

بسمه تعالی

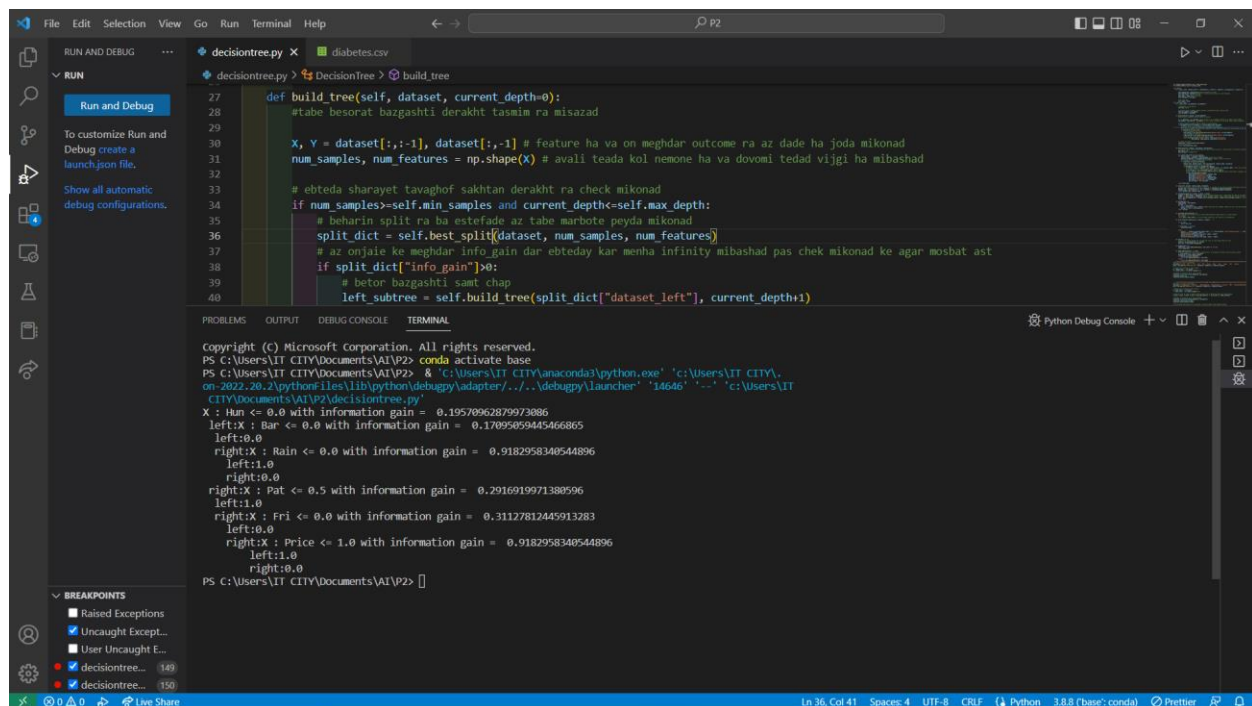
پروژه پیاده سازی درخت تصمیم برای تشخیص دیابت

آیسا میاهی نیا

استاد عبدی

برای این پروژه ابتدا می بایست کلاس نود تعریف می کردیم که این کلاس باید شامل اتریوت هایی بود که بعداً در ساختن بازگشتی درخت با آن ها نیاز پیدا می کردیم. سپس یک کلاس هم برای ساخت خود درخت تصمیم داریم که در آن توابعی برای ساده تر شدن کار داریم. ابتدا تابع اصلی که برای ساخت درخت می باشد، که با توجه به شروطی که داریم ابتدا چک می کند شرط توقف ساختن درخت رعایت شده است یا خیر. این شروط برای این می باشند که درخت ما خیلی بزرگ نشود چون اگر دیتاست بزرگی داشته باشیم، عمق درخت خیلی زیاد می شود. حال که شرایط برقرار است باید به دنبال بهترین ویژگی برای split کردن بگردد. که با استفاده از فرمول انتروپی و `information gain` توابع هر یک پیاده سازی شدند. یک فور روی همه ویژگی ها می زنیم و برای گسسته سازی همه مقادیر یونیک را به دست می آوریم و انتروپی و `information gain` را به ازای آن مقادیر یونیک محاسبه کرده، و بهترین ویژگی را انتخاب می کنیم و با استفاده از آن با کمک تابع `split` درخت را تیکه کرده و به طور بازگشتی به کار خود ادامه می دهیم. تابع `print` هم که درخت را نشان می دهد و همچنین نشان دهید که در هر نود براساس کدام ویژگی و با چه مقدار انتروپی و `information gain` داده ها را جدا کرده است.

برای تست کردن ابتدا از داده های رستوران که موجود در اسلاید ها بود استفاده کردیم به این منظور یک فایل CSV با اطلاعات این جدول درست کردیم و از انجایی که قرار بود همه داده های جدول به عنوان داده آموزشی استفاده شود پس نیازی به جداسازی نداریم و میتوانید درخت ساخته شده را در تصویر 1 مشاهده کنید.



```
def build_tree(self, dataset, current_depth=0):
    #tabe besorat bazgashti derakht tasam ra misazad
    X, Y = dataset[:, :-1], dataset[:, -1] # feature ha va on megdar outcome ra az dade ha joda mikonad
    num_samples, num_features = np.shape(X) # avall teada kol nemone ha va dovomi tedad vijgi ha mibashad

    # ebteda sharayet tavaghof sakhtan derakht ra check mikonad
    if num_samples > self.min_samples and current_depth < self.max_depth:
        # beharin split ra ba estefade az tabe marbote peyda mikonad
        split_dict = self.best_split(dataset, num_samples, num_features)
        # az onjaie ke megdar info gain dar ebteday kar menha infinity mibashad pas chek mikonad ke agar moshat ast
        if split_dict["info_gain"] > 0:
            # betor bazgashti samt chap
            left_subtree = self.build_tree(split_dict["dataset_left"], current_depth+1)
```

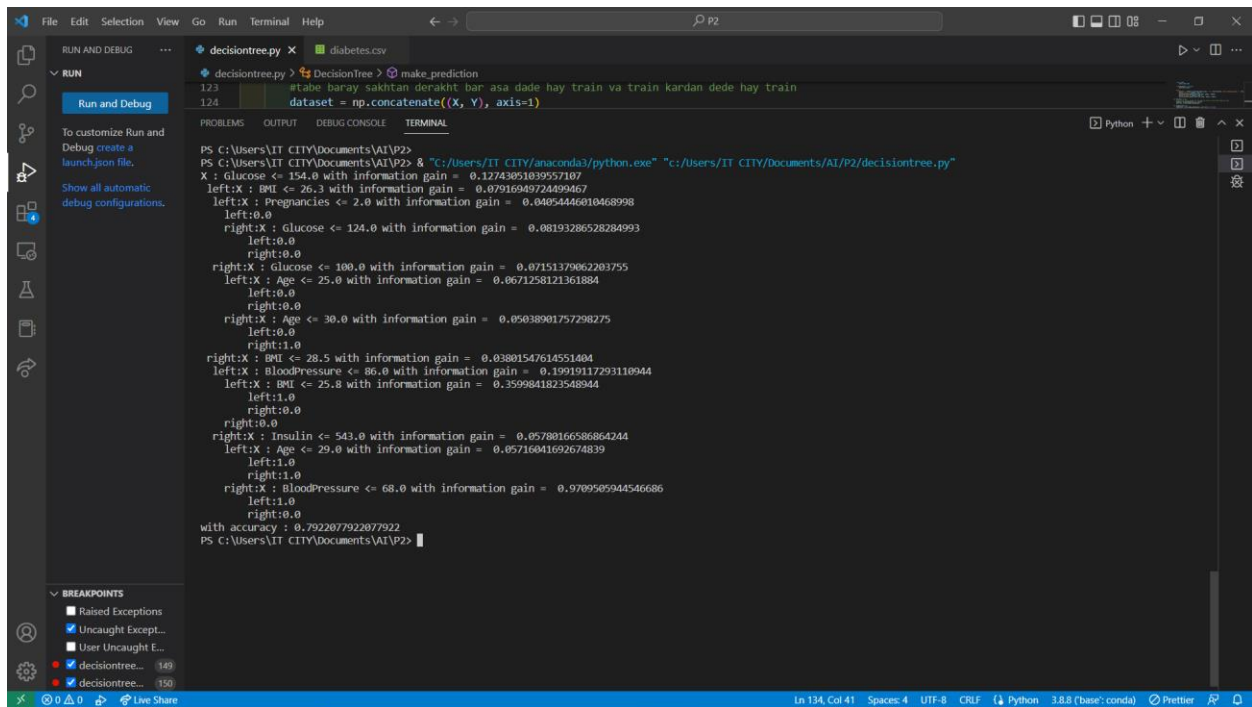
```
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Users\IT CITY\Documents\AI\p2> conda activate base
PS C:\Users\IT CITY\Documents\AI\p2> & "c:\Users\IT CITY\anaconda\python.exe" "c:\Users\IT CITY\on-2022-20-2\python\lib\python\debugpy\adapter\..\..\debugpy\launcher" "14646" "-..." "c:\Users\IT CITY\Documents\AI\p2\decisiontree.py"
X : Hun <= 0.0 with information gain = 0.19570962879973086
left:X : Bar <= 0.0 with information gain = 0.17095059445466865
left:0.0
right:X : Rain <= 0.0 with information gain = 0.9182958340544896
left:1.0
right:0.0
right:X : Pat <= 0.5 with information gain = 0.2916919971380596
left:1.0
right:X : Fri <= 0.0 with information gain = 0.31127812445913283
left:0.0
right:X : Price <= 1.0 with information gain = 0.9182958340544896
left:1.0
right:0.0
PS C:\Users\IT CITY\Documents\AI\p2> []
```

تصویر 1 - خروجی درخت به ازای داده های رستوران

یکی از چالشهایی که برای این قسمت داشتم نحوه مقداردهی جدول داده های رستوران بود چون بعضی مقادیر گسسته و برخی پیوسته بودند که من برای مقدار دهی همه را گسسته در نظر گرفتم و به هر کدام اعدادی را از 0 تا n (بسته به تنوع مقادیر) نسبت دادم.

حال داده های دیابت را با استفاده از تابع آماده `train_test_split` به دو قسمت آموزشی و آزمایشی (20 درصد آزمایشی و 80 درصد آموزشی) تقسیم کردم و با کمک داده های آموزشی درخت تصمیم را ساختم که می توانید آن را در تصویر 2 مشاهده کنید. سپس برای امتحان کردن آن داده های آزمایشی را روی درخت امتحان کردم و از آنجایی که جواب درست را هم از داده های اصلی میدانیم میتوانیم، صحت آن را با استفاده از تابع آماده `accuracy_score` بسنجیم که همانطور که باز در تصویر 2 میتوان `accuracy` آن را مشاهده کرد

که مقدار نسبتاً خوبی می باشد.



```
File Edit Selection View Go Run Terminal Help
P2
decisiontree.py x diabetes.csv
decisiontree.py > DecisionTree > make_prediction
123 #tabe baray sakhtan derakht bar asa dade hay train va train kardan dede hay train
124 dataset = np.concatenate((X, Y), axis=1)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\IT CITY\Documents\AI\P2>
PS C:\Users\IT CITY\Documents\AI\P2> & "C:/Users/IT CITY/anaconda3/python.exe" "C:/Users/IT CITY/Documents/AI/P2/decisiontree.py"
X : glucose <= 154.0 with information gain = 0.12743051039557107
left:X : BMI <= 26.3 with information gain = 0.07916949724499467
left:X : Pregnancies <= 2.0 with information gain = 0.04054446010468998
left:0.0
right:X : Glucose <= 124.0 with information gain = 0.08193286528284993
left:0.0
right:0.0
right:X : Glucose <= 100.0 with information gain = 0.07151379062203755
left:X : Age <= 25.0 with information gain = 0.0671258121361884
left:0.0
right:0.0
right:X : Age <= 30.0 with information gain = 0.05038901757298275
left:0.0
right:1.0
right:X : BMI <= 28.5 with information gain = 0.03801547614551404
left:X : BloodPressure <= 86.0 with information gain = 0.19919117293110944
left:X : BMI <= 25.8 with information gain = 0.3599841823548944
left:1.0
right:0.0
right:0.0
right:X : Insulin <= 543.0 with information gain = 0.05780166586864244
left:X : Age <= 29.0 with information gain = 0.05716041692674839
left:1.0
right:1.0
right:X : BloodPressure <= 68.0 with information gain = 0.9709505944546686
left:1.0
right:0.0
with accuracy : 0.7922077922077922
PS C:\Users\IT CITY\Documents\AI\P2>
```

تصویر 2 – درخت تصمیم داده های دیابت و مقدار accuracy

من در درخت بالا عمق ماکسیمم را 3 در نظر گرفتم اگر این عمق را کمتر مثلاً 2 در نظر بگیریم، مقدار accuracy ما کمتر می شود از طرفی اگر هم خیلی زیاد در نظر بگیریم درخت ما خیلی بزرگ می شود و زمان زیادی می برد اجرا کردن کد. باید با توجه به داده هایی که داریم عمق درخت را انتخاب کنیم.

بیش برآزشی که استفاده کردم همین محدود کردن عمق و محدود کردن تعداد نمونه در هر نود می باشد که از نوع pre pruning می باشد چون قبل از اینکه درخت کامل بشود این شرایط روی آن اعمال شده است. و از زیاد شدن عمق و حتی یک سری داده هایی که مقادیر نزدیک بهم دارند و مثل نویز می باشند و تاثیری در جداسازی ندارند، جلوگیری می کند تا accuracy ما بیشتر شود.

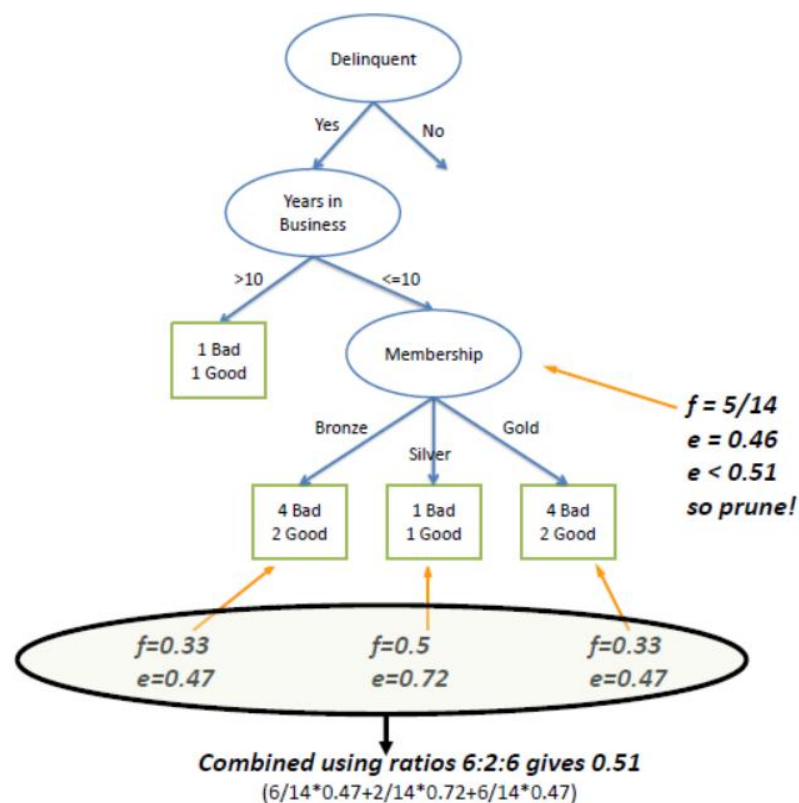
می توان از روش های post pruning هم استفاده کرد، که کمی پیاده سازی آن دشوار می باشد یکی از این روش ها استفاده از Error estimation می باشد که با استفاده از فرمول زیر می توان آن را محاسبه کرد.

$$e = \left(f + \frac{z^2}{2N} + z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) / \left(1 + \frac{z^2}{N} \right)$$

Where:

- f is the error on the training data
- N is the number of instances covered by the leaf
- z from normal distribution

که اگر e محاسبه شده برای یک نود از e محاسبه شده فرزندانش کمتر بود، فرزندانش آن را هرس می کنیم. همانند مثال زیر:



The error rate at the parent node is 0.46 and since the error rate for its children (0.51) increases with the split, we do not want to keep the children.