به نام خدا

تمرین اول یادگیری ماشین

پوریا نخعی ۹۹۴۴۳۰۱۴

توضیح کلی الگوریتم ID5:

این الگوریتم درخت تصمیم را به صورت افزایشی میسازد به این صورت که داده را یکی یکی دریافت میکند و بهترین فیچری که در هر عمق درخت میتواند قرار بگیرد را در مجموع داده های قبلی و جدید محاسبه میکند و اگر تغییری در بهترین فیچر مشاهده کند فیچر بهترین جدید را از پایین درخت به بالا می آورد. تشخیص اینکه یک فیچر بهترین است یا خیر با استفاده از معیار آنتروپی بعد از تقسیم صورت میگیرد.

توضیح بخش های اصلی کد و الگوریتم های pull up و merge:

کد پیاده سازی شده دو کلاس اصلی دارد که کلاس اول Id5_Tree و دیگری Id5_Node است. کلاس اول مسئولیت نگهداری ریشه درخت را دارد و همچنین اینکه بتواند درخت را چاپ کند و داده را یکی یکی به ریشه دهد تا آموزش ببیند. کلاس دوم معرف نود درخت است که در آن نام فیچر مربوط به نود، پیش بینی نود(درصورت برگ بودن)، فرزندان، پدر و عمق و داده هایی که به این نود رسیده اند نگهداری میشوند.
تابع update_attribute: بهترین فیچر را با استفاده از انتروپی از تابع find_best_attribute دریافت میکند و در صورتی که قبلا فیچری برای این نود انتخاب نشده است تابع expand را فراخوانی میکند تا درخت با استفاده از آن گسترش یابد(نود فعلی به چند شاخه شکسته شود). اما اگر بهترین فیچر با فیچر قبلی این نود یکسان نبود تابع pull_up را به صورت بازگشتی رو فرزندانش فراخوانی میکند. اما اگر هیچ یک از دوشرط قبلی برقرار نبود تنها propagation انجام میدهد تا داده جدید در درخت نفوذ کند.

الگوریتم pull_up: این الگوریتم به صورت بازگشتی پیاده شده است و شروط اولیه اش این است که اگر فیچر مورد نظر رسیدیم بازگشت انجام میدهیم یا اگر در برگ بودیم ابتدا درخت را با فیچر مورد نظر expand میکنیم و سپس بازگشت انجام میدهیم. حال پس از بازگشت فرض این است که تمام فرزندان نود فعلی تبدیل به فیچر مورد نظر برای pull up شده اند. در این نقطه باید به ازای هر شاخه در نود فعلی یک درخت ایجاد کنیم و در هر یک از درخت های ایجاد شده جای دو فیچر را عوض کنیم حال باید درخت های جدید بدست آمده را merge کنیم. عملیات merge هم چیزی جز جمع زدن شاخه های عمق اول دو درخت نیست. تنها نکته ای که پس از merge باید به آن دقت شود این است که اگر نودی ایجاد شد که کاملا pure است آن را collapse کنیم(contraction).

آموزش و آزمون درخت تصمیم

آزمون:

ابتدا داده های تکراری را از مجموعه دادگان حذف کردیم و سپس ۲۵ درصد داده را به داده آزمون و ۷۵ درصد را به داده آموزش تخصیص داده ایم. سپس داده آموزش را به الگوریتم داده ایم تا درخت تصمیم موردنظر را بسازد. پس از آموزش داده آزمون را برای پیش بینی به درخت داده ایم و دقت ۵۷,۴۴ درصد را بدست آوردیم.

```
acc= 57.446808510638306%
```

آزمون با هرس درخت:

سپس درخت را هرس کردیم که بهترین نتیجه آن با عمق ۵ بود. به این صورت که شاخه های عمق ۵ به بعد درخت را حذف میکنیم و براساس بیشترین لیبل موجود در نود های عمق ۵ پیش بینی را انجام میدهیم.

پس از آن بر روی داده تست دقت ۶۱,۷ درصد را بدست آوردیم. مشخصا عملیات هرس درخت باعص تعمیم پذیری بیشتر درخت میشود زیرا از یادگیری بیش از حد جزئیات جلوگیری میکند و اجازه نمیدهد درخت به داده آموزش بایاس شود و در نتیجه بیش برازش رخ نمیدهد.

```
acc_prune= 61.702127659574465%
```

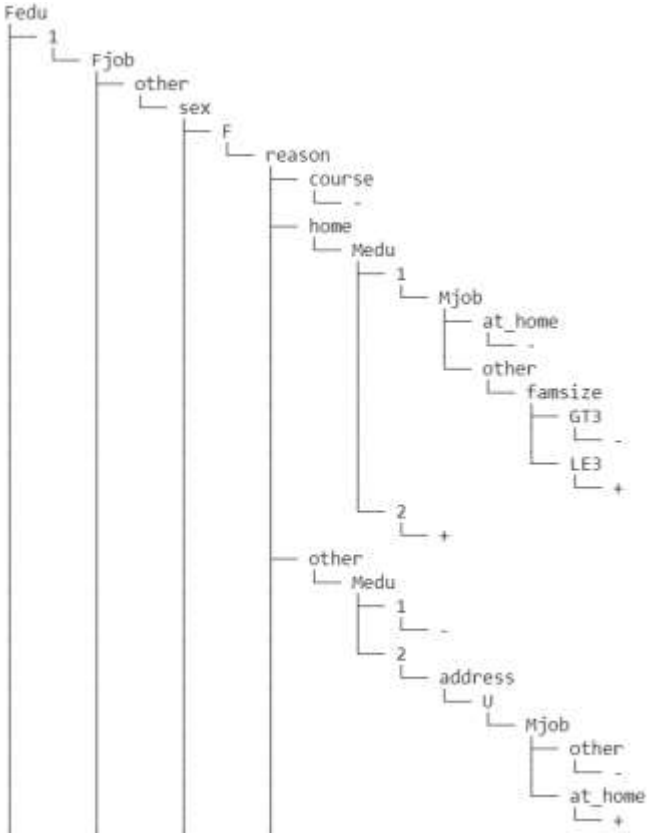
ارزیابی 5-fold cross validation :

پس از تقسیم کل دادگان به ۵ قسمت و تکرار آموزش روی ۴ قسمت و آزمون روی ۱ قسمت باقی مانده و میانگین گیری دقت هر تکرار، دقت گزارش شده ۵۳,۲۱ درصد است. که دلیلش میتواند این باشد که داده کمی داریم و زمانی که داده بزرگ و بزرگتر میشود تعمیم پذیری کاهش یافته و درنتیجه باید درخت روی دادگان بزرگتری آموزش میدید.

```
5-Fold cross validation acc = 53.214414414414414%
```

بخشی از درخت ساخته شده با استفاده از این الگوریتم و درخت هرس شده را در شکل های صفحه بعد مشاهده میکنید. برای نمایش درخت، از کتابخانه anytree در پایتون استفاده شده است.

بخشی از درخت اصلی :



بخشی از درخت هرس شده: