MACHINE LEARNING ASSIGNMENT 1

NAME : ASHOK SAI SUDIREDDY
ID : 700734963
Video Link:
https://drive.google.com/file/d/1ZYcYDMTvuv7-KrUdA370UCJLV1_LikER/view?usp=sharing
GIT HUB LINK :
https://github.com/AshokSai1999/Machine-Learning.git

Q1) Sort the list of ages, find min and max, average, median and range

```
import statistics
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
# Sorts the age in ascending order
ages.sort()
# Displays sorted values
print ("Sorted age:", ages)
# We use min method to display minimum age
print ("Minimum age:", min(ages))
# We use max method to display maximum age
print ("Maximum age:", max(ages))
# Using append method to insert the min and max values of age to the list
ages.append(min(ages))
ages.append(max(ages))
#Displays the list again with new values
print ("Adding  min and max values :",ages)
# Median
median_age = statistics.median(ages)
print ("Median:", mdn_age)
# Average age
average_age= sum(ages)/len(ages)
print ("Average = ", average)
# Range of ages
rangeof_age=max(ages)-min(ages)
print ("Range = ", rangeof_age)
```

```
[61]: #Question 1
import statistics
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
# Sorts the age in ascending order
ages.sort()
# Displays sorted values
print ("Sorted age:", ages)
# We use min method to display minimum age
print ("Minimum age:", min(ages))
# We use max method to display maximum age
print ("Maximum age:", max(ages))
# Using append method to insert the min and max values of age to the list
ages.append(min(ages))
ages.append(max(ages))
#Displays the list again with new values
print ("Adding  min and max values :",ages)
# Median
median_age = statistics.median(ages)
print ("Median:", mdn_age)
# Average age
average_age= sum(ages)/len(ages)
print ("Average = ", average)
# Range of ages
rangeof_age=max(ages)-min(ages)
print ("Range = ", rangeof_age)
```

```
Sorted age: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
Minimum age: 19
Maximum age: 26
Adding  min and max values : [19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]
Median: 24.0
Average =  22.75
Range =  7
```

Q2) Create a dictionary

dog = {'name':'jimmy','color':'black','breed':'poodle','legs':'4','age':'3'}

print ("Dog Dictionary Created:",dog)

Creating Student dictionary with given keys and values

Student ={'first_name':'Ashok','last_name':'Reddy','Gender':'Male','age':'21','marital_status':'single', 'skills':'musician','Country':'India','City':'vizag','Address':'2/18'}

print (" Dictionary Created for Student:",student)

Creating dictionary for skills

skills = {'bowler':'1','musician':'2','coder':'3'}

print (" Dictionary Created for Skills:",skills)

Finding the length of student dictionary

print ("Length of student:", len(student))

Check the datatype of skills

print (" skills Datatype:",type(skills))

To get values of skills dictionary

print ("Values of skills:",skills.values())

Adding one more item to skills

skills['cricketer'] = 4

print ("New skill added:",skills)

Get dog and student key and values

print ("Dog keys:",dog.keys())

print ("Student values:",student.values())

In [21]:
```
#Question 2
dog = {'name':'jimmy','color':'black','breed':'poodle','legs':'4','age':'3'}
print ("Dog Dictionary Created:",dog)
# Creating Student dictionary with given keys and values
student = {'first_name':'Ashok','last_name':'Reddy','Gender':'Male','age':'21','marital_status':'single',
'skills':'musician','Country':'India','City':'vizag','Address':'2/18'}
print (" Dictionary Created for Student:",student)
# Creating dictionary for skills
skills = {'bowler':'1','musician':'2','coder':'3'}
print (" Dictionary Created for Skills:",skills)
# Finding  the length of student dictionary
print ("Length of student:", len(student))
# Check the datatype of skills
print (" skills Datatype:",type(skills))
#  To get values of skills dictionary
print ("Values of skills:",skills.values())
# Adding one more item to skills
skills['cricketer'] = 4
print ("New skill added:",skills)
# Get dog and student key and values
print ("Dog keys:",dog.keys())
print ("Student values:",student.values())
```

```
Dog Dictionary Created: {'name': 'jimmy', 'color': 'black', 'breed': 'poodle', 'legs': '4', 'age': '3'}
 Dictionary Created for Student: {'first_name': 'Ashok', 'last_name': 'Reddy', 'Gender': 'Male', 'age': '21', 'marital_status':
'single', 'skills': 'musician', 'Country': 'India', 'City': 'vizag', 'Address': '2/18'}
 Dictionary Created for Skills: {'bowler': '1', 'musician': '2', 'coder': '3'}
Length of student: 9
 skills Datatype: <class 'dict'>
Values of skills: dict_values(['1', '2', '3'])
New skill added: {'bowler': '1', 'musician': '2', 'coder': '3', 'cricketer': 4}
Dog keys: dict_keys(['name', 'color', 'breed', 'legs', 'age'])
Student values: dict_values(['Ashok', 'Reddy', 'Male', '21', 'single', 'musician', 'India', 'vizag', '2/18'])
```

Q3) Create tuple of sisters and brothers

Sisters = ('Surekha', 'Sreelekha','Durga','Swetha')

Brothers = ('Sarath','Srikanth','Ashok','Sandeep')

Creating a tuple as siblings and joining the sister's and brother's tuple

siblings = Sisters + Brothers

Displays siblings' output and length of siblings

print("Siblings:", siblings)

print("Length of Siblings:", len(siblings))

Creating another tuple as family_members and adding father and mother name to it

family_members = siblings + ('Lakshmi Narayana','Madhavi')

Displays family_members output

print("Family_members:",family_members)

In [22]:
```
#Question 3
Sisters = ('Surekha', 'Sreelekha','Durga','Swetha')
Brothers = ('Sarath','Srikanth','Ashok','Sandeep')
# Creating a tuple as siblings and joining the sister's and brother's tuple
siblings = Sisters + Brothers
# Displays siblings' output and length of siblings
print("Siblings:", siblings)
print("Length of Siblings:", len(siblings))
# Creating another tuple as family_members and adding father and mother name to it
family_members = siblings + ('Lakshmi Narayana','Madhavi')
# Displays family_members output
print("Family_members:",family_members)
```

```
Siblings: ('Surekha', 'Sreelekha', 'Durga', 'Swetha', 'Sarath', 'Srikanth', 'Ashok', 'Sandeep')
Length of Siblings: 8
Family_members: ('Surekha', 'Sreelekha', 'Durga', 'Swetha', 'Sarath', 'Srikanth', 'Ashok', 'Sandeep', 'Lakshmi Narayana', 'Madh
avi')
```

Q4) Length of the set

```
it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
print("Length of it_companies:", len(it_companies))
#Adding twitter to it_companies
it_companies.add('Twitter')
print("After adding another item:",it_companies)
#Adding multiple it_companies
it_companies.update({'Tcs','Accenture','Delloit','IBM'})
print("After adding multiple items:",it_companies)
#Remove
it_companies.remove('Accenture')
print("After removing one company:",it_companies)
#Discard
it_companies.discard('Accenture')
print("After discarding company:",it_companies)
# If any item is not present Discard will not raise any error
#Joining A & B
A = {19, 22, 24, 20, 25, 26}
B = {19, 22, 20, 25, 26, 24, 28, 27}
print("Join Of A and B:", A.union(B))
#Intersection
print("Intersection of A and B:", A.intersection(B))
#Subset
print("Subset of A and B:", A.issubset(B))
#Disjoint
print("Disjoint:", A.isdisjoint(B))
#Converting list to set
age = [22, 19, 24, 25, 26, 24, 25, 24]
print("Converting list to set:", set(age))
#Length of set
print("Length of set:",len(set(age)))
#Length of list
print("Length of list:",len(age))
#Symmetric diff- returns values which are not in common with other set
print("Symmetric diff:",A.symmetric_difference(B))
#delete set
A.clear()
print(A)
B.clear()
print(B)
```

In [24]:
```
#Questioin 4
it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
print("Length of it_companies:", len(it_companies))
#Adding twitter to it_companies
it_companies.add('Twitter')
print("After adding another item:",it_companies)
#Adding multiple it_companies
it_companies.update({'Tcs','Accenture','Delloit','IBM'})
print("After adding multiple items:",it_companies)
#Remove
it_companies.remove('Accenture')
print("After removing one company:",it_companies)
#Discard
it_companies.discard('Accenture')
print("After discarding company:",it_companies)
# If any item is not present Discard will not raise any error
#Joining A & B
A = {19, 22, 24, 20, 25, 26}
B = {19, 22, 20, 25, 26, 24, 28, 27}
print("Join Of A and B:", A.union(B))
#Intersection
print("Intersection of A and B:", A.intersection(B))
#Subset
print("Subset of A and B:", A.issubset(B))
#Disjoint
print("Disjoint:", A.isdisjoint(B))
#Converting list to set
age = [22, 19, 24, 25, 26, 24, 25, 24]
print("Converting list to set:", set(age))
#Length of set
print("Length of set:",len(set(age)))
```

```
#Length of list
print("Length of list:",len(age))
#Symmetric diff- returns values which are not in common with other set
print("Symmetric diff:",A.symmetric_difference(B))
#delete set
A.clear()
print(A)
B.clear()
print(B)
```

```
Length of it_companies: 7
After adding another item: {'Microsoft', 'Amazon', 'IBM', 'Twitter', 'Oracle', 'Facebook', 'Apple', 'Google'}
After adding multiple items: {'Microsoft', 'Accenture', 'Twitter', 'Delloit', 'Oracle', 'Facebook', 'Apple', 'Google', 'Tcs',
'Amazon', 'IBM'}
After removing one company: {'Microsoft', 'Twitter', 'Delloit', 'Oracle', 'Facebook', 'Apple', 'Google', 'Tcs', 'Amazon', 'IB
M'}
After discarding company: {'Microsoft', 'Twitter', 'Delloit', 'Oracle', 'Facebook', 'Apple', 'Google', 'Tcs', 'Amazon', 'IBM'}
Join Of A and B: {19, 20, 22, 24, 25, 26, 27, 28}
Intersection of A and B: {19, 20, 22, 24, 25, 26}
Subset of A and B: True
Disjoint: False
Converting list to set: {19, 22, 24, 25, 26}
Length of set: 5
Length of list: 8
Symmetric diff: {27, 28}
set()
set()
```

Q5) Calculate area of circle and circumference of circle

Initializing r

r = int(input("enter r:"))

Calculating area of circle and circumference of circle

_area_of_circle = 3.14*r*r

_circum_of_circle = 2*3.14*r

Displays area of circle and circumference of circle

print("Area of Circle:",_area_of_circle)

print("Circumference of Circle:",_circum_of_circle)

In [39]:
```
#Question 5
# Initializing r
r = int(input("enter r:"))
# Calculating area of circle and circumference of circle
_area_of_circle = 3.14*r*r
_circum_of_circle = 2*3.14*r
# Displays area of circle and circumference of circle
print("Area of Circle:",_area_of_circle)
print("Circumference of Circle:",_circum_of_circle)
```

```
enter r:30
Area of Circle: 2826.0
Circumference of Circle: 188.4
```

Q6) Unique words using split method

Unique
statement = "I am a teacher and I love to inspire and teach people"
Using split method to separate the words and get the unique values
spt=set(statement.split(" "))
print(spt)
print ("Length:",len(spt))

```
: #Question 6
# Unique
statement = "I am a teacher and I love to inspire and teach people"
# Using split method to separate the words and get the unique values
spt=set(statement.split(" "))
print(spt)
print ("Length:",len(spt))
```

```
{'I', 'and', 'to', 'a', 'teacher', 'people', 'inspire', 'teach', 'am', 'love'}
Length: 10
```

Q7) Used tab and escape to display them in the given format

a= "Name\t Age\tCountry\tCity\t\nAsabeneh 250\tFinland\tHelsinki"

print(a)

```
#Question 7
a= "Name\t Age\tCountry\tCity\t\nAsbeneh  250\tFinland\tHelsinki"
print(a)
```

```
Name     Age    Country City
Asbeneh  250    Finland Helsinki
```

Q8) Use the string formatting method to display the following:

radius = 10

area = 3.14 * radius **2

Using String Format method

SFM = "the area of a circle with radius {} is {} meters

square.".format(radius, area)

print(SFM)

MACHINE LEARNING ASSIGNMENT 1

```
In [60]: #Question 8
         radius = 10
         area = 3.14 * radius **2
         # Using String Format method
         SFM = "the area of a circle with radius {} is {} meters square.".format(radius, area)
         print(SFM)
```

```
the area of a circle with radius 10 is 314.0 meters square.
```

Q9) Write a program, which reads weights (lbs.) of N students into a list and convert these weights to kilograms in a separate list using Loop. N: No of students (Read input from user)

```
# N number of students
N = int(input("Enter Number of students : "))
# lb to kg conversion value
convertion_value = 0.4536
lbs = []
kgs= []

# Students Weight in lbs
for i in range(0,N):
    lbs.append(int(input("Enter student Weight in lbs : ")))

print("lbs: ",lbs)

# converted weights from lbs to kg
for weight in lbs:
    kgs.append(round(weight * convertion_value,2))

print("weight in kgs : ", kgs)
```

```
Enter Number of students : 3
Enter student Weight in lbs : 150
Enter student Weight in lbs : 155
Enter student Weight in lbs : 160
lbs:  [150, 155, 160]
weight in kgs :  [68.04, 70.31, 72.58]
>
```

10)

f	1	2	3	6	6	7	10	11
label	1	1	0	0	0	1	1	1

Test set

Train set

(i) using KNN classifier where k=3

$d = \sqrt{(x-x_1)^2}$

$(6,6)$ $(6,3)$ $(6,2)$ $(6,1)$ are the points need to be calculated

i.e;

$d = \sqrt{(6-6)^2} = 0$ $(6,6)$

$d = \sqrt{(6-3)^2} = \sqrt{9} = 3$ $(6,3)$

$d = \sqrt{(6-2)^2} = \sqrt{4^2} = 4$ $(6,2)$

$d = \sqrt{(6-1)^2} = \sqrt{5^2} = 5$ $(6,1)$

} Nearest

i.e, (0,0,1)

max = 0 (output also 0)

Calculating for rest points and the predicted values is 0.

(ii) confusion matrix

Accuracy = $(TP+TN)/(TN+FP+FN+TP)$

sensitivity = $TP/(TP+FN)$

specificity = $TN/(FP+TN)$

0	TN=1	FP=0
1	FN=3	TP=0

$A = (0+1)/(1+0+3+0)$

$= 1/4 \Rightarrow 25\%$.

$S = 0/(0+3) = 0$

$SP = 1/(0+1) = 1$