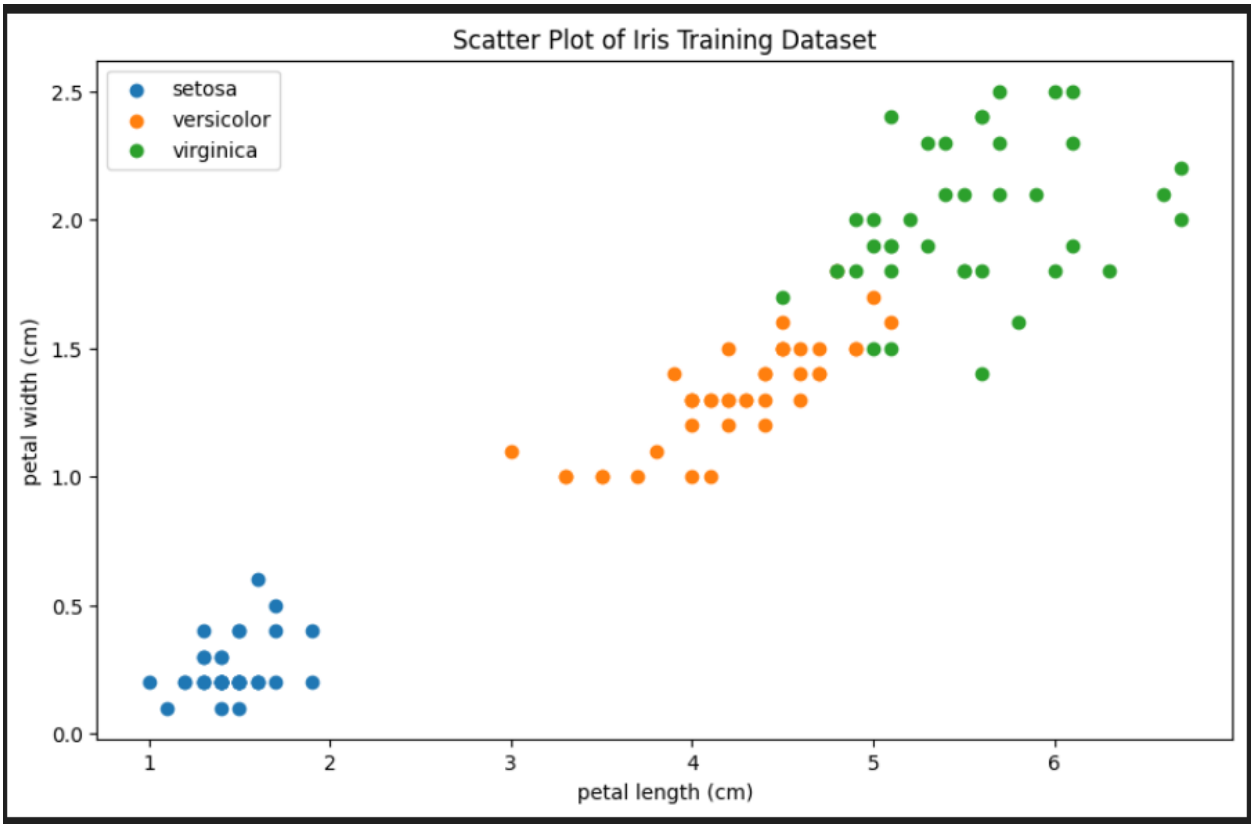
یادگیری ماشین

رویا شاهرودی 40240112093 تمرین سری 2-گزارش

الف) ابتدا کتابخانه های لازم جهت اضافه کردن داده ها، پردازش بر روی آنها و ساخت مدل های ماشین لرنینگ را اضافه میکنیم. numpy و pandas برای عملیات‌های عددی و پردازش داده‌ها، matplotlib برای رسم نمودار و sklearn مجموعه ای از توابع و کلاس های آماده را برای یادگیری ماشین فراهم میکند که در اینجا برای بارگذاری داده‌ها، تقسیم داده‌ها و پیاده‌سازی مدل ها از آنها استفاده میکنیم.

در بخش بعدی مجموعه داده iris را دانلود میکنیم و ویژگی ها و برچسب های داده ها را با x و y مشخص میکنیم. سپس داده‌ها به دو بخش train (80%) و test (20%) تقسیم می‌شوند.

ب) در این قسمت دو ویژگی 1 و 3 از مجموعه آموزشی انتخاب می‌شوند و داده‌ها به صورت نمودار scatter نمایش داده می‌شوند. هر کدام از گونه‌های مختلف گل‌ها با رنگ‌های متفاوت نشان داده شده‌اند تا بتوان تفاوت‌های موجود بین دسته‌ها را مشاهده کرد.



پ)

Criterion : ***{“gini”, “entropy”, “log_loss”}, default=”gini”***

معیاری برای سنجش کیفیت تقسیم‌بندی. مقدار 'gini' از Gini impurity استفاده میکند و 'entropy' و ’log_loss‘ از معیار اطلاعات Shannon Information Gain استفاده میکند.

Splitter : ***{“best”, “random”}, default=”best”***

استراتژی انتخاب تقسیم‌بندی در هر گره. مقدار 'best' بهترین تقسیم ممکن را انتخاب می‌کند و 'random' تقسیمات تصادفی را بررسی می‌کند.

max_depth : ***int, default=None***

حداکثر عمق درخت. اگر مقدار None باشد، گسترش گره‌ها تا زمانی که همه برگ‌ها خالص شوند یا تعداد نمونه‌ها به حداقل برسد، ادامه پیدا می‌کند.

min_samples_split : ***int or float, default=2***

حداقل تعداد نمونه‌هایی که برای تقسیم یک گره لازم است. مقدار پیش‌فرض 2 به این معناست که هر گره‌ای که حداقل 2 نمونه دارد می‌تواند تقسیم شود.

min_samples_leaf : ***int or float, default=1***

حداقل تعداد نمونه هایی که در هر برگ باید وجود داشته باشد. مقدار بیشتر باعث می‌شود که درخت تمایل کمتری به ایجاد برگ‌های کوچک داشته باشد و این می‌تواند باعث کاهش overfitting شود.

min_weight_fraction_leaf : ***float, default=0.0***

حداقل کسر وزنی از مجموع وزن ها که در هر برگ باید باشد. این پارامتر برای جلوگیری از ایجاد برگ‌های کوچک به کار می‌رود. در حالت صفر نمونه ها وزن برابر دارند.

max_features : *int, float or {“sqrt”, “log2”}, default=None*

تعداد ویژگی‌های مورد استفاده در جستجوی بهترین تقسیم. اگر None باشد، تمام ویژگی‌ها استفاده می‌شوند. این پارامتر می‌تواند به جلوگیری از overfitting کمک کند.

random_state : *int, RandomState instance or None, default=None*

عدد رندوم برای کنترل تکرارپذیری نتایج. اگر مقدار None باشد، هر بار اجرای مدل نتایج متفاوتی تولید می‌کند.

max_leaf_nodes : *int, default=None*

حداکثر تعداد برگ‌های درخت. این پارامتر می‌تواند به کاهش پیچیدگی مدل و جلوگیری از overfitting کمک کند.

min_impurity_decrease : *float, default=0.0*

حداقل کاهش ناخالصی که باید با تقسیم یک گره به دست آید تا تقسیم انجام شود. این پارامتر می‌تواند به کاهش overfitting کمک کند.

class_weight : *dict, list of dict or “balanced”, default=None*

وزن‌های مرتبط با کلاس‌ها. اگر None باشد، تمام کلاس‌ها وزن یکسان دارند. می‌توان از balanced استفاده کرد تا وزن‌ها به صورت خودکار بر اساس توزیع داده‌ها تعیین شوند.

ccp_alpha : *non-negative float, default=0.0*

پارامتر Cost-Complexity Pruning. مقدار بزرگتر باعث ایجاد درخت‌های کوچکتر میشود.

monotonic_cst : *array-like of int of shape (n_features), default=None*

این پارامتر برای تعیین محدودیت های یکنواختی در ویژگی ها استفاده میشود. برای تعیین رابطه یکنواخت میان ویژگی ها و هدف به کار می‌رود.

ت) برای انجام ادامه عملیات و محاسبه دقت در حالات مختلف یک تابع میسازیم که داده ها، لیبل آنها و طبقه بند مورد نظر را بعنوان ورودی میگیرد و با استفاده از آن طبقه بند برای داده های ورودی لیبل آنها را پیش بینی میکند و سپس با استفاده از تابع محاسبه دقت در کتابخانه sklearn دو لیبل پیش بینی شده توسط طبقه بند و لیبل اصلی داده را با هم مقایسه میکند و نتیجه را بعنوان خروجی تابع باز میگردانیم. با ساخت این تابع میتوانیم دقت را به راحتی برای هر نوع داده و طبقه بندی محاسبه کنیم.

سپس با استفاده از کتابخانه sklearn طبقه بندی درخت تصمیم را با معیار gini ایجاد میکنیم و با استفاده از داده های train آن را آموزش میدهیم.

با توجه به خواسته صورت سوال با استفاده از تابع دقت پیاده سازی شده یکبار دقت را برای داده های train محاسبه میکنیم و یکبار برای داده های test.

در ادامه درخت تصمیم را با استفاده از معیار entropy و همچنین معیار log_loss محاسبه میکنیم و مانند حالت قبل دقت این طبقه بند ها را بر روی مجموعه داده های آموزش و تست نیز محاسبه میکنیم.

ث) در این بخش با استفاده از کتابخانه sklearn طبقه بند random forest را با معیار gini فراخوانی کرده و با استفاده از داده های train آن را آموزش میدهیم و در آخر مانند بخش ها قبل دقت این طبقه بند را بر روی داده های آموزش و تست محاسبه میکنیم که در همه این حالات دقت محاسبه شده 100 درصد است.

```
train accuracy: 100.00%
test accuracy: 100.00%
```