("Enter the name of the text file: ")
search_word = input("Enter the word to search for: ")
search_word_in_file(file_path, search_word)
```

```
Enter the name of the text file: File.txt
Enter the word to search for: Hello
Lines containing the word 'Hello':
Hello All,hope you are enjoying learning Pythonexam
```

5.Write a program that reads a text file and calculates various statistics, such as the number of lines, the number of words, and the average word length.

```python
def calculate_statistics(filename):
    file =open(filename,"r")
    lines = file.readlines()
    line_count = len(lines)
    word_count = 0
    total_word_length = 0
    for line in lines:
        words = line.split()
        word_count += len(words)
        total_word_length += sum(len(word) for word in words)
        if(word_count) > 0:
            average_word_length = total_word_length / word_count
        else:
            print("0")
    print(f"Number of lines: {line_count}")
    print(f"Number of words: {word_count}")
    print(f"Average word length: {average_word_length:.2f}")
filename = 'File.txt'
calculate_statistics(filename)
```

```
Number of lines: 1
Number of words: 7
Average word length: 6.43
```

6.Write a program that appends a new line of text to an existing text file. The program should prompt the user to enter the text to be appended and then add it as a new line at the end of the file.

```
file = open("File.txt","rb")
file1 = open("File.txt","r")
print(file1.read())
```

```
Hello All,hope you are enjoying learning Pythonexam
```

```
def append_to_file(file_path):
    text = input("Enter the text to append: ")

    with open(file_path, 'a') as file:
        file.write(text + '\n')
    print("Text appended successfully!")
file_path = 'File.txt'
append_to_file(file_path)
```

```
Enter the text to append: exam
Text appended successfully!
```

```
file = open("File.txt","rb")
file1 = open("File.txt","r")
print(file1.read())
```

```
Hello All,hope you are enjoying learning Pythonexam
```

Project:4:

Online Food Ordering System:

Create a program that simulates an online food ordering system. Users can browse restaurants, view menus, and place orders for delivery or pickup

introduction:

- An online food ordering system allows your business to accept and manage orders placed online for delivery or takeaway.

- Customers browse a digital menu, either on an app or website and place and pay for their order online.
- Our online ordering system will help you transform your website into a money-making machine.
- No matter how much your business grows, you will always be able to take free unlimited orders with zero costs.
- Power your business with our free restaurant online ordering system & you'll never have to worry about fees or commissions.

Program:

```
class Restaurant:
    def __init__(self, name, cuisine):
        self.name = name
        self.cuisine = cuisine
        self.menu = []

    def add_item_to_menu(self, item):
        self.menu.append(item)


class MenuItem:
    menu = {
        "pizza": 3.00,
        "nachos": 4.50,
        "popcorn": 6.00,
        "fries": 2.50,
        "chips": 1.00,
        "pretzel": 5.00,
        "soda": 3.00,
```

```
        "lemonade": 4.25
    }

    def display_menu(self):
        if not self.menu:
            print("Menu is empty.")
            return
        print("---------MENU----------")
        for key, value in self.menu.items():
            print(f"{key:10}:${value:.2f}")
        print("-------------------------")

    def get_user_order(self):
        cart = []
        while True:
            food = input("Select an item (q to quit): ").lower()
            if food == "q":
                break
            elif self.menu.get(food) is not None:
                cart.append(food)
        return cart
class Order:
    def __init__(self, restaurant, items, delivery, address):
        self.restaurant = restaurant
        self.items = items
        self.delivery = delivery
        self.address = address
```

```
    def calculate_total(self):
        total = 0
        for food in self.items:
            total += MenuItem.menu.get(food)
        return total

class OnlineFoodOrderingSystem:
    def __init__(self):
        self.restaurants = []

    def add_restaurant(self, restaurant):
        self.restaurants.append(restaurant)

    def browse_restaurants(self):
        print("Available Restaurants:")
        for i, restaurant in enumerate(self.restaurants):
            print(f"{i+1}. {restaurant.name}")

        selection = int(input("Select a restaurant: "))
        if 1 <= selection <= len(self.restaurants):
            return self.restaurants[selection - 1]
        else:
            print("Invalid selection.")
            return None

    def place_order(self, restaurant, items, delivery):
```

```
        if delivery:
            address = input("Enter delivery address: ")
        else:
            address = ""

        order = Order(restaurant, items, delivery, address)
        total = order.calculate_total()
        print("Order placed successfully!")
        print(f"Restaurant: {restaurant.name}")
        print("Items:")
        for item in items:
            print(f"- {item}: ${MenuItem.menu.get(item):.2f}")
        print(f"Delivery: {'Yes' if delivery else 'No'}")
        if delivery:
            print(f"Address: {address}")
        print(f"Total: ${total:.2f}")

# Creating restaurants and adding them to the system
system = OnlineFoodOrderingSystem()

restaurant1 = Restaurant("Restaurant A", "Italian")
restaurant1.add_item_to_menu(MenuItem())
system.add_restaurant(restaurant1)

restaurant2 = Restaurant("Restaurant B", "Chinese")
restaurant2.add_item_to_menu(MenuItem())
system.add_restaurant(restaurant2)
```

```
# Simulating user interaction
selected_restaurant = system.browse_restaurants()
if selected_restaurant is not None:
    menu_item = MenuItem()
    menu_item.display_menu()  # Display menu only once
    print()
    selected_items = menu_item.get_user_order()
    delivery = True if input("Do you want delivery? (yes/no): ").lower() == "yes" else False
    system.place_order(selected_restaurant, selected_items, delivery)
```

```
Available Restaurants:
1. Restaurant A
2. Restaurant B
Select a restaurant: 1
---------MENU----------
pizza     :$3.00
nachos    :$4.50
popcorn   :$6.00
fries     :$2.50
chips     :$1.00
pretzel   :$5.00
soda      :$3.00
lemonade  :$4.25
-----------------------

Select an item (q to quit): pizaa
Select an item (q to quit): fries
Select an item (q to quit): popcorn
Select an item (q to quit): q
Do you want delivery? (yes/no): yes
Enter delivery address: 10/375,nagulabavi street
Order placed successfully!
Restaurant: Restaurant A


Items:
- fries: $2.50
- popcorn: $6.00
Delivery: Yes
Address: 10/375,nagulabavi street
Total: $8.50
```

Assignment-01

Name : Online Shopping Cart

Description:

Design a simple online shopping cart system using object-oriented programming concepts. Create the following classes:

Product:

Attributes: name, price, quantity
Methods:
Constructor to initialize the attributes
Getter and setter methods for each attribute
ShoppingCart:

Attributes: products (a list of Product objects)
Methods:
Constructor to initialize the products list
Method to add a product to the cart
Method to remove a product from the cart
Method to calculate the total price of all products in the cart
Method to display the contents of the cart
Main:

Create an instance of the ShoppingCart class
Add a few products to the cart
Display the contents of the cart
Calculate and display the total price of the products in the cart

```python
class Product:
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

    def get_name(self):
        return self.name

    def set_name(self, name):
        self.name = name

    def get_price(self):
        return self.price

    def set_price(self, price):
        self.price = price

    def get_quantity(self):
        return self.quantity

    def set_quantity(self, quantity):
        self.quantity = quantity


class ShoppingCart:
    def __init__(self):
        self.products = []

    def add_product(self, product):
        self.products.append(product)

    def remove_product(self, product):
        self.products.remove(product)
```

```
    def calculate_total_price(self):
        total_price = 0
        for product in self.products:
            total_price += product.get_price() * product.get_quantity()
        return total_price

    def display_cart(self):
        for product in self.products:
            print(f"Product: {product.get_name()}, Price: {product.get_price()}, Quantity: {product.get_quantity()}")

shopping_cart = ShoppingCart()

# Adding products to the cart
product1 = Product("Item 1", 30, 7)
product2 = Product("Item 2", 38, 5)

shopping_cart.add_product(product1)
shopping_cart.add_product(product2)

# Displaying the contents of the cart
shopping_cart.display_cart()

# Calculating and displaying the total price
total_price = shopping_cart.calculate_total_price()
print("Total Price:", total_price)
```

```
Product: Item 1, Price: 30, Quantity: 7
Product: Item 2, Price: 38, Quantity: 5
Total Price: 400
```

Assignment-2

```
Name : Social Media Platform

Description:

Design and develop a social media platform using object-oriented programming concepts.

Create a class called "User" with the following attributes and methods:

Attributes: userId (integer), name (string), email (string), friends (list of User objects)
Methods: getUserId(), getName(), getEmail(), addFriend(user), removeFriend(user), viewFriends()
Create a class called "Post" with the following attributes and methods:

Attributes: postId (integer), user (User object), content (string), likes (integer)
Methods: getPostId(), getUser(), getContent(), getLikes(), like(), unlike()
Create a class called "SocialMediaPlatform" with the following attributes and methods:

Attributes: users (list of User objects), posts (list of Post objects)
Methods: addUser(user), removeUser(user), createPost(user, content), removePost(post), displayPosts()
```

```python
class User:
    def __init__(self, userId, name, email):
        self.userId = userId
        self.name = name
        self.email = email
        self.friends = []

    def getUserId(self):
        return self.userId

    def getName(self):
        return self.name

    def getEmail(self):
        return self.email

    def addFriend(self, user):
        self.friends.append(user)

    def removeFriend(self, user):
        self.friends.remove(user)

    def viewFriends(self):
        for friend in self.friends:
            print(f"Friend: {friend.getName()}, Email: {friend.getEmail()}")


class Post:
    def __init__(self, postId, user, content):
        self.postId = postId
        self.user = user
        self.content = content
        self.likes = 0
```

```
    def getPostId(self):
        return self.postId

    def getUser(self):
        return self.user

    def getContent(self):
        return self.content

    def getLikes(self):
        return self.likes

    def like(self):
        self.likes += 1

    def unlike(self):
        if self.likes > 0:
            self.likes -= 1

ass SocialMediaPlatform:
    def __init__(self):
        self.users = []
        self.posts = []

    def addUser(self, user):
        self.users.append(user)

    def removeUser(self, user):
        self.users.remove(user)

    def createPost(self, user, content):
        postId = len(self.posts) + 1
        post = Post(postId, user, content)
        self.posts.append(post)
```

```
    def removePost(self, post):
        self.posts.remove(post)

    def displayPosts(self):
        for post in self.posts:
            print(f"Post ID: {post.getPostId()}")
            print(f"User: {post.getUser().getName()}, Email: {post.getUser().getEmail()}")
            print(f"Content: {post.getContent()}")
            print(f"Likes: {post.getLikes()}\n")
social_media = SocialMediaPlatform()

# Create users
user1 = User(1, "Nandhu", "Nandhu@example.com")
user2 = User(2, "Gowthu", "Gowthu@example.com")
user3 = User(3, "poojitha(ammu)", "pooji@example.com")

# Add users to the social media platform
social_media.addUser(user1)
social_media.addUser(user2)
social_media.addUser(user3)

# Create posts
social_media.createPost(user1, "Hello")
social_media.createPost(user2, "How are you!")
social_media.createPost(user3, "happy days from today on words!")

# Add friends
user1.addFriend(user2)
user1.addFriend(user3)
user2.addFriend(user3)

# Display posts
social_media.displayPosts()
```

```python
# Like posts
post1 = social_media.posts[0]
post1.like()
post1.like()

post2 = social_media.posts[1]
post2.like()

# Display posts with updated likes
social_media.displayPosts()

# Remove user and post
social_media.removeUser(user2)
social_media.removePost(post1)

# Display posts and friends after removal
social_media.displayPosts()
user1.viewFriends()
```

```
Post ID: 1
User: Nandhu, Email: Nandhu@example.com
Content: Hello
Likes: 0

Post ID: 2
User: Gowthu, Email: Gowthu@example.com
Content: How are you!
Likes: 0

Post ID: 3
User: poojitha(ammu), Email: pooji@example.com
Content: happy days from today on words!
Likes: 0
```

```
Post ID: 3
User: poojitha(ammu), Email: pooji@example.com
Content: happy days from today on words!
Likes: 0

Post ID: 2
User: Gowthu, Email: Gowthu@example.com
Content: How are you!
Likes: 1

Post ID: 3
User: poojitha(ammu), Email: pooji@example.com
Content: happy days from today on words!
Likes: 0

Friend: Gowthu, Email: Gowthu@example.com
Friend: poojitha(ammu), Email: pooji@example.com
```

Assignment-3:

Topics:

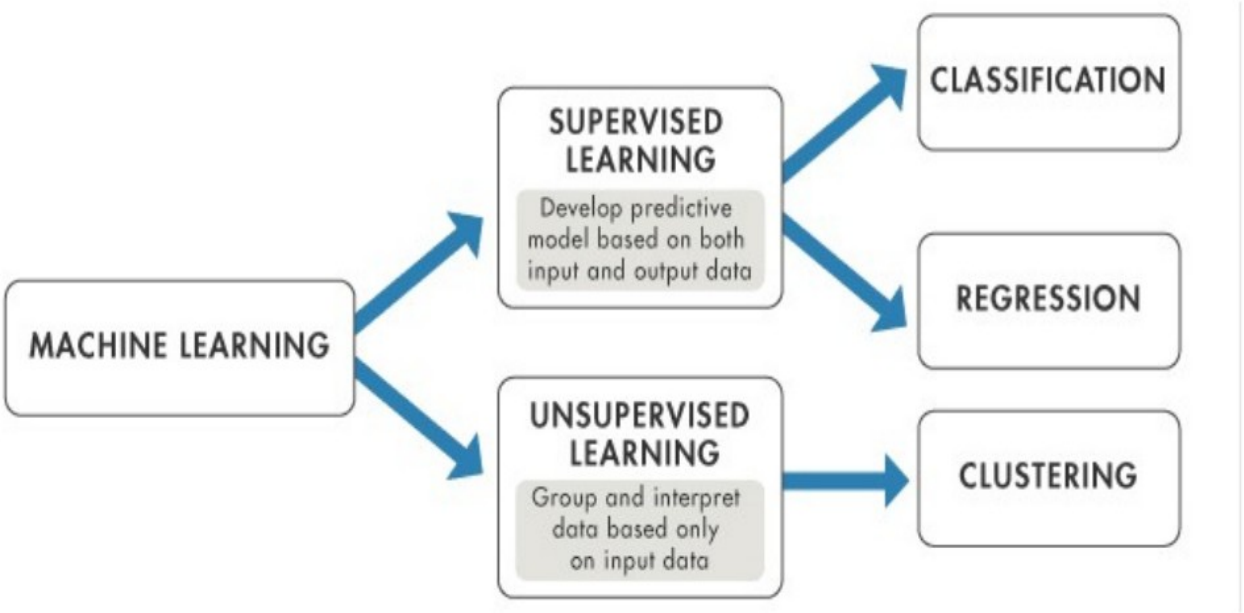
• Supervised Machine Learning

• Unsupervised Machine Learning

• Limitations of Supervised and Unsupervised Machine Learning

• Types of Supervised Machine Learning and Uses.

MACHINE LEARNING



Supervised Machine Learning

Supervised machine learning learns patterns and relationships between input and

output data. It contains a model that is able to predict with the help of a labeled

dataset. A labeled dataset is one where you already know the target answer.

It is defined by its use of labeled data. A labeled data is a dataset that contains a lot

of examples of Features and Target. Supervised learning uses algorithms that learn

the relationship of Features and Target from the dataset. This process is referred to

as Training or Fitting.

Example:suppose you are given a basket filled with different kinds of fruits. Now the

first step is to train the machine with all the different fruits one by one like this:

• If the shape of the object is rounded and has a depression at the top, is red in

color, then it will be labeled as –Apple.

• If the shape of the object is a long curving cylinder having Green-Yellow color,

then it will be labeled as –Banana.Unsupervised Machine Learning:

• It requirs unstructure data

• No need of target and explict algorithm

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that nformation without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.

Examples:suppose it is given an image having both dogs and cats which it has never

seen. the machine has no idea about the features of dogs and cats so we can't

categorize it as 'dogs and cats '. But it can categorize them according to their

similarities, patterns, and differences, i.e., we can easily categorize the picture into

two parts. The first may contain all pics having dogs in them and the second part may

contain all pics having cats in them.

Limitations of Supervised and Unsupervised Machine Learning

In supervised learning, the algorithm “learns” from the training dataset by iteratively

making predictions on the data and adjusting for the correct answer. While supervised

learning models tend to be more accurate than unsupervised learning models, they require upfront human intervention to label the data appropriately. For example, a supervised learning model can predict how long your commute will be based on the time of day, weather conditions and so on. But first, you’ll have to train it to know that rainy weather extends the driving time.

Unsupervised learning models, in contrast, work on their own to discover the inherent

structure of unlabeled data. Note that they still require some human intervention for

validating output variables. For example, an unsupervised learning model can identify that online shoppers often purchase groups of products at the same time. However, a data analyst would need to validate that it makes sense for a recommendation

engine to group baby clothes with an order of diapers, applesauce and sippy cups.

Types of Supervised Machine Learning and Uses.

Supervised learning can be further divided into two types:

• Classification

• Regression

Classification : Classification is a type where algorithms learn from the data to predict an outcome or event in the future.Classification algorithms are used for predicting discrete outcomes, if the outcome can take two possible values such as True or False, Default or NoDefault, Yes or No, it is known as Binary Classification. When the outcome contains more than two possible values, it is known as Multiclass Classification. There are many machine learning algorithms that can be used for classification tasks.

Some of them are:

- Logistic Regression
- Decision Tree Classifier
- K Nearest Neighbor Classifier
- Random Forest Classifier
- Neural Networks

Regression :where algorithms learn from the data to predict continuous values such

as sales, salary, weight, or temperature. There are many machine learning algorithms

that can be used for regression tasks. Some of them are:

- Linear Regression
- Decision Tree Regressor
- K Nearest Neighbor Regressor
- Random Forest Regressor
- Neural Networks

Project:5

9 : Estimating Insurance Claim Amounts : Problem Statement: Estimating Insurance Claim Amounts Project Description: Develop a regression model to estimate the claim amounts for insurance policies based on customer profiles, policy details, and historical claim data. Domain: Insurance Dataset Link:https://www.kaggle.com/competitions/allstate-claims-severity/data?select=train.csv

```
# importing Libraries
# importing Pandas Library as pd
import pandas as pd

# importing Numpy Library as np
import numpy as np

# importing matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# imporing seaborn as sns
import seaborn as sns

```

```
# Loading the dataset using pandas module and assign it as df
df_test = pd.read_csv('test.csv')


# Printing the dataset
df_test
```

	id	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	...	cont5	cont6	cont7	cont8	cont9	cont10
0	4	A	B	A	A	A	A	A	A	B	...	0.281143	0.466591	0.317681	0.61229	0.34365	0.38016
1	6	A	B	A	B	A	A	A	A	B	...	0.836443	0.482425	0.443760	0.71330	0.51890	0.60401
2	9	A	B	A	B	B	A	B	A	B	...	0.718531	0.212308	0.325779	0.29758	0.34365	0.30529
3	12	A	A	A	A	B	A	A	A	A	...	0.397069	0.369930	0.342355	0.40028	0.33237	0.31480
4	15	B	A	A	A	A	B	A	A	A	...	0.302678	0.398862	0.391833	0.23688	0.43731	0.50556
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
125541	587617	A	A	A	B	A	A	A	A	A	...	0.281143	0.438917	0.815941	0.39455	0.48740	0.40666
125542	587621	A	A	A	A	B	B	A	B	A	...	0.674529	0.346948	0.424968	0.47669	0.25753	0.26894
125543	587627	B	B	A	A	B	A	A	A	B	...	0.794794	0.808958	0.511502	0.72299	0.94438	0.83510
125544	587629	A	A	A	A	A	B	A	B	A	...	0.302678	0.372125	0.388545	0.31796	0.32128	0.36974
125545	587634	A	B	A	A	A	A	A	A	B	...	0.413817	0.221699	0.242044	0.25461	0.31399	0.25183

125546 rows × 131 columns

```
# Loading the dataset using pandas module and assign it as df
df_train = pd.read_csv('train.csv')

# Printing the dataset
df_train
```

	id	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	...	cont6	cont7	cont8	cont9	cont10	cont11
0	1	A	B	A	B	A	A	A	A	B	...	0.718367	0.335060	0.30260	0.67135	0.83510	0.569745
1	2	A	B	A	A	A	A	A	A	B	...	0.438917	0.436585	0.60087	0.35127	0.43919	0.338312
2	5	A	B	A	A	B	A	A	A	B	...	0.289648	0.315545	0.27320	0.26076	0.32446	0.381398
3	10	B	B	A	B	A	A	A	A	B	...	0.440945	0.391128	0.31796	0.32128	0.44467	0.327915
4	11	A	B	A	B	A	A	A	A	B	...	0.178193	0.247408	0.24564	0.22089	0.21230	0.204687
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
188313	587620	A	B	A	A	A	A	A	A	B	...	0.242437	0.289949	0.24564	0.30859	0.32935	0.223038
188314	587624	A	A	A	A	A	B	A	A	A	...	0.334270	0.382000	0.63475	0.40455	0.47779	0.307628
188315	587630	A	B	A	A	A	A	A	B	B	...	0.345883	0.370534	0.24564	0.45808	0.47779	0.445614
188316	587632	A	B	A	A	A	A	A	A	B	...	0.704364	0.562866	0.34987	0.44767	0.53881	0.863052

```
df = pd.concat([df_train, df_test], join='outer', axis=0)
print(df)
```

```
            id cat1 cat2 cat3 cat4 cat5 cat6 cat7 cat8 cat9  ...     cont6  \
0            1    A    B    A    B    A    A    A    A    B  ...  0.718367
1            2    A    B    A    A    A    A    A    A    B  ...  0.438917
2            5    A    B    A    A    B    A    A    A    B  ...  0.289648
3           10    B    B    A    B    A    A    A    A    B  ...  0.440945
4           11    A    B    A    B    A    A    A    A    B  ...  0.178193
...        ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...       ...
125541  587617    A    A    A    B    A    A    A    A    A  ...  0.438917
125542  587621    A    A    A    A    B    B    A    B    A  ...  0.346948
125543  587627    B    B    A    A    B    A    A    A    B  ...  0.808958
125544  587629    A    A    A    A    A    B    A    B    A  ...  0.372125
125545  587634    A    B    A    A    A    A    A    A    B  ...  0.221699

          cont7    cont8    cont9   cont10    cont11    cont12    cont13  \
0      0.335060  0.30260  0.67135  0.83510  0.569745  0.594646  0.822493
1      0.436585  0.60087  0.35127  0.43919  0.338312  0.366307  0.611431
2      0.315545  0.27320  0.26076  0.32446  0.381398  0.373424  0.195709
3      0.391128  0.31796  0.32128  0.44467  0.327915  0.321570  0.605077
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 313864 entries, 0 to 125545
Columns: 132 entries, id to loss
dtypes: float64(15), int64(1), object(116)
memory usage: 318.5+ MB
```

```
df.isnull().sum()
```

```
id              0
cat1            0
cat2            0
cat3            0
cat4            0
              ...
cont11          0
cont12          0
cont13          0
cont14          0
loss       125546
Length: 132, dtype: int64
```

```
df.dropna(inplace = True)
```

```
df.isnull().sum()
```

```
id         0
cat1       0
cat2       0
cat3       0
cat4       0
          ..
cont11     0
cont12     0
cont13     0
cont14     0
loss       0
Length: 132, dtype: int64
```

```
print(df.size)
print(df.shape)
```

```
24857976
(188318, 132)
```

```
# Summary stats for Numerical Column:
df.describe()
```

	id	cont1	cont2	cont3	cont4	cont5	cont6	cont7
count	188318.000000	188318.000000	188318.000000	188318.000000	188318.000000	188318.000000	188318.000000	188318.000000
mean	294135.982561	0.493861	0.507188	0.498918	0.491812	0.487428	0.490945	0.484970
std	169336.084867	0.187640	0.207202	0.202105	0.211292	0.209027	0.205273	0.178450
min	1.000000	0.000016	0.001149	0.002634	0.176921	0.281143	0.012683	0.069503
25%	147748.250000	0.346090	0.358319	0.336963	0.327354	0.281143	0.336105	0.350175
50%	294539.500000	0.475784	0.555782	0.527991	0.452887	0.422268	0.440945	0.438285
75%	440680.500000	0.623912	0.681761	0.634224	0.652072	0.643315	0.655021	0.591045
max	587633.000000	0.984975	0.862654	0.944251	0.954297	0.983674	0.997162	1.000000

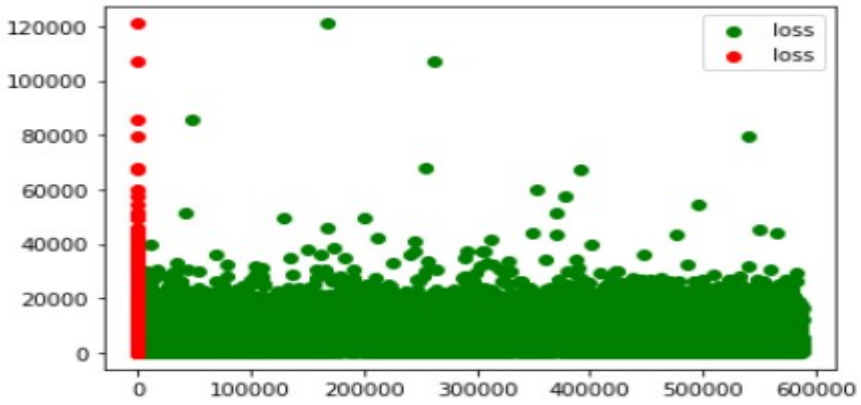
```
# Summary stats for Categorical Column:
from warnings import filterwarnings
filterwarnings('ignore')
df.describe(include = [np.object])
```

	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	cat10	...	cat107	cat108	cat109
count	188318	188318	188318	188318	188318	188318	188318	188318	188318	188318	...	188318	188318	188318
unique	2	2	2	2	2	2	2	2	2	2	...	20	11	84
top	A	A	A	A	A	A	A	A	A	A	...	F	B	BI
freq	141550	106721	177993	128395	123737	131693	183744	177274	113122	160213	...	47310	65512	152918

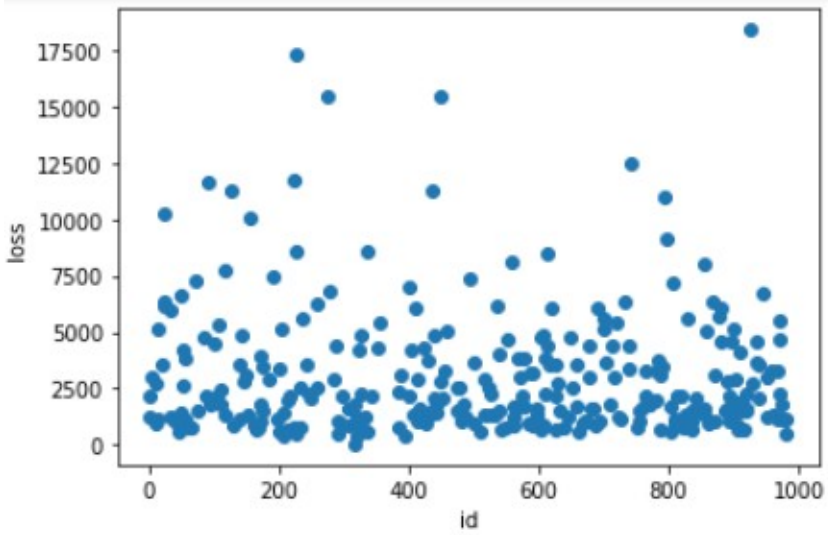
rows × 116 columns

Data Visuvalization

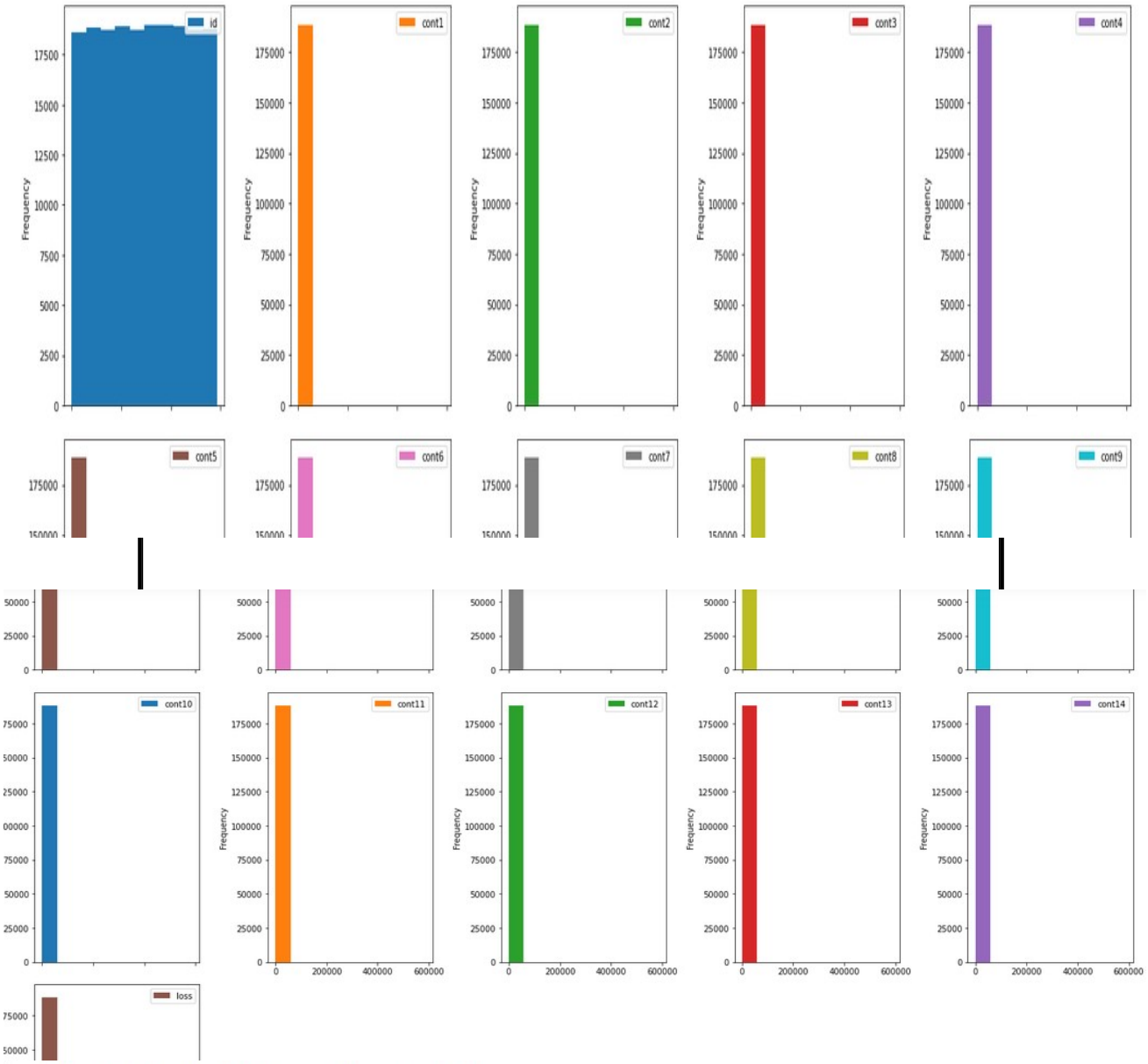
```
plt.scatter(x = 'id',y = 'loss',data = df,color = 'g')
plt.scatter(x = 'cont1',y = 'loss',data = df,color = 'r')

#Describe all the elements of a graph
plt.legend()

#show the graph
plt.show()
```



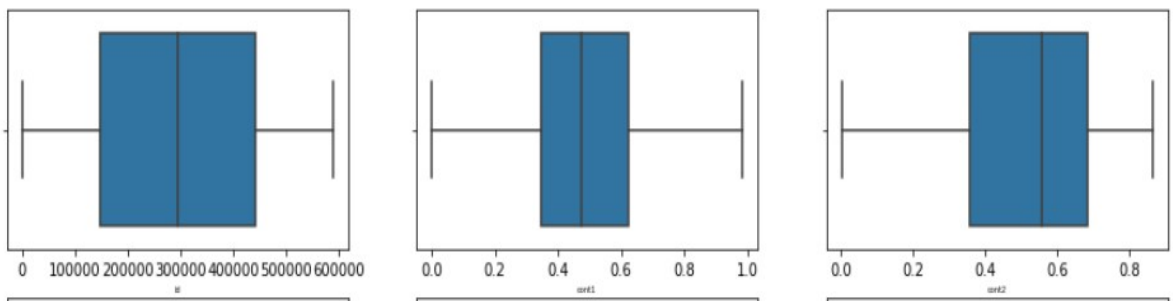
```
for i in df.columns[:-1]:
    plt.xlabel(i)
    plt.ylabel("loss")
    plt.scatter(df[i][:300],df["loss"][:300])
    plt.show()
```



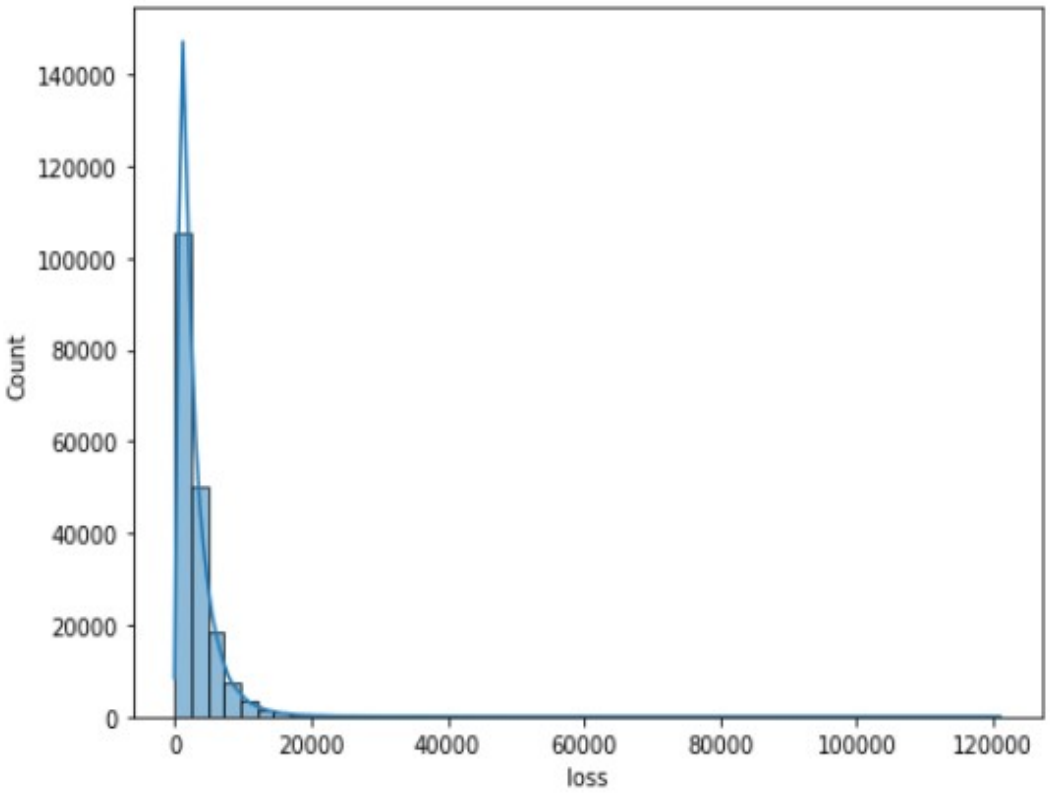
```python
# plot multiple histogram for all the numerical column in my data set:
df.plot.hist(subplots = True , layout = (4,5),figsize=(20,20))
# rearrange the plot:
plt.tight_layout()
plt.show()
```




```python
#To identify the Outliers in Numerical Columns:
#Subplots()
fig, ax = plt.subplots(3,3,figsize =(15,9))
for variable,subplot in zip(df_num.columns,ax.flatten()):
    z = sns.boxplot(x = df_num[variable], orient = 'h',whis = 1.5,ax = subplot)

    z.set_xlabel(variable,fontsize = 5)
```



```
3 sns.histplot(data=y, kde=True, bins=50)
4 plt.tight_layout()
```



```
# Label Encoding

# The label Encoding consider a level in a Categorical variable by 'Alphabetical Order'

from sklearn.preprocessing import LabelEncoder
# Create an Instance
labelencoder = LabelEncoder()

#Fit the Encoder
df['Encoded_performance_of_cat1'] = labelencoder.fit_transform(df['cat1'])

# Display the data
df
```

	id	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	...	cont7	cont8	cont9	cont10	cont11	cont12
0	1	A	B	A	B	A	A	A	A	B	...	0.335060	0.30260	0.67135	0.83510	0.569745	0.594646
1	2	A	B	A	A	A	A	A	A	B	...	0.436585	0.60087	0.35127	0.43919	0.338312	0.366307
2	5	A	B	A	A	B	A	A	A	B	...	0.315545	0.27320	0.26076	0.32446	0.381398	0.373424
3	10	B	B	A	B	A	A	A	A	B	...	0.391128	0.31796	0.32128	0.44467	0.327915	0.321570

```
print(df['cat1'].value_counts())
print('\n')
df['Encoded_performance_of_cat1'].value_counts()
```

```
A    141550
B     46768
Name: cat1, dtype: int64



0    141550
1     46768
Name: Encoded_performance_of_cat1, dtype: int64
```

```
x = np.array(df["cont1"]).reshape(-1,1)
y = np.array(df["loss"]).reshape(-1,1)
```

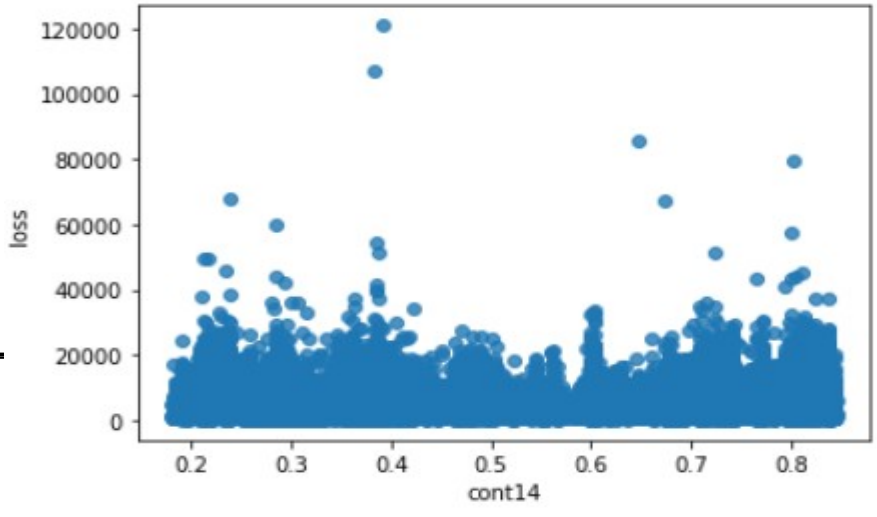
```
from sklearn.linear_model import LinearRegression
linear = LinearRegression()
linear.fit(x,y)
linear.predict([[12]])
```

```
array([[1214.35169353]])
```

```
## multiple Regression
x = df.drop("loss",axis = 1)
y = df["loss"]
```

```
sns.regplot(x="cont14",y="loss",data = df)
```

```
<AxesSubplot:xlabel='cont14', ylabel='loss'>
```



```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(df_num.drop("loss",axis = 1),df_num["id"],train_size = .75)
```

```
xtrain.shape
```

```
(141238, 15)
```

```
ytrain.shape
```

```
(141238,)
```

```
xtest.shape
```

```
(47080, 15)
```

```
ytest.shape
```

```
(47080,)
```

```
ytest.isnull().sum()
```

```
0
```

```
model = LinearRegression()
```

```
model.fit(xtrain,ytrain)
```

```
LinearRegression()
```

```
y_pred = model.predict(xtest)
```

```
ytest.head(10)
```

```
19407      61089
59649     186897
157826    491890
74298     232234
168874    526714
100911    315340
148414    462856
188085    586944
24040      75564
42362     132985
Name: id, dtype: int64
```

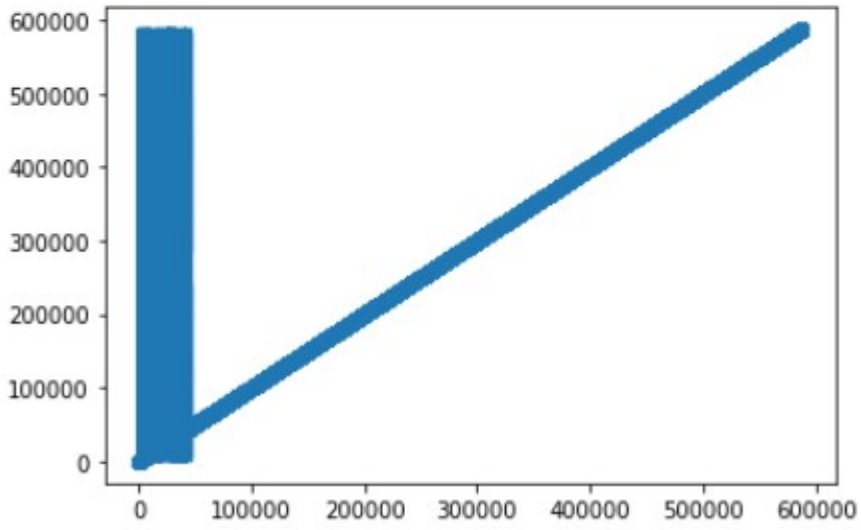
```
xtest.head(10)
```

```
1 model.score(xtrain,ytrain)
```

```
1.0
```

```
1 plt.scatter(xtest["id"],ytest)
2 plt.plot(y_pred)
```

```
[<matplotlib.lines.Line2D at 0x7f1d3773a5c0>]
```



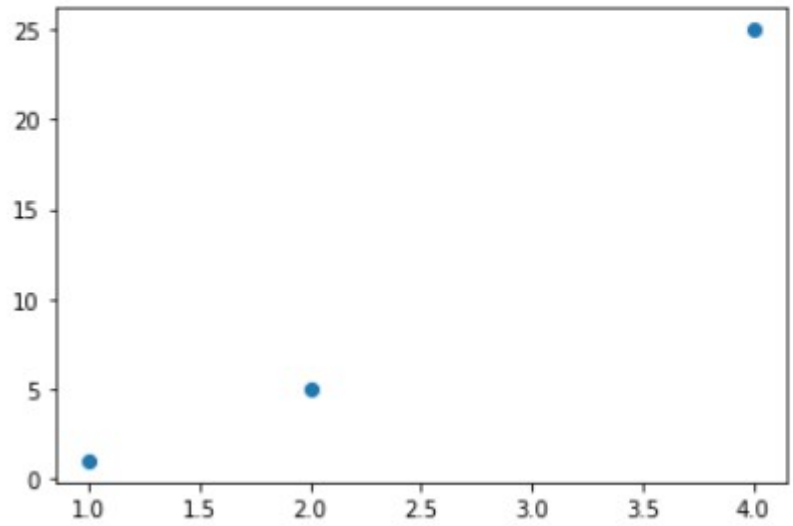
```python
for i in y_pred[:10]:
    print(i)
```

```
61089.000000001055
186897.00000000047
491889.999999999
232234.00000000032
526713.9999999988
315339.99999999994
462855.9999999992
586943.9999999986
75564.00000000102
132985.00000000073
```

```python
for i in df.columns[:-1]:
    plt.xlabel(i)
    plt.ylabel("loss")
    plt.scatter(df[i][:300],df["loss"][:300])
    plt.show()
```

```
1 plt.scatter(x_train[1],x_train[2])
```

```
<matplotlib.collections.PathCollection at 0x7f1d374656a0>
```



```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
print(poly.fit(x_train,y))
print(lin.fit(x_train,y))
lin.intercept_
```

```
PolynomialFeatures()
LinearRegression()

array([3031.94493943])
```

```
from sklearn.metrics import mean_absolute_error,mean_squared_error,mean_squared_log_error
```

```
mean_absolute_error(ytest,y_pred)
```

```
6.963696855254863e-10
```

```
mean_squared_error(ytest,y_pred)
```

```
6.436651276460588e-19
```

```
model.score(xtrain,ytrain)
```

```
1.0
```

Classification

Project 6:

6 : Predict Bank Credit Risk : Project Title Predict Bank Credit Risk using South German Credit Data Domain Banking

Problem Statement: Normally, most of the bank's wealth is obtained from providing credit loans so that a marketing bank must be able to reduce the risk of non-performing credit loans. The risk of providing loans can be minimized by studying patterns from existing lending data. One technique that you can use to solve this problem is to use data mining techniques. Data mining makes it possible to find hidden information from large data sets by way of classification. The goal of this project, you have to build a model to predict whether the person, described by the attributes of the dataset, is a good (1) or a bad (0) credit risk

Dataset Link : https://www.kaggle.com/competitions/south-german-credit-prediction/data

Dataset Fields

The below list consists of a detailed breakdown of all the features in the Dataset. 'Kredit' is our target variable, the one whose value must be predicted. P.S. The feature names are in German to preserve the authenticity of the data. In the dataset, they can be replaced using the DataDescription.csv file and the Pandas library.

Id = Id of individual entries, for evaluation

laufkont = status
1 : no checking account
2 : … < 0 DM 3 : 0<= … < 200 DM 4 : … >= 200 DM / salary for at least 1 year

laufzeit = duration

moral = credit_history
0 : delay in paying off in the past
1 : critical account/other credits elsewhere
2 : no credits taken/all credits paid back duly
3 : existing credits paid back duly till now
4 : all credits at this bank paid back duly

verw = purpose
0 : others
1 : car (new)
2 : car (used)
3 : furniture/equipment
4 : radio/television
5 : domestic appliances
6 : repairs
7 : education
8 : vacation
9 : retraining
10 : business

hoehe = amount

sparkont = savings
1 : unknown/no savings account
2 : … < 100 DM 3 : 100 <= … < 500 DM 4 : 500 <= … < 1000 DM 5 : … >= 1000 DM

beszeit = employment_duration
1 : unemployed
2 : < 1 yr 3 : 1 <= … < 4 yrs 4 : 4 <= … < 7 yrs 5 : >= 7 yrs

rate = installment_rate
1 : >= 35
2 : 25 <= … < 35
3 : 20 <= … < 25
4 : < 20

famges = personal_status_sex
1 : male : divorced/separated
2 : female : non-single or male : single
3 : male : married/widowed
4 : female : single

buerge = other_debtors
1 : none
2 : co-applicant
3 : guarantor

wohnzeit = present_residence
1 : < 1 yr 2 : 1 <= … < 4 yrs 3 : 4 <= … < 7 yrs 4 : >= 7 yrs

verm = property
1 : unknown / no property
2 : car or other
3 : building soc. savings agr./life insurance
4 : real estate

alter = age

weitkred = other_installment_plans
1 : bank
2 : stores
3 : none

wohn = housing
1 : for free
2 : rent
3 : own

bishkred = number_credits
1 : 1
2 : 2-3
3 : 4-5
4 : >= 6

beruf = job
1 : unemployed/unskilled - non-resident
2 : unskilled - resident
3 : skilled employee/official
4 : manager/self-empl./highly qualif. employee

pers = people_liable
1 : 3 or more
2 : 0 to 2

telef = telephone
1 : no
2 : yes (under customer name)

gastarb = foreign_worker
1 : yes
2 : no

kredit (target column) = credit_risk
0 : bad
1 : good

```
# importing Libraries
# importing Pandas Library as pd
import pandas as pd

# importing Numpy Library as np
import numpy as np

# importing matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# imporing seaborn as sns
import seaborn as sns

```

```
# Loading the dataset using pandas module and assign it as df
df = pd.read_csv('bankcredit.csv')

# Printing the dataset
df
```

	Id	laufkont	laufzeit	moral	verw	hoehe	sparkont	beszeit	rate	famges	...	verm	alter	weitkred	wohn	bishkred
0	0	1	18	4	2	1049	1	2	4	2	...	2	21	3	1	1
1	1	1	9	4	0	2799	1	3	2	3	...	1	36	3	1	2
2	2	2	12	2	9	841	2	4	2	2	...	1	23	3	1	1
3	3	1	12	4	0	2122	1	3	3	3	...	1	39	3	1	2
4	5	1	10	4	0	2241	1	2	1	3	...	1	48	3	1	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
795	993	1	18	4	0	3966	1	5	1	2	...	1	33	1	1	3
796	994	1	12	0	3	6199	1	3	4	3	...	2	28	3	1	2
797	997	4	21	4	0	12680	5	5	4	3	...	4	30	3	3	1
798	998	2	12	2	3	6468	5	1	2	3	...	4	52	3	2	1
799	999	1	30	2	2	6350	5	5	4	3	...	2	31	3	2	1

800 rows × 22 columns

```
df.columns = ['Id','status', 'duration', 'credit_history', 'purpose', 'amount', 'savings', 'employment_duration',
```

```
# By using tails we are getting last 5 values
df.tail(10)
```

	Id	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex	...	property	age	other_ins
790	987	1	12	2	3	674	2	4	4	4	...	2	20	
791	989	2	24	2	0	2718	1	3	3	2	...	2	20	
792	990	1	18	2	6	750	1	1	4	2	...	1	27	
793	991	2	24	2	1	12579	1	5	4	2	...	4	44	
794	992	1	18	2	1	7511	5	5	1	3	...	2	51	
795	993	1	18	4	0	3966	1	5	1	2	...	1	33	
796	994	1	12	0	3	6199	1	3	4	3	...	2	28	
797	997	4	21	4	0	12680	5	5	4	3	...	4	30	

```
1 # Information about the Dataset:
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Id                      800 non-null    int64
 1   status                  800 non-null    int64
 2   duration                800 non-null    int64
 3   credit_history          800 non-null    int64
 4   purpose                 800 non-null    int64
 5   amount                  800 non-null    int64
 6   savings                 800 non-null    int64
 7   employment_duration     800 non-null    int64
 8   installment_rate        800 non-null    int64
 9   personal_status_sex     800 non-null    int64
 10  other_debtors           800 non-null    int64
 11  present_residence       800 non-null    int64
 12  property                800 non-null    int64
 13  age                     800 non-null    int64
```

```
# isnull() method is used to check whether the dataset contain null values or not
# sum() is used to find count of null values
print(df.isnull().sum())
#getting size of the dataset(rows multiplied by column 1000x17)
print(df.size)
# getting rows and columnes in a dataset
print(df.shape)
```

```
Id                          0
status                      0
duration                    0
credit_history              0
purpose                     0
amount                      0
savings                     0
employment_duration         0
installment_rate            0
personal_status_sex         0
other_debtors               0
present_residence           0
property                    0
age                         0
other_installment_plans     0
housing                     0
number credits              0
```

```
17600
(800, 22)
```

```
# getting all statctical values like mean,median,count,min,max,standard deviation
# for numerical values
df.describe()
```

	Id	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate
count	800.00								
mean	478.10								
std	278.88								
min	0.00								
25%	238.75								
50%	472.00								
75%	707.25								
max	999.00								

8 rows × 22 c

```
df.columns
```

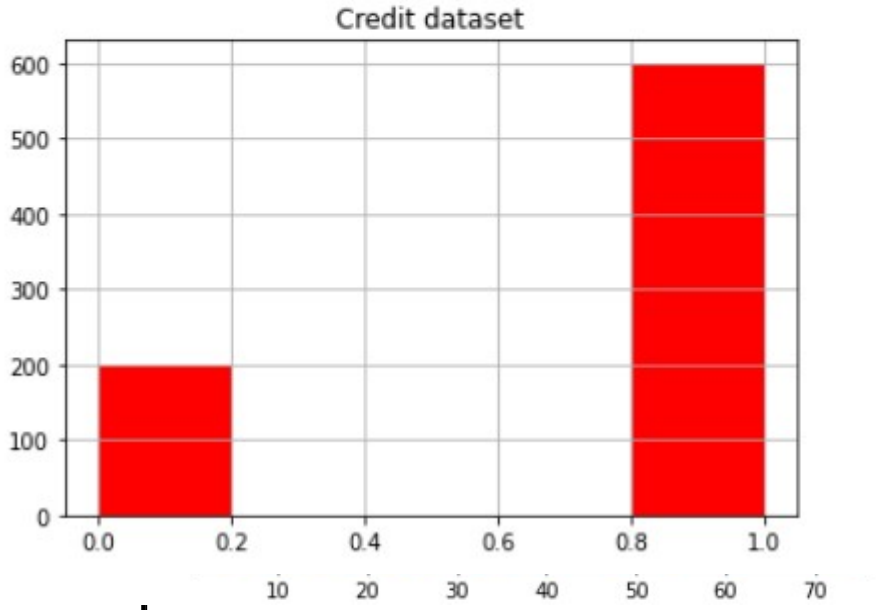
```
Index(['Id', 'status', 'duration', 'credit_history', 'purpose', 'amount',
       'savings', 'employment_duration', 'installment_rate',
       'personal_status_sex', 'other_debtors', 'present_residence', 'property',
       'age', 'other_installment_plans', 'housing', 'number_credits', 'job',
       'people_liable', 'telephone', 'foreign_worker', 'credit_risk'],
      dtype='object')
```

```
df.drop(['Id'],inplace=True,axis=1)
```

```
df
```

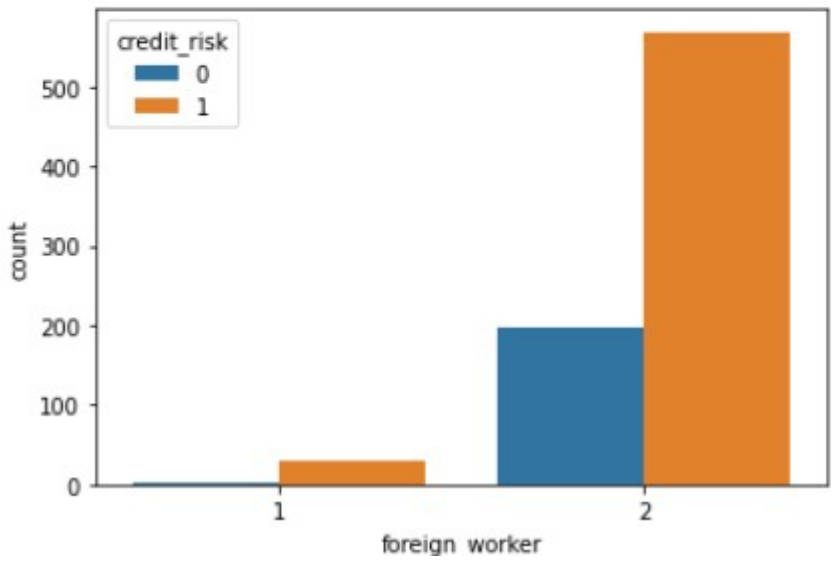
	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex	other_debtors	...	property	age
0	1	18	4	2	1049	1	2	4	2	1	...	2	21
1	1	9	4	0	2799	1	3	2	3	1	...	1	36
2	2	12	2	9	841	2	4	2	2	1	...	1	23
3	1	12	4	0	2122	1	3	3	3	1	...	1	39

```
plt.hist(x = df["credit_risk"],color ='r',bins=5)
plt.title("Credit dataset")
# plot the grid:
plt.grid()
# display the plot:
plt.show()
```



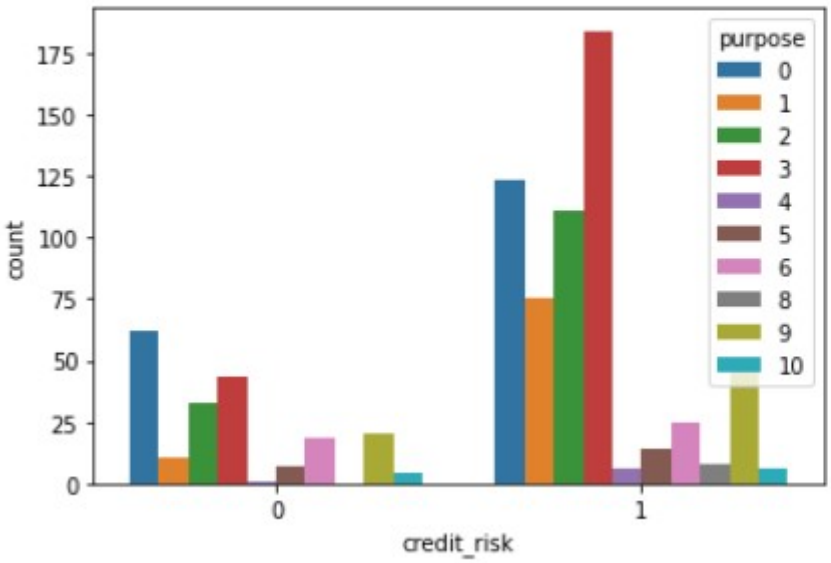

```
from warnings import filterwarnings
filterwarnings('ignore')
sns.countplot(df['foreign_worker'],hue=df['credit_risk'])
```

```
<AxesSubplot:xlabel='foreign_worker', ylabel='count'>
```

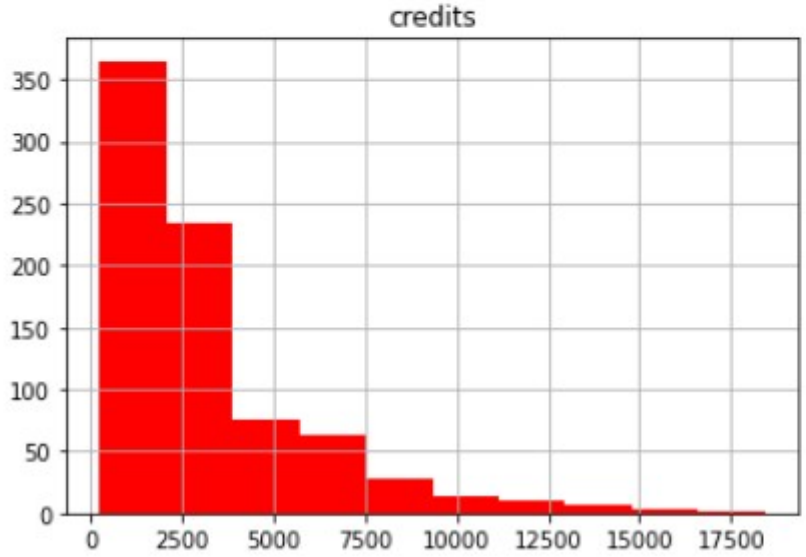


```
1 sns.countplot(df['credit_risk'],hue=df['purpose'])
```

```
<AxesSubplot:xlabel='credit_risk', ylabel='count'>
```



```python
#Plot the Histogram with multiple bins  and add Grid:

plt.hist(x = df['amount'],color = 'r')
plt.title('credits')

#Plot the Grid
plt.grid()
plt.show()
```



```
# check the bad loans
df[df['credit_risk']==0]
```

ersonal_status_sex	other_debtors	...	property	age	other_installment_plans	housing	number_credits	job	people_liable	telephone	foreign_worker	credit_risk
2	1	...	3	23	3	1	1	3	2	2	2	0
2	1	...	1	30	3	2	2	3	2	1	2	0
2	1	...	1	24	3	2	1	2	2	1	2	0
4	1	...	3	25	1	2	1	2	2	1	2	0
2	1	...	1	28	3	2	1	2	2	1	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
2	1	...	1	33	1	1	3	3	2	2	2	0
3	1	...	2	28	3	1	2	3	2	2	2	0
3	1	...	4	30	3	3	1	4	2	2	2	0
3	1	...	4	52	3	2	1	4	2	2	2	0
3	1	...	2	31	3	2	1	3	2	1	2	0

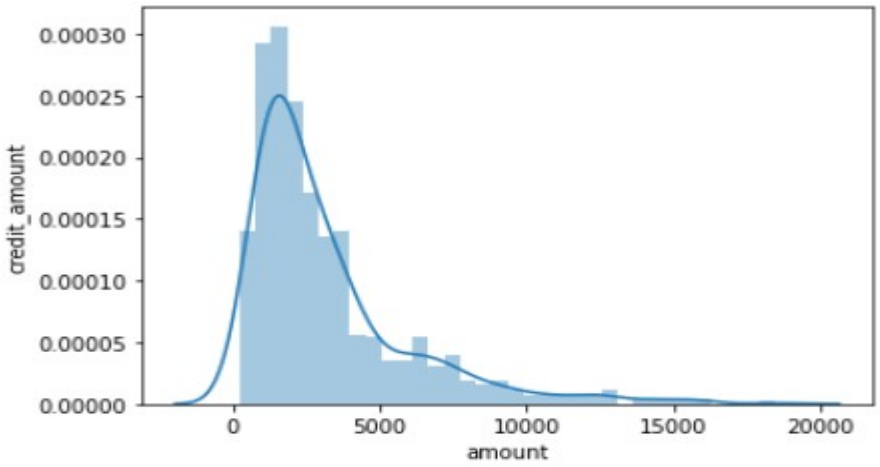
```
df["credit_risk"].value_counts()

```

```
1    600
0    200
Name: credit_risk, dtype: int64
```

DATA TRANSFORMATION

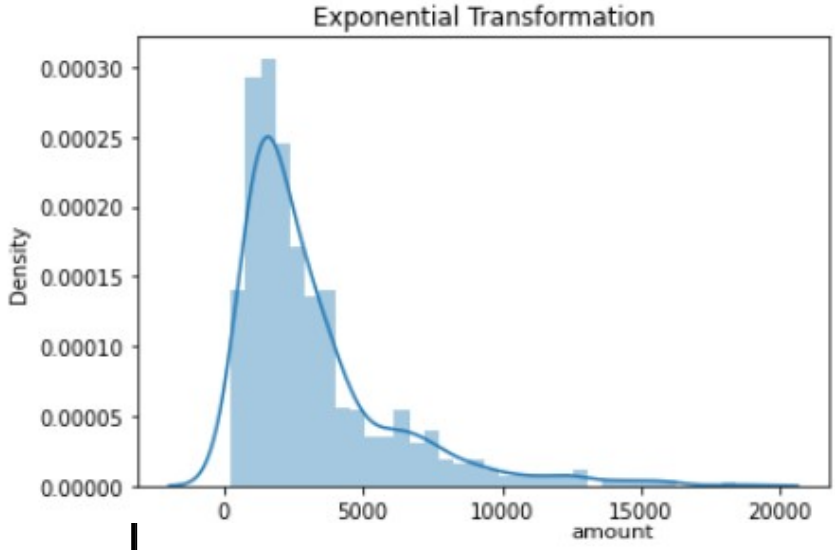
```
sns.distplot(df['amount'])
plt.ylabel('credit_amount')
print('Skewness : ',df['amount'].skew())
plt.show()
```

```
Skewness :  2.0472701844801806
```



```
displacement = np.exp(log_amount)
print('Skewness after log Transformtion : ',displacement.skew())
# plot the Distribution
sns.distplot(displacement)
plt.ylabel('Density')
plt.title('Exponential Transformation')
plt.show()
```

```
Skewness after log Transformtion :  2.04727018448018
```




```
df.drop(['age','duration'],axis=1,inplace=True)
df
```

	status	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_s
0	1	4	2	1049	1	2	4	
1	1	4	0	2799	1	3	2	
2	2	2	9	841	2	4	2	
3	1	4	0	2122	1	3	3	
4	1	4	0	2241	1	2	1	
...	...	...	...	...	...	...	...	
795	1	4	0	3966	1	5	1	
796	1	0	3	6199	1	3	4	
797	4	4	0	12680	5	5	4	
798	2	2	3	6468	5	1	2	
799	1	2	2	6350	5	5	4	

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
xtrain, xtest, ytrain, ytest = train_test_split(df.drop("credit_risk",axis=1), df["credit_risk"],train_size = .75
```

```
xtrain.shape, xtest.shape
```

```
((600, 18), (200, 18))
```

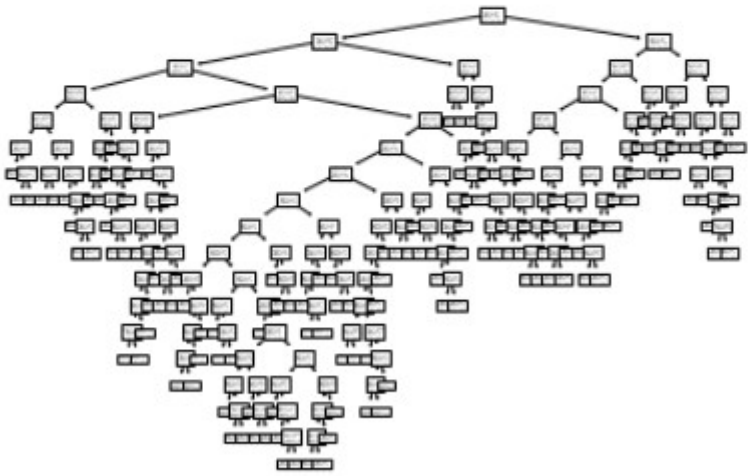
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
dtc = DecisionTreeClassifier()
```

```
dtc.fit(xtrain, ytrain)
```

```
DecisionTreeClassifier()
```

```
plot_tree(dtc)
```

```
Text(329.40000000000003, 102.67999999999999, 'gini = 0.0\nsampl
Text(329.40000000000003, 126.84, 'gini = 0.0\nsamples = 19\nval
Text(329.40000000000003, 163.07999999999998, 'X[1] <= 3.0\ngini
Text(326.70000000000005, 151.0, 'gini = 0.0\nsamples = 2\nvalue
Text(332.1, 151.0, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]')]
```



```
dtc.score(xtrain,ytrain)
```

```
1.0
```

```
y_pred = dtc.predict(xtest)
```

```
print(classification_report(ytest, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.42      0.52      0.46        44
           1       0.86      0.79      0.82       156

    accuracy                           0.73       200
   macro avg       0.64      0.66      0.64       200
weighted avg       0.76      0.73      0.74       200
```

```
dtc.score(xtest, ytest)
```

```
0.735
```

```
DecisionTreeClassifier?
```

```
from sklearn.model_selection import GridSearchCV
```

```
params = {
    'criterion' :  ["gini", "entropy", "log_loss"],
    "splitter" : ["best", "random"],
    "max_features" : ["auto", "sqrt", "log2"],
    "min_samples_leaf" : [1,2,5,6]
}
```

```
grid = GridSearchCV(estimator = dtc, param_grid = params)
```

```
grid.fit(xtrain,ytrain)
```

```
/home/student/my_project_env/lib/python3.6/site-packages/sklearn/model_selection/_vali
ng: Estimator fit failed. The score on this train-test partition for these parameters
```

```
dtc.score(xtrain,ytrain)
```

```
1.0
```

```
dtc.score(xtest, ytest)
```

```
0.735
```

```
dtc2 = DecisionTreeClassifier(
    criterion= 'gini',
    max_features = 'sqrt',
    splitter = 'random',
    min_samples_leaf = 6)
```

```
dtc2.fit(xtrain,ytrain)
```

```
DecisionTreeClassifier(max_features='sqrt', min_samples_leaf=6,
                       splitter='random')
```

```
from sklearn.preprocessing import MinMaxScaler
min = MinMaxScaler()
```

```
for column in df.columns:
  df[column] = min.fit_transform(df[[column]])
```

```
df.describe()
```

	status	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex
count	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000
mean	0.549583	0.645625	0.278500	0.162886	0.285938	0.598750	0.650833	0.562500
std	0.416977	0.274967	0.268053	0.153672	0.397354	0.306018	0.378132	0.232248
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.250000	0.500000	0.100000	0.061296	0.000000	0.500000	0.333333	0.333333
50%	0.333333	0.500000	0.200000	0.110818	0.000000	0.500000	0.666667	0.666667
75%	1.000000	1.000000	0.300000	0.201235	0.500000	1.000000	1.000000	0.666667

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
```

```
lr = LogisticRegression()
dtc = DecisionTreeClassifier()
sv = SVC()
gnb =GaussianNB()
```

```
g = GridSearchCV(estimator= dtc ,param_grid = {})
```

```
x = df.drop("credit_risk",axis=1)
y = df["credit_risk"]
```

```
svc = GaussianNB()
```

```
svc.fit(x,y)
```

```
GaussianNB()
```

```
1 svc.score(x,y)
```

```
0.76
```

```
1 x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   status                  800 non-null    int64
 1   credit_history          800 non-null    int64
 2   purpose                 800 non-null    int64
 3   amount                  800 non-null    int64
 4   savings                 800 non-null    int64
 5   employment_duration     800 non-null    int64
 6   installment_rate        800 non-null    int64
 7   personal_status_sex     800 non-null    int64
 8   other_debtors           800 non-null    int64
 9   present_residence       800 non-null    int64
 10  property                800 non-null    int64
 11  other_installment_plans 800 non-null    int64
```

```
rfc2 = RandomForestClassifier(criterion = 'log_loss', n_estimators =  1000)
```

```
rfc2.fit(x,y)

rfc2.score(x,y)
```

```
1.0
```

1.0

```
rfc?
```

```
pr = {
    'n_estimators' : [10,100,1000,600],
    'criterion':["log_loss", "entropy", "gini"]
}
```

```
g = GridSearchCV(estimator=rfc, param_grid = pr)
```

```
g.fit(x,y)
```

```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'criterion': ['log_loss', 'entropy', 'gini'],
                         'n estimators': [10, 100, 1000, 600]})
```


```
rfc2 = RandomForestClassifier(criterion = 'log_loss', n_estimators =  1000)
```

```
rfc2.fit(x,y)

rfc2.score(x,y)
```


```
1.0
```

Clustering

Project 6:

Email Spam Detection : Email Spam Detection: Cluster email data to detect spam or malicious messages. Procedure: Apply K-means on email features like sender, subject, content, and attachments to group similar emails. Domain: Cybersecurity Dataset Link: https://www.kaggle.com/code/mfaisalqureshi/email-spam-detection-98-accuracy/input?select=spam.csv

```
# importing Libraries
# importing Pandas Library as pd
import pandas as pd

# importing Numpy Library as np
import numpy as np

# importing matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# imporing seaborn as sns
import seaborn as sns
```

```
# Loading the dataset using pandas module and assign it as df
df = pd.read_csv('spam.csv')

# Printing the dataset
df
```

```
#Dropping the last 3 columns
data.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'], axis=1, inplace=True)
```

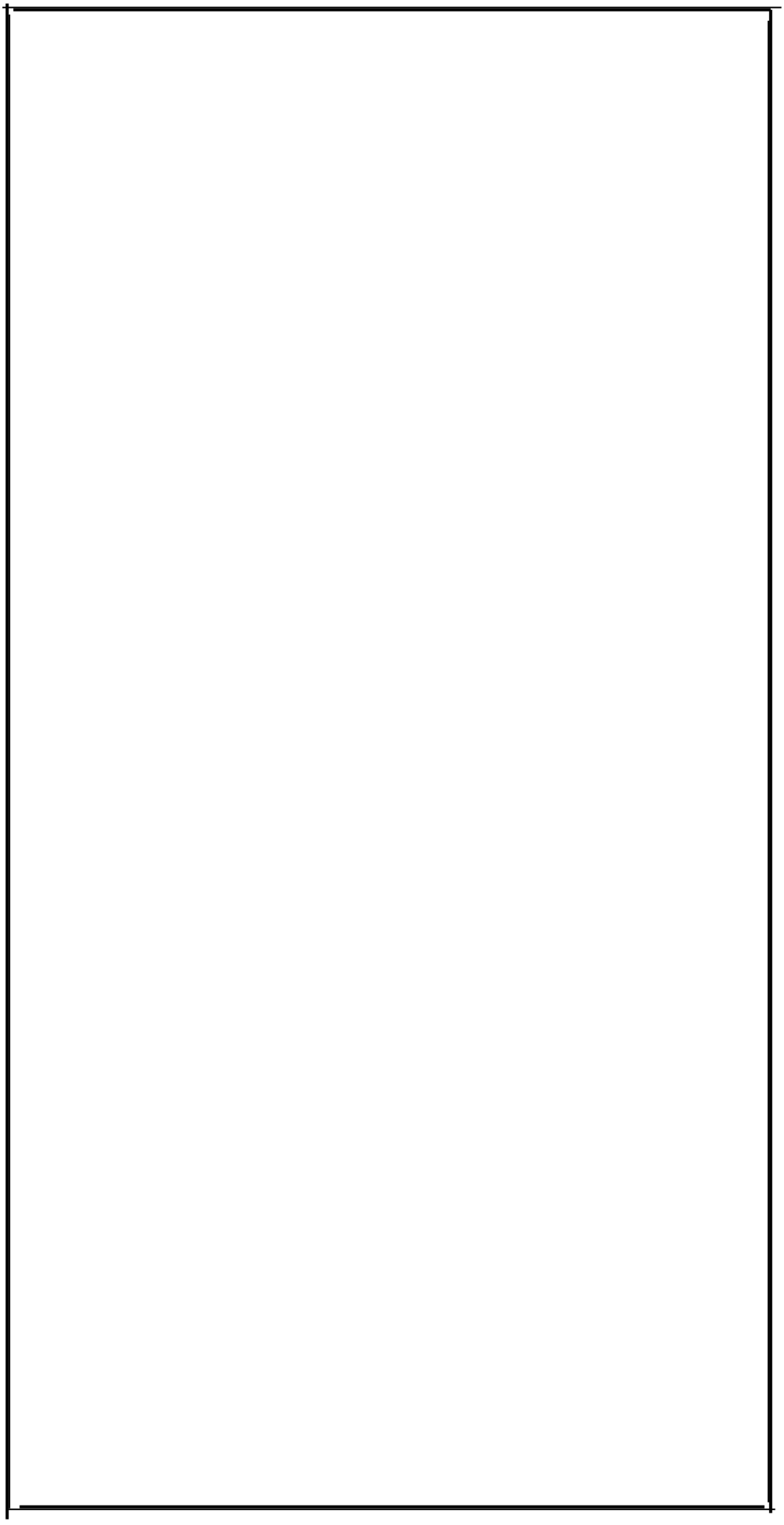
```
#Renaming the columns to be understandable
data.rename(columns={'v1': 'Target', 'v2': 'Email'}, inplace=True)
data.head()
```

	Target	Email
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

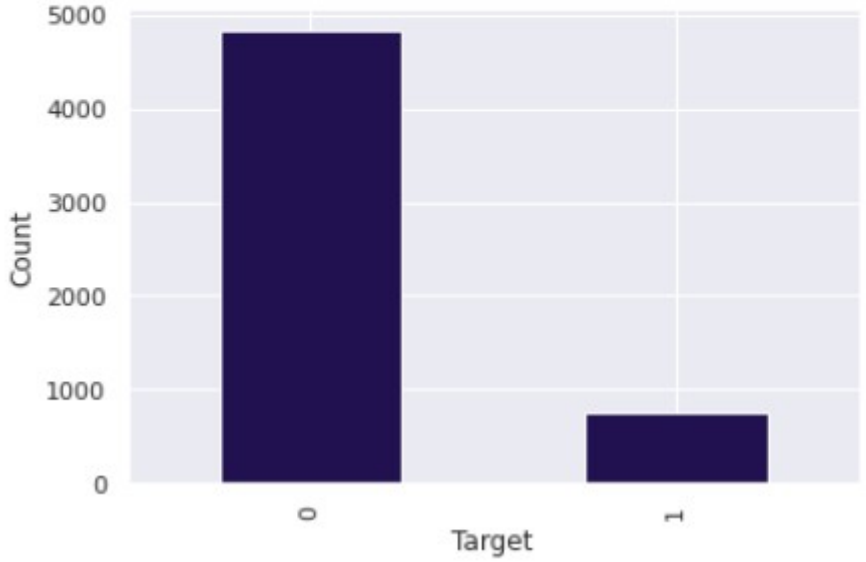
```
#Mapping the target labels to 0 and 1
data['Target']=data['Target'].map({'ham': 0, 'spam': 1})

data.head()
```

	Target	Email
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
import matplotlib.pyplot as plt
import seaborn as sns
#Setting a color palette
palette=sns.color_palette('magma')
sns.set(palette=palette)
```



```
#Plotting Spam(1) vs Not Spam(0) value counts
data['Target'].value_counts().plot(kind='bar')
plt.xlabel('Target')
plt.ylabel('Count')
plt.show()
```



```
#Splitting the data
from sklearn.model_selection import train_test_split
X_train, X_valid, y_train, y_valid= train_test_split(data['Email'], data['Target'], test_size=0.2, random_state
X_train.head()
```

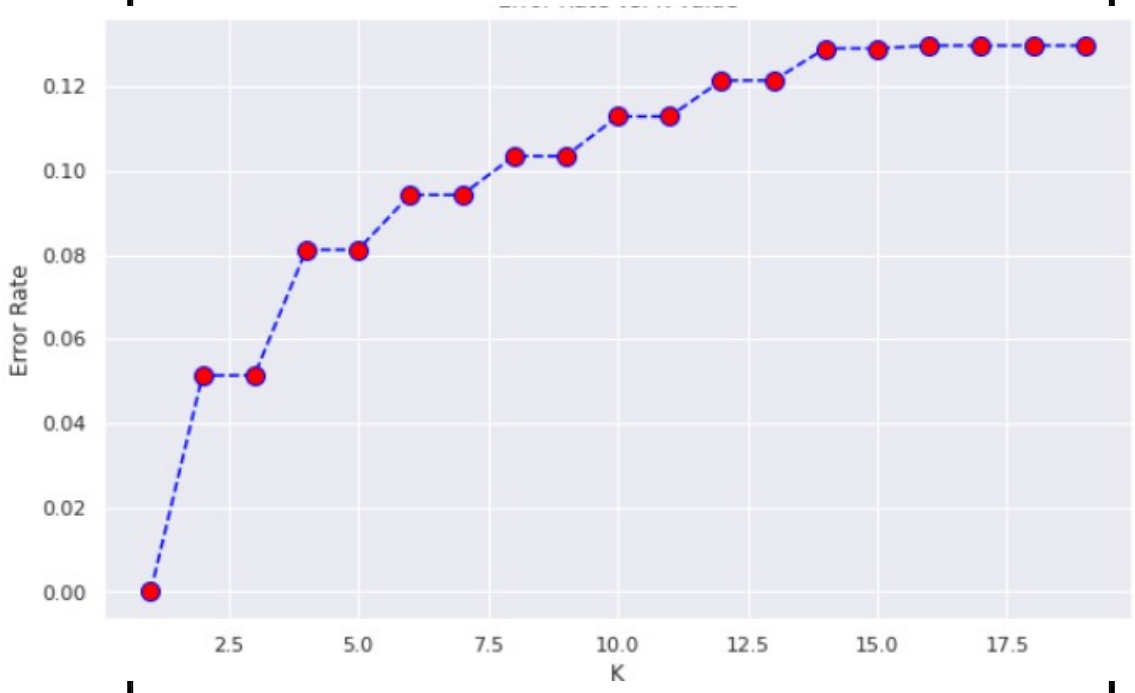
```
1114    No no:)this is kallis home ground.amla home to...
3589    I am in escape theatre now. . Going to watch K...
3095    We walked from my moms. Right on stagwood pass...
1012       I dunno they close oredi not... ÌÏ v ma fan...
3320                              Yo im right by yo work
Name: Email, dtype: object
```

```
#Funtion to build and visualise a confusion matrix
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
def my_confusion_matrix(y_test, y_pred, plt_title, accuracy_title):
    cm=confusion_matrix(y_test, y_pred)
    print(f'{accuracy_title} Accuracy Score:', '{:.2%}'.format(accuracy_score(y_valid, y_pred)))
    print(classification_report(y_test, y_pred))
    sns.heatmap(cm, annot=True, fmt='g', cbar=False, cmap='magma')
    plt.xlabel('Predicted Values')
    plt.ylabel('Actual Values')
    plt.title(plt_title)
    plt.show()
    return cm
```

```
X_train.isnull().any()
```

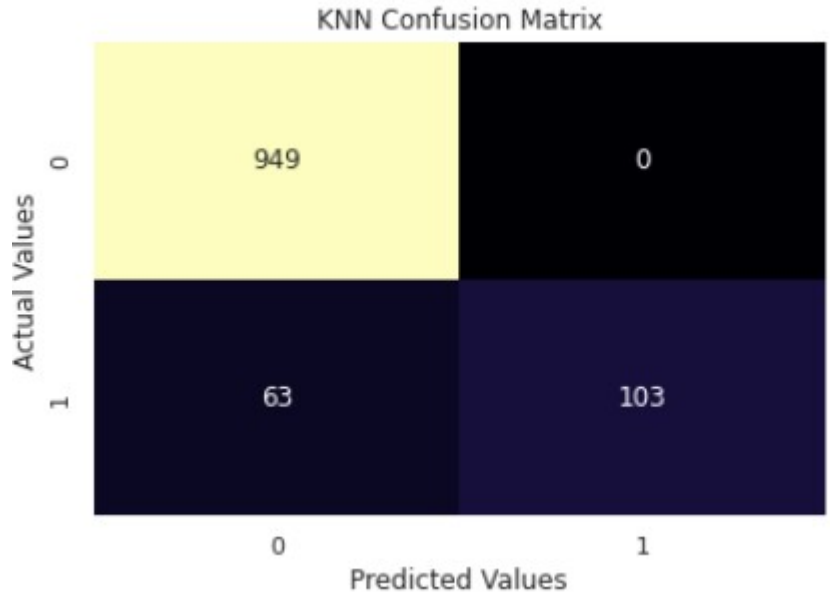
```
False
```

```python
#Visualization to find the best K value
from sklearn.neighbors import KNeighborsClassifier
#To find the optimal k value: K=((Sqrt(N)/2)
#Visualisation for the Error Rate/K-value
error_rate = []
for i in range(1,20):
    knn = KNeighborsClassifier(n_neighbors=i, metric = 'minkowski', p=1)
    knn.fit(tfidf_matrix_train, y_train)
    pred_i_knn = knn.predict(tfidf_matrix_train)
    error_rate.append(np.mean(pred_i_knn != y_train))
plt.figure(figsize=(10,6))
plt.plot(range(1,20),error_rate,color='blue', linestyle='dashed',
         marker='o',markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
plt.show()
```



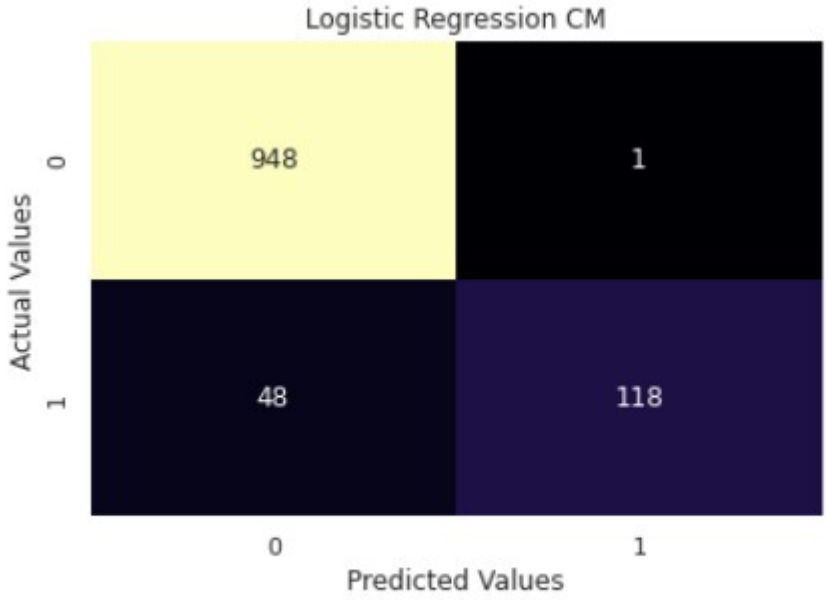
```python
#Fitting the KMM model
knn_classifier = KNeighborsClassifier(n_neighbors = 1, metric = 'minkowski', p=1)
knn_classifier.fit(tfidf_matrix_train, y_train)
y_pred_knn=knn_classifier.predict(tfidf_matrix_valid)
cm_knn=my_confusion_matrix(y_valid, y_pred_knn, 'KNN Confusion Matrix', 'KNN')
```

```
KNN Accuracy Score: 94.35%
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       949
           1       1.00      0.62      0.77       166

    accuracy                           0.94      1115
   macro avg       0.97      0.81      0.87      1115
weighted avg       0.95      0.94      0.94      1115
```



```python
#Training the model
from sklearn.linear_model import LogisticRegression
log_reg_classifier=LogisticRegression(solver='liblinear')
log_reg_classifier.fit(tfidf_matrix_train, y_train)
y_pred_log=log_reg_classifier.predict(tfidf_matrix_valid)
my_confusion_matrix(y_valid, y_pred_log, 'Logistic Regression CM', 'Logistic Regression:')
```

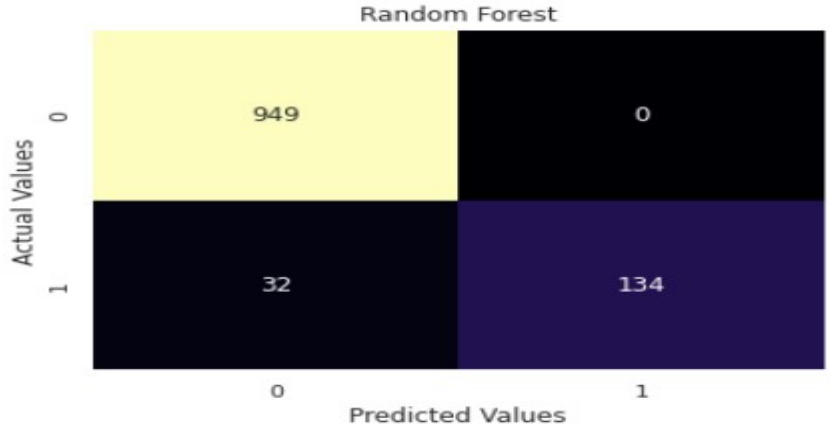
```
Logistic Regression: Accuracy Score: 95.61%
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       949
           1       0.99      0.71      0.83       166

    accuracy                           0.96      1115
   macro avg       0.97      0.85      0.90      1115
weighted avg       0.96      0.96      0.95      1115
```




```
array([[948,   1],
       [ 48, 118]])
```

```python
#Random Forest
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(tfidf_matrix_train, y_train)
y_pred_rfc=rfc.predict(tfidf_matrix_valid)
print(my_confusion_matrix(y_valid, y_pred_rfc, 'Random Forest', 'Random Forest'))
```

```
Random Forest Accuracy Score: 97.13%
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       949
           1       1.00      0.81      0.89       166

    accuracy                           0.97      1115
   macro avg       0.98      0.90      0.94      1115
weighted avg       0.97      0.97      0.97      1115
```




```
[[949   0]
 [ 32 134]]
```

4.STUDENT FEEDBACK

I am very thankful for giving me this opportunity to the SURE Trust. I enjoyed the course and learned a lot from it. The content is well organized and focused on both theory and practical situations. I particularly enjoyed the teaching style of trainer who have teach this course to us. Because, he teach this course in understanding manner. And I learned life skills, how to prepare resume from 'LST' Session. I loved it. And 'EACH ONE PLANT ONE' program inspires me a lot in SURE Trust. I really thankful to SURE Trust.

5.UNIQUENESS OF THE CROUSE

- Faculty is committed in delivering the course
- Concepts are learnt theoretically and practically
- Thought provoking assignments will be given
- Preparation of course report, which will help us to refer in the future
- Concentrate on each and every student in the course

6.Concluding Remarks

A lot of experience, knowledge and exposure that I have handy. All disclosures were awaken myself in a boost of self-confidence to face life more challenging now. Practical is a complement to the science or theory learned. I conclude that the SURE Trust training program has provided many benefits to students. I really proud to be a student of SURE Trust.