به نام خدا

گزارش پروژه اختیاری درس هوش مصنوعی

دانشجو :

محمد مهدی شرف بیانی - 401521372

توضیح کد های نوشته شده :

کلاس KNN_Classifier :

کلاس KNNClassifierپیاده‌سازی سفارشی از الگوریتم **kنزدیک‌ترین همسایه (K-Nearest Neighbors) یا KNN)**است. این الگوریتم از روش یادگیری مبتنی بر نمونه (Instance-Based Learning) استفاده می‌کند، به این معنی که در مرحله آموزش، فقط داده‌های ورودی ذخیره می‌شوند و در هنگام پیش‌بینی، شباهت نمونه جدید با داده‌های آموزش‌یافته سنجیده می‌شود.

جزئیات عملکرد کلاس:

1. **سازنده کلاس(__init__)**
   - مقدار kکه تعداد همسایه‌های مورد بررسی است را ذخیره می‌کند.
2. **متد fit(X, y)**
   - داده‌های ورودی (X) و برچسب‌های مربوط به آن (y) را به آرایه‌های numpyتبدیل و ذخیره می‌کند.
3. **متد predict(X)**
   - برای هر نمونه در مجموعه داده‌ی تست، فاصله آن با تمام نمونه‌های آموزش محاسبه می‌شود.
   - فاصله اقلیدسی (Euclidean Distance) برای اندازه‌گیری شباهت استفاده شده است.
   - kنمونه‌ای که کمترین فاصله را دارند، انتخاب شده و برچسب‌های آن‌ها جمع‌آوری می‌شود.
   - برچسب غالب (بیشترین تعداد وقوع) از بین این kنمونه، به عنوان پیش‌بینی نهایی انتخاب می‌شود.

متد Preproccess Data :

متد load_and_preprocess_dataوظیفه بارگذاری و پیش‌پردازش داده‌های مربوط به وام‌ها را بر عهده دارد. این متد شامل چندین مرحله کلیدی است که در ادامه توضیح داده شده‌اند:

۱ .بارگذاری داده‌ها

- داده‌ها از فایل CSV خوانده می‌شوند و شکل اولیه دیتافریم چاپ می‌شود تا بررسی شود که داده‌ها به درستی بارگذاری شده‌اند.

۲ .بررسی و انتخاب ستون‌های مورد نیاز

- تنها ستون‌های ضروری شامل 'grade'، 'term'، 'home_ownership'، 'emp_length'و 'bad_loans'حفظ می‌شوند.
- در صورت عدم وجود هر یک از این ستون‌ها، یک خطای ValueErrorایجاد می‌شود.

۳ .مدیریت مقادیر از دست رفته

- برای ویژگی 'emp_length'مقدار NaNبا مقدار 'missing'جایگزین می‌شود.
- سایر ویژگی‌های دسته‌ای با مقدار 'unknown'جایگزین می‌شوند.

۴ .متعادل‌سازی مجموعه داده

- توزیع کلاس 'bad_loans'بررسی می‌شود.

- برای جلوگیری از عدم توازن در کلاس‌ها، از هر کلاس تعداد برابر نمونه انتخاب می‌شود.

۵ .تبدیل ویژگی‌های دسته‌ای به مقادیر عددی

- ستون‌های دسته‌ای با استفاده از LabelEncoderرمزگذاری می‌شوند.

۶ .تقسیم داده‌ها به مجموعه‌های آموزشی، اعتبارسنجی و تست

- داده‌ها به سه بخش تقسیم می‌شوند :
  - **٪۷۰ آموزش(train)**
  - **٪۱۵ اعتبارسنجی(validation)**
  - **٪۱۵ تست(test)**
- تقسیم داده‌ها به صورت **Stratified**انجام می‌شود تا توزیع کلاس‌ها حفظ شود.

۷ .استانداردسازی ویژگی‌های عددی

- داده‌های ورودی با استفاده از StandardScalerنرمال‌سازی می‌شوند تا مدل‌ها بهتر عمل کنند.

۸ .خروجی متد

متد در نهایت موارد زیر را بازمی‌گرداند:

متد در نهایت موارد زیر را بازمی‌گرداند:

- **X_train_scaled، X_val_scaled،X_test_scaled**← داده‌های ویژگی‌های استانداردشده
- **y_train، y_val،y_test**← برچسب‌های دسته‌بندی‌شده
- **نام ستون‌ها، دیکشنری رمزگذارهای ویژگی‌های دسته‌ای، و استانداردساز(Scaler)**

این متد **فرآیند کاملی از پردازش داده‌ها را شامل می‌شود** و تضمین می‌کند که داده‌ها **پاک‌سازی، متعادل‌سازی و آماده استفاده در مدل‌های یادگیری ماشین** هستند.

متد های یادگیری :

متد یادگیری درخت تصمیم :

- این تابع یک مدل **درخت تصمیم** (Decision Tree)را با مقدار **حداکثر عمق d**آموزش می‌دهد.
- ابتدا یک نمونه از DecisionTreeClassifierبا مقدار max_depth=dایجاد می‌شود.
- سپس مدل با استفاده از مجموعه آموزشی X_train) و y_train)آموزش داده می‌شود.
- در نهایت، مدل آموزش‌دیده‌شده بازگردانده می‌شود.

ویژگی‌ها:

- **کنترل پیچیدگی مدل :**مقدار max_depthتعیین می‌کند که درخت چقدر عمیق باشد. عمق بیش از حد ممکن است به **overfitting**منجر شود.
- **بازتولیدپذیری :**مقدار random_state=42ثابت نگه داشته شده تا نتایج یکسانی در اجرای مجدد داشته باشیم.

متد یادگیری KNN :

- این تابع یک مدل **-Kنزدیک‌ترین همسایه** (KNN)را با مقدار kمشخص آموزش می‌دهد.
- ابتدا یک نمونه از کلاس KNNClassifierبا مقدار kایجاد می‌شود.
- سپس مدل بر روی داده‌های آموزشی X_train) و y_train)آموزش داده می‌شود.
- در نهایت، مدل آموزش‌یافته بازگردانده می‌شود.

ویژگی‌ها:

- **کنترل دقت و تعمیم مدل :**مقدار kتعیین می‌کند که تصمیم‌گیری بر اساس چند همسایه نزدیک صورت گیرد. مقادیر کوچک ممکن است به **overfitting**منجر شوند، در حالی که مقادیر بسیار بزرگ می‌توانند باعث **underfitting**شوند.

متد یادگیری AdaBoost :

عملکرد:

- این تابع یک مدل **AdaBoost**را با n_estimatorsمشخص آموزش می‌دهد.
- ابتدا یک نمونه از AdaBoostClassifierبا تعداد nضعیف‌آموز (weak learners) ایجاد می‌شود.
- سپس مدل روی مجموعه آموزشی X_train) و y_train) آموزش داده می‌شود.
- در نهایت، مدل آموزش‌دیده‌شده بازگردانده می‌شود.

ویژگی‌ها:

- **افزایش دقت مدل :**مقدار n_estimatorsتعیین می‌کند که چندین مدل ضعیف در کنار هم استفاده شوند تا قدرت کلی افزایش یابد.
- **مقاومت در برابر :overfitting** آدا بوست به تدریج روی نمونه‌های دشوار تمرکز می‌کند، که ممکن است باعث کاهش دقت در داده‌های نویزی شود.

متد یادگیری Random Forest :

عملکرد:

- این تابع یک مدل **جنگل تصادفی** (Random Forest)را با استفاده از **جستجوی شبکه‌ای (GridSearchCV)**آموزش می‌دهد.
- یک **شبکه جستجو** (GridSearchCV)برای پیدا کردن بهترین مقادیر n_estimators(تعداد درخت‌ها) و max_depth(حداکثر عمق درخت‌ها) اجرا می‌شود.
- مدل نهایی با بهترین **هایپرپارامترها** آموزش داده شده و همراه با بهترین مقادیر هایپرپارامتر بازگردانده می‌شود.

ویژگی‌ها:

- **تنظیم خودکار هایپرپارامترها :**به جای تعیین دستی مقدار n_estimatorsو max_depth، بهترین مقادیر آن‌ها از طریق جستجوی شبکه‌ای انتخاب می‌شود.
- **افزایش دقت و تعمیم مدل :**جنگل تصادفی با استفاده از **نمونه‌گیری تصادفی و رای‌گیری میان چندین درخت تصمیم**، دقت و پایداری را افزایش می‌دهد.

تابع Main :

در تابع main به ازای هر کدام از 3 مدل DT,KNN,AdaBoost برای tune کردن هایپرپارامتر هایشان در قالب یک حلقه هربار یک مدل را با train data اموزش دادیم و سپس دقت آن را با validation data اندازه گرفتیم و هایپر پارامتری که بیشترین دقت را دارد انتخاب کرده ایم و سپس best_model را با آن ساختیم

برای مدل RF نیز که برای tune کردن هایپرپارامتر ها از gridsearch با cv=5 استفاده کردیم پس دیگر نیازی به جدا کردن دیتای آموزش و ولیدیشن نیست و آن ها را ادغام میکنیم و به تابع train_rf می دهیم و در آنجا با gridsearch بهترین مدل را می سازد و خروجی می دهد.

نحوه کارکرد گرید سرچ :

Cv=5 یعنی دیتای آموزش را به 5 قسمت تقسیم میکند(برای همین نیاز نیست که دیتای ترین را با ولیدیشن جدا کنیم) و هربار 4 قسمت را برای آموزش و یک قسمت را برای ولیدیشن انتخاب می کند .

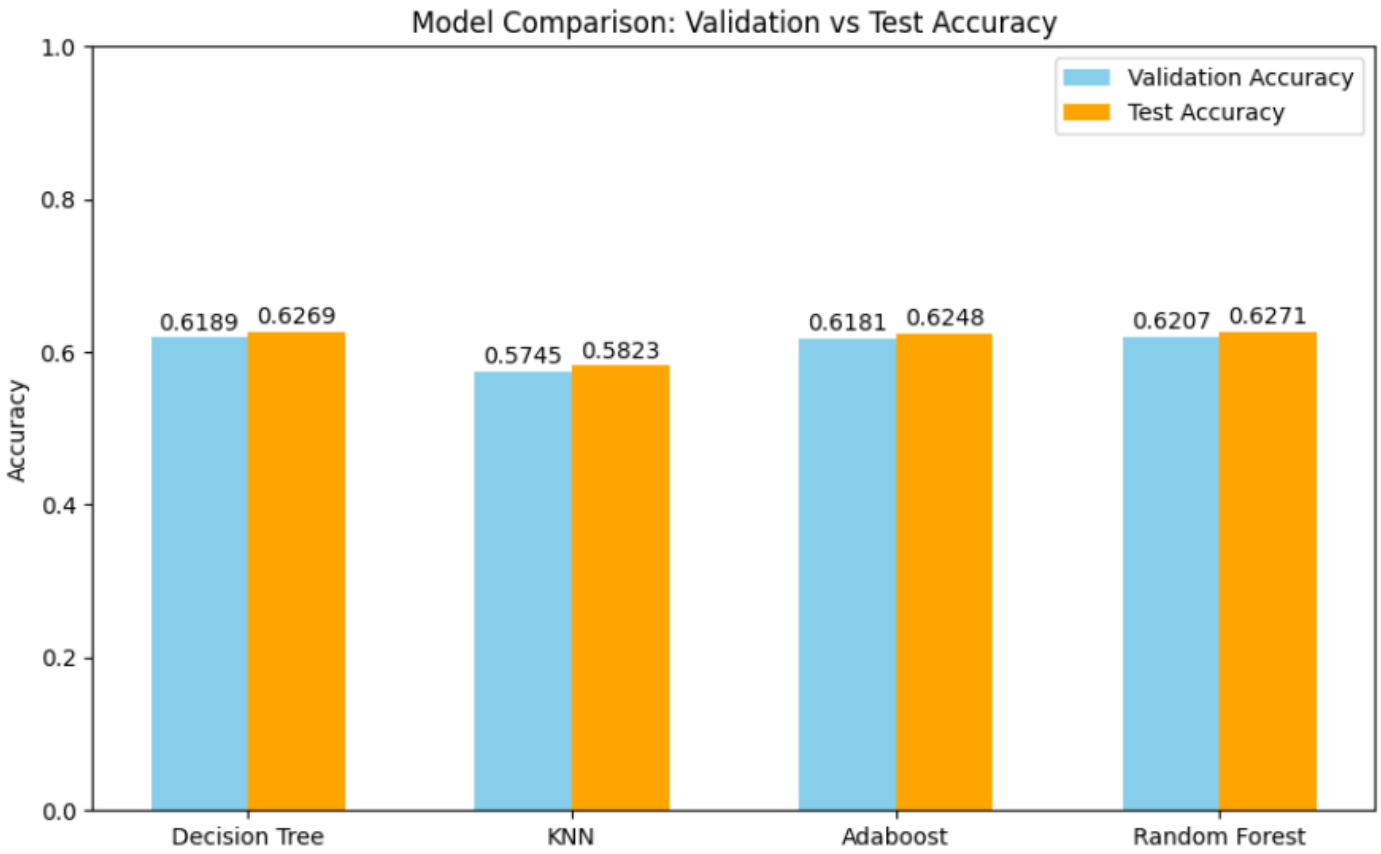
```
Data: [Fold 1] [Fold 2] [Fold 3] [Fold 4] [Fold 5]

Round 1: [Train] [Train] [Train] [Train] [Valid]
Round 2: [Train] [Train] [Train] [Valid] [Train]
Round 3: [Train] [Train] [Valid] [Train] [Train]
Round 4: [Train] [Valid] [Train] [Train] [Train]
Round 5: [Valid] [Train] [Train] [Train] [Train]
```

برای هر ترکیب از هایپرپارامتر ها این 5 پیمایش را انجام می دهد و در هر قسمت دقت را حساب کرده و در نهایت برای آن با یافتن میانگین این دقت ها دقت کلی برای ان ترکیب از هایپرپارامتر هارا میابد و در ترکیبی را انتخاب میکند که بیشترین میانگین دقت را داشته باشد و مدل بهینه را با همین ترکیب می سازد.

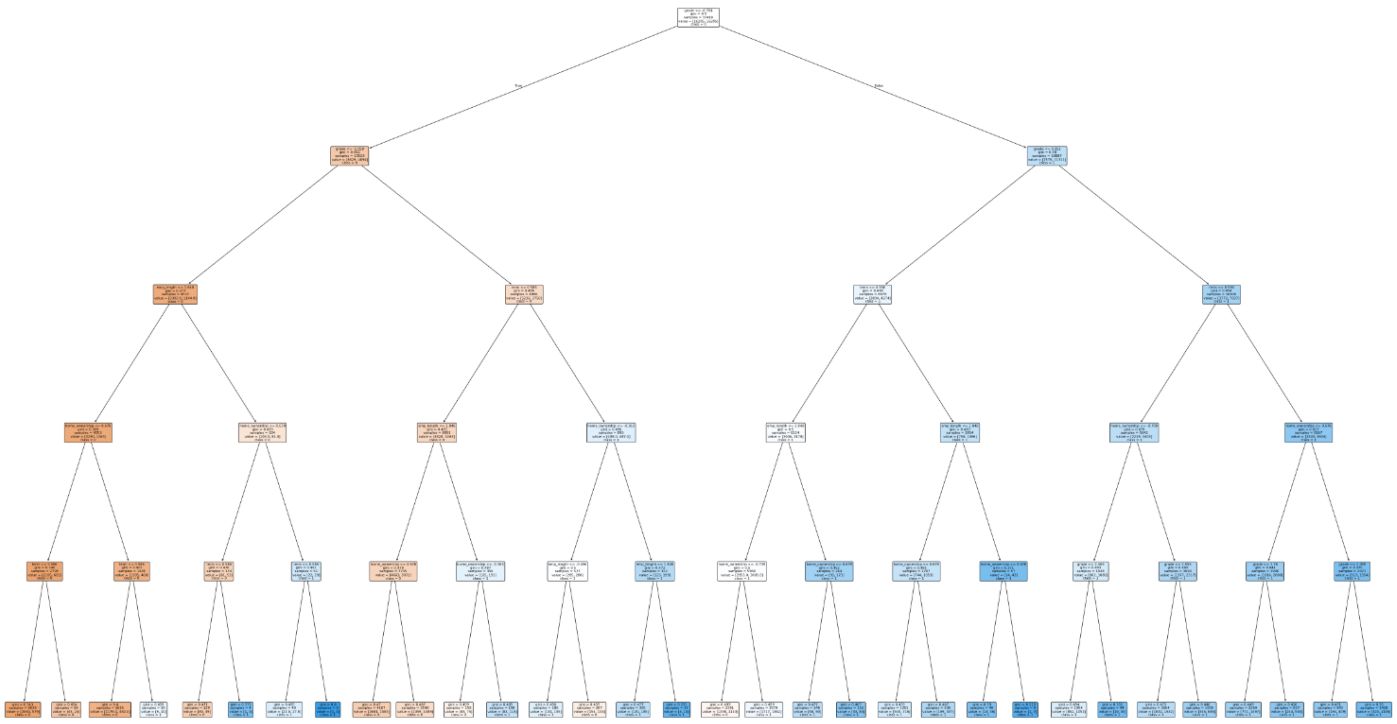
نتایج :

همانطور که مشاهده میشود هیچکدام از مدل ها overfit نشده اند:

دقت روی داده تست : dt>rf>adaboost>knn

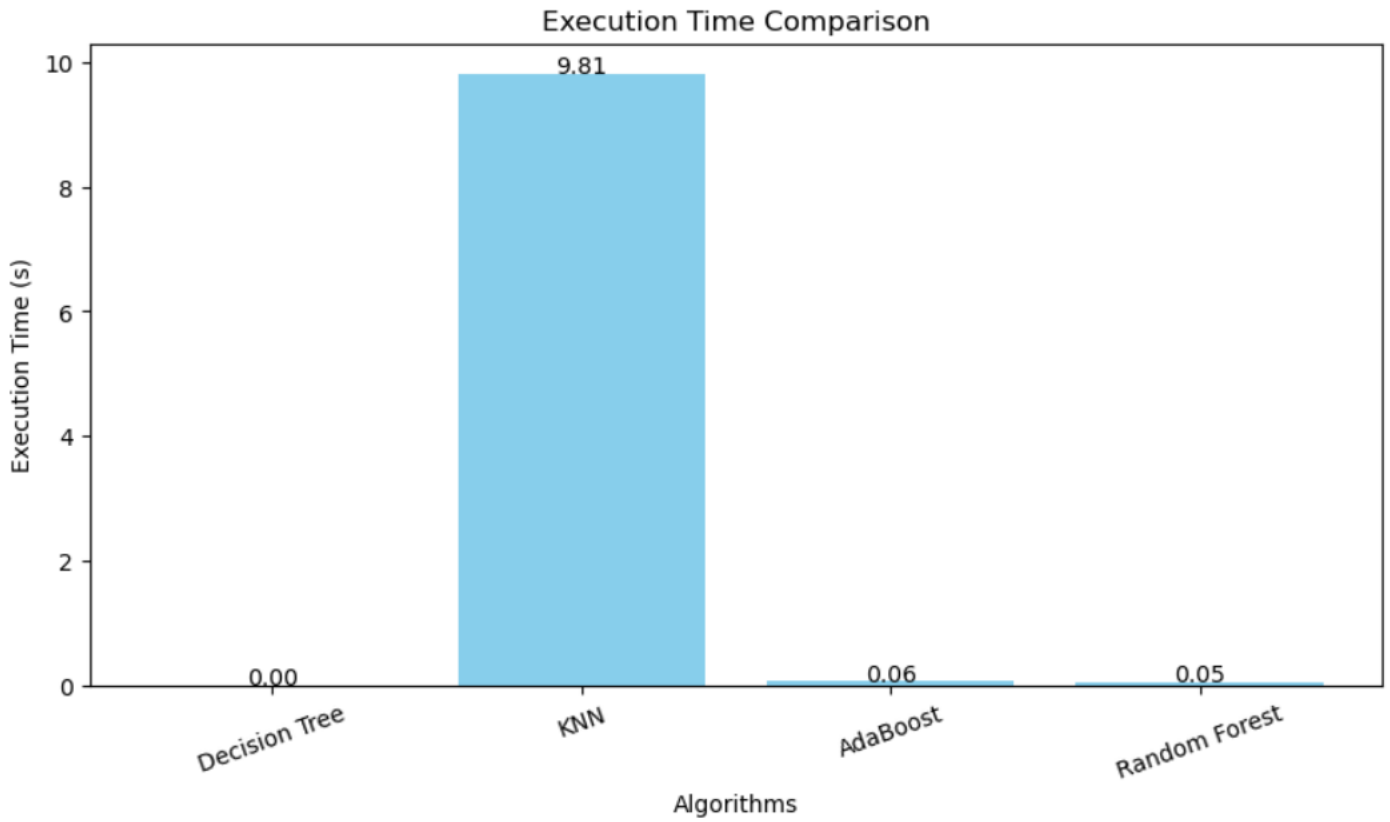


نمایش گرافیکی درخت تصمیم بهینه :



نمودار زمان اجرا برای هر مدل :

زمان اجرا(time complexity) : knn>rf>adaboost>dt



ضمیمه : مقایسه دقیق تر این 4 مدل :

مدل	پیچیدگی محاسباتی (Training)	پیچیدگی محاسباتی (Inference)	سرعت اجرا (Training)	سرعت اجرا (Inference)
KNN (K-Nearest Neighbors)	$O(1)$ (بدون نیاز به آموزش)	$O(n$ (جستجو در کل داده‌ها) $\cdot d)$	**بسیار سریع** (چون مدلی یاد نمی‌گیرد)	**کند** (به دلیل جستجو در کل داده‌ها)
Decision Tree (DT)	$O(n$ $\cdot \log n \cdot d)$	$O(d)$ (عبور از سطح درخت)	**نسبتاً سریع**	**بسیار سریع**
Random Forest (RF)	$O(m$ $\cdot n \cdot \log n \cdot d)$ با m (تعداد درخت‌ها)	$O(m$ $\cdot d)$	**کندتر از DT** (بسته به تعداد درخت‌ها)	**نسبتاً سریع** (چون چند درخت را بررسی می‌کند)
AdaBoost	$O(T$ $\cdot n \cdot d)$ با T (تعداد تکرارها)	$O(T$ $\cdot d)$	**نسبتاً کند** (نیاز به یادگیری چندین مدل ضعیف)	**نسبتاً سریع**

1. **KNN**:

- در مرحله‌ی آموزش **عملاً نیازی به محاسبات ندارد**، چون فقط داده‌ها را ذخیره می‌کند.
- در مرحله‌ی پیش‌بینی (Inference)، باید فاصله‌ی نمونه جدید را با **تمام داده‌های آموزشی** محاسبه کند که **بسیار کند است**، به‌ویژه برای دیتاست‌های بزرگ.

2. **Decision Tree (DT)**:

- در مرحله‌ی آموزش، ساخت درخت نیازمند مرتب‌سازی داده‌ها و محاسبه‌ی بهترین ویژگی‌ها است که پیچیدگی آن حدود $(d \log n \cdot O(n$ است.
- در مرحله‌ی پیش‌بینی، فقط باید از ریشه تا یکی از برگ‌های درخت حرکت کند، که معمولاً **بسیار سریع** است ($O(d)$).

3. **Random Forest (RF)**:

- نسخه‌ای از **Bagging** است که چندین درخت را یاد می‌گیرد. هر درخت به‌طور جداگانه یاد گرفته شده و ترکیب آن‌ها خروجی را تولید می‌کند.
- در مرحله‌ی آموزش، چون چندین درخت ساخته می‌شود، زمان بیشتری نسبت به یک درخت **DT** صرف می‌شود ($(d \log n \cdot n \cdot O(m)$).
- در مرحله‌ی پیش‌بینی، میانگین نتایج چندین درخت گرفته می‌شود، که زمان بیشتری نسبت به یک درخت **DT** می‌برد، اما همچنان **نسبتاً سریع** است.

4. **AdaBoost**:

- چندین مدل ضعیف (معمولاً درخت‌های تصمیم بسیار کوچک) را به‌صورت **سریالی** آموزش می‌دهد و در هر مرحله وزن داده‌های ورودی را تغییر می‌دهد.
- این باعث افزایش **پیچیدگی یادگیری** نسبت به یک درخت تصمیم ساده می‌شود ($(d \cdot n \cdot O(T)$).
- در مرحله‌ی پیش‌بینی، نتایج **T مدل ضعیف** ترکیب می‌شوند که **کندتر از یک درخت ولی سریع‌تر از Random Forest است**.

نتیجه‌گیری

- **KNN برای پیش‌بینی بسیار کند است** و فقط در دیتاست‌های کوچک استفاده می‌شود.
- **Decision Tree سریع‌ترین مدل است**، اما دقت کمتری نسبت به مدل‌های Ensemble دارد.
- **Random Forest دقت بالاتر ولی زمان آموزش بیشتری نیاز دارد**، مخصوصاً وقتی تعداد درخت‌ها زیاد شود.
- **AdaBoost پیچیدگی بیشتری در آموزش دارد ولی پیش‌بینی آن سریع‌تر از Random Forest است**.