در این پروژه قصد داریم درخت تصمیمی پیاده سازی کنیم تا با آموزش آن روی داده های مربوط به نفرات کشتی تایتانیک بتواند با دریافت داده وضعیت آن را (زنده ماندن یا خیر) پیشبینی کند.

```
csv_data = pandas.read_csv("titanic.csv")
csv_data = csv_data.fillna("")
```

با استفاده از کتابخانه ی pandas دیتای مربوط را استخراج می کنیم.

```
def CalculateSurviveds(node):
    y = len(list(filter(lambda x: x[-1]==1, node.exs)))
    n = len(node.exs)-y
    return y,n
```

این تابع با دریافت یک نود تعداد زنده و مرده های مربوط به دیتای ذخیره شده در آن نود را برمی گرداند.

```
def entropy(children):
    result = 0
    totalcount = 0
    for j in range(len(children)):
        totalcount += len(children[j].exs)
    for i in range(len(children)):
        y,n = CalculateSurviveds(children[i])
        if(y==0):
            result +=0
            continue
        result += (y/(y+n))*(math.log2((n+y)/y))*(len(children[i].exs)/totalcount)
    return result
```

این تابع entropy مربوط به یک لایه (با دریافت لیست نود های فرزند) را طبق رابطه محاسبه می کند.

```python
def gini_index(children):
    result = 0
    totalcount = 0
    for j in range(len(children)):
        totalcount += len(children[j].exs)
    for i in range(len(children)):
        y,n = CalculateSurviveds(children[i])
        if(y==0):
            result +=0
            continue
        result += ((y/(y+n))*(1-(y/(y+n))) + (n/(y+n))*(1-(n/(y+n))))*(len(children[i].exs)/totalcount)
    return result
```

این تابع gini index را طبق رابطه محاسبه می کند.
که بسته به نیاز و شرایط بین gini index و entropy یکی به عنوان تابع محاسبه گر و مبنا برای انتخاب سوال بعدی انتخاب می شود .

```python
def whichQuestion(node, Questions, ent_gini):
    min_etp = sys.maxsize
    question = None
    for q in Questions:
        if(entropy(q.children(node))<=min_etp):
            question = q
            min_etp = ent_gini(q.children(node))

    node.attribute = question
    node.children = question.children(node)
    return node
```

این تابع با دریافت یک نود و لیست سوال های مجاز برای پرسش ، در بین سوالات با محاسبه ی (entropy یا gini index) سوالی را که کمترین مقدار entropy یا gini index را دارد انتخاب می کند و نود های حاصل از آن سوال را به عنوان فرزندان نود ورودی ، و سوال انتخاب شده را به عنوان attribute نود ورودی قرار می دهد و نود را بر می گرداند.

```
def survived(node):
    y,n = CalculateSurviveds(node)
    if(y>n):
        return True
    return False
```

این تابع با استفاده از تابع CalculateSurviveds که پیش تر توضیح داده شد در صورتی که تعداد زنده های نمونه های یک نود بیشتر از تعداد مرده ها شود درست برمی گرداند و در غیر این صورت غلط.

```
def tree(node , Questions, ent_gini, accuracy):
    if(len(node.exs)==0):
        node.state = survived(node.parent)
        return None
    elif(len(Questions)==0):
        node.state = survived(node)
        return None
    elif(NoNeed2Question(node, accuracy)):
        node.state = survived(node)
        return None
    node = whichQuestion(node, Questions,ent_gini)
    for x in node.children:
        tree(x, list(filter(lambda x: x!=node.attribute, Questions)),ent_gini , accuracy)
    return node
```

این تابع اصلی می باشد که درخت تصمیم را می سازد و به صورت بازگشتی عمل می کند که دارای چندین شرط مختلف است

- اگر در نودی نمونه ای وجود نداشته باشد وضعیت نود والد آن را برمیگرداند.
- در صورتی که تمام سوالات پرسیده شود وضعیت آن نود را (طبق وضعیت نمونه های خودش) را تعیین می کند.
- در صورتی که تابع NoNeed2Question روی نمونه های نود درست باشد(دقت نمونه ها برابر یا بیشتر از پارامتر دقت ورودی (accuracy) وضعیت نود مشخص می شود.
- و در غیر این صورت برای نود طبق تابع whichQuestion فرزندان و attribute مشخص می شود و سپس روی فرزندان ، همین تابع به صورت بازگشتی صدا زده می شود.
- پارامتر های ورودی تابع عبارت اند از node که ریشه ی درخت می باشد و در انتها باز گردانده میشود ، Questions که لیست سوال های مجاز به پرسش می باشد ، ent_gini که تابع ورودی برای مبنا جهت تعیین سوال بعدی می باشد که می تواند تابع entropy یا gini index باشد و accuracy می باشد که پارامتر ورودی برای تابع NoNeed2Question می باشد که اگر دقت نمونه ها(دقت زنده ها یا مرده ها) برابر یا مساوی آن باشد ، وضعیت آن نود تعیین می شود و دیگر سوالی در ادامه ی آن نود پرسیده نمی شود.

در انتها خروجی همان Root می باشد که هم اکنون بعد از صدا زدن تابع tree روی آن دارای فرزندان و attribute های مختلف می باشد.

```python
def machine(treenode, person):
    lists = treenode.attribute.children(node(None, [person], None, None))
    index = 0
    for i in range(len(lists)):
        if(len(lists[i].exs)!= 0):
            index = i
            break
    if(treenode.children[index].state== None):
        machine(treenode.children[index], person)
    else:
        return treenode.children[index].state
```

این تابع ماشین حاصل ساخته شده می باشد که با دریافت یک داده (انسان به صورت آرایه ای از ویژگی های آن) با استفاده از Root ساخته شده از خروجی تابع tree وضعیت آن (زنده یا مرده بودن) را پیشبینی می کند.

```python
def trainTest(data, porpotion):
    total = len(data)
    index = int(porpotion*total)
    return [data[:index] , data[index::]]
```

این تابع با دریافت نسبت (porpotion) و داده ها به عنوان ورودی داده ها را به نسبت تقسیم می کند که برای این درخت تصمیم داده ها به نسبت 80 20 (80 درصد برای آموزش و 20 درصد برای تست) تقسیم و استفاده شده اند.

```
def accuracy(test_data, root):
    total = len(test_data)
    trues = 0
    for data in test_data:
        state = None
        if(machine(root, data) == True):
            state = 1
        else:
            state = 0
        if(state == data[-1]):
            trues+=1
    return trues/total
```

این تابع با دریافت داده های تست ، دقت درخت ساخته شده را روی این داده ها محاسبه می کند و نتیجه را بر میگرداند.

همچنین برای هر attribute یک کلاس ساخته شده که دارای یک تابع children می باشند که با دریافت نود به عنوان ورودی داده های آن را طبق بازه بندی های مناسب به نود های مختلف نسبت میدهد و جواب را به عنوان لیستی از نودها بر میگرداند. در واقع یک مجموعه داده را به چند مجموعه داده سامان دهی شده تقسیم میکند.

علی شیخ عطار
استاد : دکتر عبدی
هوش مصنوعی
دانشکده ی مهندسی کامپیوتر علم و صنعت
1401