پروژه درخت تصمیم

پس از ران کردن پروژه با استفاده از داده های train درخت تصمیم ساخته میشود.
ابتدا باید data موجود که بصورت فایل csv را با استفاده از pandas بخوانیم.برای راحتی کار از map کردن استفاده میکنیم،به این معنی برای کلمه "male" معادل 0 و برای "female" معادل 1 قرار میدهیم.

```
csv_data = pandas.read_csv("titanic.csv").head(1000)
s = {'male' : 0 , 'female' : 1 , 'S':0 , 'C' : 1 , 'Q' : 2}
tickett = {'PC 17609' : 0 , '24160' : 1, '113781' : 2, '19952' : 3, '112050' : 4, '11769' : 5, '13502' : 6}
cabinn = {'A36' : 0 , 'B5' : 1, 'C22 C26' : 2, 'D7' : 3, 'E12' : 5}
csv_data['sex'] = csv_data['sex'].map(s)
csv_data['embarked'] = csv_data['embarked'].map(s)
csv_data['ticket'] = csv_data['ticket'].map(tickett)
csv_data["cabin"] = csv_data["cabin"].fillna('B5')
csv_data['cabin'] = csv_data['cabin'].map(cabinn)



csvdata = []
for row in csv_data.values:
    csvdata.append(row)
```

پس از خواندن 80 درصد داده ها به عنوان train برای ساخت درخت تابع زیر صدا زده میشود .

```
def makedecisiontree(node,attributes):
    if (len(attributes)==0):
        node.survived = check(node)
        return None
    if (len(node.examples ) ==0 ):
        node.survived = check(node.parent)
        return None
    node = whichquestion2ask(node,attributes,Entropy)
    for child in node.childs:
        makedecisiontree(child,list(filter(lambda s: s!= node.a,attributes)))
    return node

makedecisiontree(Root,ListOfattributes)
```

این method یک node به عنوان root و لیستی از توابع که این توابع هر کدام مجموعه سوالاتی هستند که قرار است پرسیده شود به عنوان ورودی میگیرد.
برای هر node یک فیلد attribute تعریف شده که آن مشخص میکند از هر node چه سوالی باید پرسیده شود.این فیلد با صدا زدن تابع whichquestion2ask با ورودی بچه های node و لیست سوال های باقی مانده مقدار دهی میشود.
به همین صورت برای بچه های node که لیستی از node ها است تابع makedecisiontree صدا زده میشود تا درخت خواسته شده ساخته شود.

```
def whichquestion2ask(Node,ListOfattributes,Entropy):
    entropy = 1
    selectedQuestion = None
    for a in ListOfattributes:
        Node.childs = a(Node)
        e = Entropy(Node.childs)
        if e < entropy:
            entropy = e
            selectedQuestion = a
    Node.childs = selectedQuestion(Node)
    Node.a = selectedQuestion
    return Node
```

این تابع قرار شد سوال مربوط به هر node را با محاسبه entropy یا gini index مشخص کند و بچه های ایجاد شده از پرسش سوال را در فیلد childs که لیستی از node است قرار دهد.

```
def Entropy(childs):
    Y = 0
    N = 0
    result = 0
    totalnumber = 0
    for j in range(len(childs)):
        totalnumber += len(childs[j].examples)
    for nd in childs:
        Y = 0
        N = 0
        for i in nd.examples:
            if i[10]==1:
                Y+=1
            else:
                N+=1
        if (N==0 or Y==0):
            result += 0
        else:
            result += (len(nd.examples)/totalnumber)*(Y/(Y+N)) *(math.log2((Y+N)/Y))
    return result
```

```
def gini_index(childs):
    Y = 0
    N = 0
    result = 0
    totalnumber = 0
    for j in range(len(childs)):
        totalnumber += len(childs[j].examples)
    for nd in childs:
        for i in nd.examples:
            if i[10]==1:
                Y+=1
            else:
                N+=1
        if (N==0 or Y==0):
            result += 0
        else:
            result += (len(nd.examples)/totalnumber)*((Y/(Y+N))*(1-(Y/(Y+N))) + (N/(Y+N))*(1-(N/(Y+N))))
    return result
```

آنتروپی و جینی ایندکس با استفاده از فرمول های بالا محاسبه میشود.

تابع makedecisiontree به صورت بازگشتی تا وقتی که لیست سوالات خالی نشده و node همچنان شامل examples میباشد صدا زده میشود و غیر این صورت برای آن node تابع check اجرا میشود که چک میکند .

```
def check(Node):
    y = 0
    n = 0
    if (Node !=None and Node.examples != None):
        for i in Node.examples:
            if (i[10] == 1):
                y+=1
            else:
                n += 1
        if y>n:
            return True
        elif n>y:
            return False
        else:
            if(Node.parent!=None):
                return check(Node.parent)
    if(Node.parent!=None):
        return check(Node.parent)
    else:
        return False
```

تابع check در examples های node پیمایش میکند و تعداد زنده و مرده بودن آن ها را چک میکند اگر تعداد زنده ها بیشتر بود وضعیت node.survived را زنده (true) و اگر تعداد مرده ها بیشتر بود وضعیت node.survived را مرده (false) می گذارد و در صورت برابر بودن متود check برای parent آن node صدا میزند.

حال پس از ساخته شدن درخت باید داده های test را به عنوان ورودی به درخت بدهیم و خروجی زنده یا مرده بودن آن را دریافت کنیم.

```
def isAlive(person,Root):
    childs = Root.a(person)
    j = 0
    selectedindex = 0
    for c in childs:
        j+=1
        if (len(c.examples)!=0):
            selectedindex = j
            break
    if(Root.childs!= None):
        if (Root.childs[selectedindex-1].childs == None):
            return Root.childs[selectedindex-1].survived
        else:
            return isAlive(person,Root.childs[selectedindex-1])

isAlive(newnode,Root)
```

تابع isAlive با گرفتن person و ریشه درخت تصمیم باید زنده یا مرده بودن person را حدس بزند.

صبا کیانوش
استاد درس : دکتر عبدی