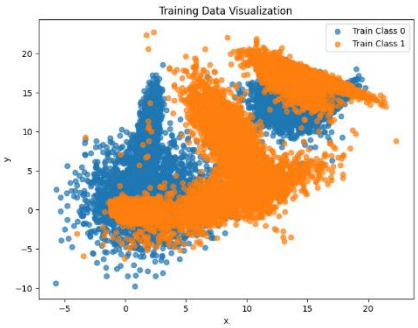
گزارش فاز سوم پروژه هوش محاسباتی

استاد درس : دکتر فضل ارثی

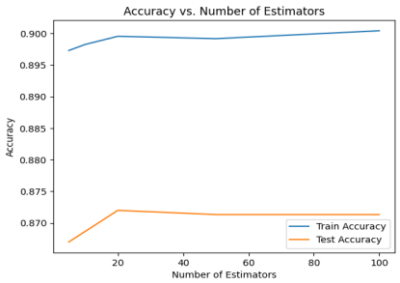
دانشجویان : غزاله بابایی – عطیه بنکدار

در این پروژه قصد داریم طبقه بند های Ensembled را بررسی کنیم. در ابتدا دیتاست آموزشی را ویژوالایزمی کنیم.



در این شکل نحوه ی توزیع دیتا در ۲ کلاس را می بینیم. دیتا در ۲ کلاس توزیع شده واین کلاس ها دارای توزیع یکنواخت نمی باشند. هم چنین میتوان مشاهده کرد که دیتاهای مربوط به یک کلاس به گونه ای پخش شده اند که میتوان 2 مرکز تجمع چگالی برای هر کلاس در نظر گرفت به گونه ای که انگار هر کلاس خود از ۲ کلاس تشکیل شده است.

در بخش **اول** الگوریتم Bagging را با کلسیفایر دسیژن تری پیاده سازی کرده ایم. درادامه دقت ها به دست آمده روی estimator های را[5, 10, 20, 50, 100] می بینیم:




```
Best Params: {'max_depth': 12, 'min_samples_leaf': 1, 'n_estimators': 20}, Best Test Accuracy: 0.8742, Best Train Accuracy: 0.9059
```

همین طور که در تصویر میبینیم دقت داده ی آموزشی کمی بیشتر از دقت داده ی تست می باشد اما با توجه به اینکه این اختلاف از ۴ درصد کمتر است (و hyperparameter tuning روی پارامتر های درخت انجام شده است) اورفیت نشده است.

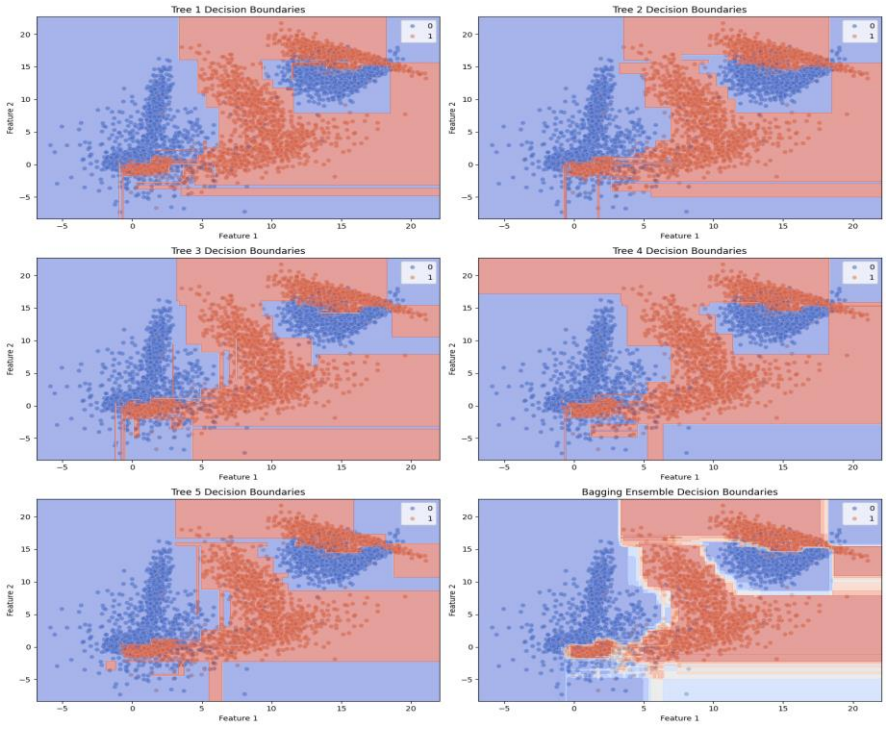
همان گونه که در نمودار و خروجی کد مشخص است بهترین دقت را در این الگوریتم ما بر روی تعداد ۲۰ estimator داریم.

خروجی ۵ کلسیفایر و الگوریتم Bagging:

همین طور که در تصویر قابل مشاهده است این کلسیفایرها تفاوت فاحشی در طبقه بندی دیتاهایی که در نواحی پرتراکم هر کدام از کلاس ها هستند ندارند و تفاوت اصلی آنها در دیتاهایی است که در فاصله یکسان ازنواحی پر تراکم هر دو کلاس دارند و تشخیص آن ها برای دسیژن تری دشوار تر می باشد. در نتیجه میتوان عملکرد درخت ها به صورت تکی را بدین صورت تحلیل کرد که هر درخت تصمیم سعی دارد داده‌ها را به‌خوبی تفکیک کند، اما به‌دلیل انعطاف‌پذیری بالا و وابستگی به نمونه داده‌های ورودی، مرزهای تصمیم‌گیری پیچیده و ناپایدارهستند. این امر منجر به بیش‌برازش و عملکرد ضعیف‌تر می‌شود.

تحلیل مدل بگینگ: بگینگ با ترکیب خروجی چندین درخت تصمیم‌گیری، خطاهای آن‌ها را جبران کرده و مدل نهایی را پایدارتر و قدرتمندتر می‌کند. در نتیجه، مرزهای تصمیم‌گیری در بگینگ هموارتر و عملکرد مدل به طور کلی بهتر از درخت‌های تکی است.

تصویر به خوبی نشان می‌دهد که بگینگ می‌تواند با کاهش واریانس و ترکیب مدل‌های ضعیف‌تر (Base Learners)، یک مدل قوی و پایدار ایجاد کند که در تفکیک کلاس‌ها عملکرد بهتری دارد.



مقایسه با sklearn bagging

خروجی کد پیاده سازی شده:

```
...
Test  - Precision: 0.8605, Recall: 0.8821, F1 Score: 0.8711
Best Params: {'max_depth': 12, 'min_samples_leaf': 1, 'n_estimators': 20}, Best Test Accuracy: 0.8742, Best Train Accuracy: 0.9059
Best Train Precision: 0.8890, Recall: 0.9052, F1 Score: 0.8970
Best Test Precision: 0.8605, Recall: 0.8821, F1 Score: 0.8711
```

خروجی sklearn :

```
Train Accuracy: 0.916125
Test Accuracy: 0.8683333333333333
Train Precision: 0.9046193673651
Test Precision: 0.8548334949886841
Train Recall: 0.9305925432756325
Test Recall: 0.8857621440536013
Train F1 Score: 0.9174221602330065
Test F1 Score: 0.8700230338927278
```

مقایسه:

عملکرد روی داده‌های آموزشی:

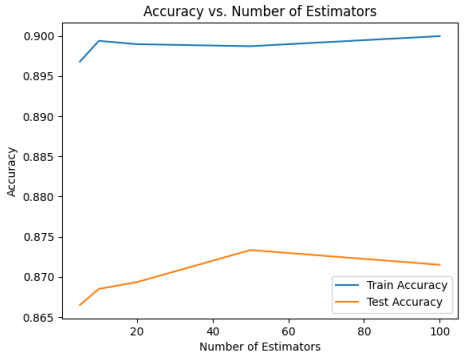
Sklearn عملکرد بهتری روی داده‌های آموزشی داشته است (دقت، یادآوری، و F1 بالاتر)، اما این ممکن است نشان‌دهنده‌ی پیچیدگی بیشتر مدل و خطر بیش‌برازش (Overfitting) باشد.

عملکرد روی داده‌های تست:

کد پیاده سازی شده کمی بهتر روی داده‌های تست عمل کرده است، که نشان‌دهنده‌ی تعمیم‌پذیری بهتر در مقایسه با sklearn است.

در **بخش دوم** پروژه الگوریتم رندوم فارست را پیاده سازی کرده ایم که عملکردی بسیار مشابه با Bagging دارد تنها با این تفاوت که تعداد ی رندوم از فیچر ها بر روی هر نود برای اسپلیت شدن انتخاب می شوند.

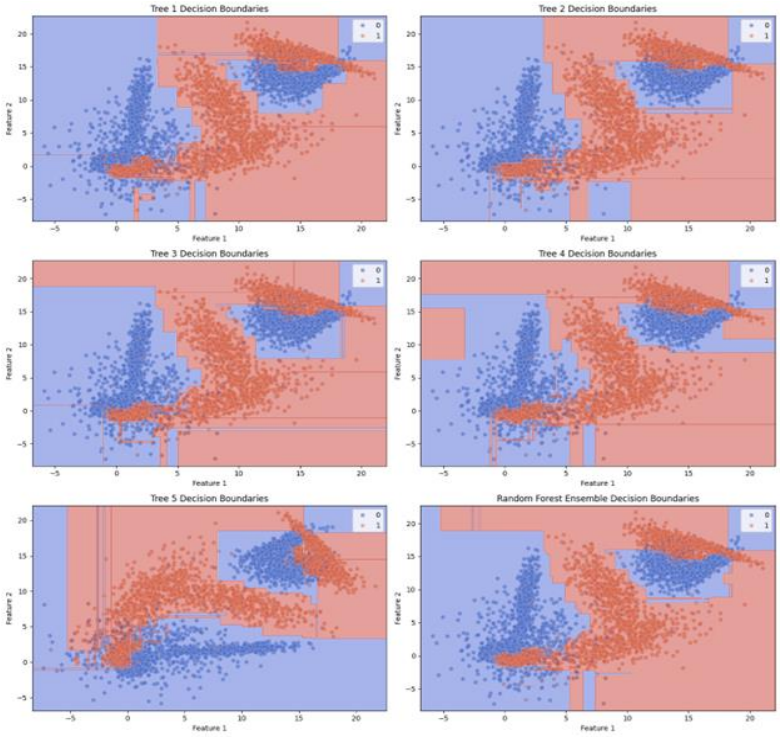
بررسی دقت های داده روی Random Forest:




```
Best Params: {'max_depth': 10, 'max_features': 2, 'min_samples_leaf': 5, 'n_estimators': 10}, Best Test Accuracy: 0.8728, Best Train Accuracy: 0.8894
```

در نمودار دقت و خروجی نهایی مشخص است که با گرفتن یک میانگین بین بهترین تعداد estimator برای هر دو داده ی آموزشی و تست می توان estimator = 10 انتخاب کرد.

تحلیل کلسیفایر های درخت تصمیم و نتیجه نهایی الگوریتم Random Forest



هر درخت تصمیم به تنهایی در Random Forest مرزهای تصمیم‌گیری بسیار مربعی و نامنظم ایجاد می‌کند. این مرزها به دلیل

تقسیم‌بندی‌های سلسله‌مراتبی و موازی با محور درخت‌های تصمیم (منظور این است که در هر مرحله از درخت تصمیم، داده‌ها فقط در امتداد یکی از محورهای ویژگی‌ها تقسیم می‌شوند)هستند.

مرزهای هر درخت متفاوت هستند به دلیل:

انتخاب تصادفی زیرمجموعه‌ای از ویژگی‌ها در هر تقسیم (ویژگی مهم جنگل تصادفی که در بگینگ وجود ندارد).

نمونه‌برداری بوت‌استرپ متفاوت برای آموزش هر درخت.

در نتیجه، برخی از درخت‌های تصمیم در نواحی خاص عملکرد خوبی دارند، درحالی‌که برخی دیگر ممکن است خطای بیشتری داشته باشند.

تحلیل مدل نهایی Random Forest

مرز تصمیم‌گیری نهایی مجموعه (نمودار پایین سمت راست) بسیار صاف‌تر و قابل تعمیم‌تر از هر درخت منفرد است.

اثر مجموعه باعث میانگین‌گیری مرزهای تصمیم هر درخت می‌شود، که باعث کاهش بیش‌برازش (Overfitting) و ایجاد مرزهای قوی‌تر می‌شود.

برخلاف درخت‌های منفرد که در برخی نواحی دچار بیش‌برازش شده‌اند، مجموعه مرزهای نامنظم و پرنویز را هموار کرده و پیش‌بینی کلی بهتری ارائه می‌دهد.

مقایسه با خروجی‌های بگینگ:

نکات کلیدی مقایسه بین Random Forest و Bagging به شرح زیر است:

انتخاب ویژگی‌ها:

- جنگل تصادفی در هر تقسیم ویژگی‌های تصادفی را انتخاب می‌کند، که باعث کاهش همبستگی بین درخت‌ها می‌شود.
- در بگینگ، از تمام ویژگی‌ها استفاده می‌شود که باعث افزایش همبستگی درخت‌ها می‌گردد.

این ویژگی تصادفی بودن در جنگل تصادفی کمک می‌کند تا تعمیم‌پذیری بهتری ایجاد شود.

مرزهای تصمیم‌گیری:

- در بگینگ، مرز تصمیم‌گیری مجموعه نیز نسبت به درخت‌های منفرد صاف‌تر است، اما ممکن است کمی بیش‌برازش کند اگر ویژگی‌ها همبستگی زیادی داشته باشند.
- در جنگل تصادفی، مرز تصمیم‌گیری هموارتر و مقاوم‌تر است به دلیل کاهش همبستگی بین درخت‌ها.

کاهش واریانس:

- هر دو روش بگینگ و جنگل تصادفی واریانس را نسبت به یک درخت منفرد کاهش می‌دهند.
- اما جنگل تصادفی به دلیل کاهش همبستگی بین درخت‌ها، واریانس کمتری دارد.

بایاس:

- هر دو روش بایاس مشابهی دارند، زیرا هر دو به درخت تصمیم به‌عنوان یادگیرنده پایه متکی هستند.

تعمیم‌پذیری:

- جنگل تصادفی معمولاً در داده‌های پرنویز یا داده‌های با ابعاد بالا بهتر عمل می‌کند.

مقایسه با sklearn RandomForest

خروجی کد پیاده سازی شده:

```
Test  - Accuracy: 0.8690, Precision: 0.8695, Recall: 0.8690, F1 Score: 0.8690
Best Params: {'max_depth': 10, 'max_features': 2, 'min_samples_leaf': 1, 'n_estimators': 20}, Best Test Accuracy: 0.8723, Best Train Accuracy: 0.8897
Best Train Precision: 0.8961, Recall: 0.8957, F1 Score: 0.8957
Best Test Precision: 0.8695, Recall: 0.8690, F1 Score: 0.8690
```

خروجی sklearn :

```
Training Set Metrics:
Accuracy: 0.9240
Precision: 0.9241
Recall: 0.9240
F1 Score: 0.9240

Test Set Metrics:
Accuracy: 0.8733
Precision: 0.8737
Recall: 0.8733
F1 Score: 0.8734
```

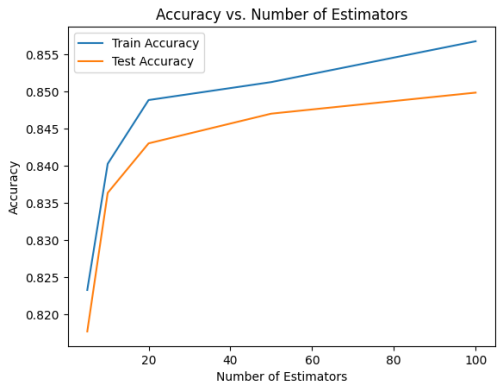
مقایسه: در خروجی sklearn کمی اورفیت دیده می شود و به طور کلی دقت ان در داده ی آموزشی بالا تر بوده و اما تفاوت چشمگیری در دقت داده ی تست دیده نمی شود. به طور کلی قابلیت generalization کد پیاده سازی شده بیشتر و دقت پیاده سازی با کتابخانه کمی بیشتر است.

در بخش سوم الگوریتم ada boost رو بررسی کردیم و نتایج خرو جی آن بدین ترتیب است :

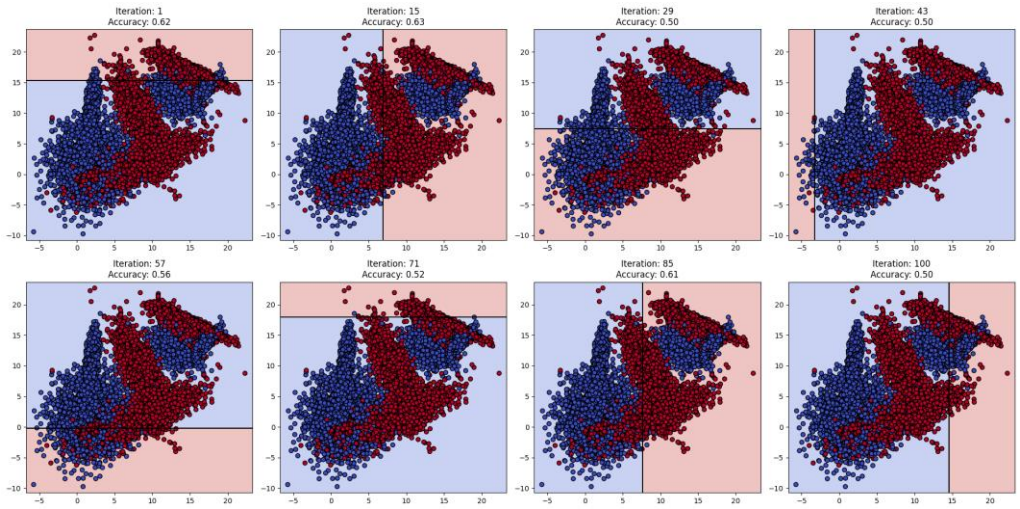
```
Training Metrics:
Accuracy: 0.8512
Precision: 0.8518
Recall: 0.8512
F1 Score: 0.8512

Testing Metrics:
Accuracy: 0.8470
Precision: 0.8474
Recall: 0.8470
F1 Score: 0.8469
```

که نشان میدهد مدل اورفیت نشده و قابلیت نسبتا خوبی در generalization دارد. نمودار دقت این الگوریتم هم این موضوع را به خوبی نشان می دهد(۱ درصد اختلاف دقت داده ی آموزشی و تست قابل چشم پوشی است)



تحلیل ۸ iteration از ADABOOST



Iteration 1 (دقت ۰/۶۲):

مرز تصمیم اولیه ساده و خطی است و نمی‌تواند داده‌ها را به خوبی تفکیک کند. داده‌های زیادی در سمت اشتباه مرز قرار گرفته‌اند.

Iteration 15 (دقت ۰/۶۳):

مرز تصمیم کمی تغییر کرده است اما هنوز عملکرد مدل بهبود جزئی داشته و دقت اندکی افزایش یافته است.

Iterations 29، ۴۳ و ۷۱ (دقت ۰/۵۰ - ۰/۵۲):

در این مراحل، مرز تصمیم به شدت نامناسب است و دقت به ۵۰٪ کاهش یافته که نشان‌دهنده عملکرد ضعیف مدل در این تکرارهاست. این می‌تواند ناشی از وزن‌دهی نادرست به داده‌ها باشد.

Iteration 57 و Iteration 85 (دقت ۰/۵۶ و ۰/۶۱):

مرز تصمیم منطقی‌تر شده و مدل شروع به بهبود کرده است. در این مرحله، برخی از اشتباهات اصلاح شده و داده‌ها به شکل بهتری تفکیک می‌شوند.

Iteration 100 (دقت ۰/۵۰):

در مرحله پایانی، مرز تصمیم دوباره عملکرد ضعیفی دارد و دقت به ۵۰٪ بازمی‌گردد. این نشان می‌دهد که الگوریتم در برخی تکرارها با نوسان مواجه شده است.

تحلیل کلی : این تصویر مربوط به مراحل اولیه یادگیری است و مدل هنوز به دقت بهینه خود نرسیده است.

در مراحل ابتدایی، الگوریتم آدا بوست سعی می‌کند مرزهای تصمیم ساده‌ای ایجاد کند و به تدریج خطاها را اصلاح کند.

نتایج نهایی نشان می‌دهند که در تکرارهای بعدی (پس از Iteration 100)، مدل توانسته خطاها را به حداقل برساند و به دقت ۸۵٪ برسد. این دقت در تصویر مشاهده نمی‌شود، زیرا مراحل نهایی یادگیری (که بهترین عملکرد مدل را نشان می‌دهند) در این تصویر نمایش داده نشده‌اند.

در بخش **چهارم** نیز با استفاده از الگوریتم stacking ۴ لرنر را با هم ترکیب کرده تا از پیش بینی نتایج همه ی آنها استفاده بکنیم.

لرنر های استفاده شده و مزایا آنها:

۱. Logistic Regression

مزایا:

- سادگی و سرعت بالا: اجرای سریع و ساده، به‌خصوص برای داده‌های کوچک و خوش‌ساختار.
- قابل تفسیر بودن: ضریب ویژگی‌ها (Coefficients) به شما اجازه می‌دهد تأثیر هر ویژگی بر پیش‌بینی نهایی را تفسیر کنید.
- مناسب برای داده‌های خطی: برای مسائلی که مرز تصمیم خطی باشد، به خوبی عمل می‌کند.
- مقاومت در برابر Overfitting: به خصوص زمانی که از Regularization استفاده شود (مثل L1 و L2).
- پشتیبانی از احتمال: خروجی به‌صورت احتمال کلاس‌ها است که در بسیاری از مسائل کاربرد دارد.

مورد استفاده:

- مسائل خطی مانند طبقه‌بندی باینری یا چندکلاسه (One-vs-All).
- زمانی که نیاز به مدلی ساده و قابل تفسیر باشد.

۲. CatBoostClassifier (الگوریتم تقویتی درختی)

مزایا:

- تعامل با ویژگی‌های دسته‌ای (Categorical Features): بدون نیاز به تبدیل داده‌های دسته‌ای به عدد (مانند One-Hot Encoding).
- سرعت بالا: نسبت به بسیاری از الگوریتم‌های بوستینگ دیگر سریع‌تر عمل می‌کند.
- کنترل Overfitting: به کمک پارامترهایی مانند depth و iterations از بیش‌برازش جلوگیری می‌کند.
- پشتیبانی از داده‌های پیچیده و غیرخطی: برای داده‌هایی که الگوهای غیرخطی دارند، بسیار مناسب است.

عملکرد بالا بدون نیاز به تنظیم زیاد: بهینه‌سازی پیش‌فرض‌ها به گونه‌ای انجام شده که نتایج خوبی ارائه دهد.

مورد استفاده:

- مسائل پیچیده طبقه‌بندی که داده‌ها دارای ویژگی‌های غیرخطی یا دسته‌ای هستند.
- زمانی که دقت مدل در اولویت است و می‌خواهید از الگوریتم‌های تقویتی استفاده کنید.

۳. Linear Discriminant Analysis (تحلیل تفکیک خطی - LDA)

مزایا:

- مدل خطی و سریع: مانند رگرسیون لجستیک، عملکرد سریع و محاسبات ساده دارد.
- کاهش ابعاد (Dimensionality Reduction): می‌تواند ویژگی‌ها را به فضایی با ابعاد کمتر نگاشت کند.
- مفروضات آماری قدرتمند: بر اساس توزیع گاوسی داده‌ها (Normal Distribution)، که برای داده‌های توزیع‌شده خطی مناسب است.
- تفسیرپذیری بالا: مشابه رگرسیون لجستیک، به‌راحتی قابل تفسیر است.

مورد استفاده:

- داده‌های خطی که فرض توزیع گاوسی بر روی داده‌ها برقرار است.
- زمانی که نیاز به ترکیب کاهش ابعاد و طبقه‌بندی دارید.

۴. GaussianNB (نایو بیزین با توزیع گاوسی)

مزایا:

- سرعت بسیار بالا: یکی از سریع‌ترین الگوریتم‌های یادگیری ماشین، به دلیل محاسبات ساده.
- مناسب برای داده‌های کوچک و پراکنده: حتی با حجم داده‌های کم، به خوبی عمل می‌کند.
- تعامل با داده‌های پیوسته: فرض توزیع گاوسی برای ویژگی‌های پیوسته بسیار مناسب است.

- ساده و قابل تفسیر: بر اساس احتمالات عمل می‌کند که خروجی آن تفسیرپذیر است.
- پشتیبانی از چندکلاسه بودن: به‌راحتی برای طبقه‌بندی چندکلاسه (Multiclass) قابل استفاده است.

مورد استفاده:

- زمانی که فرض مستقل بودن ویژگی‌ها (Naive Assumption) تقریباً برقرار باشد.
- داده‌های با توزیع گاوسی و ساده.
- مسائل طبقه‌بندی سریع مانند فیلترهای ایمیل اسپم یا داده‌های کم‌حجم.

با توجه به مزایا این لرنر ها انتظار داریم که از ترکیب ا نها یک متا مدل مناسب برای ترین کردن داده های خود ایجاد کنیم.

نتایج حاصل:

```
2024-12-18 12:13:34,073 - INFO - Training Set Metrics:
2024-12-18 12:13:34,076 - INFO - Training Accuracy: 0.882
2024-12-18 12:13:34,101 - INFO - Training Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88     11984
           1       0.87      0.90      0.88     12016

    accuracy                           0.88     24000
   macro avg       0.88      0.88      0.88     24000
weighted avg       0.88      0.88      0.88     24000
...

2024-12-18 12:13:36,145 - INFO - Test F1 Score: 0.872449781495583
2024-12-18 12:13:36,148 - INFO - Test Precision: 0.8732842548076923
2024-12-18 12:13:36,151 - INFO - Test Recall: 0.8725
```

که دقت روی داده های تست و آموزشی و اختلاف نا چیز آنها نشان میدهد که مدل از تعمیم پذیری مناسبی برخوردار است.

همچنین نتیجه ی دقت هر کدام از لرنر ها نشان میدهد که هیچ کدام اورفیت نشده اند:

```
Model 1: Train Accuracy = 0.5858, Test Accuracy = 0.5878
Model 2: Train Accuracy = 0.8820, Test Accuracy = 0.8725
Model 3: Train Accuracy = 0.5838, Test Accuracy = 0.5847
Model 4: Train Accuracy = 0.6248, Test Accuracy = 0.6218
```