

MACHINE LEARNING ASSIGNMENT - 1

NAME : SUPRIYA SAMA

700744510

Video link -

https://drive.google.com/file/d/1JOBymeNVUTv6n-HmtKmjPVoFPHBABT7o/view?usp=share_link

1. For the given list sorting can be done by using sort() function in ascending order by default. min(), max(), append() functions can be used to find minimum, maximum and to add the ages to the list. Median can be find by using ages[(length+1)//2]. Average of the ages can be calculated by dividing sum of all ages by the length. Range can be calculated by subtracting minimum age from maximum age.

The screenshot shows a Jupyter Notebook window titled "jupyter ML- Assignment 1.1". The URL in the address bar is "localhost:8888/notebooks/ML-%20Assignment%201.ipynb". The notebook has a single cell labeled "In [4]:" containing the following Python code:

```
#given List of ages
ages=[19,22,19,24,20,25,26,24,25,24]
#sorting the list
ages.sort()
#finding min and max age
minAge = min(ages)
maxAge = max(ages)
#adding min and max ages to list
ages.append(minAge)
ages.append(maxAge)
ages.sort()
length = len(ages)
#finding median age
medianAge = ages[ (length+1)//2 ] if length%2==1 else (ages[length//2]+ages[(length//2)+1])/2
#finding average age
sumofAges = sum(ages)
averageofAges = sumofAges//length
#finding range of ages
rangeofAges = maxAge-minAge
#printing the results
print(minAge)
print(maxAge)
print(medianAge)
print(averageofAges)
print(rangeofAges)
```

The code defines a list of ages, sorts it, and then calculates the minimum, maximum, median, average, and range of the ages. The results are printed at the end. The notebook is running in a Python 3 kernel.

A screenshot of a Jupyter Notebook window titled "jupyter ML- Assignment 1.1". The code cell contains the following Python script:

```
sumofAges = sum(ages)
averageofAges = sumofAges/len(ages)
#finding range of ages
rangeofAges = maxAge-minAge
#printing the results
print(minAge)
print(maxAge)
print(medianAge)
print(averageofAges)
print(rangeofAges)

19
26
24.0
22
7
```

The notebook interface includes a toolbar with file operations like File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a code editor with various icons. The status bar at the bottom shows system information like weather, battery, and date.

2. Create dog dictionary using dict() and add the given features to the dictionary and print those values. Create student dictionary by using dict() and add given details to the dictionary and print the results. Length can be calculated by using len(student). Data type can be checked using type(). Add the skills by using append() function. To get dictionary keys as list use list(dog.keys()) and use list(dog.values()) for dictionary values.

A screenshot of a Jupyter Notebook window titled "jupyter ML-Assignment 1.2". The code cell contains the following Python script:

```
#creating dictionary for dog
dog = dict()
#adding name, color, breed, legs, age to the dictionary
dog["name"]='romeo'
dog["color"]='white'
dog["breed"]='shihzu'
dog["legs"]=4
dog["age"]=6
#printing dog dictionary
print("Dog dictionary is : ",dog)
print()

#creating student dictionary
student = dict()
#adding student information
student["first_name"]='supriya'
student["last_name"]='sama'
student["gender"]='female'
student["age"]=21
student["marital status"]='single'
student["skills"]=['python','java','AWS']
student["country"]='india'
student["city"]='Nizamabad'
student["address"]='street no 4'
#printing student dictionary
print("student dictionary is : ",student)
```

The notebook interface includes a toolbar with file operations like File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a code editor with various icons. The status bar at the bottom shows system information like weather, battery, and date.

```

printing student dictionary
print("student dictionary is : ",student)

printing the length of student dictionary
print("length of student dictionary is : ",str(len(student)))

value of student skills and datatype
print("student skills are : ",end=" ")
print(student["skills"])
print("type of student skills is : ",type(student["skills"]))

adding two skills
student["skills"].append("numpy")
student["skills"].append(".net")
printing modified student skills
print("modified student skills : ",student["skills"])

getting dictionary keys as list
dog_keys=list(dog.keys())
print("dog dictionary keys : ",dog_keys)
student_keys=list(student.keys())
print("student dictionary keys : ",student_keys)
getting dictionary values as a list
dog_values=list(dog.values())
print("dog dictionary values : ",dog_values)
student_values=list(student.values())
print("student dictionary values : ",student_values)

```

```

getting dictionary keys as list
dog_keys=list(dog.keys())
print("dog dictionary keys : ",dog_keys)
student_keys=list(student.keys())
print("student dictionary keys : ",student_keys)
getting dictionary values as a list
dog_values=list(dog.values())
print("dog dictionary values : ",dog_values)
student_values=list(student.values())
print("student dictionary values : ",student_values)

Dog dictionary is : {'name': 'romeo', 'color': 'white', 'breed': 'shihtzu', 'legs': 4, 'age': 6}

student dictionary is : {'first_name': 'supriya', 'last_name': 'sama', 'gender': 'female', 'age': 21, 'marital status': 'single', 'skills': ['python', 'java', 'AWS'], 'country': 'india', 'city': 'Nizamabad', 'address': 'street no 4'}
length of student dictionary is : 9
student skills are : ['python', 'java', 'AWS']
type of student skills is : <class 'list'>
modified student skills : ['python', 'java', 'AWS', 'numpy', '.net']
dog dictionary keys : ['name', 'color', 'breed', 'legs', 'age']
student dictionary keys : ['first_name', 'last_name', 'gender', 'age', 'marital status', 'skills', 'country', 'city', 'address']
dog dictionary values : ['romeo', 'white', 'shihtzu', 4, 6]
student dictionary values : ['supriya', 'sama', 'female', 21, 'single', ['python', 'java', 'AWS', 'numpy', '.net'], 'india', 'Nizamabad', 'street no 4']

```

3. Create a separate tuples for brothers and sisters containing names. Join both the tuples and assign it to siblings. Length can be find using len() function. Add father and mother names to the siblings by using '+' and assign the result to the family_members.

The screenshot shows a Jupyter Notebook interface running on a Windows desktop. The notebook has multiple tabs open, all titled 'ML Assignment'. The active cell (In [3]) contains Python code that defines tuples for brothers and sisters, joins them into a single tuple for siblings, and adds parents to the family. The output of the code is displayed below the code cell.

```
#creating a tuple containing names of sisters
sisters=("suchi","sony","aishu","raji")
#creating a tuple containing names of brothers
brothers=("aryan","karthik")
#printing brothers names
print("brothers are : ",brothers)
#printing sisters names
print("sisters are : ",sisters)

#joining brothers and sisters
siblings=brothers+sisters
#printing sibling names
print("siblings are : ",siblings)
#printing number of siblings
print("number of siblings are : ",len(siblings))

#adding names of father and mother
family_members=("prathap","savitha")*siblings
#printing family members names
print("family members are : ",family_members)

brothers are : ('aryan', 'karthik')
sisters are : ('suchi', 'sony', 'aishu', 'raji')
siblings are : ('aryan', 'karthik', 'suchi', 'sony', 'aishu', 'raji')
number of siblings are : 6
family members are : ('prathap', 'savitha', 'aryan', 'karthik', 'suchi', 'sony', 'aishu', 'raji')
```

4. Length of `it_companies` can be find using `len()` function. Add ‘Twitter’ to the set by using `it_companies.add("Twitter")`. We can insert and remove companies from the set by using `update(companies)` and `remove(companies)`. The `remove()` method will raise an error if the specified item does not exist whereas `discard()` method does not. We use union, intersection operators to join and to find intersections. We check if the A and B are subset or disjoint sets by using `A.issubset(B)` and `A.isdisjoint(B)`. We join the sets using union operator. To delete the sets we use `clear()` method. Convert list to set by using `age_set = set(age)`.

In [3]:

```
#given List of it companies
it_companies = {"Facebook", "Google", "Microsoft", "Apple", "IBM", "oracle", "Amazon"}
A = {19, 22, 24, 20, 25, 26}
B = {19, 22, 20, 25, 26, 24, 28, 27}
age = [22, 19, 24, 25, 26, 24, 25, 24]
#printing the length of it companies
print("length of it_companies : ",len(it_companies))
#adding twitter to it companies
it_companies.add("Twitter")
#printing List after adding twitter to it companies
print("it_companies after appending twitter : ",it_companies)

#inserting the multiple it companies at once to the set it_companies
companies = ["NCR", "Wipro", "TCS"]
it_companies.update(companies)
#printing List of it companies after appending multiple companies
print("it_companies after appending multiple companies : ",it_companies)

#removing one company from the set
it_companies.remove("oracle")
#printing the List after removing one company
print("it_companies after removing one company : ", it_companies)

#Difference between remove and discard method
```

2°C Rain and snow

localhost:8888/notebooks/ML%20Assignment%201.ipynb

jupyter ML Assignment 1.4 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) Logout

File + % Run ▶ Code

```
#Difference between remove and discard method
#The remove() method will raise the error if the specified item does not exist in the list whereas discard() method will not.

#joining A and B
C = A.union(B)
#printing the results
print("joining A and B is : ",C)

#finding intersection of A and B
I = A.intersection(B)
#printing the results
print("Intersection pf A and B is : ", I)

#checking if A is subset of B
check = A.issubset(B)
if check:
    print("A is subset of B")
else:
    print("A is not a subset of B")

#checking if A and B are disjoint sets
check1 = A.isdisjoint(B)
if check1:
    print("A and B are disjoint sets")
else:
    print("A and B are not disjoint sets")
```

2°C Rain and snow

localhost:8888/notebooks/ML%20Assignment%201.ipynb

jupyter ML Assignment 1.4 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) Logout

File + % Run ▶ Code

The screenshot shows a Jupyter Notebook window titled "jupyter ML Assignment 1.4". The code cell contains the following Python script:

```
else:  
    print("A and B are not disjoint sets")  
  
#joining A with B  
A_join_B = A.union(B)  
#joining B with A  
B_join_A = B.union(A)  
#printing the results  
print("A join B is :", A_join_B)  
print("B join A is :", B_join_A)  
  
#symmetric difference between A and B  
D = A.symmetric_difference(B)  
print("symmetric difference between A and B is :", D)  
  
#deleting all the sets completely  
it_companies.clear()  
A.clear()  
B.clear()  
  
#converting ages List to set  
age_set = set(age)  
#printing the length of the list  
print("length of ages list is : ", len(age))  
#printing the length of set  
print("length of ages set is : ", len(age_set))  
print("length of ages list is greater than the ages set")
```

The screenshot shows the same Jupyter Notebook window after running the code. The output cell displays the results of the operations:

```
length of it_companies : 7  
it_companies after appending twitter : {'Facebook', 'Google', 'Amazon', 'Microsoft', 'oracle', 'Twitter', 'IBM', 'Apple'}  
it_companies after appending multiple companies : {'Facebook', 'IBM', 'oracle', 'NCR', 'Twitter', 'TCS', 'Google', 'Amazon', 'Microsoft', 'Wipro', 'Apple'}  
it_companies after removing one company : {'Facebook', 'IBM', 'NCR', 'Twitter', 'TCS', 'Google', 'Amazon', 'Microsoft', 'Wipro', 'Apple'}  
joining A and B is : {19, 20, 22, 24, 25, 26, 27, 28}  
Intersection pf A and B is : {19, 20, 22, 24, 25, 26}  
A is subset of B  
A and B are not disjoint sets  
A join B is : {19, 20, 22, 24, 25, 26, 27, 28}  
B join A is : {19, 20, 22, 24, 25, 26, 27, 28}  
symmetric difference between A and B is : {27, 28}  
length of ages list is : 8  
length of ages set is : 5  
length of ages list is greater than the ages set
```

5. Area of circle can be calculated using $3.14*(r**2)$ and assign this value to variable `_area_of_circle_`. Circumference can be calculated by using $2*3.14*r$ and assign this value to variable `_circum_of_circle_`. We can calculate the area of circle for user input by using $3.14*(n**2)$.

The screenshot shows a Jupyter Notebook interface with multiple tabs open, all titled "ML Assignment 1.5". The active cell (In [6]) contains Python code to calculate the area and circumference of a circle, both with a radius of 30 meters, and then to calculate the area with user input. The output shows the results for each case.

```
In [6]: #area of circle  
#Given radius of circle 30 meters  
r=30  
#assign variable name "_area_of_circle"  
_area_of_circle=3.14*(r**2)  
#printing the results  
print("area of circle :", _area_of_circle)  
  
#circumference of circle  
#assign variable name "_circum_of_circle"  
_circum_of_circle=2*3.14*r  
#printing the results  
print("circumference of circle :",_circum_of_circle)  
  
#area of circle for user input  
#taking user input  
n=int(input("Enter the value "))  
#assign variable name "area"  
area=3.14*(n**2)  
#printing the results for user input  
print("Area of circle with user input :",area)  
  
area of circle : 2826.0  
circumference of circle : 188.4  
Enter the value 23  
Area of circle with user input : 1661.060000000002
```

6. Assign given string to the variable 'a' and split the given string using a.split(). Use set to remove duplicates. Count the number of words using len() function and print the values.

The screenshot shows a Jupyter Notebook interface with multiple tabs open, all titled "ML Assignment 1.6". The active cell (In [2]) contains Python code to assign a given string to variable 'a', split it into words using .split(), use a set to remove duplicates, and then count the words using len(). The output shows the result as 10.

```
In [2]: #assign variable to the given string  
a="I am a teacher and I love to inspire and teach people"  
#split the given string  
b=a.split()  
#using set to remove duplicates  
set1=set(b)  
#using len to count number of words  
count=len(set1)  
#printing the result  
print(count)
```

In []: |

7. Assign given string to the variable ‘txt’. Use ‘/’ to give space between the string. Print the output.

The screenshot shows a Jupyter Notebook window titled "jupyter ML Assignment 1.7". In the code cell (In [6]), the following Python code is written:

```
In [6]: #tab escape
#declaring a string to the txt variable
txt = "Name \tAge\tCountry \tCity\nAsabeneh\t250\tFinland \tHelsinki"
#printing the results
print(txt)
```

The output cell shows the printed string:

Name	Age	Country	City
Asabeneh	250	Finland	Helsinki

The system tray at the bottom indicates it's raining at 1°C, the date is 1/21/2023, and the time is 4:55 PM.

8. Assign given value to the variable ‘radius’. We can find the area of the circle by using $3.14 * radius ** 2$. Print the output using format().

The screenshot shows a Jupyter Notebook window titled "jupyter ML Assignment 1.8". In the code cell (In [2]), the following Python code is written:

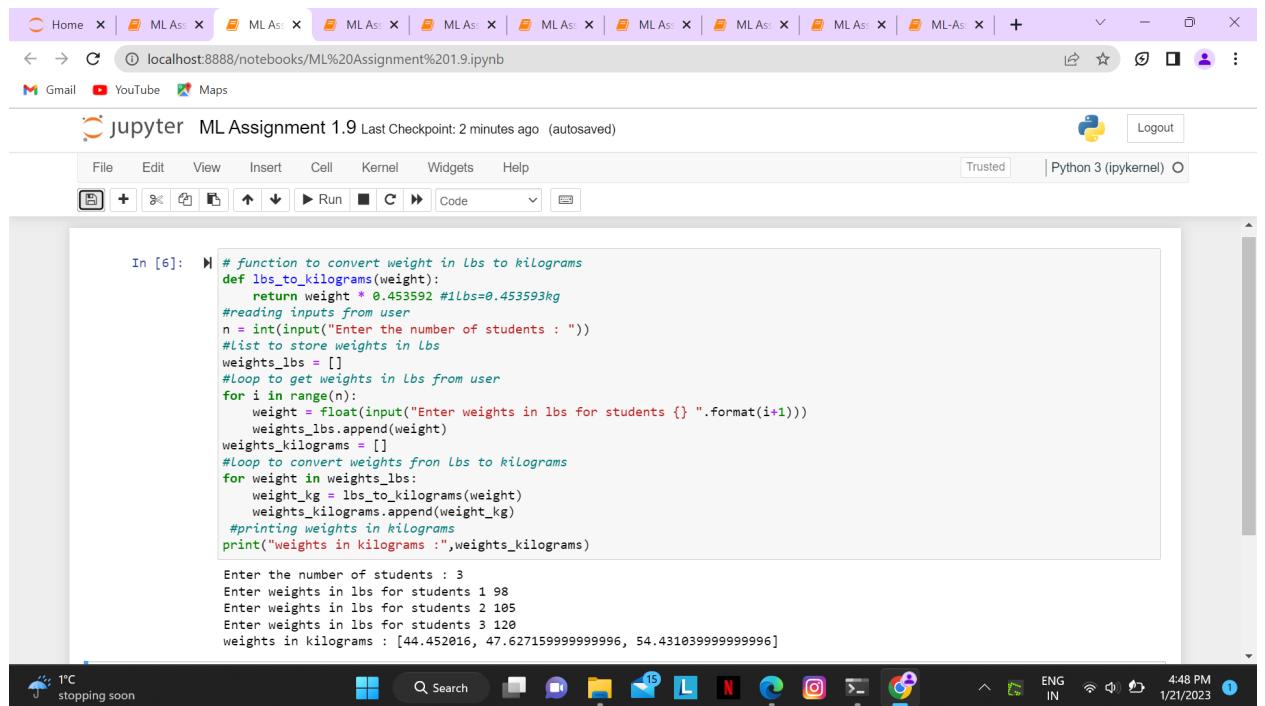
```
In [2]: #assigned variable radius
radius=10
#assigned area of circle formula
area=3.14 * radius ** 2
#printing the output using string format
print("The area of circle with radius {} is {} meters square".format(radius,area))
```

The output cell shows the printed string:

The area of circle with radius 10 is 314.0 meters square

The system tray at the bottom indicates it's raining at 1°C, the date is 1/21/2023, and the time is 4:53 PM.

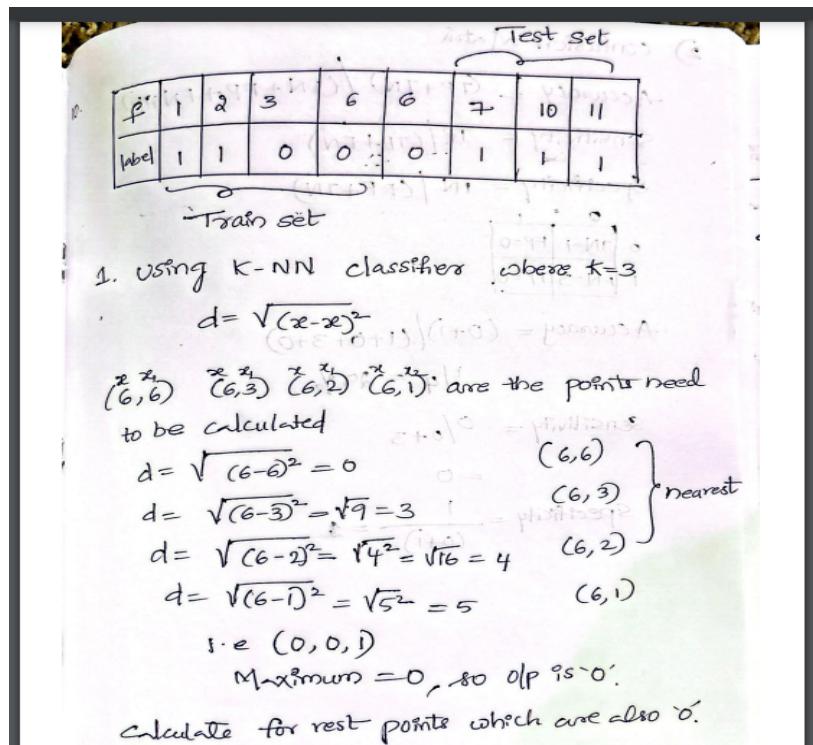
9. Assign input to the variable 'n'. Use for loop for appending the elements into the list. Convert weights from lbs to kilograms using for loop.



```
In [6]: # function to convert weight in lbs to kilograms
def lbs_to_kilograms(weight):
    return weight * 0.453592 #1lbs=0.453593kg
#reading inputs from user
n = int(input("Enter the number of students : "))
#list to store weights in lbs
weights_lbs = []
#loop to get weights in lbs from user
for i in range(n):
    weight = float(input("Enter weights in lbs for students {} ".format(i+1)))
    weights_lbs.append(weight)
weights_kilograms = []
#loop to convert weights from lbs to kilograms
for weight in weights_lbs:
    weight_kg = lbs_to_kilograms(weight)
    weights_kilograms.append(weight_kg)
#printing weights in kilograms
print("weights in kilograms :",weights_kilograms)

Enter the number of students : 3
Enter weights in lbs for students 1 98
Enter weights in lbs for students 2 105
Enter weights in lbs for students 3 120
weights in kilograms : [44.452016, 47.627159999999996, 54.43103999999999]
```

- 10.



	1	2	3	4	5	6	7	8	9	10	11
label	1	1	0	0	0	0	1	1	1	1	1

Test set

Train set

1. Using K-NN classifiers where $k=3$

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

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

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

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

$$d = \sqrt{(6-2)^2 + (6-2)^2} = \sqrt{16} = 4$$

$$d = \sqrt{(6-1)^2 + (6-1)^2} = \sqrt{52} = 5$$

$x = (0,0,1)$
 $\text{Maximum} = 0, \text{ so O/P is } 0.$

Calculate for rest points which are also 0.

2) confusion matrix

$$\text{Accuracy} = \frac{TP+TN}{(TN+FP+FN+TP)}$$

$$\text{Sensitivity} = \frac{TP}{(TP+FN)}$$

$$\text{Specificity} = \frac{TN}{(FP+TN)}$$

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

$$\text{Accuracy} = \frac{(0+1)}{(1+0+3+0)} = \frac{1}{4} = 25\%$$

↳ 0.25 = 25% (0.25)

$$\text{Sensitivity} = \frac{0}{0+3} = 0$$

$$\text{Specificity} = \frac{1}{(0+1)} = \frac{1}{1} = 1$$

$$1 = \frac{1}{1} = 1$$

$$(1, 0, 0) = 1$$