MACHINE LEARNING – CODE

Amir Shokri

Exercise 3 – AND Decision tree with python

A AND B

A	B	RESULT
F	F	F
F	T	F
T	F	F
T	T	T

```
def myAnd(a, b):
        if a == False:
                return False
        elif a == True:
                if b == False:
                        return False
                elif b == True:
                        return True
```

Amir Shokri

Exercise 3 – OR Decision tree with python

A OR B

A	B	RESULT
F	F	F
F	T	T
T	F	T
T	T	T

```
def myOr(a, b):
        if a == False:
                if b == False:
                        return False
                elif b == True:
                        return True
        elif a == True:
                return True
```

Amir Shokri

Exercise 3 – XOR Decision tree with python

A XOR B

A	B	RESULT
F	F	F
F	T	T
T	F	T
T	T	F

```
def myXor(a, b):
        if a == False:
                if b == False:
                        return False
                elif b == True:
                        return True
        elif a == True:
                if b == False:
                        return True
                elif b == True:
                        return False
```

Amir Shokri

Exercise 3 – NOR Decision tree with python

A NOR B

A	B	RESULT
F	F	T
F	T	F
T	F	F
T	T	F

```
def myNor(a, b):
        if a == False:
                if b == False:
                        return True
                elif b == True:
                        return False
        elif a == True:
                if b == False:
                        return False
                elif b == True:
                        return False
```

Amir Shokri

Exercise 3 – NAND Decision tree with python

A NAND B

A	B	RESULT
F	F	T
F	T	T
T	F	T
T	T	F

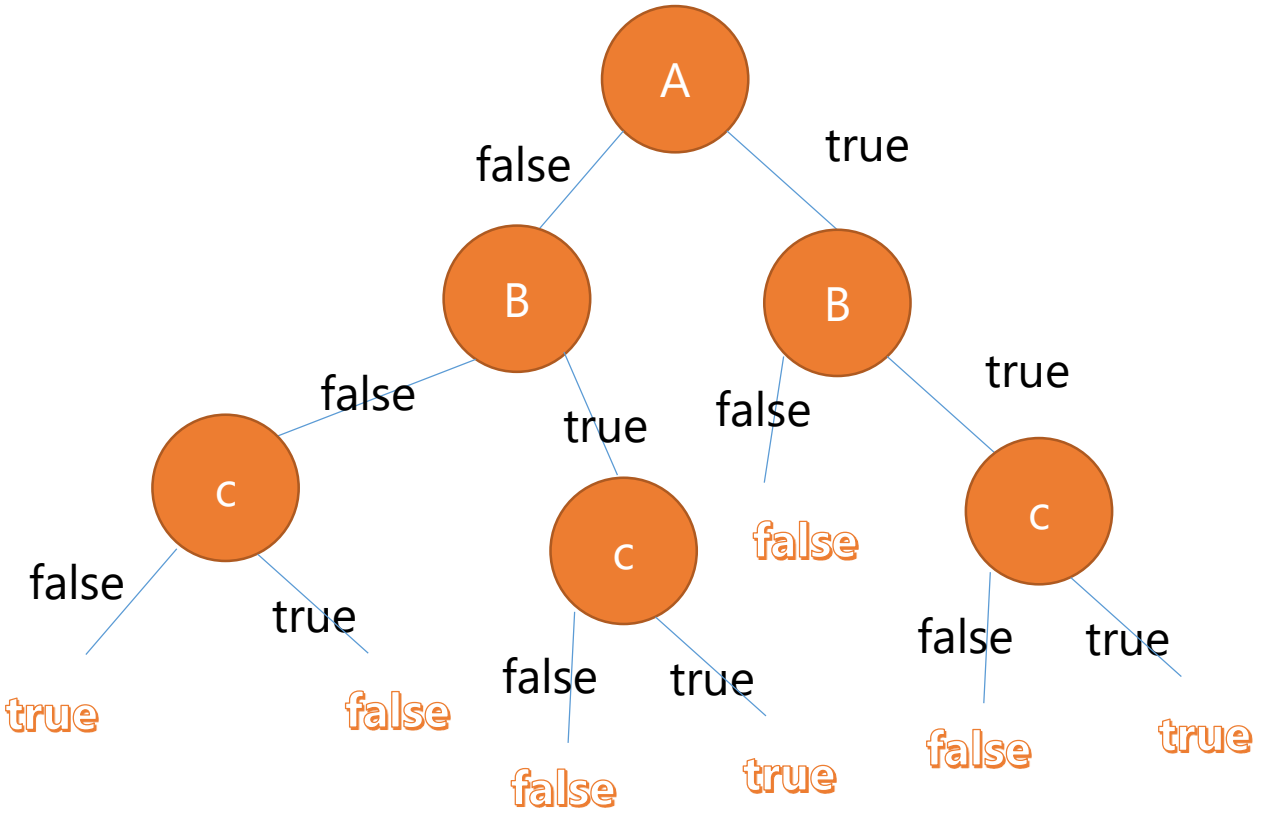
```
def myNand(a, b):
        if a == False:
                return True
        elif a == True:
                if b == True:
                        return False
                elif b == False:
                        return True
```

Amir Shokri

Exercise 3 – Decision Tree

(A AND B) OR (B OR C) XOR (A NAND B)

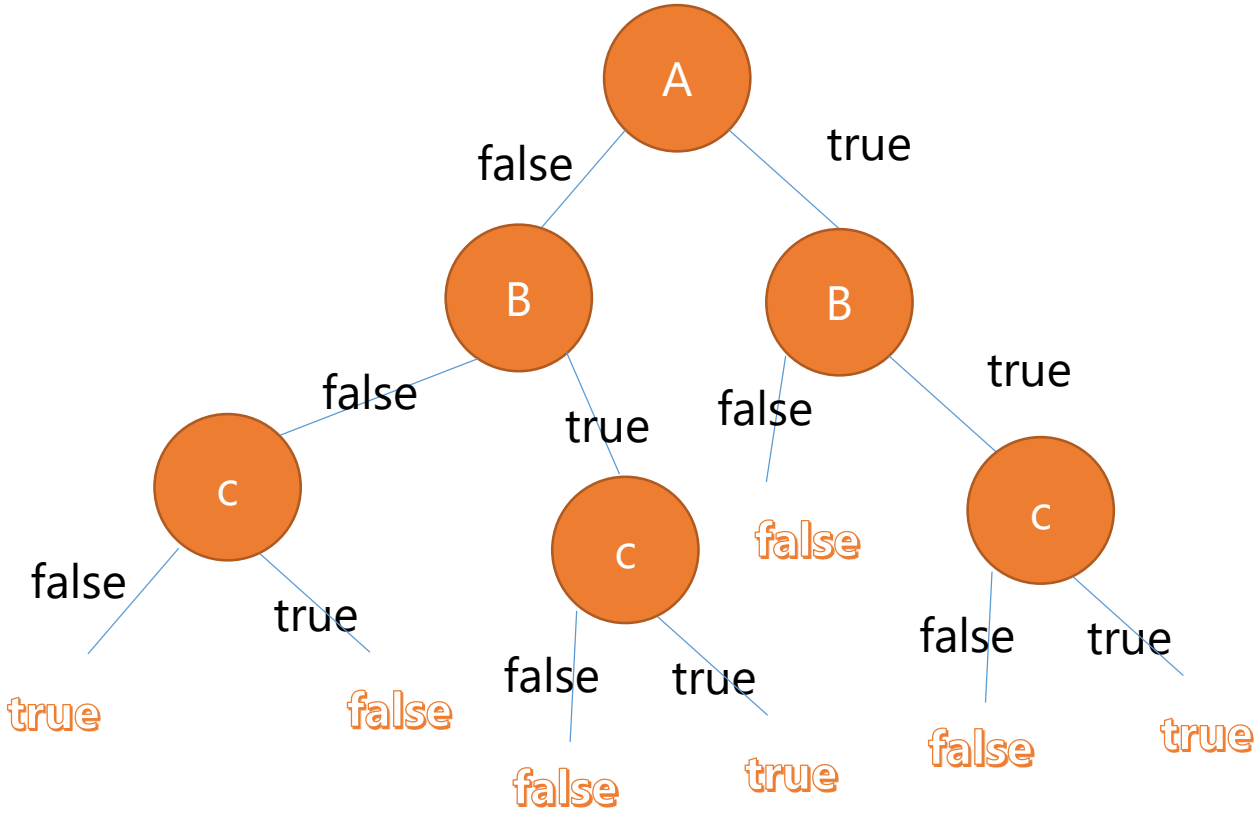
A	B	C	RESULT
F	F	F	T
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	F
T	T	F	F
T	T	T	T



Amir Shokri

Exercise 3 – Decision Tree

(A AND B) OR (B OR C) XOR (A NAND B)



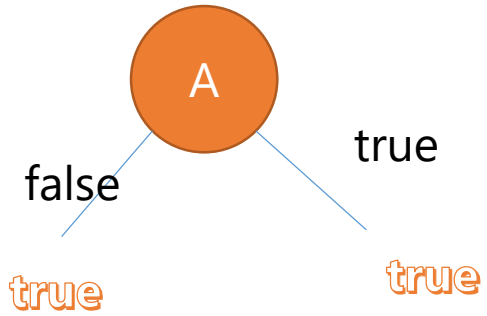

```
def sectionOne(a, b, c):
        if a == True:
                if b == True:
                        if c == True:
                                return True
                        elif c == False:
                                return False
                elif b == False:
                        return False
        elif a == False:
                if b == False:
                        if c == True:
                                return False
                        elif c == False:
                                return True
                elif b == True:
                        if c == True:
                                return True
                        elif c == False:
                                return False
```

Amir Shokri

Exercise 3 – Decision Tree

(A AND B OR C) OR (C NAND B)

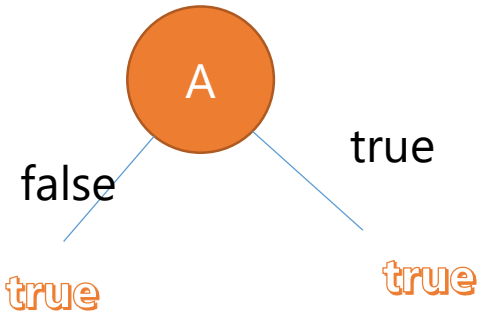
A	B	C	RESULT
F	F	F	T
F	F	T	T
F	T	F	T
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	T
T	T	T	T



Amir Shokri

Exercise 3 – Decision Tree

(A AND B OR C) OR (C NAND B)



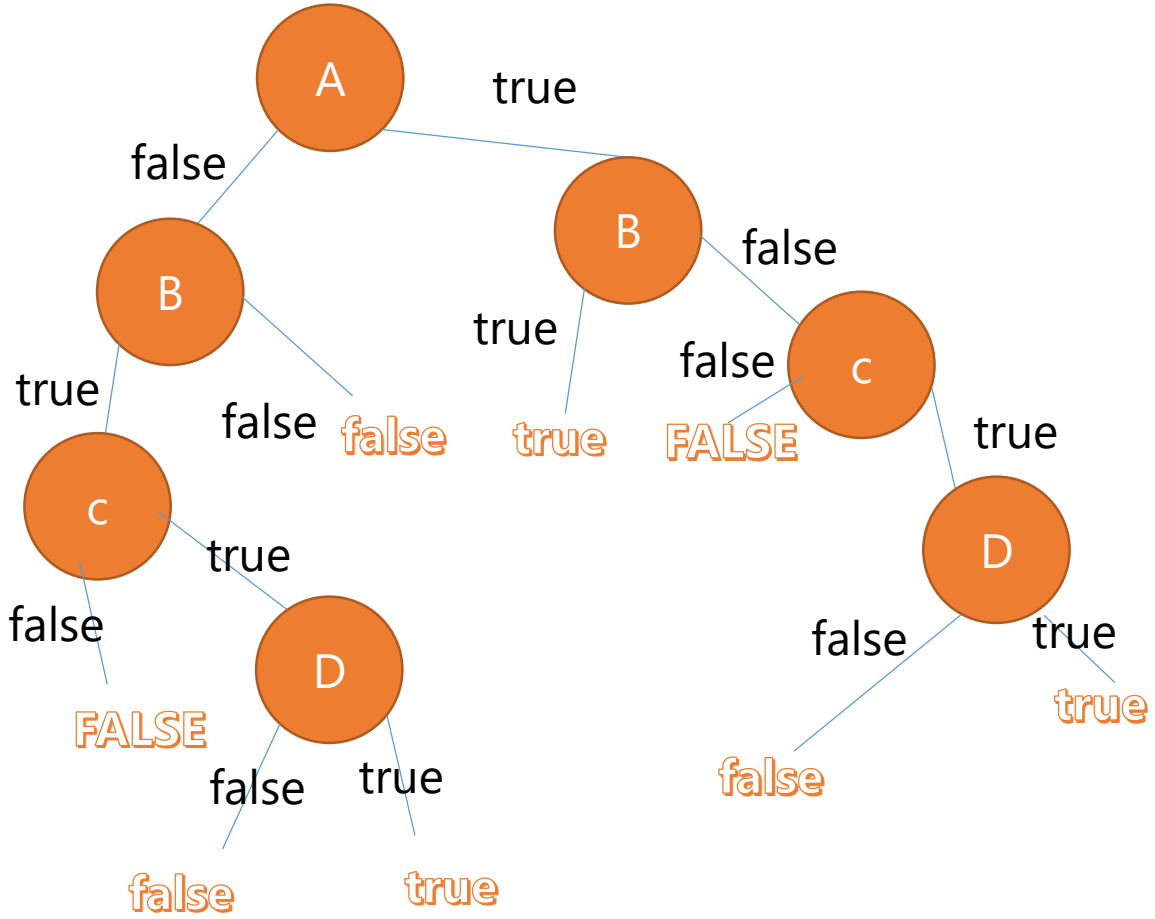

```
def sectionTwo(a, b, c):
        if a == True:
                return True
        elif a == False:
                return True
```

Amir Shokri

Exercise 3 – Decision Tree

(A XOR B) AND (B OR C) AND (C AND D)

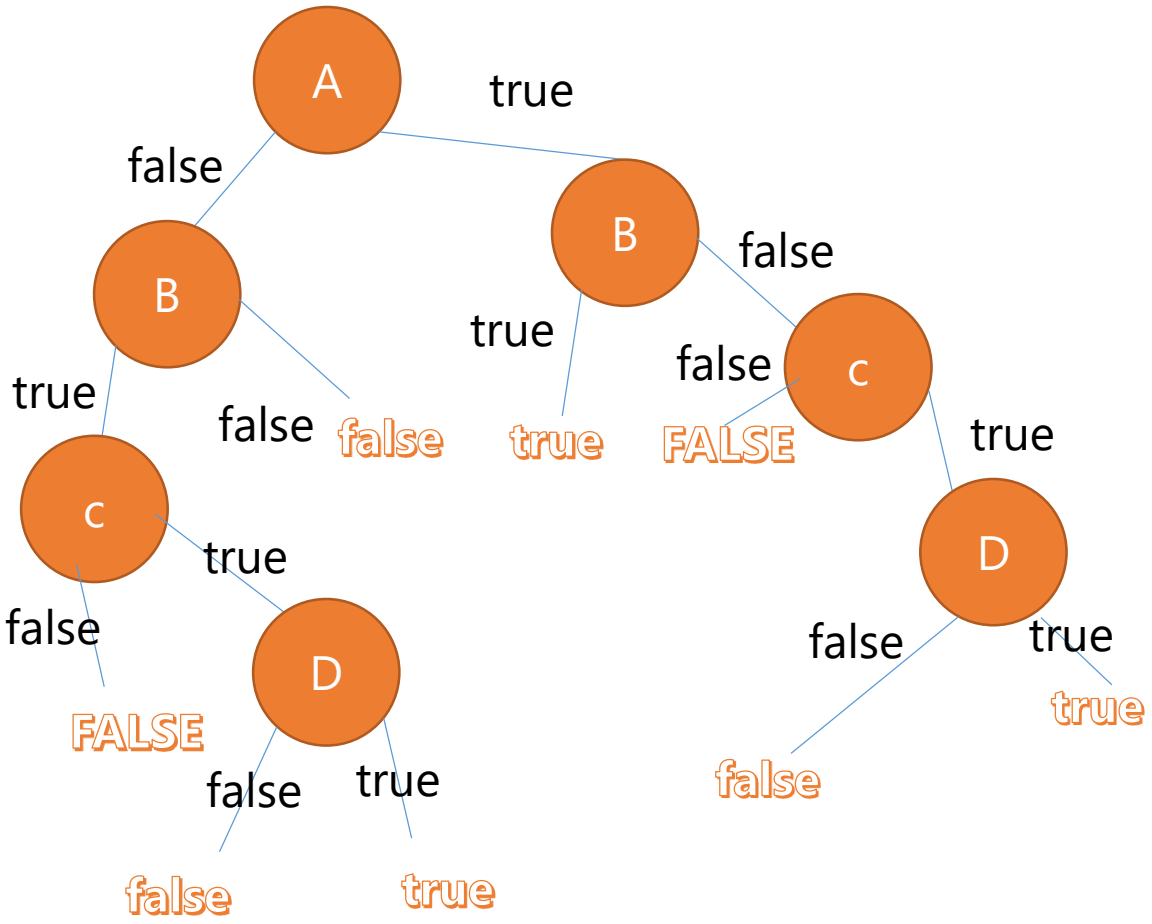
A	B	C	D	RESULT
F	F	F	F	F
F	F	F	T	F
F	F	T	F	F
F	F	T	T	F
F	T	F	F	F
F	T	F	T	F
F	T	T	F	F
F	T	T	T	T
T	F	F	F	F
T	F	F	T	F
T	F	T	F	F
T	F	T	T	T
T	T	F	F	F
T	T	F	T	F
T	T	T	F	F
T	T	T	T	F



Amir Shokri

Exercise 3 – Decision Tree

(A XOR B) AND (B OR C) AND (C AND D)




```
def sectionThree(a, b, c, d):
        if a == True:
                if b == True:
                        return True
                elif b == False:
                        if c == False:
                                return False
                        elif c == True:
                                if d == False:
                                        return False
                                elif d == True:
                                        return True
        elif a == False:
                if b == False:
                        return False
                elif b == True:
                        if c == False:
                                return False
                        elif c == True:
                                if d == False:
                                        return False
                                elif d == True:
                                        return True
```

Amir Shokri

Exercise 3 – FIND-S

Example	Size	Color	Shape	Class/Label
1	big	red	circle	No
2	small	red	triangle	No
3	small	red	circle	Yes
4	big	blue	circle	No
5	small	blue	circle	Yes

$h_0 = < \emptyset , \emptyset , \emptyset >$
$h_1 = < \emptyset , \emptyset , \emptyset >$
$h_2 = < \emptyset , \emptyset , \emptyset >$
$h_3 = <$ small, red, circle $>$
$h_4 = <$ small, red, circle $>$
$h_5 = <$ big, ?, circle $>$

Exercise 3 – FIND-S

```
def myFindS(data, arributeCount, exampleCount):
        h = ['null', 'null', 'null']
        print("h[0] : ", h)
        for i in range(exampleCount):
            if data[ i ][ arributeCount -1 ] == 'yes':
                for j in range(arributeCount - 1):
                    if h[j] == 'null':
                        h[j] = data[i][j]
                    elif h[j] != data[i][j]:
                        h[j] = '?'
            print("h[", i+1, "] : ", h)

arributeCount = 4
exampleCount = 5
myData = [ ['big', 'red', 'circle', 'no'], ['small', 'red', 'triangle',
'no'],['small', 'red', 'circle', 'yes'],['big', 'blue', 'circle', 'no'],['small',
'blue', 'circle', 'yes'] ]
myFindS(myData, arributeCount, exampleCount)
```

```
h[0] :  ['null', 'null', 'null']
h[ 1 ] :  ['null', 'null', 'null']
h[ 2 ] :  ['null', 'null', 'null']
h[ 3 ] :  ['small', 'red', 'circle']
h[ 4 ] :  ['small', 'red', 'circle']
h[ 5 ] :  ['small', '?', 'circle']
```

Amir Shokri

Exercise 3 – FIND-S

```
def myFindS(data, arributeCount, exampleCount):
        h = ['null', 'null', 'null']
        print("h[0] : ", h)
        for i in range(exampleCount):
            if data[ i ][ arributeCount -1 ] == 'yes':
                for j in range(arributeCount - 1):
                    if h[j] == 'null':
                        h[j] = data[i][j]
                    elif h[j] != data[i][j]:
                        h[j] = '?'
            print("h[", i+1, "] : ", h)

arributeCount = 4
exampleCount = 5
myData = [ ['big', 'red', 'circle', 'no'], ['small', 'red', 'triangle',
'no'],['small', 'red', 'circle', 'yes'],['big', 'blue', 'circle', 'no'],['small',
'blue', 'circle', 'yes'] ]
myFindS(myData, arributeCount, exampleCount)
```

```
h[0] :  ['null', 'null', 'null']
h[ 1 ] :  ['null', 'null', 'null']
h[ 2 ] :  ['null', 'null', 'null']
h[ 3 ] :  ['small', 'red', 'circle']
h[ 4 ] :  ['small', 'red', 'circle']
h[ 5 ] :  ['small', '?', 'circle']
```

Amir Shokri

Exercise 3 – ID3 ENTROPY

```
import math

def myEntropy(positiveCount, negativeCount, allCount):
        a = - (negativeCount / allCount) * (math.log((negativeCount / allCount), 2.0))
        b = - (positiveCount / allCount) * (math.log((positiveCount / allCount), 2.0))
        return a + b
```

Entropy: $\sum_{i=1} -p_i * \log_2(p_i)$

Amir Shokri

Exercise 3 – KNN

```
import random
import math

def sortRowWise(m):
    for i in range(len(m)):
        for j in range(len(m[i])):
            for k in range(len(m[i]) - j - 1):
                if (m[i][k] > m[i][k + 1]):
                    t = m[i][k]
                    m[i][k] = m[i][k + 1]
                    m[i][k + 1] = t

    return m

myNumbers = []

for x in range(1,50):
            myNumbers.append(x)
```

Amir Shokri

Exercise 3 – KNN

```
w, h = 3, 100;
myData = [[0 for x in range(w)] for y in range(h)]

for x in range(0, 99):
        myData[x][0] = random.choice(myNumbers)
        myData[x][1] = random.choice(myNumbers)

print("43 : ", myData[43])

for x in range(0,99):
        myData[x][2] = (myData[43][0]-(myData[x][0])^2) + (myData[43][1]-
(myData[x][0])^2)

myData = sorted(myData, key=lambda x: x[2])

print("[", myData[0][0], "][", myData[0][1], "]")
print("[", myData[1][0], "][", myData[1][1], "]")
print("[", myData[2][0], "][", myData[2][1], "]")
```

Amir Shokri

Exercise 3 – READ NEWS STRING

```
def freq(str):
    unique_words = set(str)
    for words in unique_words :
        print('Frequency of ', words , 'is :', str.count(words))


file1 = open("news/1.txt","r")
file2 = open("news/2.txt","r")
file3 = open("news/3.txt","r")
file4 = open("news/4.txt","r")
file5 = open("news/5.txt","r")

string1 = ""
string2 = ""
string3 = ""
string4 = ""
string5 = ""
```

Amir Shokri

Exercise 3 – READ NEWS STRING

```
for line in file1:
    string1 = string1 + line
string1 = string1.split(" ")

for line in file2:
    string2 = string2 + line
string2 = string2.split(" ")

for line in file3:
    string3 = string3 + line
string3 = string3.split(" ")

for line in file4:
    string4 = string4 + line
string4 = string4.split(" ")

for line in file5:
    string5 = string5 + line
string5 = string5.split(" ")

string = string1 + string2 + string3 + string4 + string5;

print(freq(string))
```

Amir Shokri