به نام خدا

دانشگاه تهران

دانشکدگان علوم و فناوری های میان رشته ای

Machine Learning

تمرین دوم

نام و نام خانوادگی	محمدحسین مازندرانیان
شماره دانشجویی	۸۳۰۴۰۲۰۶۶
تاریخ ارسال گزارش	18/08/1403

سوال ۱) پیاده سازی Random Forest

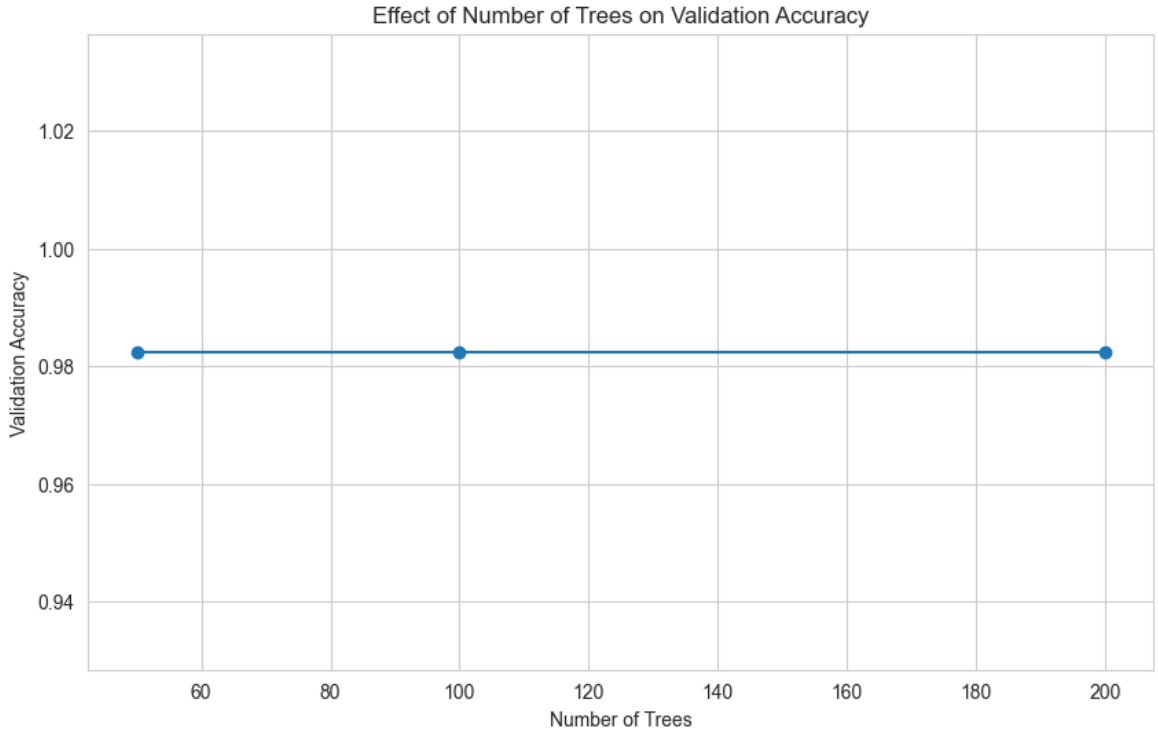
ابتدا داده‌های موجود به سه مجموعه تقسیم می‌شوند: ٪۶۰ برای آموزش، ٪۲۰ برای اعتبارسنجی، و ٪۲۰ برای آزمون. این تقسیم‌بندی به ارزیابی دقیق‌تر عملکرد مدل روی داده‌هایی که در مراحل مختلف دیده نشده‌اند، کمک می‌کند.

در مرحله بعد، تابعی به نام calculate_accuracy تعریف می‌شود که به‌طور خاص دقت مدل را برای مقادیر مختلف یک هایپرپارامتر محاسبه می‌کند. این تابع به ما اجازه می‌دهد تا در هر مرحله، تنها یک پارامتر را تغییر دهیم و اثرات آن را بر دقت مدل ارزیابی کنیم. این تابع به‌ویژه در تحلیل اثرات مقادیر مختلف برای پارامترهای max_depth و min_samples_splitاستفاده می‌شود.

برای یافتن بهترین ترکیب از پارامترها، یک جستجوی شبکه‌ای (Grid Search) انجام می‌شود که در آن سه هایپرپارامتر کلیدی (تعداد درخت‌ها n_estimators، عمق حداکثری درخت‌ها max_depth، و حداقل تعداد نمونه‌ها برای انشعاب (min_samples_split)با استفاده از اعتبارسنجی پنج‌گانه ارزیابی می‌شوند GridSearchCV .بهترین ترکیب پارامترها را از میان گزینه‌های تعیین‌شده انتخاب کرده و به این ترتیب، مدل بهینه برای مراحل بعدی آماده می‌شود.

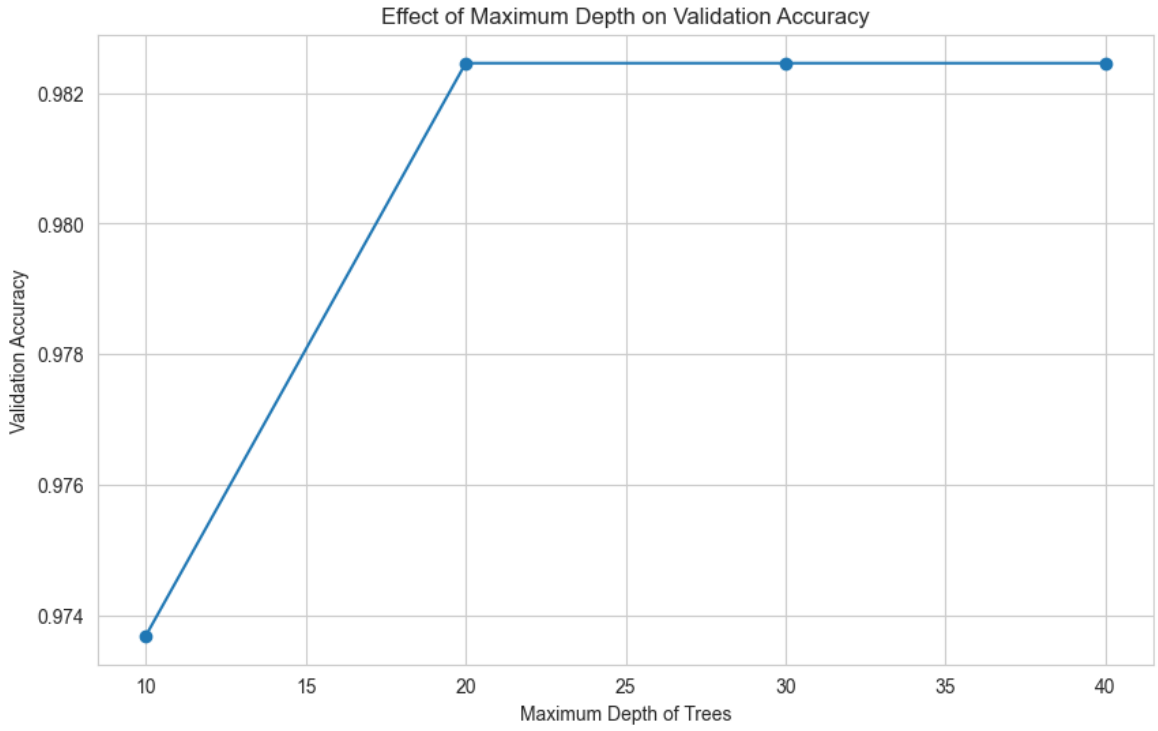
پس از یافتن بهترین مدل، این مدل روی داده‌های آموزش دوباره آموزش داده شده و سپس روی داده‌های اعتبارسنجی و آزمون ارزیابی می‌شود. در این ارزیابی، معیار دقت برای هر دو مجموعه اعتبارسنجی و آزمون محاسبه شده و نتایج گزارش می‌شود. به‌علاوه، ماتریس درهم‌ریختگی و گزارش کامل طبقه‌بندی که شامل معیارهای دقت (precision) ، یادآوری (recall)، و امتیاز F1 برای هر کلاس است، نیز ارائه می‌شود تا دید کاملی از عملکرد مدل به دست آید.

در نهایت، اثر هر یک از هایپرپارامترهای اصلی بر دقت اعتبارسنجی به صورت جداگانه بررسی می‌شود. نمودارهای مربوط به هر پارامتر رسم شده‌اند تا روند تغییرات دقت با افزایش تعداد درخت‌ها، عمق درخت‌ها، و مقدار min_samples_split مشخص شود.



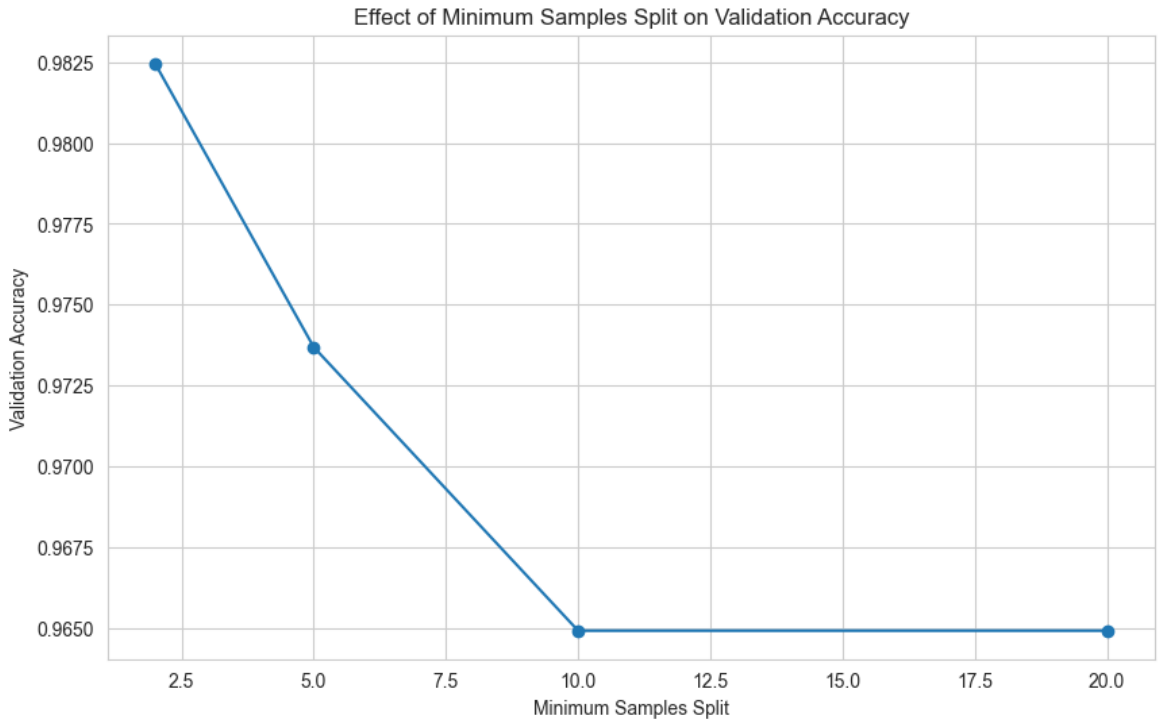
شکل ۱

در شکل شماره ۱ تاثیر دقت تعداد درخت های مدل بر دقت را مشاهده می کنیم. با توجه به این نمودار تعداد درخت های مختلفی که به random forest دادیم تاثیری در دقت مدل نداشته است.



شکل ۲

در شکل شماره ۲ تاثیر تعداد ماکسیمم درخت ها بر دقت ولیدیشن را مشاهده میکنیم که از یک عددی به بعد دقت مدل ما زیاد نمی شود و بهترین مقدار همان ۲۰ می باشد.



شکل ۳

در شکل ۳ تاثیر مقادیر minimum samples split را بر روی مقدار دقت رو داده های ولیدیشن رو مشاهده میکنیم که هر چه این عدد بالاتر می رود دقت مدل ما کمتر میشود و بهترین مقدار همان ۲ می باشد.

سوال ۲) پیاده سازی Adaboost روی درخت تصمیم

Adaboost Python

```
class CustomAdaBoost:
    def __init__(self, n_estimators=50, max_depth=1):
        self.n_estimators = n_estimators
        self.max_depth = max_depth
        self.alphas = []
        self.models = []

    def fit(self, X, y):
        N, _ = X.shape
        weights = np.ones(N) / N

        for _ in range(self.n_estimators):
            stump = DecisionTreeClassifier(max_depth=self.max_depth)
            stump.fit(X, y, sample_weight=weights)
            y_pred = stump.predict(X)

            misclassified = (y_pred != y)
            error = np.sum(weights * misclassified) / np.sum(weights)

            alpha = 0.5 * np.log((1 - error) / (error + 1e-10))
            self.alphas.append(alpha)
            self.models.append(stump)

            weights *= np.exp(alpha * misclassified * 2)
            weights /= np.sum(weights)

    def predict(self, X):
        final_prediction = sum(alpha * model.predict(X) for alpha, model in zip(self.alphas,
self.models))
        return np.sign(final_prediction)
```

کد CustomAdaBoost یک پیاده‌سازی دستی از الگوریتم Adaboost است که برای ترکیب چندین مدل ساده یا ضعیف (در اینجا درخت تصمیم) و بهبود دقت مدل نهایی طراحی شده است. هدف اصلی این کد افزایش دقت طبقه‌بندی با تقویت مدل‌های ضعیف‌تر است که به طور مستقل عملکرد خوبی ندارند اما با ترکیب مناسب، یک مدل قوی تشکیل می‌دهند.

توضیح اجزای اصلی CustomAdaBoost

۱. **تعریف کلاس CustomAdaBoost:** کلاس CustomAdaBoostاز ابتدا به منظور پیاده‌سازی دستی Adaboostایجاد شده است. این کلاس دو پارامتر کلیدی دارد:

- n_estimators: تعداد مدل‌های ضعیف (درخت‌های تصمیم) که در ترکیب Adaboost استفاده می‌شوند.
- max_depth: عمق حداکثری هر درخت تصمیم که برای کنترل پیچیدگی مدل‌های ضعیف به کار می‌رود.

۲. **تابع __init__:** در این قسمت، مقداردهی اولیه برای تعداد مدل‌ها و عمق هر مدل انجام می‌شود. سپس، لیست‌هایی برای نگه‌داشتن مدل‌ها (self.models) و وزن‌های هر مدل (self.model_weights) ایجاد می‌شود تا هر مدل و وزن اختصاصی آن برای محاسبه نهایی آماده شود.

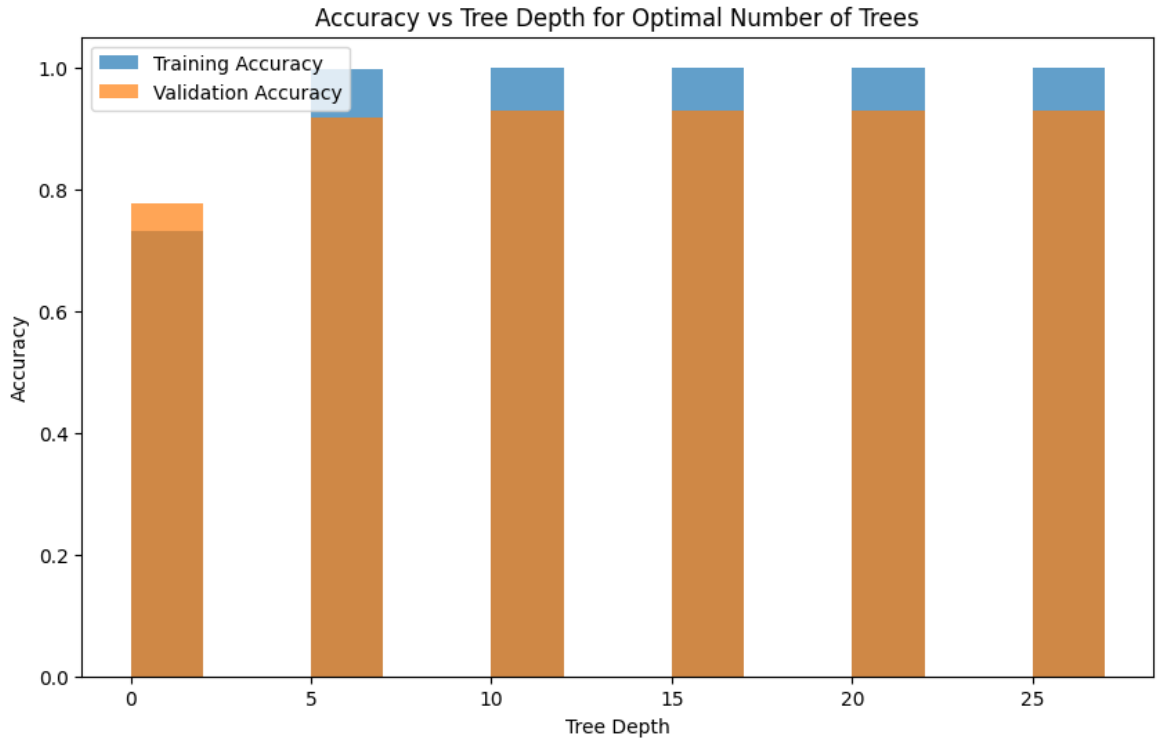
۳. **تابع fit:** این تابع مدل Adaboost را با استفاده از داده‌های آموزشی آموزش می‌دهد.

- در ابتدا وزن‌های نمونه‌ها برای داده‌های آموزشی تنظیم می‌شوند. این وزن‌ها در هر تکرار تغییر می‌کنند تا تمرکز بیشتری روی نمونه‌های دشوارتر ایجاد شود.
- در هر تکرار، یک مدل ضعیف با وزن‌دهی به داده‌ها آموزش داده می‌شود. سپس خطای مدل برای داده‌های وزنی محاسبه می‌شود.
- وزنی برای مدل در نظر گرفته می‌شود که نشان‌دهنده میزان اهمیت آن مدل در پیش‌بینی نهایی است؛ مدل‌های با دقت بالاتر وزن بیشتری دریافت می‌کنند.
- سپس، وزن‌های نمونه‌ها به‌روزرسانی می‌شوند. نمونه‌هایی که به درستی طبقه‌بندی نشده‌اند، وزن بیشتری دریافت می‌کنند تا مدل‌های بعدی به آن‌ها توجه بیشتری داشته باشند.
- مدل ضعیف و وزن آن به لیست مدل‌ها و وزن‌ها اضافه می‌شوند.

۴. **تابع predict:** این تابع برای پیش‌بینی داده‌های جدید با استفاده از مدل‌های تقویتی استفاده می‌شود.

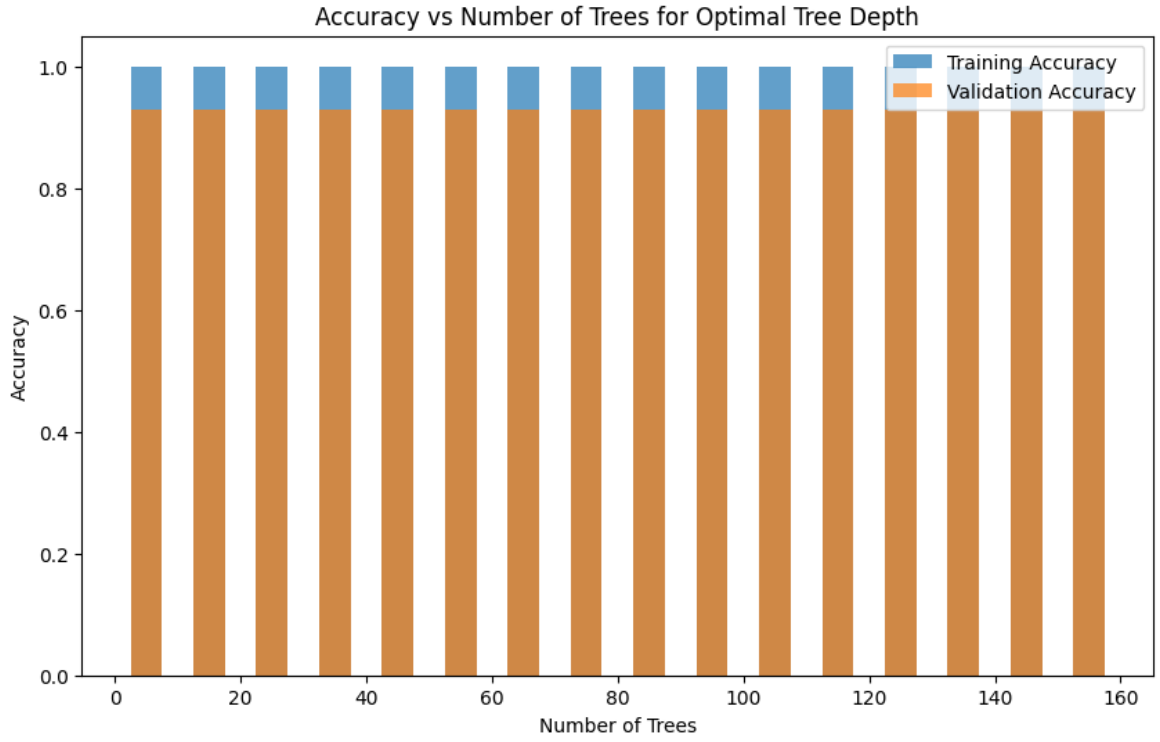
- در این تابع، هر مدل ضعیف پیش‌بینی خود را روی داده‌ها انجام می‌دهد و این پیش‌بینی‌ها بر اساس وزن مدل ترکیب می‌شوند.
- در نهایت، رای‌گیری وزنی انجام می‌شود تا پیش‌بینی نهایی Adaboost تعیین گردد.

۵. **عملکرد کلی** Adaboost : با این مکانیزم، از چندین مدل ضعیف با دقت پایین استفاده می‌کند و با افزایش وزن نمونه‌های سخت‌تر، مدل‌ها را وادار می‌کند که روی این نمونه‌ها تمرکز بیشتری کنند. نتیجه این فرآیند، یک مدل نهایی است که دقت بالاتری روی داده‌ها دارد و بهتر می‌تواند داده‌های جدید را دسته‌بندی کند.



شکل ۴

شکل شماره ی ۴ تاثیر تعداد عمق درخت روی دقت مدل را نشان می دهد. همانطور که مشخص است از عدد ۵ بیشتر روی دقت مدل تاثیری ندارد.



شکل ۵

شکل شماره ی ۵ تاثیر تعداد درختان روی دقت مدل را نشان می دهد که نتیجه میگیریم تعداد درخت تاثیر آنچنانی روی این مدل ندارد.