Machine Learning (Assignment # 1)

CS-5710

Name: Satya Ishyanth Kadali

Id: 700735513

CRN: 11813

Video link:
https://drive.google.com/drive/folders/1RLHZ5lY0HByWKbaFn29dUVo3ajMKOHIO?usp=sharing

Modifying Lists, Sets, Tuples, Dictionaries

Question 1

Created a list of 10 students ages.

• Sorting the list by using sort() method .

and finding the min and max age by using min() and max() methods.

• By using append() the min age and the max age are added to the list. Basically append() adds values to the list at last

• Finding the median age by using floor division // operator we can find index of the list to calculate median.

• Average is sum of all the values of the list using sum() method divided by length of the list by using len() method.

• Already we have max and min values, so max- min gives range of the ages.

In [3]:

```
#Question 1
#List for 10 student ages
ages = [19,22,19,24,20,25,26,24,25,24]

#Sorting list by using sort()
ages.sort()
print(ages)

#Finding min and max
n = min(ages)
m = max(ages)
print('min=',n,', max=',m)

#adding min and max ages to the list
#Using append() to add the values into the list
ages.append(n)
ages.append(m)
print(ages)

#finding median
a = len(ages)
if a%2 == 0:
    median = (ages[a//2] + ages[a//2-1])/2
else:
    median = ages[n//2]
print('Median Age=', median)

# Finding the avgerage age
s = sum(ages)
a = len(ages)
avg = s/a
print('Average Age=',avg)

#Finding range of the ages
range_ages = m-n
print('Range of the ages=',range_ages)
print(len(ages))
```

```
[19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
min= 19 , max= 26
[19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]
Median Age= 24.0
Average Age= 22.75
Range of the ages= 7
12
```

Question 2

• Created an empty dictionary called dog and added name, color, breed, legs, age to the dog dictionary
• Created a student dictionary and add first_name, last_name, gender, age, marital status, skills, country, city and address as keys for the dictionary.
• By using len() method to find length of the student dictionary.
• By using get(), printing the value of skills and checking the data type with type() method.
• Modify the skills values by appending to the skills key .
• Using keys() we can print all the keys in the dictionary.
• Using values() we can print all the values in the dictionary.

In [20]:

```
#Question 2
#Creating a dictionary
dog = {}
#adding keys and values
dog.update({"name":"Snoopy", "color":"white", "breed":"samoyed", "legs":"short", "age":"17"})
print(dog,'\n')

#creating a student dictionary
std = {
    "first_name":"Ishyanth",
    "last_name":"Kadali",
    "gender":"male",
    "age":"23",
    "martial_status":"Single",
    "skills":["python", "tensorflow", "keras", "pandas"],
    "country":"USA",
    "city":"Warrensburg",
    "Address":"501 Joll St"
    }
print(std,'\n')
#finding Lenght of dictionary
print(len(std),'\n')

#type check
v = std.get("skills")
print(v)
print(type(v),'\n')

#Modifying values
std["skills"].append("java")
std["skills"].append("C++")
print(std,'\n')

#Dictionary keys as list
print(std.keys(),'\n')
#Dictionary values as list
print(std.values())

```

```
{'name': 'Snoopy', 'color': 'white', 'breed': 'samoyed', 'legs': 'short', 'age': '17'}

{'first_name': 'Ishyanth', 'last_name': 'Kadali', 'gender': 'male', 'age': '23', 'martial_status': 'Single', 'skills': ['pyt
hon', 'tensorflow', 'keras', 'pandas'], 'country': 'USA', 'city': 'Warrensburg', 'Address': '501 Joll St'}

9

['python', 'tensorflow', 'keras', 'pandas']
<class 'list'>

{'first_name': 'Ishyanth', 'last_name': 'Kadali', 'gender': 'male', 'age': '23', 'martial_status': 'Single', 'skills': ['pyt
hon', 'tensorflow', 'keras', 'pandas', 'java', 'C++'], 'country': 'USA', 'city': 'Warrensburg', 'Address': '501 Joll St'}

dict_keys(['first_name', 'last_name', 'gender', 'age', 'martial_status', 'skills', 'country', 'city', 'Address'])

dict_values(['Ishyanth', 'Kadali', 'male', '23', 'Single', ['python', 'tensorflow', 'keras', 'pandas', 'java', 'C++'], 'US
A', 'Warrensburg', '501 Joll St'])
```

Question 3

• Created 2 tuples containing names of sisters and brothers.

• Joined brothers and sisters tuples by using + operator and assigned it to siblings.

• Used len() method to find out the length of the tuple.

• We can't modify the siblings tuple. So I created a parents tuple with name of father and mother and assign it to family_members by joining siblings and parents tuples.

In [6]:

```
#Question 3
#creating tuples
brothers = ('Phani','Deep','Karthik','Chintu')
sisters = ('Vinela','Hanisha', 'Bharani')
#join tuples
siblings = brothers + sisters
print(siblings)
#lenght of tuples
print('Number of siblings =',len(siblings),'\n')

#we can't modify tuples
parents = ('Lakshmana Rao','Satya Veni')
family_members = siblings + parents
print(family_members)
```

```
('Phani', 'Deep', 'Karthik', 'Chintu', 'Vinela', 'Hanisha', 'Bharani')
Number of siblings = 7

('Phani', 'Deep', 'Karthik', 'Chintu', 'Vinela', 'Hanisha', 'Bharani', 'Lakshmana Rao', 'Satya Veni')
```

Question 4

• len() to find the length of the set it_companies

• Added 'Twitter' to it_companies by append() method.

• Inserted multiple IT companies at once to the set it_companies by appending values in list format.

• Removed one of the companies from the set it_companies using remove() function.

• Difference between remove and discard is if the item doesn't exist, the remove() function will raise an error, but the discard() method won't.

• Joining A and B using union()

• By using intersection() we can find A intersection B

• We can know A subset of B or not by issubset().

• We can know A and B disjoint sets or not by using isdisjoint().

• Using update() method joined A with B and B with A.

• Common values by using symmetric difference between A and B

• Clear() deletes the sets completely.

• By converting the ages to a set, it removes the duplicated and comparing the length of the list and the set will results list is greater than set as there are no duplicates in set.

In [23]:

```
#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]

#Finding the length of it_companies
print("length of it_companies:", len(it_companies))
it_companies.add('Twitter')
print(it_companies)

#adding companies
it_companies.update(['Tesla','Razer','Alienware'])
print(it_companies)

#removing a company
it_companies.remove('IBM')
print(it_companies)

'''Difference between remove and discard is if the item doesn't exist, the remove() function
   will raise an error, but the discard() method won't.'''

#joining 2 sets
C = A.union(B)
print(C)

#Intersection
D = A.intersection(B)
print(D)

#Subset
E = A.issubset(B)
print(E)

#Disjoint sets
F = A.isdisjoint(B)
print(F)

#Joining sets
A.update(B)
print(A,'\n')
B.update(A)
print(B,'\n')

#Symmetric Diff
A = {19, 22, 24, 20, 25, 26}
B = {19, 22, 20, 25, 26, 24, 28, 27}
G = A.symmetric_difference(B)
print('symmetric_difference A nad B is', G)

#Deleiting the sets
A.clear()
B.clear()
print(A)
print(B)

#Coverting ages to set
age_set = set(age)
print(age_set)

#compare
x = len(age)
z = len(age_set)
print("Length of the list is greater than length of the set:", x>z)
```

```
length of it_companies: 7
{'Twitter', 'Apple', 'IBM', 'Microsoft', 'Oracle', 'Amazon', 'Facebook', 'Google'}
{'Oracle', 'Razer', 'Google', 'Alienware', 'Amazon', 'Twitter', 'Tesla', 'Microsoft', 'IBM', 'Facebook', 'Apple'}
{'Oracle', 'Razer', 'Google', 'Alienware', 'Amazon', 'Twitter', 'Tesla', 'Microsoft', 'Facebook', 'Apple'}
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26}
True
False
{19, 20, 22, 24, 25, 26, 27, 28}

{19, 20, 22, 24, 25, 26, 27, 28}

symmetric_difference A nad B is {27, 28}
set()
set()
{19, 22, 24, 25, 26}
Length of the list is greater than length of the set: True
```

Question 5

Radius = 30 meters.

Using ** operator for calculating exponential values

• Took the input radius from user and calculated the area.

In [8]:

```
#Question 5
# calculating radius and circumference of a circle
r = 30
print('radius of a circle', r)
_area_of_circle_ = 3.14*r**2
_circum_of_circle_ = 2*3.14*r
print('Area of the circle',_area_of_circle_)
print('Circumference of the circle',_circum_of_circle_, '\n')

#New radius input
radius = float(input('enter the new radius of the circle = '))

_area_of_circle_ = 3.14*radius**2

print('Area of the circle',_area_of_circle_)

```

```
radius of a circle 30
Area of the circle 2826.0
Circumference of the circle 188.4

enter the new radius of the circle = 14
Area of the circle 615.44
```

Question 6

"I am a teacher and I love to inspire and teach people"

There are 10 unique words in the sentence.

By split methods and set we can produce unique words.

In [11]:

```
#Question 6
sen= "I am a teacher and I love to inspire and teach people"

#By using split methods and set

words=set(sen.split(' '))

#finding No. of unique words
print('Number of Unique words = ',len(words))

print('Unique Words in the sentence are ')
for w in words:
    print(w)

```

```
Number of Unique words =  10
Unique Words in the sentence are
love
people
teach
am
a
teacher
inspire
and
to
I
```

Question 7
By using tab escape sequence

In [17]:
```
#Question 7
#tab escape sequence
print('Name\t\tAge\tCountry\t\tCity')
print('Asabeneh\t250\tFinland\t\tHelsinki')
```

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

Question 8
By using string formatting method
Variables and sentences can be displayed in one print statement.

In [18]:
```
#Question 8
#string formatting method
radius = 10
area = 3.14*radius**2
print("The area of a circle with radius ",radius," is ",int(area)," meters suqare.")
```

```
The area of a circle with radius  10  is  314  meters suqare.
```

Question 9
Python program, which reads weights (lbs.) of N students into a list and convert the weights to kilograms in a separate list using for Loop.

In [19]:
```
#Question 9
#Reading input from user
n = int(input('Enter the number of students: '))
lbs = []
kg = []
for i in range(0,n):
    t = float(input('Enter the weight: '))
    lbs.append(t)

#concerting lbs to kgs
ki = 0.45359237
for y in lbs:
    kg.append(y * ki)
print(kg)
```

```
Enter the number of students: 5
Enter the weight: 150
Enter the weight: 155
Enter the weight: 145
Enter the weight: 148
Enter the weight: 160
[68.0388555, 70.30681735, 65.77089365, 67.13167076, 72.57477920000001]
```

Question 10

- Importing libraries which are numpy to work with arrays, pandas to import and feed the dataset to the model.
- Importing scikit learn to work with KNN classifier and confusion matrix and to split the data into test data and training data.

In [38]:

```
#Question 10
#importing libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix

```

In [39]:

```
#reading the dataset
dataset=pd.read_csv("dataset.csv")
print(dataset)
X= dataset['f'].values
y= dataset['output'].values

```

```
    f output
0   1    dot
1  10    dot
2   6  cross
3   7    dot
4   2    dot
5   6  cross
6  11    dot
7   3  cross
```

- Using reshape() method to convert data into 2 dimensional array.
- Split the data into 50:50 ratio.
- Using StandardScaler()to normalize the data.
- Creating a K Nearest Neighbor classifier to predict the output.
- Giving the k=3 to KNN model.

- Fitting the data into the model to train.

In [19]:
```
#using reshape to convert data into 2 dimensional array
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size = 0.5)
X_train = np.array(X_train).reshape(-1,1)
X_test = np.array(X_test).reshape(-1,1)
```

In [20]:
```
#Data normalization
norm= StandardScaler()
X_train= norm.fit_transform(X_train)
X_test= norm.transform(X_test)
```

In [21]:
```
#creating KNN model
model= KNeighborsClassifier(n_neighbors=3, metric= 'euclidean' )
#Data fitting into the model
model.fit(X_train, y_train)
```

Out[21]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=3)
```

- Predicting the test set result by giving test data.
- Put a threshold if the values in X_test less than 1.5 then it will be classified as **cross** or else **dot.**
- Checking the predicted results by using confusion matrix.

In [57]:
```
#Predicting the test set result
predict_output = model.predict(X_test)
print(X_test)
thresults = []
for i in X_test:
    if i < 1.5:
        thresults.append('cross')
    else:
        thresults.append('dot')
print("Output:",thresults)
```

```
[[ 2.74562589]
 [ 0.78446454]
 [ 2.35339362]
 [-0.39223227]]
Output: ['dot', 'cross', 'dot', 'cross']
```

In [69]:
```
#checking the output by confusion matrix
results= confusion_matrix(y_test, thresults)
print("Confusion matrix:\n",results)
```

```
Confusion matrix:
 [[2 0]
 [0 2]]
```

- Here, we got 2 true positives and 2 true negatives.

In [70]:
```
#finding model accuracy
count=sum(sum(results))
accuracy=(results[0,0]+results[1,1])/count
print ('Accuracy=', accuracy)
```

```
Accuracy= 1.0
```

In [71]:
```
# finding model sensitivity
sense = results[0,0]/(results[0,0]+results[0,1])
print('Sensitivity=', sense )
```

```
Sensitivity= 1.0
```

In [72]:
```
#finding model specificity
specificity = results[1,1]/(results[1,0]+results[1,1])
print('Specificity=', specificity)
```

```
Specificity= 1.0
```

- The model got 100% accuracy.
- The model got 100% sensitivity as it is correctly predicting positives instances.
- The model got 100% specificity as it is correctly predicting negative instances.

Video link:
https://drive.google.com/drive/folders/1RLHZ5lY0HByWKbaFn29dUVo3ajMKOHIO?usp=sharing