Artificial Intelligence with Python-Assignement

PART 1-

Q1

```
n=int(input("Enter number of elements: "));
a=[];
b=[];
print("Enter the numbers: ");
for i in range (0,n): #taking input of integer array
    a.append(int(input()));
print("Square roots are: ");
for i in range (0,n):
    b.append(a[i]**0.5);
print(b);
```

```
Enter number of elements: 3
Enter the numbers:
1
5
6
Square roots are:
[1.0, 2.23606797749979, 2.449489742783178]
```

Q2

```
for i in range (1,100):
    if(i%3==0 and i%5==0):
        print(i);
```

```
15
30
45
60
75
90
```

Q3

```
l=str(input("Enter any letter :"));
c=l;
if len(l)!=1:
    print("invalid input");
elif l in 'aeiouAEIOU':
    print(l,"is a vowel");
elif ('a'<=l and l<='z') or ('A'<=l and l<='Z'):
    print(l,"is a consonant");
else:
    print("invalid input")
```

```
Enter any letter :w
w is a consonant
```

Q4

```python
a=input("Enter the first character :");
b=input("Enter the second character :");
if len(a)!=1 or len(b)!=1 :
    print("invalid input")
else:
    print("the distance between them is",abs(ord(a)-ord(b)))
```

```
Enter the first character :a
Enter the second character :d
the distance between them is 3
```

Q5

```python
def func(str):
    v=0;
    for char in str :
        if char in 'aeiouAEIOU':
            v+=1;
    return v;
l=input("Enter a string :");
print("string has",func(l),"vowels")
```

```
Enter a string :wgsgfdfu
string has 1 vowels
```

Q6

```python
for i in range (65,91) :
    print(chr(i));
```

```
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
```

```
R
S
T
U
V
W
X
Y
Z
```

Q7

```
n=int(input("Enter number of elements: "));
l=[];
sum=0;
print("Enter the numbers: ");
for i in range (0,n):
    l.append(int(input()));
for i in range (0,n) :
    if l[i]%2==0 :
        sum+=l[i];
print("sum of even numbers is",sum)
```

```
Enter number of elements: 4
Enter the numbers:
1
2
3
4
sum of even numbers is 6
```

Q8

```
for i in range (0,11) :
    if i%3!=0 :
        print("square of",i,"is",i*i);
```

```
square of 1 is 1
square of 2 is 4
square of 4 is 16
square of 5 is 25
square of 7 is 49
square of 8 is 64
square of 10 is 100
```

Q9

```
a=input("Enter first string :");
a=list(i for i in a); #list comprehension for converting string to
list
b=input("Enter second string :");
b=list(i for i in b);
```

```
for i in range (0,len(a)) :
    if a[i]=='A':
        a[i]='a'; #because string is immutable and was giving error
here
for i in range (0,len(b)) :
    if b[i]=='A':
        b[i]='a';
c=''.join(a+b);
print("New string is",c);
```

```
Enter first string :ASDSaaaa
Enter second string :fsgccAawg sfaA
New string is aSDSaaaafsgccaawg sfaa
```

Q10

```
n=int(input("Enter number of elements: "));
a=[];
print("Enter the numbers: ");
for i in range (0,n):
    a.append(int(input()));
odd_a=list(i for i in a if i%2==1);
print("Odd numbers are",odd_a);
```

```
Enter number of elements: 1
Enter the numbers:
2
Odd numbers are []
```

Q11

```
a=input("Enter first string :");
print(a.swapcase());
```

```
Enter first string :AtvesDAFCcd
aTVESdafcCD
```

PART 2- Iris Classifier

```
import pandas as pd
data=pd.read_csv("Iris.csv");
x = data.drop(columns = ["variety"]).values #last column of variety is
dropped
y = data["variety"].values #only last column left as a list of correct
labels
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier as D
from sklearn.neighbors import KNeighborsClassifier as K
from sklearn.svm import SVC as S
from sklearn.ensemble import RandomForestClassifier as R
from sklearn.metrics import accuracy_score, confusion_matrix
classifier=[ (D(), "Decision Tree"), (K(),"K neighbour"),
           (S(),"Support Vector Machine"),(R(),"Random forest
```

```python
classifier")]
size=[0.4,0.3,0.2,0.1];
for c,n in classifier:
     for i in size:
        X_train, X_test, y_train, y_test =
train_test_split(x,y,test_size=i,random_state=1);
        c.fit(X_train, y_train);
        pred=c.predict(X_test);
        acc=accuracy_score(y_test, pred);
        print(n,"for test size",i,"has accuracy", acc*100,"%");
        #if i==0.2:
        cm=confusion_matrix(y_test, pred);
        print("the confusion matrix is :");
        print(cm);
```

```
Decision Tree for test size 0.4 has accuracy 96.66666666666667 %
the confusion matrix is :
[[19  0  0]
 [ 0 20  1]
 [ 0  1 19]]
Decision Tree for test size 0.3 has accuracy 95.55555555555556 %
the confusion matrix is :
[[14  0  0]
 [ 0 17  1]
 [ 0  1 12]]
Decision Tree for test size 0.2 has accuracy 96.66666666666667 %
the confusion matrix is :
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
Decision Tree for test size 0.1 has accuracy 100.0 %
the confusion matrix is :
[[5 0 0]
 [0 6 0]
 [0 0 4]]
K neighbour for test size 0.4 has accuracy 98.33333333333333 %
the confusion matrix is :
[[19  0  0]
 [ 0 21  0]
 [ 0  1 19]]
K neighbour for test size 0.3 has accuracy 97.77777777777777 %
the confusion matrix is :
[[14  0  0]
 [ 0 18  0]
 [ 0  1 12]]
K neighbour for test size 0.2 has accuracy 100.0 %
the confusion matrix is :
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

```
K neighbour for test size 0.1 has accuracy 100.0 %
the confusion matrix is :
[[5 0 0]
 [0 6 0]
 [0 0 4]]
Support Vector Machine for test size 0.4 has accuracy
98.33333333333333 %
the confusion matrix is :
[[19  0  0]
 [ 0 20  1]
 [ 0  0 20]]
Support Vector Machine for test size 0.3 has accuracy
97.77777777777777 %
the confusion matrix is :
[[14  0  0]
 [ 0 17  1]
 [ 0  0 13]]
Support Vector Machine for test size 0.2 has accuracy
96.66666666666667 %
the confusion matrix is :
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
Support Vector Machine for test size 0.1 has accuracy 100.0 %
the confusion matrix is :
[[5 0 0]
 [0 6 0]
 [0 0 4]]
Random forest classifier for test size 0.4 has accuracy
96.66666666666667 %
the confusion matrix is :
[[19  0  0]
 [ 0 20  1]
 [ 0  1 19]]
Random forest classifier for test size 0.3 has accuracy
95.55555555555556 %
the confusion matrix is :
[[14  0  0]
 [ 0 17  1]
 [ 0  1 12]]
Random forest classifier for test size 0.2 has accuracy
96.66666666666667 %
the confusion matrix is :
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
Random forest classifier for test size 0.1 has accuracy 100.0 %
the confusion matrix is :
[[5 0 0]
```

```
 [0 6 0]
 [0 0 4]]
```

Part 3- Haar cascade Algorithms

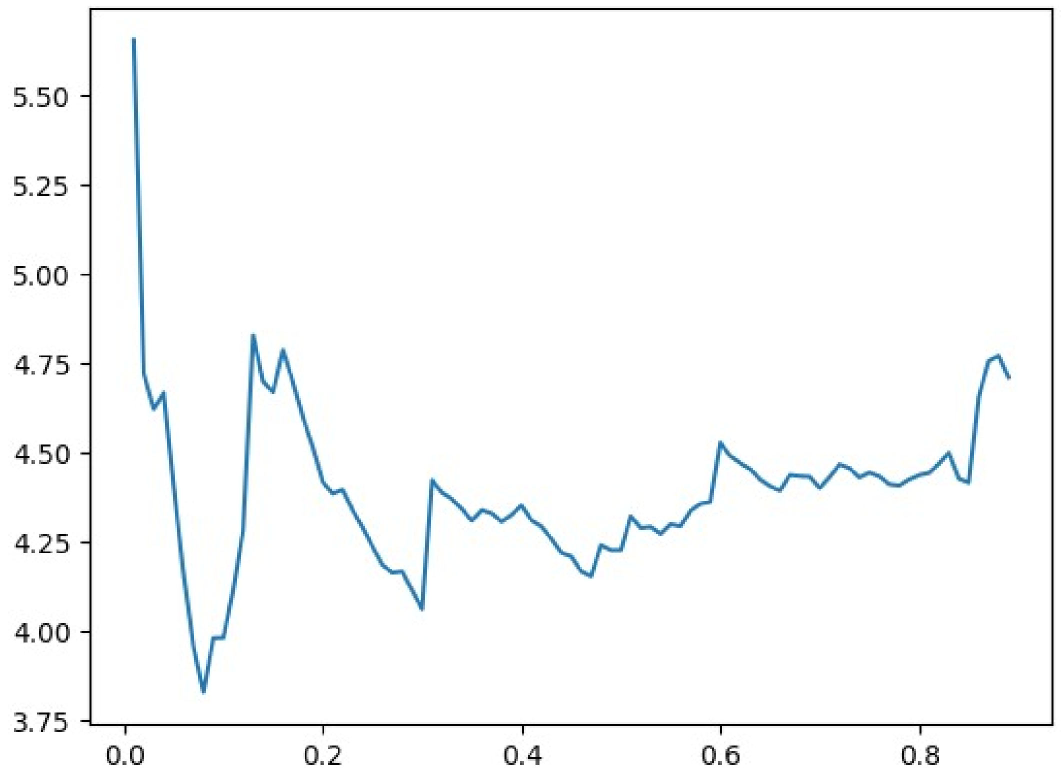
```
import cv2
cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
img=cv2.imread("pic.jpg"); #name of the image may change
g=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY);
faces = cascade.detectMultiScale(g, 1.3, 5)
for (x,y,w,h) in faces:
    marked_image = cv2.rectangle(img, (x, y), (x + w, y + h),
(255,0,0),2);
cv2.imshow('img',marked_image)
cv2.waitKey(5000)
cv2.destroyAllWindows()
```

Part 4- Boston Housing Price

```
import math
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv("boston.csv",sep=",");
x=data.iloc[:, :12];#this dataset had a column named 'black' because
at that time racial segregation was
                    #prevelant in USA, we will not be considering that
column in x
y=data.iloc[:, 13];
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(strategy="mean")
x=imputer.fit_transform(x)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
#size=[0.1,0.2,0.3,0.4,0.5]
l=list()
I=list()
for i in range(1,90,1):
    i=i/100
    xtr, xt, ytr, yt =
train_test_split(x,y,test_size=i,random_state=1)
    m=LinearRegression()
    m.fit(xtr, ytr)
    pred=m.predict(xt)
    error=mean_squared_error(yt,pred)
    l.append(math.sqrt(error))
    I.append(i)
    if((i*100)%10==0):
        print("The mean squared error for test size=",i,"is", error)
        print("The standard deviation for test size=",i,"is",
math.sqrt(error))
```

```
plt.plot(I,l)
plt.show()
```

```
The mean squared error for test size= 0.1 is 15.836059073217788
The standard deviation for test size= 0.1 is 3.9794546200726786
The mean squared error for test size= 0.2 is 19.512610393629064
The standard deviation for test size= 0.2 is 4.417308048306012
The mean squared error for test size= 0.3 is 16.484329681514076
The standard deviation for test size= 0.3 is 4.060089861260964
The mean squared error for test size= 0.4 is 18.931321487752694
The standard deviation for test size= 0.4 is 4.351013845961961
The mean squared error for test size= 0.5 is 17.856902786813038
The standard deviation for test size= 0.5 is 4.225742868042617
The mean squared error for test size= 0.6 is 20.49353443117692
The standard deviation for test size= 0.6 is 4.526978510129789
The mean squared error for test size= 0.7 is 19.357359794380255
The standard deviation for test size= 0.7 is 4.399699966404556
The mean squared error for test size= 0.8 is 19.679829541046544
The standard deviation for test size= 0.8 is 4.436195390314379
```



This graph shows the variation of standard deviation with test size.