به نام خدا

Diagnosis of disease based on fuzzy algorithms

در این مسئله ما قصد داریم از دیتا ست موجود ماتریس های R_o و R_c را به دست بیاوریم که ماتریس R_o به این معنی است که در هر بیماری چه علائمی ممکن است بروز پیدا کند و میزان بروز آن ها که حاوی دانش پزشکی در خصوص بیماری ها و ارتباطی که بیماری و **علائم ها با هم دارند میباشد .**

صرفا با این رابطه ما نمی توانیم اکتفا کنیم مثلا اگر فردی سردرد و یا بدن درد و سرفه داشته باشد این علائم ممکن است در چندین بیماری مثل آنفولانزا و کرونا و ... مشترک باشند، اما اگر فاکتوری مثل حس بویایی را در نظر بگیریم که فقط در یکی از این بیماری ها بود نشان دهنده تمایز بین آن ها است پس ما در ابتدا باید به این که چه نشانه هایی ممکن است بروز پیدا کنند و در مرحله بعد به فاکتور هایی که باعث متمایز شدن آنها هستند بپردازیم .مثلا بدن درد در همه و حس بویایی به ندرت و فقط در یکی، پس ماتریس R_c مشخص کننده میزان تمیز بودن بیماری با توجه به علائم از سایر بیماری ها است .

Occurance Relation :R_o

Confirmability relation :R_c

روش کار:

ابتدا داده های آموزش و تست را به نسبت 30 به 70 تقسیم می کنیم .

در مرحله اول با استفاده از داده های ترین و ماتریس R_o را به دست می آوریم. روش کار به این صورت است که در تمام ردیف های که لیبل مشترک دارند میانگین ستون هر علایم را به عنوان میزان بروز آن علائم در فلان بیماری در نظر میگیریم .

کد مربوطه :

```
def calculate_ocurrence_matrix(self):
    for index, class_label in enumerate(self.data.class_list):
```

```
        row_with_class_label =
self.data.trainData[self.data.trainLabel == class_label]
        row_r_ocurrence = [np.mean(column) for column in
row_with_class_label.T]
        self.r_ocurrence[index] = row_r_ocurrence
```

توضیحات کد :

انتخاب ردیف های با لیبل برابر

```
self.data.trainData[self.data.trainLabel == class_label]
```

محاسبه میانگین برای هر ستون

```
[np.mean(column) for column in row_with_class_label.T]
```

```
[[0.08297655 0.35484703 0.03729387 0.45895412 0.49113123 0.06570564
 0.04111625]
 [0.08036556 0.74582671 0.08317794 0.49319154 0.42075957 0.06224497
 0.17010047]
 [0.16122072 0.80526062 0.0527652  0.48946917 0.38661581 0.06265922
 0.11768476]
 [0.07174784 0.83412162 0.2795389  0.15253502 0.30376214 0.13524881
 0.14780322]]
```

در ادامه باید ماترست R_c که بیانگر میزان قابل تایید بودن بروز هر کدام از بیماری ها در بیمارمان است را ایجاد کنیم .

برای ساخت R_c بهترین الفا کاتی که در پروژه قبل به دست آوردیم را استفاده میکنیم . به این صورت که این آلفا کات را روی داده های ترین اعمال میکنیم و داده های که کمتر از این مقدار دارند را 0 و بقیه را 1 قرار میدهیم . در واقع انگار دیتا ست را به مقادیر 0 و 1 تغییر

میدهیم . و سپس در ادامه برای محاسبه میزان متمایز بودن هر بیماری نسبت به علایم اول ردیف های مربوط به هر بیماری را انتخاب میکنیم و در این ردیف ها میزان یا درصد هر علایم به این صورت به دست می آوریم که تعداد هر علایمی که مقدار 1 دارند تقسیم بر تعداد کل علایمی که 1 شده اند به دست می آوریم . و در ادامه برای هر بیماری یک ردیف داریم که این ردیف ها شامل میزان بروز هر علایم در آن بیماری است که با کنار هم قرار دادن این ردیف ها ماتریس ما ایجاد میشود.که تعداد سطر های آن در واقع بیماری ها و ستون های آن علایم ما هستند. که هر سلول در این ماتریس بیان کننده میزان متمایز بودن هر بیماری با توجه به علایم است .

.0 0.43646409 0.25966851 .0 0.29281768 0.01104972]]
[.0
.0 0.10526316 0.10526316 .0 0.73684211 .0]
[0.05263158
.0 .0 0.06666667 .0 0.86666667 0.06666667]
[.0
0.04347826 .0 .0 0.13043478 0.7826087 .0]
[[0.04347826

کد مربوطه :

```
def do_alpha_cut(self):
   for index, row in enumerate(self.data.trainData):
       for i, item in enumerate(row):
                       self.data.trainData[index][i] = item >=
self.alpha_cut and 1 or 0
```

این متد در واقع آلفا کاتی که از مرحله قبل به دست آورده ایم را روی داده های تست اعمال میکند .و بعد از اعمال آلفا کات با استفاده از متد زیر ماتریس r_c را به محاسبه می کنیم

```
def calculate_confumability_matrix(self):
   self.do_alpha_cut()
   for index, class_label in enumerate(self.data.class_list):
                                        rows_with_class_label     =
self.data.trainData[self.data.trainLabel == class_label]
                    row_r_confumability   =   [np.sum(column)   /
np.sum(rows_with_class_label) for column in
                                rows_with_class_label.T]

       self.r_confumability[index] = row_r_confumability
```

این متد ماتریس r_o را برای ما محاسبه میکند .

بعد از به دست آوردن ماتریس های R_O و R_c ما می توانیم با composition کردن آنها با داده های مربوط به هر بیمار و به دست آوردن ماتریس های R2 , R1 که در واقع بیان کننده میزان احتمال ابتلا بیمار به هر بیماری و دیگری در واقع تایید بر ماتریس اول است نوع بیماری بیمار ورودی را حدس بزنیم . به این صورت که بعد از composition ماتریس های که ما داریم هر سطر بیان کننده یک بیمار و ستون های هر سطر میزان امکان ابتلای آن بیمار به یک بیماری را بیان میکند .

برای اجرای کد کافی است که فایل های موجود در فایل زیپ را به همین صورت که هست در یک پروژه پایتون قرار داده و فایل main را اجرا کنید . در فایل زیپ برای فاز دوم پروژه 3

فایل وجود دارد که شامل FuzzyClusteringPartTwoAndThree , main , services که با ترتیب برای اجرا و بعضی متد های مورد نیاز برای لود و تنظیمات مربوط به دیتابیس و کلاس های مربوط به کلاسترین در فایل FuzzyClusteringPartTwoAndThree قرار دارند .
بعد از اجرا کد با حذف بعضی از ستون ها در میزان acc به دست آمده تغییراتی مشاهده می شد ولی باز هم acc قابل قبولی نداشتیم

نتیجه با دیتا بیس کامل :
Acc train is 53.002070393374744
Acc test is 25.6198347107438

با حذف ستون اول :
Acc train is 85.16949152542372
Acc test is 46.61016949152542

با حذف دو ستون اول :
Acc train is 74.74120082815735
Acc test is 33.057851239669425

با حذف سه ستون اول :

Acc train is 74.3801652892562

Acc test is 41.32231404958678

.

.

.

توضیحات کد فاز سوم :

در این مرحله ما قصد داریم که الگوریتم c-mean را با استفاده از کتابخانه های آماده های پایتون بر روی دیتاست موجود اعمال کنیم . برای این کار ما از کتابخانه fcmeans استفاده کردیم که کد آن به شرح زیر است .

```
import numpy as np
from fcmeans import FCM
from matplotlib import pyplot as plt
from servises import *


raw_data = load_data('DataSets/hcvdat0.csv', array=True)
raw_data[:, 1] = [x[0] for x in raw_data[:, 1]]
data = Data(data=raw_data, data_range=(4, 13), label_range=1,
bias=False, normal=True)


fcm = FCM(n_clusters=5)
fcm.fit(data.trainData)
fcm_centers = fcm.centers
fcm_labels = fcm.predict(data.trainData)
acc = accuracy_metric(fcm_labels,data.trainLabel )

print(acc)
```

نتیجه بعد از اجرا های متفاوت :

accuracy train is 21.325051759834366

accuracy train is 23.96694214876033

حذف ستون اول :

accuracy train is 19.06779661016949

accuracy train is 15.254237288135593

حذف ستون دوم :

accuracy train is 26.91511387163561

accuracy train is 23.140495867768596

البته قابل ذکر است که که با توجه به انتخاب تعداد کلاس های مورد انتظار نتایج متفاوتی مشاهده میشد ولی در اینجا به دلیل وجود تنها 5 کلاس ما تعداد را 5 در نظر گرفتیم .