به نام خدا

محسن رحیمی

این پروژه در محیط google colab پیاده سازی شده است.
در ابتدا کلاس های Node و Tree تعریف شده اند. هر Node یک parent و یک value و تعدادی children دارد. هر درخت یک ریشه دارد و یک متود برای چاپ کردن درخت.
در ابتدا دیتای رستوران به صورت دستی وارد شده است. که هر کدام از مقادیر رشته ای به یک عدد تخصیص داده شده است.
سپس تابع prune تعریف شده که هر نود را اگر بیشتر از 80 درصد ورودی های آن یک مقدار خروجی داشتند یک برگ در نظر می گیرد.
در بلاک بعد توابع importance و importance_gini_index و calc_B را داریم که در کامنت های همان block مشخص شده است که چه کاری انجام می دهند.
تابع plurality_value مقدار اکثریت (0 یا 1) خروجی ها را مشخص می کند.
تابع decision_tree_learn پیاده سازی شبه کد داخل جزوه برای درخت تصمیم است که هم بر اساس gini index و هم بر اساس entropy کار می کند.
تابع predict یک درخت و یک ردیف از دیتاست تست می گیرد و در درخت آن را پیمایش و نتیجه را بر می گرداند.
تابع accuracy مقدار miss ها را به دست آورده و درصد دقت را بر می گرداند (در اینجا اگر دیتا از لحاظ نتیجه unbalance باشد، ممکن است این نوع accuracy معتبر نباشد و باید از روش های دیگر استفاده کرد. ولی در این دیتاست مقدار خروجی های راضی و ناراضی در حدود هم هستند و نگرانی از این بابت وجود ندارد.)
در قسمت بعد داده های رستوران به دو قسمت train و test تقسیم شده اند که نسبت آنها به هم 8 به 2 است. برای اینکه از داده های با نتیجه ۱ و ۰ به یک نسبت داده برای train و test انتخاب شود. جدا از هم به صورت رندم انتخاب می کنیم و سپس با هم ترکیب می کنیم.
در ادامه نتیجه را برای داده های رستوران به دست می آوریم که دقت ۷۵ درصد برای هر دو حالت gini-index و entropy می دهد.
در ادامه دستاست هواپیما را داریم.
ابتدا مقادیر پیوسته را گسسته کرده و مقادیر را به صورت عددی map می کنیم. برای گسسته سازی یک تابع spectrum_to_descrete_mapper داریم که با کمک آن یک بازه پیوسته را می توانیم به تعداد تکه های دلخواه تقسیم کنیم.
سپس مانند داده های رستوران، داده های هواپیما را به train و test تقسیم می کنیم.
و در نهایت برای entropy به دقت 82.75828882151966 درصد و برای gini-index به 81.40609210336 می رسیم.
در نهایت هم از توابع آماده کتابخانه scikitlearn استفاده شده تا دقت implementation با scikitlearn مقایسه شود که در این حالت 94.4516625763919 دقت خروجی است.
ایده برای بهتر کردن نتیجه:
۱- بهبود سیستم حرص کردن به صورتی که تعداد نمونه های موجود در node هم در نظر گرفته شود
۲- همه توابع را در قالب یک کلاس پیاده سازی کنیم که استفاده از آن راحت تر باشد.