# MACHINE LEARNING (CS-5710)

# ASSIGNMENT - 1

**Name: Neeraj Kumar Kajuluri**

**Student ID: 700742091**

## Git hub Link :- https://github.com/NeerajKumarKajuluri/ML-Assigment-1

## Video Link: - https://drive.google.com/file/d/16KtKN4GSKquvz93K0niOB-td_22_KbGa/view?usp=share_link

### QUESTION 1

The following is a list of 10 students ages:
ages = [19, 22, 19, 24, 20, 25, 26,
24,25,24]

• Sort the list and find the min and max age

• Add the min age and the max age again to the list

• Find the median age (one middle item or two middle items divided by two)

• Find the average age (sum of all items divided by their number)

• Find the range of the ages (max minus min)

**Source code & Outputs**

```
In [24]: #Sort the list and find the min and max age
         ages=[19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
         ages.sort() #Sorting the list using sort() method
         print("the sorted listed of ages: ",ages)

         min_age=min(ages) #minimum of ages in the given list
         max_age=max(ages) #maximum of ages in the given list

         print("minimum age: ",min_age)
         print("maximum age: ",max_age)

         the sorted listed of ages:  [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
         minimum age:  19
         maximum age:  26
```

In the code sample above, we established a list called "ages" and added some members to it. Then we used the built-in method sort, which by defaults sorts list values in ascending order. The other built-in functions minimum and maximum were also utilized to determine the minimum and maximum elements in the supplied list.

In [25]:
```
#Add the min age and the max age again to the list
print("original length: ",len(ages))
ages.append(min_age) #appending minimum age again to the list of ages
ages.append(max_age) #appending maximum age again to the list of ages

print("Length after appending: ",len(ages))
```

```
original length:  10
Length after appending:  12
```

In this code snippet we used the append() method in list methods to append the minimum and maximum ages into the list and we printed the length of the new list after appending the new elements.

In [3]:
```
# Find the median age (one middle item or two middle items divided by two)
print("ages: ",ages)
length=len(ages)
if(length%2==0):
    #find one of the two middle indices of the list
    ind=length//2-1
    #calculating median
    med=(ages[ind]+ages[ind+1])/2
else:
    #in the case of dd length, the median is just the middle element
    med=ages[length//2]
print("median: ",med)
```

```
ages:  [19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]
median:  24.0
```

Using the if-else conditional statements, we calculated the list's median and printed it.

In [4]:
```
#Find the average age (sum of all items divided by their number)

total=sum(ages) #find the sum of all eages in the list usinf sum() function
avg=total/length
print("average: ",avg)
```

```
average:  22.75
```

In [5]:
```
#Find the range of the ages (max minus min)
range=max(ages)-min(ages)
print("Range: ",range)
```

```
Range:  7
```

In this snippet of code, we found the sum of the elements in the supplied list using the sum() method, then we computed the average. The range of the list was determined in the subsequent snippet.

## Question 2

• Create an empty dictionary called dog

• Add name, colour, breed, legs, age to the dog dictionary

• Create a student dictionary and add first name, last name, gender, age, marital status, skills, country, city and address as keys for the dictionary

• Get the length of the student dictionary

• Get the value of skills and check the data type, it should be a list

• Modify the skills values by adding one or two skills

• Get the dictionary keys as a list

• Get the dictionary values as a list

**Source code & Output**

```
In [6]: #Create an empty dictionary called dog
        dog=dict() #intializing an empty dictionary
        print("length: ",len(dog))

        length:  0

In [7]: #Add name, color, breed, legs, age to the dog dictionary
        dog["name"]="snoopy"
        dog["color"]="black"
        dog["breed"]="poodle"
        dog["legs"]=4
        dog["age"]=2

        print(dog)

        {'name': 'snoopy', 'color': 'black', 'breed': 'poodle', 'legs': 4, 'age': 2}

In [8]: ''' Create a student dictionary and add first_name, last_name, gender,
            age, marital status, skills, country, city and address as keys for the dictionary '''
        #initializing empty student dictionary
        student={}

        #adding irst_name, last_name, gender, age, marital status, skills, country, city
        student["first_name"]="Neeraj Kumar"
        student["last_name"]="kajuluri"
        student["gender"]="Male"
        student["age"]=22
        student["Marital_status"]="Unmarried"
        student["Skills"]=["C","C++","Java","Python","Cyber Security","Machine Learning","Cloud Computing"]
        student["country"]="India"
        student["City"]="Tanuku"
        student["address"]="Tanuku"

        print("Student details: ")
        print(student)

        Student details:
        {'first_name': 'Neeraj Kumar', 'last_name': 'kajuluri', 'gender': 'Male', 'age': 22, 'Marital_status': 'Unmarried', 'Skills':
        ['C', 'C++', 'Java', 'Python', 'Cyber Security', 'Machine Learning', 'Cloud Computing'], 'country': 'India', 'City': 'Tanuku',
        'address': 'Tanuku'}
```

This code sample uses the dictionary function to initialize a new dictionary before adding keys and values to it. Then we printed a second dictionary that contained student information as keys and values.

```
In [9]: #Get the length of the student dictionary
dict_len=len(student)
print("Student length: ",dict_len)

Student length:  9

In [10]: #Get the value of skills and check the data type, it should be a list
student_skills=student["Skills"] #Getting the value of skills of student
print(student_skills)
skills_type=type(student_skills) #getting the data type of skills key
print(skills_type)

['C', 'C++', 'Java', 'Python', 'Cyber Security', 'Machine Learning', 'Cloud Computing']
<class 'list'>

In [11]: #Modify the skills values by adding one or two skills
student["Skills"].append("HTML") #appending a new skill to the value of skills of student
print(student["Skills"])
student["Skills"].append("CSS")
print(student["Skills"])

['C', 'C++', 'Java', 'Python', 'Cyber Security', 'Machine Learning', 'Cloud Computing', 'HTML']
['C', 'C++', 'Java', 'Python', 'Cyber Security', 'Machine Learning', 'Cloud Computing', 'HTML', 'CSS']

In [12]: #Get the dictionary keys as a list
keys_list=[] #initializing an empty list to store keys
for k,v in student.items():
    keys_list.append(k)
print("The keys are: ")
print(keys_list)

The keys are:
['first_name', 'last_name', 'gender', 'age', 'Marital_status', 'Skills', 'country', 'City', 'address']

In [13]: #Get the dictionary values as a lisT
values_list=[] #initializing an empty list to store values
for k,v in student.items():
    values_list.append(v)
print("The values are: ")
print(values_list)

The values are:
['Neeraj Kumar', 'kajuluri', 'Male', 22, 'Unmarried', ['C', 'C++', 'Java', 'Python', 'Cyber Security', 'Machine Learning', 'Clo
ud Computing', 'HTML', 'CSS'], 'India', 'Tanuku', 'Tanuku']
```

In this snippet of code, the length of the dictionary was determined using the length function, and the datatype of the skills key was determined using the type() function. The skills key was then updated by adding certain values to that. The dictionary values were then printed after the dictionary keys.

## Question 3

- Create a tuple containing names of your sisters and your brothers (imaginary siblings are fine)
- Join brothers and sister's tuples and assign it to siblings
- How many siblings do you have?
- Modify the sibling's tuple and add the name of your father and mother and assign it to family member.

**Source code & Output**

In [14]:
```
#Create a tuple containing names of your sisters and your brothers (imaginary siblings are fine)

brothers=() #initializing an empty tuple to store brothers
sisters=() #initializing an empty tuple to store sisters

#adding elemets to brothers
brothers = brothers+("Siva",)
brothers += ("Lokesh",)
brothers += ("Bharath",)
brothers += ("Kishore",)

print("type: ",type(brothers))
print("Brothers: ")
print(brothers)

print()
#adding elements to sisters
sisters+=("Likhitha","Lahari","Vinitha","Sravya",)

print("type: ",type(sisters))
print("Sisters: ")
print(sisters)
```

```
type:  <class 'tuple'>
Brothers:
('Siva', 'Lokesh', 'Bharath', 'Kishore')

type:  <class 'tuple'>
Sisters:
('Likhitha', 'Lahari', 'Vinitha', 'Sravya')
```

We formed two tuples with this code and then added the values to the tuples. The other tuple was then given new elements, and it was given back to the original tuple. We added values to the tuples using the compound assignment operator since tuples are immutable.

In [15]:
```
#Join brothers and sisters tuples and assign it to siblings
siblings=() #Initializing an empty tuple
siblings=(brothers+sisters) #concatenating two tuples using '+' operator
print("type: ",type(siblings))
print("siblings: ")
print(siblings)
```

```
type:  <class 'tuple'>
siblings:
('Siva', 'Lokesh', 'Bharath', 'Kishore', 'Likhitha', 'Lahari', 'Vinitha', 'Sravya')
```

In [16]:
```
#How many siblings do you have?
print("total no.of siblings: ",len(siblings))
```

```
total no.of siblings:  8
```

In [17]:
```
#Modify the siblings tuple and add the name of your father and mother and assign it to family_member
family_member=() #initializing an empty tuple
family_member += siblings + ("Veera bhadra rao","Kalpana")

print("type: ",type(family_member))
print("family member: ")
print(family_member)
```

```
type:  <class 'tuple'>
family member:
('Siva', 'Lokesh', 'Bharath', 'Kishore', 'Likhitha', 'Lahari', 'Vinitha', 'Sravya', 'Veera bhadra rao', 'Kalpana')
```

In this code we created two more tuples and added them to the family member tuple finally and printed it.

## QUESTION 4

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]

- Find the length of the set it companies
- Add 'Twitter' to it companies
- Insert multiple IT companies at once to the set it companies

• Remove one of the companies from the set it companies
• What is the difference between remove and discard
• Join A and B
• Find A intersection B
• Is A subset of B
• Are A and B disjoint sets
• Join A with B and B with A
• What is the symmetric difference between A and B
• Delete the sets completely
• Convert the ages to a set and compare the length of the list and the set.

**Source code & Output**

In [18]:
```
#Find the length of the set it_companies
it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
print("length: ",len(it_companies))
```

```
length:  7
```

In [19]:
```
#Add 'Twitter' to it_companies
it_companies.add('Twitter')
print("length: ",len(it_companies))
print(it_companies)
```

```
length:  8
{'IBM', 'Oracle', 'Amazon', 'Google', 'Twitter', 'Microsoft', 'Facebook', 'Apple'}
```

In [53]:
```
#Insert multiple IT companies at once to the set it_companies
dup_set={'Adobe','HSBC','EPAM','Infosys'} #initializing a set
it_companies.update(dup_set) #adding the second set

print(it_companies)
```

```
{'Google', 'Oracle', 'HSBC', 'Apple', 'Infosys', 'IBM', 'Twitter', 'Amazon', 'EPAM', 'Facebook', 'Adobe', 'Microsoft'}
```

In [54]:
```
#Remove one of the companies from the set it_companies
it_companies.remove('EPAM') #removing the company 'EPAM'
print(it_companies)
```

```
{'Google', 'Oracle', 'HSBC', 'Apple', 'Infosys', 'IBM', 'Twitter', 'Amazon', 'Facebook', 'Adobe', 'Microsoft'}
```

In [66]:
```
#Join A and B
A = {19, 22, 24, 20, 25, 26}
B = {19, 22, 20, 25, 26, 24, 28, 27}
A.union(B) #all unique elements and all common elements will be printed once
B.union(A)
print("A: ",A)
print("B: ",B)
```

```
A:  {19, 20, 22, 24, 25, 26}
B:  {19, 20, 22, 24, 25, 26, 27, 28}
```

In this code snippet we calculated the length of the set and added some more elements to the sets and printed the new set. Then we added multiple values into the set at a once and printed it again. Then we checked the remove function from sets built-in functions and printed the resultant set. After that we applied the set operation called union and printed the resultant.

```
In [67]: #Find A intersection B
         A = {19, 22, 24, 20, 25, 26}
         B = {19, 22, 20, 25, 26, 24, 28, 27}
         int_Set=A.intersection(B)
         print("The common elements are: ")
         print(int_Set)

         The common elements are:
         {19, 20, 22, 24, 25, 26}

In [68]: #Is A subset of B
         res=A.issubset(B)
         print(res) #returns true or false

         True

In [69]: #Are A and B disjoint sets
         res1=A.isdisjoint(B)
         if res1==True:
             print("A and B are dijoint")
         else:
             print("No")

         No

In [70]: #Join A with B and B with A
         A = {19, 22, 24, 20, 25, 26}
         B = {19, 22, 20, 25, 26, 24, 28, 27}
         A.union(B) #all unique elements and all common elements will be printed once
         B.union(A)
         print("A: ",A)
         print("B: ",B)

         A:  {19, 20, 22, 24, 25, 26}
         B:  {19, 20, 22, 24, 25, 26, 27, 28}

In [71]: #What is the symmetric difference between A and B
         sym_dif=A.symmetric_difference(B)
         print(sym_dif)

         {27, 28}
```

In this code snippet we used the other set functions such as intersection, subset, disjoint and set symmetric difference and printed the results.

```
In [72]: # Delete the sets completely
         A.clear() #deleting setA completelt from memory
         B.clear() ##deleting setA completelt from memory
         print(A)
         print(B)

         set()
         set()

In [74]: #Convert the ages to a set and compare the length of the list and the set.
         set_age=set(ages)
         print("List ages length: ",len(ages))
         print("set ages length: ",len(set_age))

         #the lengths are differnet because the set removes the duplicated ages

         List ages length:  10
         set ages length:  6
```

Here in this snippet, we deleted the sets completely and printed the empty sets and then converted the list into sets using the set function and proved that the sets will not allow duplicates.

## QUESTION 5

The radius of a circle is 30 meters.

• Calculate the area of a circle and assign the value to a variable name of *area of circle*

• Calculate the circumference of a circle and assign the value to a variable name of circumcircle

• Take radius as user input and calculate the area.

**Source code & Output**

```
In [75]: #Calculate the area of a circle and assign the value to a variable name of area_of_circle
         #initializing radius ad rad
         rad=30
         #calcluating area of circle
         area_of_circle=3.14*rad*rad
         print("Area: ",area_of_circle)

         Area:  2826.0

In [76]: #Calculate the circumference of a circle and assign the value to a variable name of circum_of_circle
         _circum_of_circle=2*3.14*rad
         print("Circumference: ",_circum_of_circle)

         Circumference:  188.4

In [22]: rad_inp=int(input("Enter radius"))
         new_area=3.14*rad_inp*rad_inp
         print("Area from user input radius: ",new_area)

         Enter radius5
         Area from user input radius:  78.5
```

In this code snippet we calculated the area of the circle and then circumference of the circle, by taking the radius as user input using the input function.

## QUESTION 6

“I am a teacher and I love to inspire and teach people”

• How many unique words have been used in the sentence? Use the split methods and set to get the unique words.

**Source code & Output**

```
In [79]: s="I am a teacher and I love to inspire and teach people" #initializing a string
         count=0 #initializing the count of unique wors to 0
         unique_s=s.split() #using split to split each word in the given string
         for word in unique_s:
             if s.count(word)==1: #getting the count of each word in th estring
                 count+=1
         print("total no.of unique words: ",count)

         total no.of unique words:  6
```

In this code snippet we used the split method to split the given string at each space character. Then we compared each word with all the other words in the string and counted the number of unique words in the given string.

## QUESTION 7

Use a tab escape sequence to get the following lines.

**Name Age Country City**

**Asabeneh 250 Finland Helsinki**

**Source code & Output**

```
In [80]: #\t escape sequence
         print("Name\tAge\tCountry\tCity")
         print("Asabensh\t250\tFinland\tHelsinki")

         Name    Age     Country City
         Asabensh        250     Finland Helsinki
```

In this code snippet we used '\t' the escape character to get 3 spaces between the words.

## QUESTION 8

Use the string formatting method to display the following:
radius = 10
area = 3.14 * radius ** 2
"The area of a circle with radius 10 is 314 meters square."

**Source code & Output**

```
In [83]: radius=10 #initializing radius to 10
         area=3.14*radius**2
         print("The radius of a circle with radius {} is {} meters square".format(radius,int(area)))

         The radius of a circle with radius 10 is 314 meters square
```

## QUESTION 9

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)

Ex: L1: [150, 155, 145, 148]

Output: [68.03, 70.3, 65.77, 67.13]

**Source code & Output**

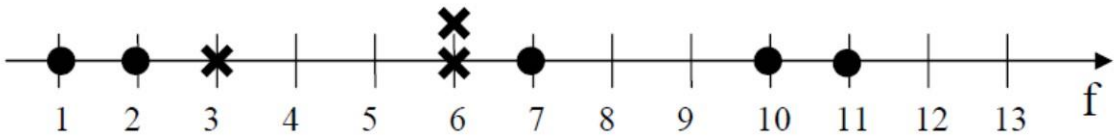
```
In [97]: n=int(input()) #initializing n with user input
         wei_g=list(map(int,input("Enter weights").split()))
         wei_kg=[]
         for x in wei_g:
             wei_kg.append(x*0.453592)
         print(wei_kg)

         4
         Enter weights150 155 145 148
         [68.0388, 70.30676, 65.77083999999999, 67.131616]
```

In this code snippet we converted a list of values in lbs to kilograms by taking the list elements from user input.

## QUESTION 10

The diagram below shows a dataset with 2 classes and 8 data points, each with only one feature value, labeled f. Note that there are two data points with the same feature value of 6. These are shown as two x's one above the other.




1. Divide this data equally into two parts. Use first part as training and second part as testing. Using KNN classifier, for K=3, what would be the predicted outputs for the test samples? Show how you arrived at your answer.
2. Compute the confusion matrix for this and calculate accuracy, sensitivity and specificity values.

**Solution:**

Given data elements are taken in the tabular form as below,

| **Feature** | **Label** |
|---|---|
| 1 | O |
| 2 | O |
| 3 | X |
| 6 | X |
| 6 | X |
| 7 | O |
| 10 | O |

| | |
|---|---|
| 11 | O |

Here, the first four rows of data are considered to be the Training dataset and the next four rows are selected as the Testing dataset.

Now, according to the KNN Classifier we shall now consider K=3 and then the distance between the testing and training data is demonstrated below.
In the below table the columns are the training dataset and rows are the testing dataset.

| | 1(O) | 2(O) | 3(X) | 6(X) |
|---|---|---|---|---|
| 6 | 5 | **4** | **3** | **0** |
| 7 | 6 | **5** | **4** | **1** |
| 10 | 9 | **8** | **7** | **4** |
| 11 | 10 | **9** | **8** | **5** |

The highlighted rows are the distance values.

Let us now assume ‘O’ as negative and ‘X’ as positive values. Now the prediction on testing data is as below:

| | True label | Predicted label | O/P |
|---|---|---|---|
| 6 | X | X | Tp |
| 7 | O | X | Fp |
| 10 | O | X | Fp |
| 11 | O | X | Fp |

Confusion matrix for the above prediction is:

| | |
|---|---|
| TN | FP |
| FN | TP |

The final confusion matrix is:

**0 3 0 1**

Accuracy of the classifier = (TP + TN) / (P + N) = 1 / 4 = 0.25
Sensitivity of the classifier = TP / P = 1/1 = 1
Specificity of the classifier = TN / N = 0/3 = 0