جنگل تصادفی:

Random Forest یک الگوریتم یادگیری گروهی است که چندین درخت تصمیم را برای پیش‌بینی ترکیب می‌کند. با ساخت مجموعه ای از درختان تصمیم و سپس میانگین گیری پیش بینی های انجام شده توسط هر درخت کار می کند. درختان با استفاده از زیرمجموعه ای از داده ها و زیر مجموعه ای از ویژگی ها ساخته می شوند که به کاهش بیش از حد برازش کمک می کند. Random Forest به ویژه در مدیریت مجموعه داده‌های با ابعاد بالا با ویژگی‌های بسیار مؤثر است و نشان داده شده است که در طیف گسترده‌ای از کاربردها به خوبی کار می‌کند.

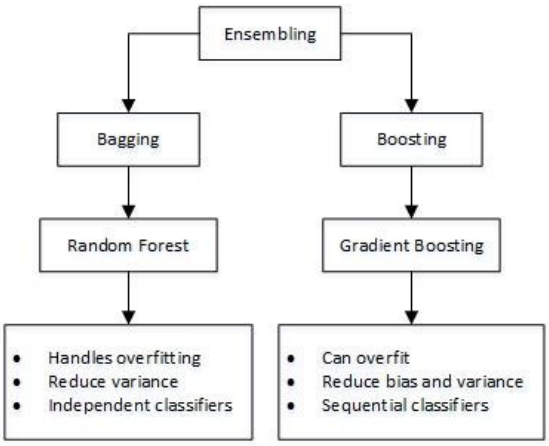
افزایش گرادیان:

Gradient Boosting یکی دیگر از الگوریتم های یادگیری گروهی است که چندین یادگیرنده ضعیف را برای پیش بینی ترکیب می کند. برخلاف Random Forest که چندین درخت را به‌طور مستقل می‌سازد، Gradient Boosting دنباله‌ای از درختان را می‌سازد که در آن هر درخت سعی می‌کند اشتباهات درخت قبلی را اصلاح کند. این کار با افزودن مکرر درختان جدید به مدل، با تمرکز هر درخت جدید بر روی نمونه هایی که توسط درختان قبلی ضعیف پیش بینی شده بودند، کار می کند. تقویت گرادیان به ویژه برای مشکلاتی که ویژگی‌های زیادی وجود دارد و روابط بین ویژگی‌ها پیچیده است، مؤثر است. Gradient boosting trees می توانند دقیق تر از Random Forest باشند. از آنجایی که ما آنها را آموزش می دهیم تا خطاهای یکدیگر را تصحیح کنند، آنها قادر به ثبت الگوهای پیچیده در داده ها هستند. با این حال، اگر داده ها نویزی باشند، درختان تقویت شده ممکن است بیش از حد بر روی هم قرار بگیرند و شروع به مدل سازی نویز کنند.

در نتیجه دو تفاوت اصلی بین درختان Gradient Boosting و Random Forest وجود دارد. اولی را به صورت متوالی، یک درخت در هر بار آموزش می دهیم تا اشتباهات قبلی را اصلاح کنیم. در مقابل، ما درختان را به طور مستقل در یک جنگل تصادفی می سازیم. به همین دلیل، ما می‌توانیم یک جنگل را به صورت موازی آموزش دهیم، اما gradient-boosting trees نه.

تفاوت اصلی دیگر در نحوه تصمیم گیری آنهاست. از آنجایی که درختان در یک جنگل تصادفی مستقل هستند، می توانند خروجی های خود را به هر ترتیبی تعیین کنند. سپس، پیش‌بینی‌های فردی را در یک پیش‌بینی جمعی جمع می‌کنیم: طبقه اکثریت در مسائل طبقه‌بندی یا مقدار متوسط در رگرسیون. از سوی دیگر، gradient-boosting trees به ترتیب ثابت اجرا می شوند و این توالی نمی تواند تغییر کند. به همین دلیل، آنها فقط ارزیابی متوالی را می پذیرند.

کدام یک معمولا نتایج بهتری دارد؟



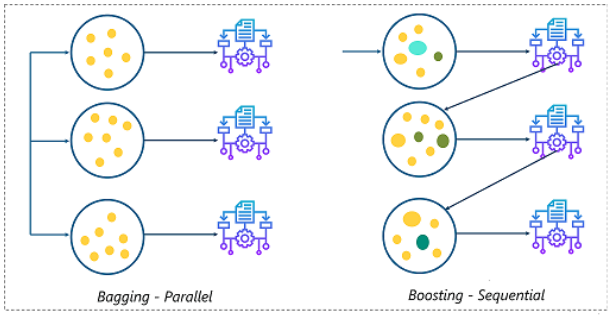
پاسخ به این سوال بستگی به مجموعه داده های خاص و مشکلی دارد که با آن برخورد می شود. به طور کلی، Gradient Boosting روی مجموعه داده های پیچیده تر با ویژگی های زیاد بهتر عمل می کند و در حالت کلی این الگوریتم را با نتیجه بهتر میشناسیم، در حالی که Random Forest هم می تواند روی مجموعه داده های ساده تر با ویژگی های کمتر خوب و حتی بهتر عمل کند. با این حال، همیشه توصیه می شود هر دو الگوریتم را امتحان کنید و عملکرد آنها را در یک مجموعه داده معین مقایسه کنید تا مشخص شود کدام یک برای آن مشکل خاص بهترین کار را دارد. همچنین شایان ذکر است که هر دو الگوریتم را می توان برای بهبود عملکرد آنها تنظیم و بهینه کرد و هر دو مستعد اورفیتینگ هستند.

۲- روش‌های Ensemble خانواده‌ای از تکنیک‌ها هستند که چندین مدل را برای بهبود عملکرد کلی پیش‌بینی یک سیستم یادگیری ماشین ترکیب می‌کنند.

Boosting:

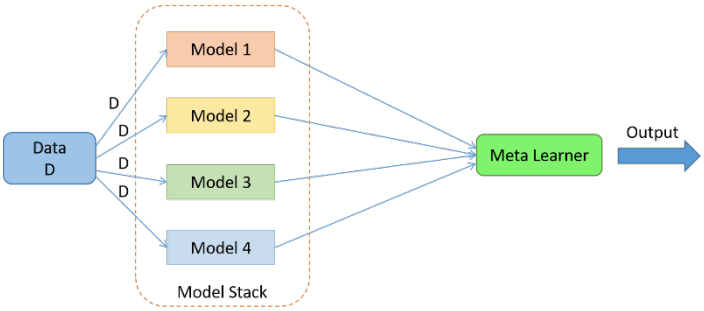
Boosting یک روش مجموعه ای است که چندین مدل ضعیف را برای ایجاد یک مدل قوی ترکیب می کند. این با آموزش متوالی مدل‌های ضعیف بر روی مجموعه داده‌های مشابه کار می‌کند و هر مدل سعی می‌کند خطاهای مدل قبلی را تصحیح کند. سپس پیش‌بینی نهایی با تجمیع پیش‌بینی‌های همه مدل‌های ضعیف انجام می‌شود. الگوریتم های تقویت کننده محبوب عبارتند از AdaBoost، Gradient Boosting و XGBoost.

Bagging:



Bagging یکی دیگر از روش‌های گروهی است که با ترکیب پیش‌بینی‌های چند مدل کار می‌کند. با این حال، بر خلاف Boosting، بسته بندی با آموزش هر مدل بر روی یک زیر مجموعه متفاوت از داده های آموزشی کار می کند. سپس پیش بینی نهایی با تجمیع پیش بینی های همه مدل ها انجام می شود. بسته بندی می تواند به کاهش بیش از حد اتصال و بهبود عملکرد مدل کمک کند. الگوریتم‌های bagging محبوب عبارتند از Random Forest و Bootstrap Aggregating.

Stacking:



Stacking یک روش گروهی پیشرفته‌تر است که پیش‌بینی‌های چند مدل را با استفاده از یک متا مدل ترکیب می‌کند. در Stacking ، چندین مدل پایه بر روی داده‌های آموزشی آموزش داده می‌شوند و پیش‌بینی‌های آن‌ها به‌عنوان ورودی برای یک متا مدل استفاده می‌شود، که یاد می‌گیرد پیش‌بینی‌های مدل‌های پایه را برای پیش‌بینی نهایی ترکیب کند. Stacking می تواند با بهره گیری از نقاط قوت مدل های مختلف به بهبود عملکرد مدل کمک کند.

به طور خلاصه، این سه روش مجموعه محبوبی هستند که برای بهبود عملکرد مدل‌های یادگیری ماشین استفاده می‌شوند. Boosting مدل‌های ضعیف را برای ایجاد یک مدل قوی ترکیب می‌کند، bagging چندین مدل را در زیر مجموعه‌های مختلف داده‌ها آموزش می‌دهد، و stacking پیش‌بینی‌های چند مدل را با استفاده از یک متا مدل ترکیب می‌کند.

	Bagging	Boosting	Stacking
Purpose	Reduce Variance	Reduce Bias	Improve Accuracy
Base Learner Types	Homogeneous	Homogeneous	Heterogeneous
Base Learner Training	Parallel	Sequential	Meta Model
Aggregation	Max Voting, Averaging	Weighted Averaging	Weighted Averaging

-۳

الگوریتم تقویت گرادیان به دلایل مختلفی از قدرت تعمیم بالایی برخوردار است:

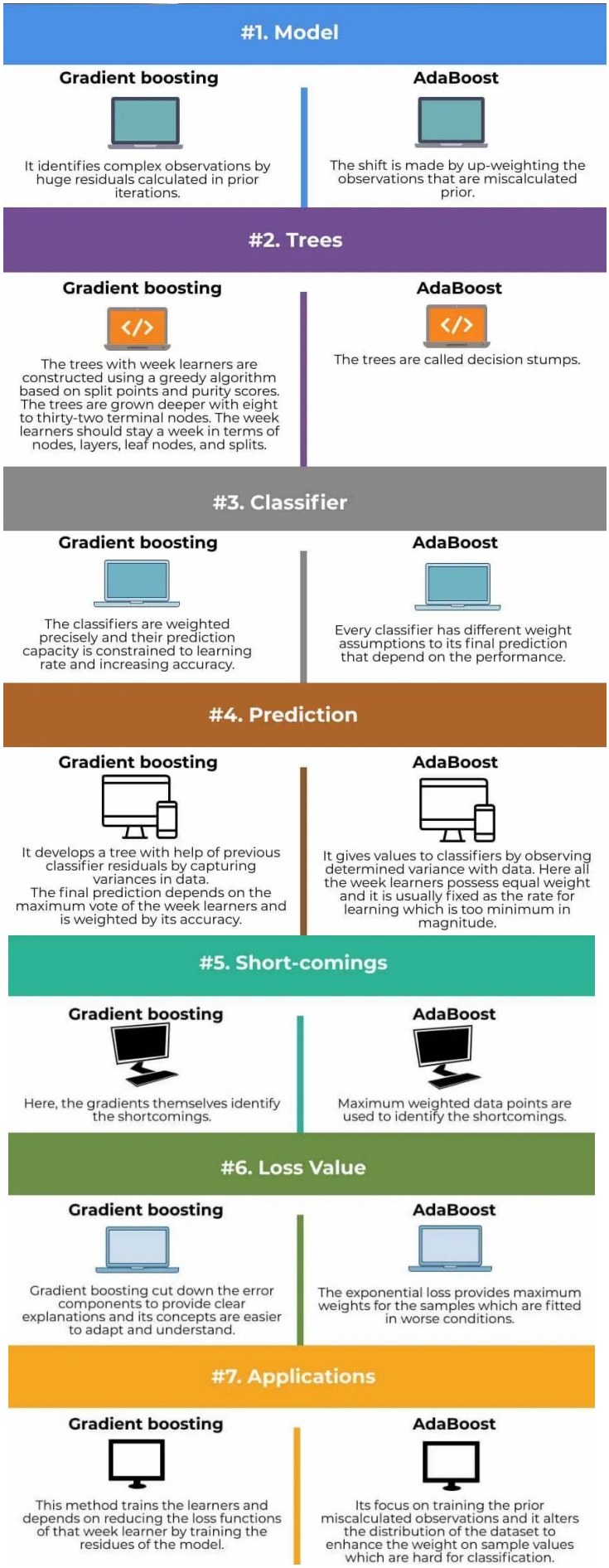
۱. Ensemble Learning: Gradient Boosting یک الگوریتم یادگیری گروهی است که چندین یادگیرنده ضعیف (درخت تصمیم) را برای تشکیل یک یادگیرنده قوی ترکیب می کند. این به الگوریتم اجازه می‌دهد تا الگوها و روابط پیچیده‌ای را در داده‌هایی که ممکن است توسط درخت‌های تصمیم فردی نادیده گرفته شوند، ثبت کند.
۲. Iterative Learning: الگوریتم به صورت تکراری یاد می‌گیرد و هر مدل بعدی بر روی خطاهای مدل قبلی تمرکز می‌کند. این به الگوریتم کمک می کند تا اشتباهات خود را اصلاح کند و پیش بینی های خود را با هر تکرار بهبود بخشد.
۳. Regularization: این الگوریتم تکنیک‌های منظم‌سازی مانند shrinkage و توقف زودهنگام را برای جلوگیری از overfitting ترکیب می‌کند. Shrinkage سهم هر درخت را در مدل نهایی کاهش می دهد، در حالی که توقف زودهنگام الگوریتم را زمانی که خطای اعتبارسنجی متوقف می شود، متوقف می کند.
۴. Feature Importance: الگوریتم می‌تواند مهم‌ترین ویژگی‌های مجموعه داده را شناسایی و اولویت‌بندی کند، و به آن اجازه می‌دهد بر مرتبط‌ترین اطلاعات تمرکز کند و نویز در داده‌ها را نادیده بگیرد.
۵. Robustness to Outliers: تقویت گرادیان نسبت به موارد پرت قوی است، زیرا تحت تأثیر مقادیر شدید در داده ها قرار نمی گیرد.
۶. Handling Missing Values: تقویت گرادیان می تواند مقادیر از دست رفته در داده ها را مدیریت کند، که یک مشکل رایج در مجموعه داده های دنیای واقعی است. می‌تواند مقادیر گمشده را در نظر بگیرد یا از آنها به عنوان یک دسته جداگانه در طول ساخت درخت استفاده کند.
۷. روابط غیر خطی: تقویت گرادیان می تواند روابط غیرخطی بین ویژگی ها و متغیر هدف را ثبت کند.
۸. Scalability: این الگوریتم مقیاس پذیر است و می تواند مجموعه داده های بزرگی را با تعداد زیادی ویژگی مدیریت کند که آن را برای برنامه های داده های بزرگ مناسب می کند.
۹. Versatility: تقویت گرادیان یک الگوریتم همه کاره است که می تواند برای کارهای طبقه بندی و رگرسیون استفاده شود. همچنین می‌توان از آن با loss functions های مختلف و معیارهای ارزیابی استفاده کرد که به آن اجازه می‌دهد برای انواع مختلف مشکلات تنظیم شود.
۱۰. Interpretability: در حالی که تفسیر مدل های مجموعه ممکن است دشوار باشد، الگوریتم تقویت گرادیان امتیازهای اهمیت ویژگی را ارائه می دهد که می تواند به شناسایی مهم ترین پیش بینی کننده ها در مدل کمک کند.

-۴

AdaBoost و Gradient Boosting هر دو روش های یادگیری گروهی هستند که در یادگیری نظارت شده استفاده می شوند. با این حال، آنها رویکردهای متفاوتی برای ایجاد یک مدل پیش بینی قوی از چندین یادگیرنده ضعیف دارند.

AdaBoost با آموزش مکرر یک سری از یادگیرندگان ضعیف (مثلاً درختان تصمیم) در زیر مجموعه‌های داده‌های آموزشی کار می‌کند. در هر تکرار، الگوریتم به نقاط داده‌ای که توسط یادگیرنده ضعیف قبلی به اشتباه طبقه‌بندی شده‌اند، وزن‌های بالاتر و به نقاطی که به درستی طبقه‌بندی شده‌اند، وزن‌های کمتری اختصاص می‌دهد. سپس یادگیرنده ضعیف بعدی بر روی این مجموعه داده وزنی آموزش داده می شود و این روند تا زمانی ادامه می یابد که تعداد از پیش تعیین شده ای از یادگیرندگان ضعیف آموزش داده شوند. سپس پیش‌بینی نهایی با ترکیب پیش‌بینی‌های همه زبان‌آموزان ضعیف با استفاده از اکثریت وزنی انجام می‌شود.

از سوی دیگر، تقویت گرادیان شامل آموزش مکرر یک سری از یادگیرندگان ضعیف نیز می شود، اما هر یادگیرنده ضعیف بعدی در مورد خطاهای باقی مانده یادگیرنده قبلی آموزش می بیند. به عبارت دیگر، الگوریتم به جای وزن دادن به داده های آموزشی، بر روی اشتباهات یادگیرنده قبلی تمرکز کرده و سعی در اصلاح آنها در تکرار بعدی دارد. سپس پیش‌بینی نهایی با ترکیب پیش‌بینی‌های همه زبان‌آموزان ضعیف با استفاده از یک میانگین ساده یا جمع وزنی انجام می‌شود.



بنابراین تفاوت اصلی بین AdaBoost و Gradient Boosting در نحوه تعیین وزن به داده ها در طول تمرین است. AdaBoost به نقاط داده تمرین وزن اختصاص می دهد، در حالی که Gradient Boosting بر روی خطاهای باقی مانده تمرکز می کند. علاوه بر این، اگر یدگیرنده های ضعیف بیش از حد پیچیده باشند، AdaBoost تمایل بیشتری به اضافه کردن دارد، در حالی که Gradient Boosting به دلیل تمرکز بر خطاهای باقی مانده، می تواند مدل های پیچیده تری را مدیریت کند.

تفاوت های دیگری هم دارند:

- **روش آموزش**: درAdaBoost ، هر مدل ضعیف به صورت مستقل از دیگر مدل‌ها آموزش داده می‌شود، در حالی که درGradient Boosting ، هر مدل ضعیف برای بهبود عملکرد مدل کلی، بر روی خطای باقی‌مانده از مدل‌های قبلی آموزش داده می‌شود.

- **نوع مدل‌های ضعیف**: درAdaBoost ، می‌توان از هر مدل ضعیفی استفاده کرد، مثلاً یک درخت تصمیم‌گیری یا یک مدل خطی؛ در حالی که درGradient Boosting ، بیشتر از مدل‌های درختی استفاده می‌شود.

- **سرعت آموزش**: به طور کلی، AdaBoostبه دلیل نیاز به چندین بار آموزش دادن به داده‌ها، از Gradient Boostingکندتر است.

- **مقاومت به داده‌های نویزی**: به طور کلی، Gradient Boosting مقاوم‌تر به داده‌های نویزی است، به دلیل اینکه مدل کلی بهبود عملکرد خود را برای تعدادی از مدل‌های ضعیف در نظر می‌گیرد و از اثر داده‌های نویزی کمتر تحت تاثیر قرار می‌گیرد.

۱- n_estimators : این هایپرپارامتر تعداد درختان را در مدل تقویت گرادیان تعیین می کند. افزایش تعداد تخمین‌گرها می‌تواند منجر به عملکرد بهتر شود، اما همچنین زمان آموزش را افزایش می‌دهد و به طور بالقوه می‌تواند منجر به اورفیتینگ شود. کاهش تعداد برآوردگرها می تواند منجر به عدم تناسب شود. در مثال داده شده، مقدار n_estimators روی ۵۰۰ تنظیم شده است که یک مقدار معقول است.

۲- Learning_rate: این فراپارامتر میزان مشارکت هر درخت در پیش‌بینی نهایی را تعیین می‌کند. نرخ یادگیری کمتر می تواند منجر به عملکرد بهتر شود و اورفیتینگ را کاهش دهد، اما زمان آموزش را نیز افزایش می دهد. همچنین نسبت عکس با n_estimators دارد. در مثال داده شده، نرخ یادگیری روی ۰.۰۵ تنظیم شده است که یک مقدار معقول است.

۳- max_features: این هایپرپارامتر حداکثر تعداد ویژگی هایی را که برای تقسیم در هر گره در نظر گرفته می شود، تعیین می کند. تنظیم این مقدار روی مقدار کمتر می‌تواند اورفیتینگ را کاهش دهد و تعمیم را بهبود بخشد، اما همچنین می‌تواند واضح بودن مدل را کاهش دهد. در مثال داده شده، max_features روی ۵ تنظیم شده است، به این معنی که تنها ۵ ویژگی برای تقسیم در هر گره در نظر گرفته شده است که نسبت به ۱۰۰ میتواند بیشتر هم بشود تا معیار ارزیابی افزایش باید.

۴- subsample: این فراپارامتر کسری از نمونه هایی را که برای آموزش هر درخت استفاده می شود را تعیین می کند. یک نمونه فرعی کوچکتر می تواند اورفیتینگ را کاهش دهد و تعمیم را بهبود بخشد، اما همچنین می تواند زمان آموزش را افزایش دهد و واضح بودن مدل را کاهش دهد. اما اگر این مقدار خیلی کم باشد و مدل نتواند با نمونه های کافی آموزش ببیند دقت و کارایی کاهش می یابد. در مثال داده شده، نمونه فرعی روی ۰.۳ تنظیم شده است، به این معنی که تنها ۳۰ درصد از نمونه ها برای آموزش هر درخت استفاده می شود میتوان مقدار آن را بیشتر کرد.

۵- max_depth: این هایپرپارامتر حداکثر عمق هر درخت را در مدل تقویت گرادیان تعیین می کند. یک max_depth کوچکتر می تواند اورفیتینگ را کاهش دهد و تعمیم را بهبود بخشد، اما همچنین می تواند بیانگر بودن مدل را کاهش دهد. در مثال داده شده، max_depth روی ۳ تنظیم شده است، به این معنی که هر درخت می تواند حداکثر عمق ۳ داشته باشد که نسبت به ۱۰۰ کم است و با کمی افزایش آن احتمالا کارایی بهبود یابد.

اگر مدل فاقد برازش باشد، افزایش پارامترهای فراپارامتر مانند n_estimators، max_depth و max_features می تواند به بهبود عملکرد کمک کند. از سوی دیگر، اگر مدل بیش از حد برازش داشته باشد، کاهش هایپرپارامترهایی مانند n_estimators، max_depth و max_features و افزایش هایپرپارامترهایی مانند learning_rate و subsample می تواند به بهبود عملکرد کمک کند.