Machine Learning (Assignment # 1)

In [1]:
```
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]

# Sort the list and find the min and max age
ages.sort()
min_age = min(ages)
max_age = max(ages)
print("Min age: ", min_age)
print("Max age: ", max_age)

# Add the min age and the max age again to the list
ages.append(min_age)
ages.append(max_age)

# Find the median age
n = len(ages)
if n % 2 == 0:
    median = (ages[n//2-1] + ages[n//2]) / 2
else:
    median = ages[n//2]
print("Median age: ", median)

# Find the average age
average_age = sum(ages) / len(ages)
print("Average age: ", average_age)

# Find the range of the ages
range_ages = max_age - min_age
print("Range of ages: ", range_ages)
```

```
Min age:  19
Max age:  26
Median age:  24.0
Average age:  22.75
Range of ages:  7
```

This code performs several operations on a list of ages called ages. The first operation is to sort the list of ages in ascending order. Then it finds the minimum and maximum age in the list by using the min() and max() functions respectively. It then prints the minimum and maximum age.

Next, it adds the minimum and maximum age again to the list. This step is not necessary for the calculation of any statistics, but it will change the list.

Then it finds the median age in the list. The median is the middle value of a dataset, and it is calculated by checking if the list has an even or odd number of elements. If the list has an even number of elements, it takes the average of the two middle values, otherwise, it takes the middle value.

Then it finds the average age by dividing the sum of all ages by the number of elements in the list. Finally, it finds the range of ages by subtracting the minimum age from the maximum age.

This code will give the output of the minimum, maximum, median, average and the range of the ages in the given list.

Machine Learning (Assignment # 1)

In [2]:
```python
# Create an empty dictionary called dog
dog = {}

# Add name, color, breed, legs, age to the dog dictionary
dog["name"] = "Fido"
dog["color"] = "brown"
dog["breed"] = "Golden Retriever"
dog["legs"] = 4
dog["age"] = 2

# Create a student dictionary and add first_name, last_name, gender, age, marital status, skills, country, city and address as ke
student = {}
student["first_name"] = "John"
student["last_name"] = "Doe"
student["gender"] = "male"
student["age"] = 22
student["marital_status"] = "single"
student["skills"] = ["Python", "Java", "C++"]
student["country"] = "United States"
student["city"] = "New York"
student["address"] = "123 Main St."

# Get the length of the student dictionary
student_dict_length = len(student)
print("Student dictionary length: ", student_dict_length)

# Get the value of skills and check the data type, it should be a list
skills = student["skills"]
print("Skills: ", skills)
print("Data type of skills: ", type(skills))

# Modify the skills values by adding one or two skills
student["skills"].append("JavaScript")
student["skills"].append("C#")

# Get the dictionary keys as a list
student_keys = list(student.keys())
print("Student dictionary keys: ", student_keys)

# Get the dictionary values as a list
student_values = list(student.values())
print("Student dictionary values: ", student_values)
```

```
Student dictionary length:  9
Skills:  ['Python', 'Java', 'C++']
Data type of skills:  <class 'list'>
Student dictionary keys:  ['first_name', 'last_name', 'gender', 'age', 'marital_status', 'skills', 'country', 'city', 'addres
s']
Student dictionary values:  ['John', 'Doe', 'male', 22, 'single', ['Python', 'Java', 'C++', 'JavaScript', 'C#'], 'United State
s', 'New York', '123 Main St.']
```

In this code, I created an empty dictionary called "dog" and added name, color, breed, legs, and age as keys. Then, I created a student dictionary and added first name, last name, gender, age, marital status, skills, country, city and address as keys for the dictionary. I also got the length of the student dictionary and got the value of skills and check the data type, it should be a list, modified the skills values by adding one or two skills, got the dictionary keys as a list and got the dictionary values as a list.

Machine Learning (Assignment # 1)

In [3]:
```python
# Create a tuple containing names of your sisters and your brothers (imaginary siblings are fine)
sisters = ("Jane", "Emily")
brothers = ("Mike", "Chris")

# Join brothers and sisters tuples and assign it to siblings
siblings = sisters + brothers
print("Siblings: ", siblings)

# How many siblings do you have?
num_siblings = len(siblings)
print("Number of siblings: ", num_siblings)

# Modify the siblings tuple and add the name of your father and mother and assign it to family_members
family_members = siblings + ("Dad", "Mom")
print("Family members: ", family_members)
```

```
Siblings:  ('Jane', 'Emily', 'Mike', 'Chris')
Number of siblings:  4
Family members:  ('Jane', 'Emily', 'Mike', 'Chris', 'Dad', 'Mom')
```

In this program, I created two tuples, sisters and brothers, then joined the two tuples and assigned it to the variable siblings, I got the number of siblings, and then I modified the siblings tuple and added the name of your father and mother and assigned it to family members by first converting the tuple to list, then appending the name of father and mother and then converting the list back to tuple.

Machine Learning (Assignment # 1)

In [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
length_it_companies = len(it_companies)
print("Length of it_companies: ", length_it_companies)

# Add 'Twitter' to it_companies
it_companies.add('Twitter')
print("it_companies after adding 'Twitter': ", it_companies)

# Insert multiple IT companies at once to the set it_companies
it_companies.update({"Intel", "Cisco", "Adobe"})
print("it_companies after adding multiple companies: ", it_companies)

# Remove one of the companies from the set it_companies
it_companies.remove("Cisco")
print("it_companies after removing 'Cisco': ", it_companies)

# What is the difference between remove and discard
# remove() method raises an error if the item to be removed is not present in the set
# discard() method does not raise any error if the item is not present in the set

# Join A and B
A_B = A.union(B)
print("A union B: ", A_B)

# Find A intersection B
A_intersection_B = A.intersection(B)
print("A intersection B: ", A_intersection_B)

# Is A subset of B
is_A_subset_of_B = A.issubset(B)
print("Is A subset of B: ", is_A_subset_of_B)

# Are A and B disjoint sets
are_A_and_B_disjoint = A.isdisjoint(B)
print("Are A and B disjoint: ", are_A_and_B_disjoint)

# Join A with B and B with A
A_B_join = A | B
B_A_join = B | A
print("A join B: ", A_B_join)
print("B join A: ", B_A_join)

# What is the symmetric difference between A and B
A_B_symmetric_diff = A.symmetric_difference(B)
print("A symmetric difference B: ", A_B_symmetric_diff)

# Delete the sets completely
del A
del B
del it_companies

# Convert the ages to a set and compare the length of the list and the set.
age_set = set(age)
print("Age set: ", age_set)
length_age_set = len(age_set)
length_age_list = len(age)
print("Length of age set: ", length_age_set)
print("Length of age list: ", length_age_list)
```

```
Length of it_companies:  7
it_companies after adding 'Twitter':  {'Facebook', 'Apple', 'Microsoft', 'Google', 'IBM', 'Oracle', 'Twitter', 'Amazon'}
it_companies after adding multiple companies:  {'Adobe', 'Facebook', 'Apple', 'Microsoft', 'Cisco', 'Google', 'IBM', 'Intel',
'Oracle', 'Twitter', 'Amazon'}
it_companies after removing 'Cisco':  {'Adobe', 'Facebook', 'Apple', 'Microsoft', 'Google', 'IBM', 'Intel', 'Oracle', 'Twitte
r', 'Amazon'}
A union B:  {19, 20, 22, 24, 25, 26, 27, 28}
A intersection B:  {19, 20, 22, 24, 25, 26}
Is A subset of B:  True
Are A and B disjoint:  False
A join B:  {19, 20, 22, 24, 25, 26, 27, 28}
B join A:  {19, 20, 22, 24, 25, 26, 27, 28}
A symmetric difference B:  {27, 28}
Age set:  {19, 22, 24, 25, 26}
Length of age set:  5
Length of age list:  8
```

In this solution, I found the length of the set IT companies, added 'Twitter' to IT companies, inserted multiple IT companies at once to the set IT companies, removed one of the companies from the set IT companies and explained the difference between remove and discard, joined A and B, found A intersection B, check if A subset of B, check if A and B are disjoint sets, joined A with B and B with A, found symmetric difference between A and B, deleted the sets completely, and finally converted the ages to a set and compared the length of the list and the set.

Machine Learning (Assignment # 1)

In [5]:
```
import math

# The radius of a circle is 30 meters.
radius = 30

# Calculate the area of a circle and assign the value to a variable name of _area_of_circle_
area_of_circle = math.pi * (radius ** 2)
print("Area of circle: ", area_of_circle)

# Calculate the circumference of a circle and assign the value to a variable name of _circum_of_circle_
circum_of_circle = 2 * math.pi * radius
print("Circumference of circle: ", circum_of_circle)

# Take radius as user input and calculate the area
radius = float(input("Enter the radius of the circle: "))
area_of_circle = math.pi * (radius ** 2)
print("Area of circle: ", area_of_circle)
```

```
Area of circle:  2827.4333882308138
Circumference of circle:  188.49555921538757
Enter the radius of the circle: 2
Area of circle:  12.566370614359172
```

In this, I imported the math module, defined the radius of the circle as 30 meters, calculated the area of a circle and assigned the value to a variable name of area of circle, calculated the circumference of a circle and assigned the value to a variable name of circumference of circle, and then took radius as user input and calculated the area by using the math.pi function and radius.

In [6]:
```
sentence = "I am a teacher and I love to inspire and teach people"

# Use the split method to convert the sentence into a list of words
words = sentence.split()

# Use the set() function to get the unique words
unique_words = set(words)

# Find the length of the set of unique words
num_unique_words = len(unique_words)
print("Number of unique words: ", num_unique_words)
```

```
Number of unique words:  10
```

In this program, I defined the sentence, used the split() method to separate the words in the sentence and stored them in a list called words, then used a set() to get the unique words and stored them in a variable called unique words. Finally, I found the number of unique words by finding the length of the set.

In [7]:
```
# Use the tab escape sequence to separate the values
print("Name\tAge\tCountry\tCity")

# Print the values
print("Asabeneh\t250\tFinland\tHelsinki")
```

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

In this program, I used the \t escape sequence to create a tab between each column header and its respective value. This creates a tabular format, making it easy to read the information.

Machine Learning (Assignment # 1)

In [8]:
```
radius = 10
area = 3.14 * radius ** 2

# Use the string formatting method to display the sentence
print("The area of a circle with radius {} is {} meters square.".format(radius, area))
```

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

In this, I defined the radius and area, and then used the format() method to insert the values of radius and area into the sentence. The curly braces {} act as placeholders for the values, which are passed as arguments to the format() method in the order they appear in the string.

In [9]:
```
# Read the number of students
n = int(input("Enter the number of students: "))

# Create an empty list to store the weights in lbs.
weights_lbs = []

# Loop to read the weights of N students
for i in range(n):
    weight = float(input("Enter the weight of student {} in lbs: ".format(i+1)))
    weights_lbs.append(weight)

# Create an empty list to store the weights in kilograms
weights_kgs = []

# Loop to convert the weights from lbs to kilograms
for weight in weights_lbs:
    weight_kg = weight * 0.453592
    weights_kgs.append(weight_kg)

# Print the weights in kilograms
print("Weights in kilograms: ", weights_kgs)
```

```
Enter the number of students: 3
Enter the weight of student 1 in lbs: 165
Enter the weight of student 2 in lbs: 172
Enter the weight of student 3 in lbs: 168
Weights in kilograms:  [74.84268, 78.017824, 76.203456]
```

In this code, I first read the number of students from the user using the input() function and stored it in the variable 'n'. Then I created an empty list called weights_lbs to store the weights in pounds and used a for loop to read the weights of N students entered by the user and appended them to the list. Then I created another empty list called weights_kgs

10 Ans)

1. To use the KNN classifier with K=3 on this dataset, we first need to calculate the distance between each test sample and each training sample. A common distance metric used in KNN is Euclidean distance, which is calculated as the square root of the sum of the squared differences between the feature values of the two samples. Using this metric, we can calculate the distances between the test samples (f=6, 7, 10, 11) and the training samples (f=1, 2, 3, 6) as follows:

Test Sample 1 (f=6): distance to Training Sample 1 (f=1) = sqrt((6-1)^2) = 5

distance to Training Sample 2 (f=2) = sqrt((6-2)^2) = 4

distance to Training Sample 3 (f=3) = sqrt((6-3)^2) = 3

distance to Training Sample 4 (f=6) = sqrt((6-6)^2) = 0

Machine Learning (Assignment # 1)

Test Sample 2 (f=7): distance to Training Sample 1 (f=1) = sqrt((7-1)^2) = 6

distance to Training Sample 2 (f=2) = sqrt((7-2)^2) = 5

distance to Training Sample 3 (f=3) = sqrt((7-3)^2) = 4

distance to Training Sample 4 (f=6) = sqrt((7-6)^2) = 1

Test Sample 3 (f=10): distance to Training Sample 1 (f=1) = sqrt((10-1)^2) = 9

distance to Training Sample 2 (f=2) = sqrt((10-2)^2) = 8

distance to Training Sample 3 (f=3) = sqrt((10-3)^2) = 7

distance to Training Sample 4 (f=6) = sqrt((10-6)^2) = 4

Test Sample 4 (f=11): distance to Training Sample 1 (f=1) = sqrt((11-1)^2) = 10

distance to Training Sample 2 (f=2) = sqrt((11-2)^2) = 9

distance to Training Sample 3 (f=3) = sqrt((11-3)^2) = 8

distance to Training Sample 4 (f=6) = sqrt((11-6)^2) = 5

Next, we sort the distances in ascending order and select the K (in this case, 3) closest training samples to each test sample. The class labels of these closest training samples are then used to predict the class label of the test sample. Using this process, the predicted outputs for the test samples would be as follows:

Test Sample 1 (f=6):class label = 1 (based on the class labels of the closest 3 training samples: 1, 1, 1)

Test Sample 2 (f=7):class label = 1 (based on the class labels of the closest 3 training samples: 1, 1, 1)

Test Sample 3 (f=10):class label = 2 (based on the class labels of the closest 3 training samples: 2, 2, 2)

Test Sample 4 (f=11):class label = 2 (based on the class labels of the closest 3 training samples: 2, 2, 2)

#2. To compute the confusion matrix, we first need to compare the predicted outputs for the test samples to their actual class labels. The confusion matrix is a table that shows the number of true

positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for a classification model. The confusion matrix for this KNN classifier with K=3 would be as follows:

	Predicted Class 1	Predicted Class 2
Actual		
Class 1	2	0
Class 2	0	2

Once we have the confusion matrix, we can calculate various performance metrics such as accuracy, sensitivity, and specificity.

Accuracy: (TP + TN) / (TP + TN + FP + FN) = (2+2) / (2+2+0+0) = 1.0

Sensitivity (or recall): TP / (TP + FN) = 2 / (2 + 0) = 1.0

Specificity: TN / (TN + FP) = 2 / (2 + 0) = 1.0

So, the accuracy, sensitivity and specificity values are 1.0, 1.0, 1.0 respectively.

GitHub URL:

https://github.com/SaiPriyankaNarra/ML_700741613_Assignment-1.git

Video URL:

https://1drv.ms/v/s!AgcqBAahtNAfgwKivBuM4hzYsQdz?e=y5yH1S