MACHINE LEARNING

Name- DHHEERAJ BOLEENENNI
ID- 700727909

Question1

```
### Question 1
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]

# Sort the list and find the min and max age
sorted_ages = sorted(ages)
min_age = min(sorted_ages)
max_age = max(sorted_ages)
print(sorted_ages)
print("min_age, max_age=",min_age,",",max_age)
# Add the min age and the max age again to the list
sorted_ages.extend((min_age, max_age))
print(sorted_ages)

# Median of Ages
sorted_ages = sorted(sorted_ages)
len_ages = len(sorted_ages)

median_age = statistics.median(sorted_ages)
print("median_age=",median_age)
# Average of Ages
avg_age = sum(sorted_ages)/len(sorted_ages)
print("avg_age=",avg_age)
# Range of Ages
range_age = max_age - min_age
print("range_age=",range_age)
```

Question 1(output)

```
[19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
min_age, max_age= 19 , 26
[19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]
median_age= 24.0
avg_age= 22.75
range_age= 7
```

Question2

```
length of student dic 9
skills= ['python', 'java', 'sql']
type of skills= <class 'list'>
update skills= ['python', 'java', 'sql', 'ML']
student keys= ['first_name', 'last_name', 'gender', 'age', 'marital status', 'skills', 'country', 'city', 'address']
student values= ['dhheeraj', 'boleenenni', 'male', 23, 'single', ['python', 'java', 'sql', 'ML'], 'United States', 'lees summit', '1104 Innovation Campus']
```

```
dog = dict()
dog['name'] = 'shadow'
dog['color'] = 'black'
dog['breed'] = 'rotwiller'
dog['legs'] = 4
dog['age'] = 4

student = {
    "first_name": "dhheeraj",
    "last_name": "boleenenni",
    "gender": "male",
    "age": 23,
    "marital status": "single",
    "skills": ["python", "java","sql"],
    "country": "United States",
    "city": "lees summit",
    "address": "1104 Innovation Campus"
}

len_student = len(student)
print("length of student dic",len_student)
skills = student['skills']
print("skills=",skills)
type_of_skills = type(skills)
print("type of skills=",type_of_skills)
student['skills'].extend(["ML"])
print("update skills=",student['skills'])
#student.update()

keys_student = list(student.keys())
print("student keys=",keys_student)
values_student = list(student.values())
print("student values=",values_student)
print("break")
```

Question2(output)

```
length of student dic 9
skills= ['python', 'java', 'sql']
type of skills= <class 'list'>
update skills= ['python', 'java', 'sql', 'ML']
student keys= ['first_name', 'last_name', 'gender', 'age', 'marital status', 'skills', 'country', 'city', 'address']
student values= ['dhheeraj', 'boleenenni', 'male', 23, 'single', ['python', 'java', 'sql', 'ML'], 'United States', 'lees summit', '1104 Innovation Campus']
```

Question3

```
brothers = ("prem","harish")
sisters = ("mounica","manu")

siblings = brothers + sisters
len_siblings = len(siblings)
print("length of sibilings=",len_siblings)
family_members = siblings + ("narssing rao","lavanya")
print("family members=",family_members)
print("break3")
```

Question3(output)

```
length of sibilings= 4
family members= ('prem', 'harish', 'mounica', 'manu', 'narssing rao', 'lavanya')
```

Question4

```
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]

len_it_companies = len(it_companies)
print("len_companies=",len_it_companies)
it_companies.add("Twitter")
it_companies.discard("Oracle")
print("remove vs discard=remove deletes the element from the list if not present it returns Key error discard deleted "
      "the element from the list otherwise return None")

join_AB = A.union(B)
intersection_AB = A.intersection(B)
print("intersection=",intersection_AB)
isSubset_AB = A.issubset(B)

A.isdisjoint(B)
A.union(B)
B.union(A)
A.difference(B)
A.clear()
B.clear()

set_ages = set(age)
print("set ages length = ",len(set_ages), "list ages length :",len(age))
print("break4")
```

Question4(output)

```
len_companies= 7
remove vs discard=remove deletes the element from the list if not present it returns Key error discard deleted the element from the list otherwise return None
intersection= {19, 20, 22, 24, 25, 26}
set ages length =  5 list ages length : 8
```

Question 5

```
r = 30
pi = 3.14
_area_of_circle_ =pi * r * r
_circum_of_circle_ = 2 * pi * r
print("area of circle=",_area_of_circle_," ,circumference of a circle=",_circum_of_circle_)
r = float(input("Enter radius of circle : "))
area_of_circle = pi * r * r
print("area of circle as per unit=",area_of_circle)
print("break5")
```

Question5(output)

```
area of circle= 2826.0  ,circumference of a circle= 188.4
Enter radius of circle : 2
area of circle as per unit= 12.56
```

Question6

```
sent = """I am a teacher and I love to inspire and teach people"""
split_sent = sent.split()
unique_words = set(split_sent)
print("unique words=",unique_words)
print("break6")
```

Question6(output)

```
unique words= {'teacher', 'people', 'teach', 'to', 'and', 'am', 'I', 'inspire', 'a', 'love'}
```

Question7

```
data = "Name\tAge\tCountry\tCity\nAsabeneh\t250\tFinland\tHelsinki"
print(data)
print("break7")
```

Question7(output)

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

Question8

```
radius = 10
area = 3.14 * radius ** 2
sent = "The area of a circle with radius {} is {} meters square.".format(radius, area)
print(sent)
```

Question 8(output)

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

Question9

```
N = int(input("Enter No. of students : "))
lbs_to_kg_convert = 0.4536
lbs_weights = []
kg_weights = []
for i in range(0,N):
    lbs_weights.append(int(input("Enter student Weight(lbs) : ")))

for weight in lbs_weights:
    kg_weights.append(round(weight * lbs_to_kg_convert,2))

print(kg_weights)
print("break9")
```

Question 9(output)

```
Enter No. of students : 4
Enter student Weight(lbs) : 150
Enter student Weight(lbs) : 155
Enter student Weight(lbs) : 145
Enter student Weight(lbs) : 148
[68.04, 70.31, 65.77, 67.13]
```

Question 10

A =

data	1	2	3	6	6	7	10	11
B = x	0	0	1	1	1	0	0	0

Splitting train and test data.

A_train = [1, 6, 7, 11]

B train = [0, 1, 0, 0]

A test = [2, 3, 6, 10]

B test = [0, 1, 1, 0]

using KNN classifies

$K = 3$

outputs of A test

distance = $\sqrt{(A - A_i)^2}$

predicting A test values using distance with 3 Neighbour values from point.

A test [0] = 2 =

i.e; $\sqrt{(2-1)^2} = 1$

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

$\sqrt{(2-7)^2} = 5$

$\sqrt{(2-11)^2} = 9$

B pred [0] = 0

A test [1] = 3

i.e; $\sqrt{(3-1)^2} = 2$

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

$\sqrt{(3-7)^2} = 4$

$\sqrt{(3-11)^2} = 8$

B pred[1] = 0

A test [1] = 6

i.e; $\sqrt{(6-1)^2} = 5$

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

$\sqrt{(6-7)^2} = 1$

$\sqrt{(6-11)^2} = 5$

B pred[2] = 0

A test [0] = 10

i.e; $\sqrt{(10-1)^2} = 9$

$\sqrt{(10-6)^2} = 4$

$\sqrt{(10-7)^2} = 3$

$\sqrt{(10-11)^2} = 1$

B pred[3] = 0

B prediction = [0, 0, 0, 0]

B test = [0, 1, 1, 0]

TP = 0/4

FP = 0/4

TN = 2/4

FN = 2/4

$$\text{Accuracy} = \left(\frac{TP + TN}{\text{Total data}}\right) = 0.125$$

$$\text{Sensitivity} = \frac{TP}{(TP + FN)} = 0$$

$$\text{specificity} = \frac{TN}{(FP + TN)} = 1$$