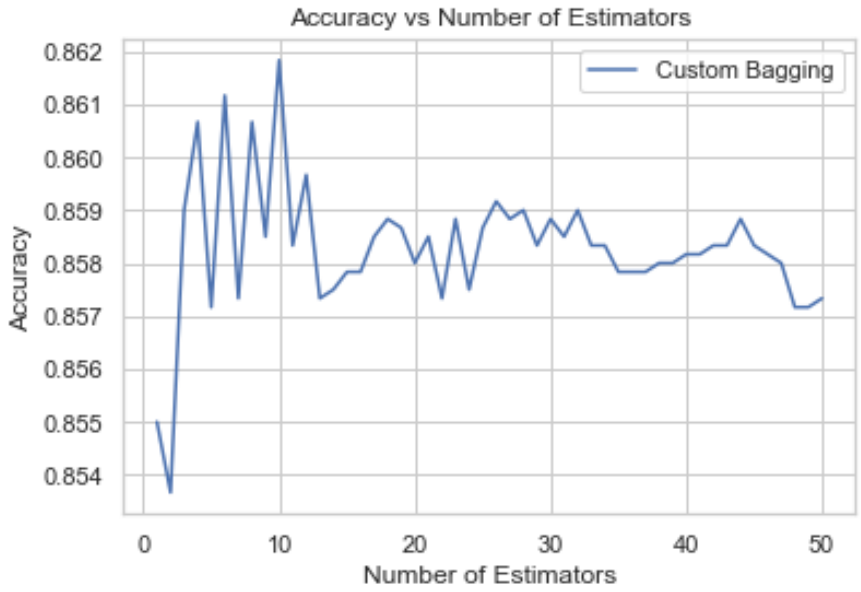
۱-Bagging:

۱.۱-custom:

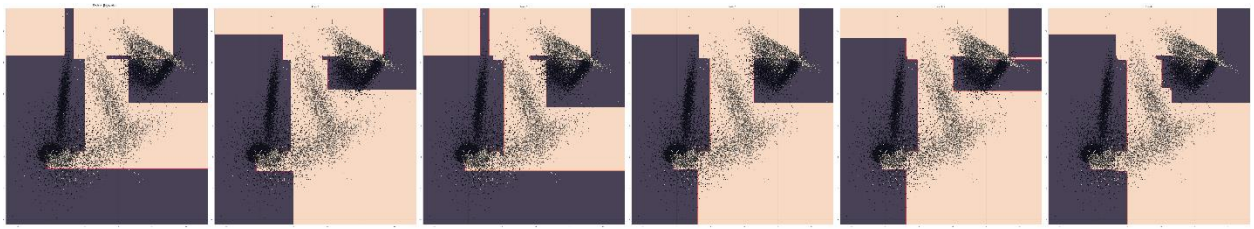
ابتدا یک درخت تصمیم با اعماق محدود و گرید سرچ زیر train میکنیم.

```
param_grid = {
    'max_depth': [1, 2, 3, 4, 5],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [1, 2, 3, 4]
}
```

سپس با این درخت به دنبال بهترین accuracy در میان ۱ تا ۵۰ estimator می گردیم.



بهترین تعداد estimatorها در این بازه ۱۰ است و سپس با آن مدل را میسازیم.



نتایج به این صورت میباشد که راستترین تصویر classifier نهایی است که مشاهده میشودبهتر و جزیی تر

است و نسبت به روند نویز دار estimatorها بهتر عمل کرده است.

Training Metrics:

Accuracy: 0.8626

Precision: 0.8657

Recall: 0.8587

F1-Score: 0.8622

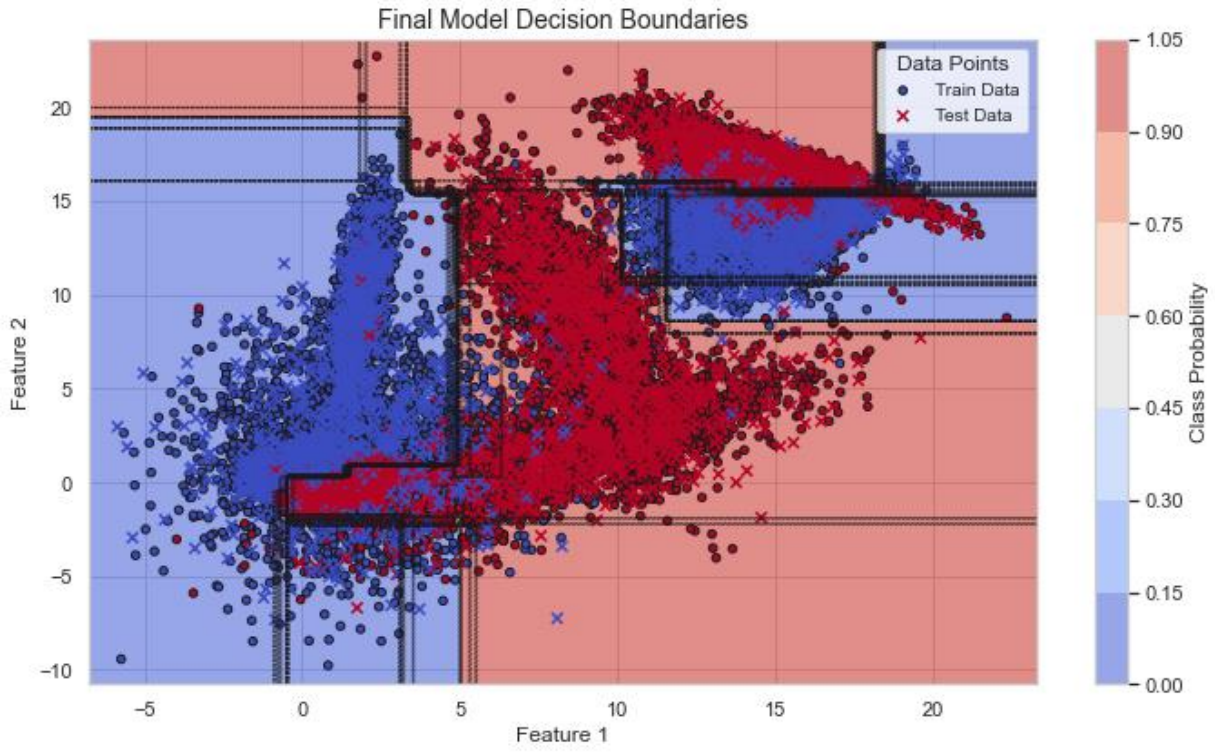
Testing Metrics:

Accuracy: 0.8618

Precision: 0.8627

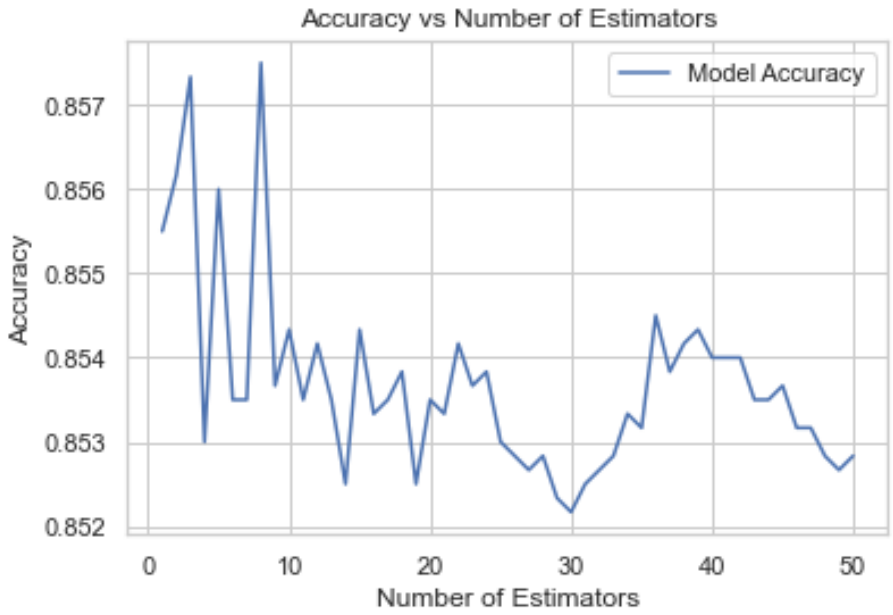
Recall: 0.8590

F1-Score: 0.8608



Classifier نهایی به همراه خطوط جدا کننده estimatorها

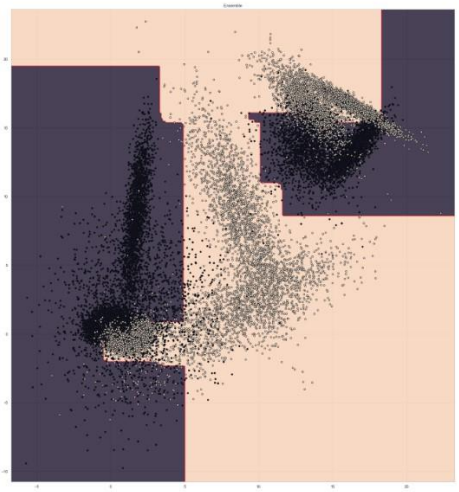
۱.۲-library:



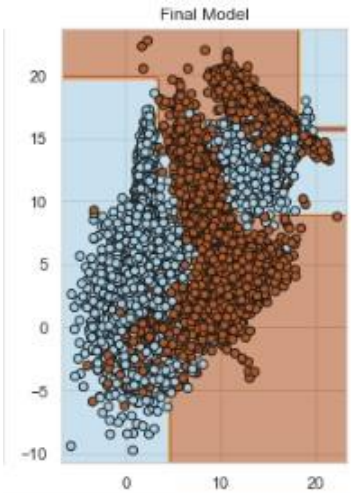
با همان درخت بخش قبل این نتایج حاصل میشود و بهترین تعداد estimatorها برای این مدل عدد ۸ میشود.

	Accuracy	Precision	Recall	F1-Score
Train	0.860083	0.858124	0.863266	0.860687
Test	0.857500	0.853117	0.861977	0.857524

مدل custom:



مدل آماده:

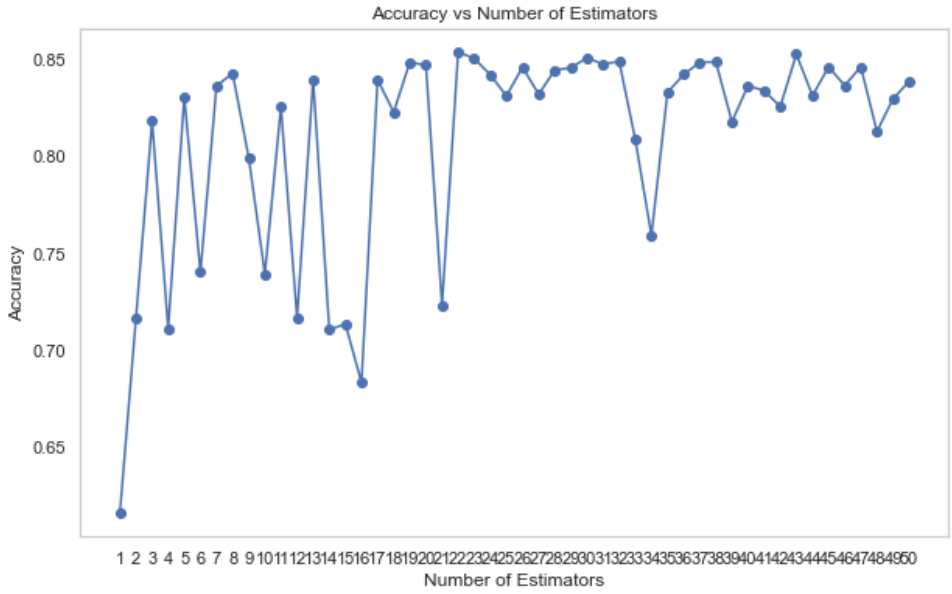


همانطور که مشاهده میشود رفتار دو الگوریتم کاملا مشابه است.

۲-Random Forest:

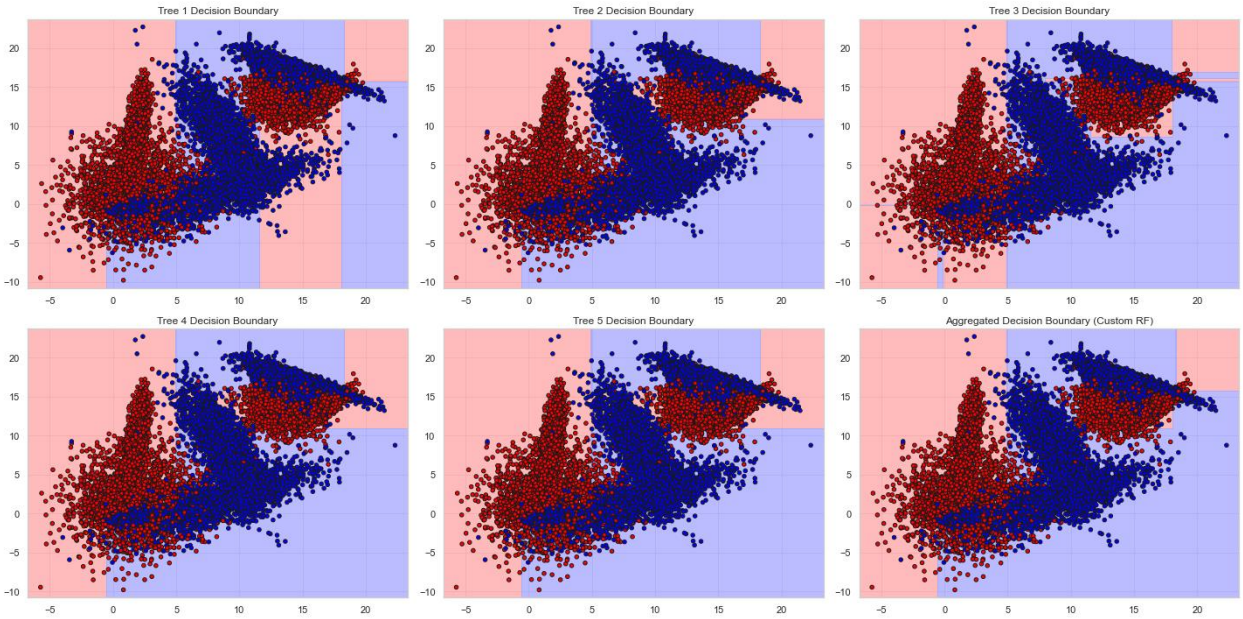
۱.۱-custom:

با grid search بخش اول مدل را در بازه ۱تا۵۰ estimator تمرین میدهیم و نتیجه زیر حاصل میشود.



با توجه به نتایج بهترین تعداد estimator برای این مدل۲۲ است. با این تعداد estimator مدل را train میکنیم:

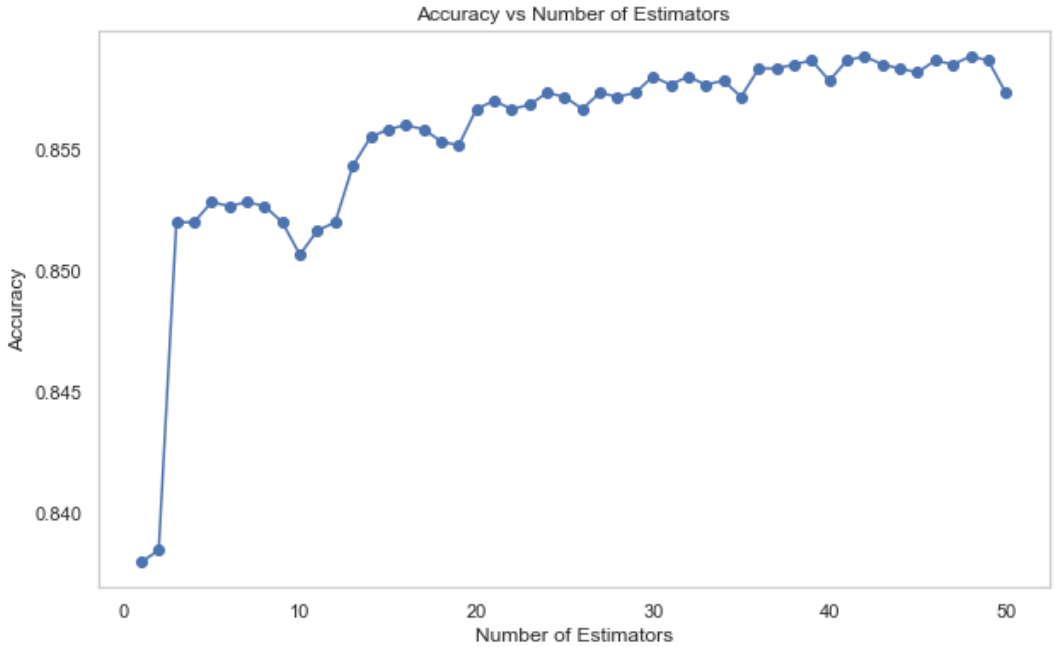
	Accuracy	Precision	Recall	F1-Score
Train	0.835042	0.805120	0.884654	0.843015
Test	0.829167	0.799145	0.877052	0.836288



همانطور که مشخص است مدل aggregated عملکرد مطلوبی دارد و الگوهای با کیفیت درخت ها را دنبال میکند و از تصمیمات اشتباه و نویزدار درخت هایی مانند درخت۱ بدور است.

۱.۲-library:

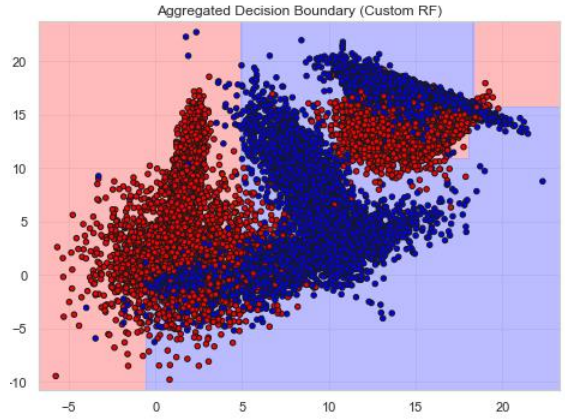
همانند بخش قبل بدنبال بهترین تعداد estimator میگردیم و نتیجه زیر حاصل میشود.

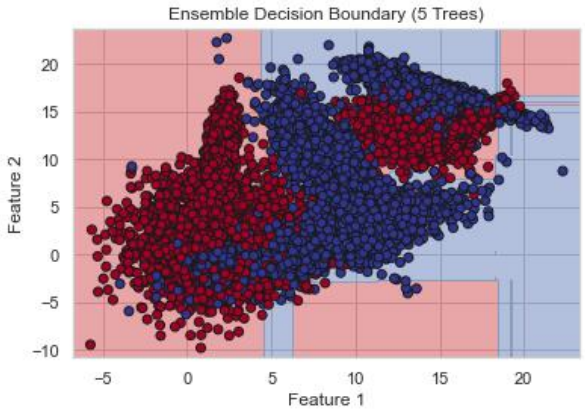


تعداد estimator ها ۴۲ تا انتخاب میشود و با این تعداد estimator مدل را train میکنیم.

Accuracy on Train Set: 0.8597916666666666

Accuracy on Test Set: 0.8528333333333333





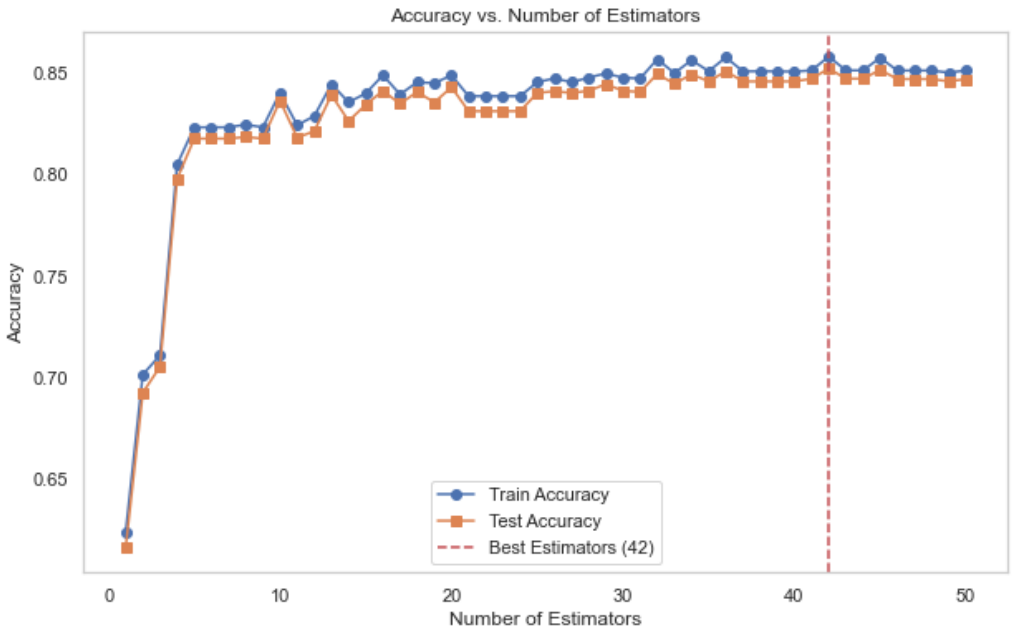
مدل custom

مدل آماده

دو مدل رفتار خیلی نزدیکی دارند ولی مدل آماده در تشخیص داده های پایین تصویر بهتر عمل کرده است.

۳-Adaboost:

ابتدا همانند بخشهای قبل به دنبال بهترین تعداد estimator میگردیم.



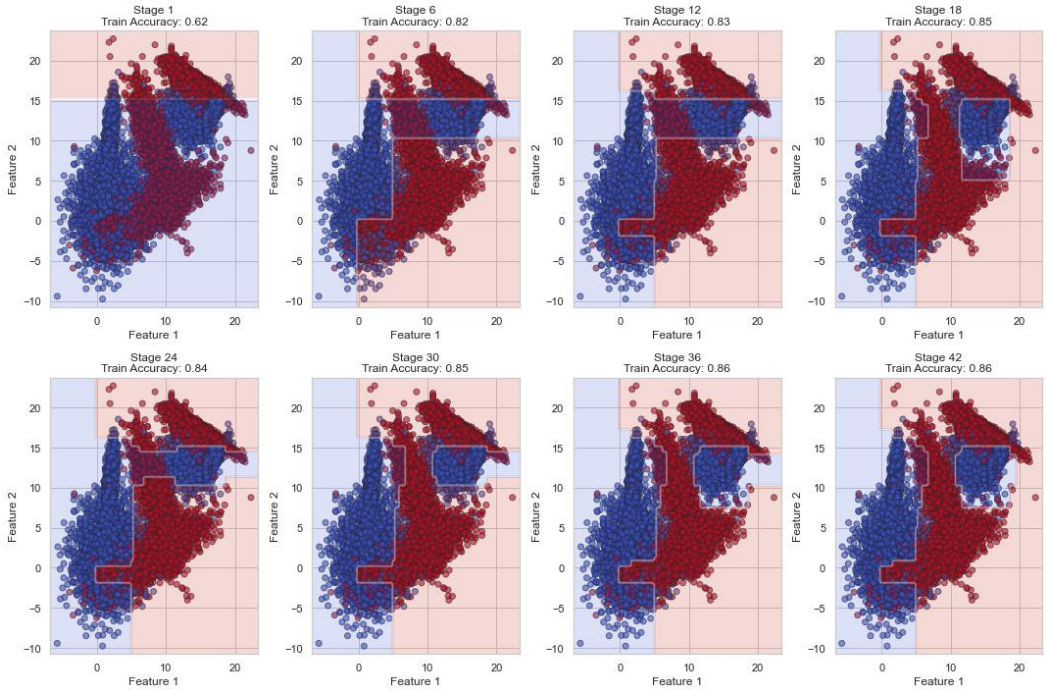
بهترین تعدادestimator برای این مدل ۴۲ است. با این تعداد و weak classifier زیر مدل را train میکنیم.

```
weak_classifier = DecisionTreeClassifier(max_depth=1)
```

نتایج:

	Accuracy	Precision	Recall	F1-Score
Train	0.858042	0.858115	0.858042	0.858036
Test	0.852167	0.852206	0.852167	0.852157

حال تغییرات نحوه classify کردن این مدل را در stageهای متفاوت بررسی میکنیم.



مشخص است که الگوریتم با گذر زمان در شکل گیری class های خود بهتر عمل میکند و کلاسهای بهتری درست میکند و رفته رفته از دسته های کلی و یکدست به دسته های دقیقتر میرسد.

۴-Stacked Learners:

با استفاده از classifierها و hyperparameter tuning زیر base model خود را میسازیم.

```
base_model_params = {
    'rf': {
        'model': RandomForestClassifier(random_state=42),
        'params': {
            'n_estimators': [100],
            'max_depth': [None]
        }
    },
    'gb': {
        'model': GradientBoostingClassifier(random_state=42),
        'params': {
            'n_estimators': [50, 100, 150],
            'learning_rate': [0.01, 0.1, 0.2],
            'max_depth': [3, 5, 7]
        }
    },
    'dtree': {
        'model': DecisionTreeClassifier(random_state=42),
        'params': {
            'max_depth': [None, 5, 10, 20],
            'min_samples_split': [2, 5, 10]
        }
    },
    'mlp': {
        'model': MLPClassifier(max_iter=500, random_state=42),
        'params': {

        }
    }
}
```

این مدل ها را بروی داده های تست train کرده و نتایج زیر حاصل میشود.

rf - Best Parameters: {'max_depth': None, 'n_estimators': 100}

rf - Train Accuracy: 1.0000, Validation Accuracy: 0.8618

rf Metrics:

	Accuracy	Precision	Recall	F1-Score
Train	1.00	1.00000	1.000000	1.000000
Test	0.86	0.85268	0.868677	0.860604

gb - Best Parameters: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 50}

gb - Train Accuracy: 0.8827, Validation Accuracy: 0.8739

gb Metrics:

	Accuracy	Precision	Recall	F1-Score
Train	0.881833	0.868556	0.900216	0.884103
Test	0.872333	0.856180	0.893467	0.874426

dtree - Best Parameters: {'max_depth': 10, 'min_samples_split': 10}

dtree - Train Accuracy: 0.8931, Validation Accuracy: 0.8657

dtree Metrics:

	Accuracy	Precision	Recall	F1-Score
Train	0.891708	0.879748	0.907790	0.893549
Test	0.862833	0.851659	0.877052	0.864169

mlp - Best Parameters: {}

mlp - Train Accuracy: 0.8712, Validation Accuracy: 0.8695

mlp Metrics:

	Accuracy	Precision	Recall	F1-Score
Train	0.871667	0.864913	0.881325	0.873042
Test	0.867667	0.856957	0.881072	0.868847

در این بخش مدل random forest به دلیل نداشتن hyperparameterهای مناسب overfit میشود.

برای جلوگیری از تاثیر منفی این مدل از Regression_Logistic استفاده میکنیم.

دلایل:

1- وزن‌دهی به مدل‌های بهتر

- Logistic Regressionبر اساس متا-ویژگی‌ها (که خروجی مدل‌های پایه هستند) یاد می‌گیرد که چگونه به هر مدل پایه وزن مناسب بدهد.
- مدل‌هایی که عملکرد ضعیف دارند (به دلیل overfitting یا سایر مشکلات) معمولاً خروجی‌هایی با دقت پایین‌تر تولید می‌کنند Logistic Regression .می‌تواند این خروجی‌ها را با وزن‌دهی کمتر نسبت به مدل‌هایی که عملکرد بهتری دارند، به حداقل برساند.

2- جلوگیری از تقویت بیش‌برازش

- Logistic Regressionیک مدل خطی و ساده است که روابط بین متا-ویژگی‌ها و خروجی نهایی را مدل می‌کند. این سادگی باعث می‌شود که از تقویت الگوهای غیرواقعی و پیچیدگی‌های بیش از حد که ممکن است در متا-ویژگی‌های ناشی از مدل‌های overfit وجود داشته باشد، جلوگیری شود.

3-محدودیت در جلوگیری کامل ازoverfitting

- Logistic Regressionنمی‌تواند کاملاً اثر مدل‌های overfit را حذف کند. اگر یک یا چند مدل پایه کاملاً نادرست باشند و خروجی‌های غیرمنطقی ارائه دهند، ممکن است همچنان بر عملکرد مدل متا تأثیر منفی بگذارند.
- برای بهبود بیشتر، می‌توان مدل‌های پایه را با روش‌های منظم‌سازی (regularization) یا بهینه‌سازی پارامترها تقویت کرد.

نقش K-Fold در تولید متا-ویژگی‌ها

- استفاده از **K-Fold Cross-Validation**برای تولید متا-ویژگی‌ها به جلوگیری از ورود اطلاعات بیش‌برازش‌شده به مدل متا کمک می‌کند. این تکنیک باعث می‌شود Logistic Regressionاز پیش‌بینی‌های واقعی‌تر برای آموزش استفاده کند، نه پیش‌بینی‌هایی که ناشی از overfittingروی داده‌های آموزشی باشند

در نهایت دقت مدل نهایی به حالت زیر در می آید.

Meta-Model Metrics:

	Accuracy	Precision	Recall	F1-Score
Train	0.875583	0.864937	0.890563	0.877563
Test	0.871833	0.860208	0.886432	0.873123