۱)‌

در این پروژه ابتدا با کمک کتابخانه sklearn یک decision tree میسازیم که با ۸۰ درصد دیتا داده شده train شده است. این دیتا دارای ۳۰۳ sample و ۱۲ feature است. سپس accuracy این درخت تصمیم را با تست روی ۲۰ درصد باقی مانده دیتا به وسیله خود تابع accuracy\_score موجود در کتابخانه sklearn بدست می آوریم.

bootstrapping چیست و چه تاثیری بر روی واریانس و انحراف معیار استاندارد دارد؟

bootstrapping یک تست است که بر اساس نمونه برداری تصادفی با امکان تکرار انجام میشود.

/////////

Accuracy این درخت تصمیم با سه بار تست به صورت زیر است.

تست اول:‌ ۵۳ درصد

تست دوم: ۵۶ درصد

تست سوم: ۵۸ درصد

میانگین: ۵۵.۶ درصد

۲.۱ ـ ۲.۲)

Overfitting چیست و چرا درخت تصمیم به آن حساس است؟ bagging سعی دارد چه مساله ای را حل کند؟

یک ماشین ممکن است در بعضی موارد آنقدر train data را خوب یاد بگیرد که فقط بتواند همان دیتا را درست پاسخ دهد و اگر موردی خارج از train data را بخواهد پیش بینی کند نتواند با دقت خوبی پیش بینی کند. در یک درخت تصمیم overfitting زمانی اتفاق می افتد که درخت تصمیم به گونه ای طراحی شده است که به طور کامل بر همه ی سمپل های داده تست fit شود. مشکلی که درخت های تصمیم دارند این است که با توجه به train data ای که به آنها داده میشود خروجی آنها میتواند بسیار متفاوت باشد (به همان دلیل overfitting) به همین واریانس بسیار زیاد میشود. یکی از روش های از بین بردن این مشکل bagging است. در این روش به صورت تصادفی و با جایگزینی, از train data ای که داریم چند دسته سمپل N تایی (به گونه ای که معمولا N به اندازه ۶۰ درصد اندازه train data اصلی است.) از train data درست میکنیم (ایجاد bootstrapped data)‌. و سپس برای هر دسته از سمپل ها یک درخت تصمیم جداگانه train میکنیم. سپس برای داده های تست, تست ها را به همه ی این درخت های تصمیم میدهیم و سپس هر جوابی که درخت های تصمیم بیشتری به آن رسیده بودند میشود جواب پیش بینی کلی مساله. با این روش یک train data ای که داشتم را به چند train data تبدیل کردیم و هر کدام از آنها را جداگانه train کردیم که با این کار جواب کلی دیگر بر یک نمونه train data بیش از حد fit نیست و برای همین با روش bagging میتوان مشکل overfitting را حل کرد.

برای پیاده سازی bagging در پروژه دیتا داده شده را به ۵ دسته ۱۵۰تایی تقسیم می کنیم و سپس برای هر دسته یک درخت تصمیم را train می کنیم. سپس هر داده تست را به این ۵ درخت میدهیم تا در مورد آن تصمیم بگیرند و هر تصمیمی که بیشتر گرفته شود میشود تصمیم نهایی.

دقت مدل در سه بار تست به صورت زیر است.

تست اول: ۵۶ درصد

تست دوم: ۶۰ درصد

تست سوم: ۶۰ درصد

میانگین: ۵۸ درصد

۲.۳)

در این بخش با شروع از اولین feature هر بار یکی از ویژگی ها را حذف میکنیم و درخت تصمیم برای دیتا باقی مانده را میسازیم.

////////////////////////

۲.۴)

در این بخش پنج ویژگی را به طور تصادفی از ۱۲ ویژگی موجود انتخاب کرده و درخت تصمیم آن را میسازیم. Accuracy درخت ایجاد شده به صورت زیر است.

///////////////////////////

۲.۵)

برای ساخت random forest باید تعداد بیشتری از درخت هایی که در بخش ۲.۴ ایجاد کردیم, درست کنیم و سپس داده تست را به همه آنها بدهیم و هر نتیجه ای که بیشتر انتخاب شد میشود نتیجه نهایی. یعنی برای ساخت جنگل تصادفی باید تعداد زیادی درخت داشته باشیم که در هر کدام به صورت تصادفی بعضی ویژگی ها حذف شده است و با دیتا های باقی مانده آموزش دیده شده اند.

ارتباط random forest و bagging چیست؟ random forest سعی دارد چه مساله ای را حل کند؟

/////////////////////

با مقایسه دقت های ثبت شده در قسمت های ۱ و ۲.۲ و ۲.۵ و با توجه به سوالات بالا چه نتیجه ای میگیرید؟

///////////////////////