سید حامد سادات 400521405

پروژه درخت تصمیم

این متن مستندات پروژه درخت تصمیم است

- **الگوریتم درخت تصمیم:**

الگوریتم به این شکل هست که وقتی info_gain ویژگی هارو بدست آورد،ویژگی که بیشترین gain را دارد برمیدارد و به تعداد تمام کلاس های داده،یک node درست میکند و همین کار را تکرار میکند

این الگوریتم بازگشتی است

الگوریتم به صورت DFS کار میکند،یعنی اول node های فرزند را میسازد تا جایی که نتوان بیشتر تقسیم کرد،بعد الگوریتم برمیگردد و node بقیه کلاس ها را میسازد

الگوریتم این کار را انجام میدهد تا جایی که به stopping criteria برسد که شامل حداکثر عمق درخت،حداکثر تعداد node ها و حداقل info gain میشود

به صورت پیشفرض حداقل gain مقدار محدودیتی برای عمق و تعداد node ها درنظر گرفته نشده اما میتوان در تابع به آنها مقدار داد

تابع info gain که برای بدست آوردن information gain استفاده میشود،از یکی از دو تابع entropy و gini index استفاده میکند که میتوان تابع را در هنگام ساخت اولیه درخت تصمیم مشخص کرد

الگوریتم تابع داخلی برای تست دارد
وقتی عملیات train تمام شد،تابع به صورت اتوماتیک فعال شده و داده های train را در هر row داخل درخت تصمیم تست میکند و نتیجه تست را به صورت فایل txt به اسم train_result.txt ذخیره میکند

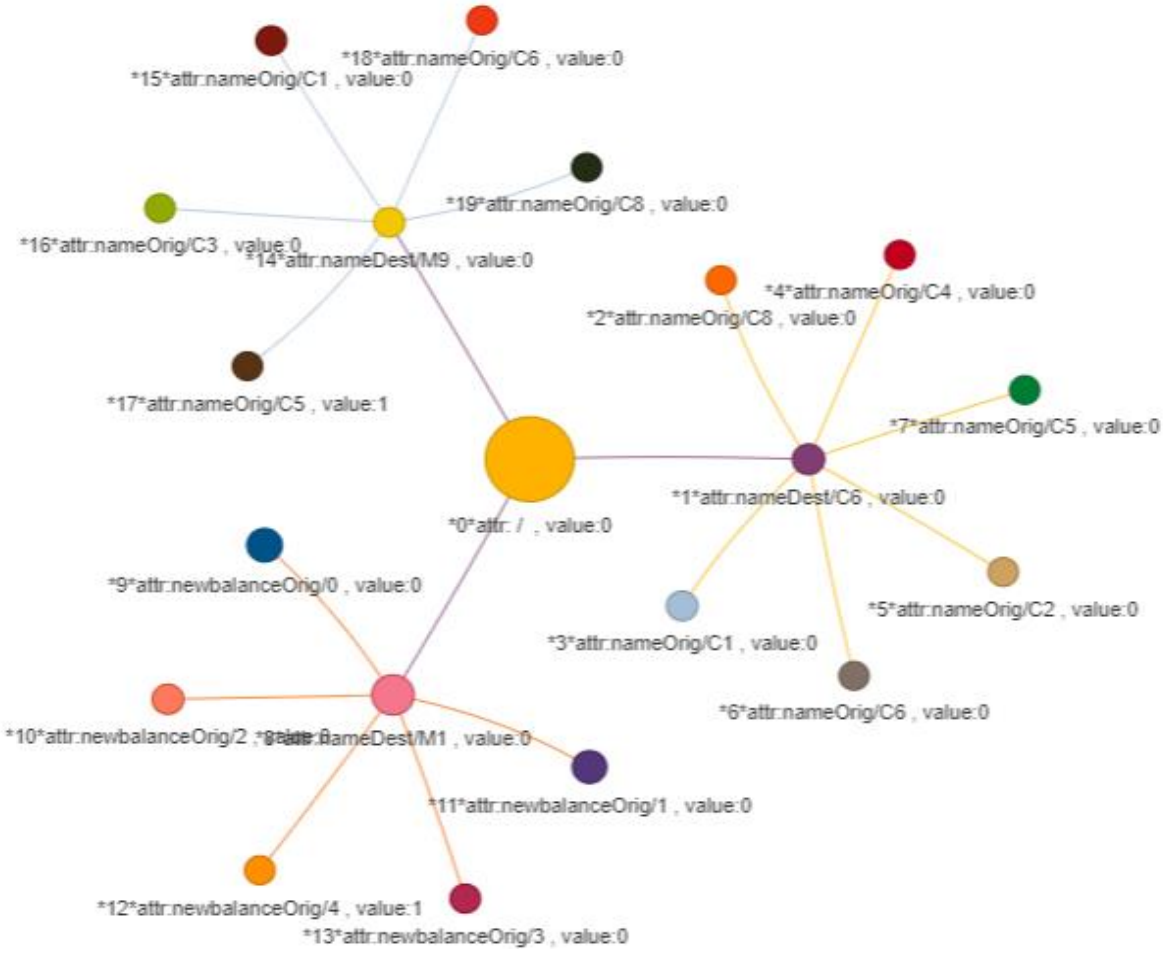
بعد از آن نیز با صدا زدن تابع test میتوان داده های مورد نظر را به آن داد و بعد از اتمام آن تابع فایل خروجی با نام test_result.txt میدهد که شامل نتیجه تست است که شامل تعداد درست،تعداد کل row و نسبت آنها(دقت حدس تابع) میشود

تابع تست هر row داده را به داخل تابع predict میدهد و در آنجا،

row داده شده،مقدار ویژگی داخل آن توسط node گرفته میشود و بر حسب کلاس آن به node مربوطه فرستاده شده تا به برگ برسیم و نتیجه را خروجی بگیریم

به دلیل استفاده از dictionary برای دسترسی به node فرزند مقادیر ممکنه برای یک کلاس میتواند غیر عددی نیز باشد

در داخل پیاده سازی درخت تصمیم یک تابع برای visualize کردن آن طراحی شده میتوان گره ها با جزییات آن را دید



این visual یک درخت تصمیم با محدود کردن عمق به 3 و حداکثر 20 تعداد node است

میتوان اطلاعات بیشتری در هر node نیز نمایش داد و رنگ آنها را بر حسب value آنها گذاشت

داده هایی که به درخت تصمیم داده میشود،تمام attribute های آن باید کلاس بندی شده باشند و آخرین ستون داده ها باید label داده باشد

الگوریتم میتواند برچسب های غیر 0 و 1 را نیز پردازش کند

مثلا برچسب هایی که 3 تایی یا 4 تایی هستند هم میتواند پردازش شود

- **داده ها:**

برای آنالیز داده ها از کتابخانه matplotlib و ydata-profiling استفاده شده

داده نیاز به آنالیز دقیق و جزئی و زمانبر برای کلاس بندی دقیق و مواثر دارد

داده ها زیاد بود و دسته بندی بازه میبایستی به صورت نمایی معکوس میبود تا واقعا مواثر باشند

داده هایی مثل step و type به دلیل گسسته بودن و کم بودن تعداد نیازی به گسسته سازی نداشتند برای همین به همان شکل ماندند

داده های عددی پیوسته به بازه های 10 تایی تقسیم شدند و تا حد امکان داده های نزدیکتر به 0 کوچکتر بودند

داده های اسمی نیز به 2 حرف اول آنها دسته بندی شدند

ولی از آنجایی که داده تعداد زیادی 0 داشت،2 دسته داده نمونه برداری شده آماده کردم به صورت زیر:

isFraud_test:

isFraud==1 ➔1000

isFraud==0 ➔2000

isFraud_train:

isFraud==1 ➔1000

isFraud==0 ➔2000

داده ها قبل از برداشتن shuffle شده و با اطمینان داده های داخل test در train وجود ندارند

نتایج train و test داده ها:

در کل داده:

داده train:دقت تقریبا 98%

داده test:دقت تقریبا 97%

نحوه تقسیم:80% اول داده به train و بقیه به test

از آنجایی که تعداد 0 ها زیاد بود احتمال اینکه این موضوع روی زیاد بودن دقت داده تاثیر گذاشته باشد زیاد است

برای همین از داده های تقسیم شده ایی که بالاتر گفتم استفاده کردم

داده های متوازن شده:

داده train:71%

داده test:60%

با زیاد کردن حجم داده،معمولا train و test دقت بهتری پیدا میکرد حتی برای پیدا کردن fraud ها

درست بازع بندی کردن داده ها نیز تاثیر داشتند اما به طور کل داده ها به نسبت ویژگی به سختی تقسیم میشدند و برای بالاتر رفتن info_gain باید تقسیم بندی داده ها دقیق تر انجام میشد

تفاوت بین gini_index و entropy نیز زیاد نبود

تفاوت دقت آنها حداکثر به 3 درصد میرسید و entropy دقت بیشتری داشت