دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکدهٔ مهندسی برق - گروه مهندسی کنترل

یادگیری ماشین
پاسخ مینی پروژه دوم

نام و نام خانوادگی: سیدمحمد حسینی
شمارهٔ دانشجویی: ۹۹۰۱۳۹۹
تاریخ: مهرماه ۱۴۰۲
GitHub
Drive Google

فهرست مطالب

فهرست تصاویر

فهرست جداول

فهرست برنامه‌ها

۱ سوال ۲

۱.۱ مشکل مورد بررسی

مقاله به چالش مدل‌های مبتنی بر SVM به منظور طبقه‌بندی چندکلاسه می‌پردازد. هایSVM سنتی ، عملکرد Binary Classification دارند این در حالی که برای مسائل چند کلاسه، نیاز به سازگار کردن مدل یا رویکرد تصمیم‌گیری، به منظور مدیریت تمام کلاس‌ها می‌باشد. رویکردهای کلاسیک موجود مانند روش‌های one-vs-one و one-vs-all ، اگرچه محبوب هستند، محدودیت‌هایی دارند. روش one-vs-one شامل حل $K(K-1)$ مسئله دودویی است که برای Kهای بزرگ باعث افزایش شدید بار محاسباتی می‌شود. از سوی دیگر، روش one-vs-all می‌تواند مناطق تصمیم‌گیری ایجاد کند که متعلق به هیچ کلاسی نمی‌باشد.

۲.۱ روش پیشنهادی

در این مقاله یک مدل به اسم Generalized Multiclass Support Vector Machine (GenSVM) پیشنهاد داده شده است که برای مدیریت طبقه‌بندی چندکلاسه بدون تجزیه مسئله به چندین طبقه‌بندی دودویی طراحی شده است. روش GenSVM از بهینه‌سازی iterative majorization ، یک تکنیک بهینه‌سازی ریاضی، برای حل مسئله طبقه‌بندی استفاده می‌کند. این روش یک تابع بزرگی ایجاد می‌کند که به حداقل رساندن آن آسان‌تر از تابع هدف اصلی است و تضمین می‌کند که به سمت راه‌حل بهینه همگرا می‌شود. الگوریتم بهینه‌سازی تکراری شامل مراحل تعیین نقطه شروع، به حداقل رساندن تابع بزرگی و تکرار تا همگرایی است. دلیل استفاده از این الگوریتم این است که عموما بهینه کردن یک تابع dual می‌تواند هزینه محاسباتی زیادی داشته باشد.

۳.۱ ارزیابی عملکرد

عملکرد GenSVM با استفاده از آزمایشات گسترده‌ای محاسبه شده است تا با بقیه مدل‌های موجود در زمینه هایSVM چندکلاسه مقایسه شود. برای ارزیابی جامع عملکرد، از مجموعه داده‌های بزرگ و کوچک استفاده شد. استفاده از مجموعه داده‌های بزرگ به منظور بررسی مقیاس‌پذیری و کارایی محاسباتی در شرایط واقعی بود، در حالی که مجموعه داده‌های کوچک به منظور بررسی دقت طبقه‌بندی و قابلیت تعمیم در شرایط مختلف مورد استفاده قرار گرفتند.

برای مقایسه روش‌ها، از شاخص adjusted Rand index (ARI) برای اندازه‌گیری دقت طبقه‌بندی و همچنین از روش‌های رتبه‌بندی برای ارزیابی کارایی نسبی استفاده شد. روش رتبه‌بندی بر اساس زمان آموزش و دقت طبقه‌بندی هر روش انجام شد. در این مطالعه، GenSVM با چندین روش موجود در زمینه هایSVM چندکلاسه مقایسه شد که شامل روش‌های one-vs-all، one-vs-one و سایر روش‌های بهینه‌سازی چندکلاسه می‌شود.

نتایج نشان داد که GenSVM به طور قابل توجهی از نظر دقت طبقه‌بندی و کارایی محاسباتی برتری دارد. GenSVM بهترین نتایج را برای سریع‌ترین روش در جستجوی شبکه‌ای داشت و زمان آموزش متوسط آن به طور قابل توجهی کمتر از سایر روش‌ها بود. علاوه بر این، این روش نشان داد که در مقابل داده‌های مختلف مقاوم و قابل توسعه است و مقیاس‌پذیری و قابلیت اطمینان آن را نشان می‌دهد. این مدل بعد از روش one-vs-one

بهترین دقت را کسب کرده است.

۴.۱ مزایا و معایب

مزایا:

- **سرعت**: GenSVM زمان محاسبات را به طور قابل توجهی نسبت به روش‌های سنتی کاهش می‌دهد.
- **دقت**: این روش از نظر دقت طبقه‌بندی از اکثر روش‌های موجود پیشی می‌گیرد.
- **مقیاس‌پذیری**: GenSVM مقاوم و مقیاس‌پذیر است و در مقابل مجموعه داده‌های مختلف عملکرد خوبی دارد.
- **حذف ناحیه ابهام**: خروجی این مدل، فضای ویژگی‌ها را به گونه‌ای افراز می‌کند که هیچ ناحیه‌ای بدون کلاس مشخص وجود نداشته باشد.
- **یک مدل به عنوان خروجی**: در این مدل به جای آموزش چندین svm برای تشخیص هر کلاس، یک مدل آموزش داده می‌شود که تمامی کلاس‌ها را پیشبینی کند.

معایب:

- **پیچیدگی**: پیاده‌سازی بهینه‌سازی تکراری و مدیریت مجموعه‌ای بزرگتر از هاhyperparameter می‌تواند پیچیده باشد .
- **منابع مورد نیاز**: علیرغم بهبودهای کارایی، نیاز اولیه به منابع محاسباتی می‌تواند بالا باشد، به ویژه برای مجموعه داده‌های بسیار بزرگ درحالی که بخواهیم تمام هایhyperparameter موجود را بررسی کنیم.

در نتیجه، روش پیشنهادی GenSVM پیشرفت قابل توجهی در زمینه طبقه‌بندی چندکلاسه با هاSVM ارائه می‌دهد، دقت و کارایی بهتری را به ارمغان می‌آورد، اگرچه با افزایش پیچیدگی و نیازهای منابع همراه است.

۵.۱ مشکل مورد بررسی

مقاله به چالش مدل‌های مبتنی بر SVM به منظور طبقه‌بندی چندکلاسه می‌پردازد. SVM های سنتی، عملکرد Binary Classification دارند این در حالی که برای مسائل چند کلاسه، نیاز به سازگار کردن مدل یا رویکرد تصمیم‌گیری، به منظور مدیریت تمام کلاس‌ها می‌باشد.

به صورت کلی سه نوع رویکرد برای هایSVM چندکلاسه وجود دارد: روش‌های heuristic که شامل one-vs-one و one-vs-all می‌باشد. روش‌های مذکور one-vs-one و one-vs-all، اگرچه محبوب و پر استفاده هستند، محدودیت‌هایی دارند. روش one-vs-one شامل حل $K(K-1)$ مسئله دودویی است که برای Kهای بزرگ باعث افزایش شدید بار محاسباتی می‌شود. از سوی دیگر، روش one-vs-all باید k مرتبه تمام دیتاست را استفاده کند تا یک مدل طبقه‌بندی ایجاد کند . این روش‌ها که مبتنی بر تبدیل مسئله به زیرمسئله‌های کوچکتر است، می‌تواند مناطق تصمیم‌گیری ایجاد کنند که متعلق به هیچ کلاسی نمی‌باشد.

روش دیگر مبتنی بر اصلاح خطا می‌باشد که حالت کلی‌تر روش قبلی می‌باشد. ایراد مدل‌های مبتنی بر این روش این است که ماتریس اصلاح باید از قبل مشخص باشد که در بسیاری از موارد انجام اینکار بسیار دشوار است. آخرین روش مبتنی بر طراحی و آموزش مدل‌هایی است که مستقیما مسئله چند کلاسه را با استفاده از یک ماشین SVM حل کند.

۶.۱ روش پیشنهادی

در این مقاله یک مدل به اسم Generalized Multiclass Support Vector Machine (GenSVM) پیشنهاد داده شده است که برای مدیریت طبقه‌بندی چندکلاسه بدون تجزیه مسئله به چندین طبقه‌بندی دودویی طراحی شده است. روش GenSVM از بهینه‌سازی iterative majorization، یک تکنیک بهینه‌سازی ریاضی، برای حل مسئله طبقه‌بندی استفاده می‌کند. این روش یک تابع بزرگی ایجاد می‌کند که به حداقل رساندن آن آسان‌تر از تابع هدف اصلی است و تضمین می‌کند که به سمت راه‌حل بهینه همگرا می‌شود. الگوریتم بهینه‌سازی تکراری شامل مراحل تعیین نقطه شروع، به حداقل رساندن تابع بزرگی و تکرار تا همگرایی است. دلیل استفاده از این الگوریتم این است که عموماً بهینه کردن یک تابع دوگان می‌تواند هزینه محاسباتی زیادی داشته باشد.

GenSVM یک طبقه‌بند تک‌ماشینه است که از فضای Simplex استفاده می‌کند. Simplex فضایی است که در آن بردارهای ویژگی‌ها به نقاطی در داخل یک مثلث نگاشت می‌شوند. یکی از مزایای استفاده از فضای Simplex این است که تصمیم‌گیری در این فضا ساده‌تر است و مرزهای تصمیم‌گیری به طور طبیعی شکل می‌گیرند.

این مدل از تابع هزینه Huber Hinge استفاده می‌کند که مشتق‌پذیر می‌باشد. نگاشت از فضای ورودی به فضای Simplex با کمینه‌سازی خطاهای طبقه‌بندی بهینه‌سازی می‌شود که با اندازه‌گیری فاصله یک نمونه از مرزهای تصمیم‌گیری در فضای Simplex محاسبه می‌شود. پیش‌بینی کلاس نیز در همین فضای Simplex انجام می‌شود.

در بهینه‌سازی تابع هزینه این مدل pـ نرم را بهینه می‌کند. مرزهای تصمیم‌گیری این مدل در فضای Simplex به مثلثی مربوط می‌شوند که نقاط درون آن به کلاس‌های مختلف تعلق دارند. تابع زیان GenSVM این امکان را فراهم می‌کند که مدل‌های تک‌ماشینه قبلی در قالب طبقه‌بندهای نوع سوم پیاده‌سازی شوند.

۷.۱ ارزیابی عملکرد

عملکرد GenSVM با استفاده از آزمایشات گسترده‌ای محاسبه شده است تا با بقیه مدل‌های موجود در زمینه هایSVM چندکلاسه مقایسه شود. برای ارزیابی جامع عملکرد، از مجموعه داده‌های بزرگ و کوچک استفاده شد. استفاده از مجموعه داده‌های بزرگ به منظور بررسی مقیاس‌پذیری و کارایی محاسباتی در شرایط واقعی بود، در حالی که مجموعه داده‌های کوچک به منظور بررسی دقت طبقه‌بندی و قابلیت تعمیم در شرایط مختلف مورد استفاده قرار گرفتند.

برای مقایسه روش‌ها، از شاخص adjusted Rand index (ARI) برای اندازه‌گیری دقت طبقه‌بندی و همچنین از روش‌های رتبه‌بندی برای ارزیابی کارایی نسبی استفاده شد. روش رتبه‌بندی بر اساس زمان آموزش و دقت طبقه‌بندی هر روش انجام شد. در این مطالعه، GenSVM با چندین روش موجود در زمینه هایSVM

چندکلاسه مقایسه شد که شامل روش‌های one-vs-all، one-vs-one و سایر روش‌های بهینه‌سازی چندکلاسه می‌شود.

نتایج نشان داد که GenSVM به طور قابل توجهی از نظر دقت طبقه‌بندی و کارایی محاسباتی برتری دارد. GenSVM بهترین نتایج را برای سریع‌ترین روش در جستجوی شبکه‌ای داشت و زمان آموزش متوسط آن به طور قابل توجهی کمتر از سایر روش‌ها بود. علاوه بر این، این روش نشان داد که در مقابل داده‌های مختلف مقاوم و قابل توسعه است و مقیاس‌پذیری و قابلیت اطمینان آن را نشان می‌دهد. این مدل بعد از روش one-vs-one بهترین دقت را کسب کرده است.

۸.۱ مزایا و معایب

مزایا:

- **سرعت**: GenSVM زمان محاسبات را به طور قابل توجهی نسبت به روش‌های سنتی کاهش می‌دهد.
- **دقت**: این روش از نظر دقت طبقه‌بندی از اکثر روش‌های موجود پیشی می‌گیرد.
- **مقیاس‌پذیری**: GenSVM مقاوم و مقیاس‌پذیر است و در مقابل مجموعه داده‌های مختلف عملکرد خوبی دارد.
- **حذف ناحیه ابهام**: خروجی این مدل، فضای ویژگی‌ها را به گونه‌ای افراز می‌کند که هیچ ناحیه‌ای بدون کلاس مشخص وجود نداشته باشد.
- **یک مدل به عنوان خروجی**: در این مدل به جای آموزش چندین SVM برای تشخیص هر کلاس، یک مدل آموزش داده می‌شود که تمامی کلاس‌ها را پیش‌بینی کند.

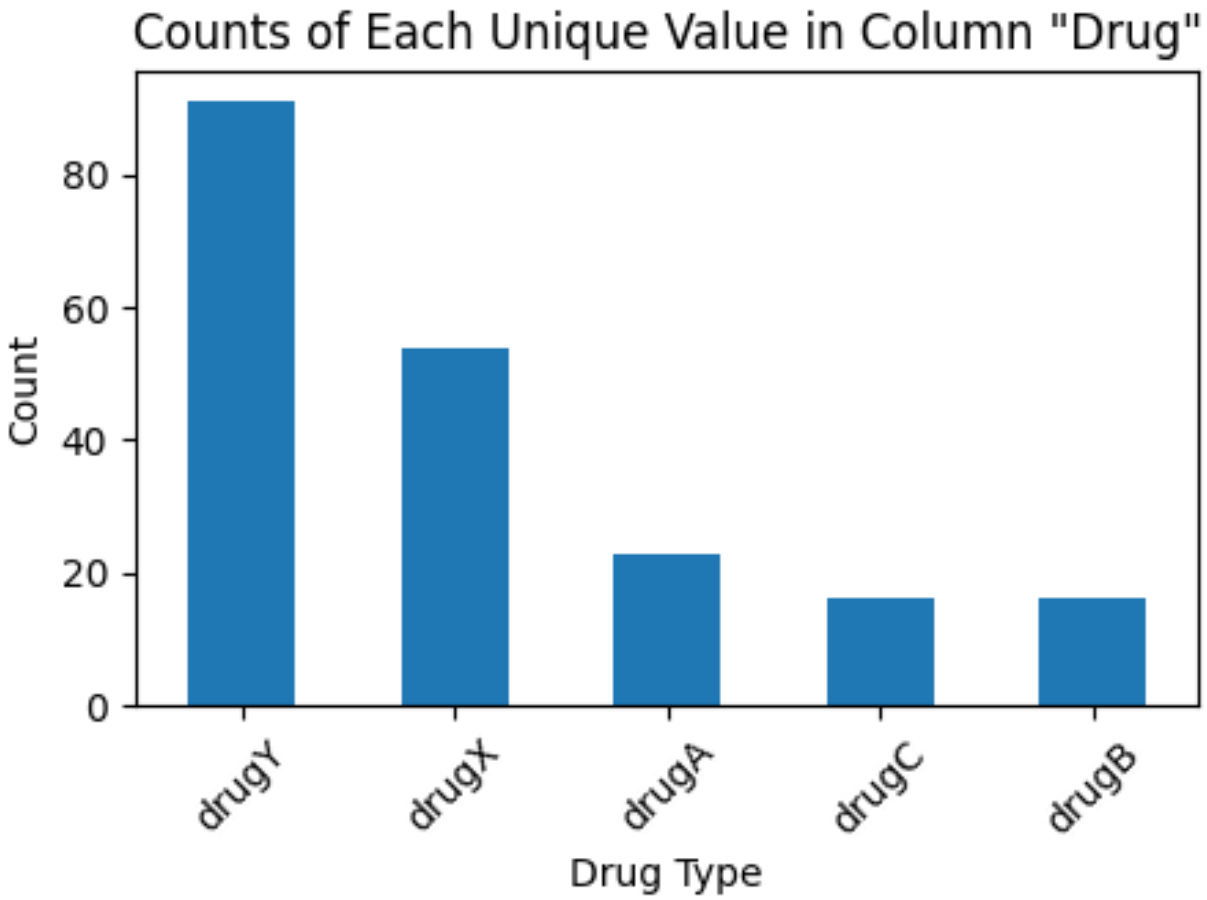
معایب:

- **پیچیدگی**: پیاده‌سازی بهینه‌سازی تکراری و مدیریت مجموعه‌ای بزرگتر از هاhyperparameter می‌تواند پیچیده باشد.
- **منابع مورد نیاز**: علیرغم بهبودهای کارایی، نیاز اولیه به منابع محاسباتی می‌تواند بالا باشد، به ویژه برای مجموعه داده‌های بسیار بزرگ درحالی که بخواهیم تمام هایhyperparameter موجود را بررسی کنیم.

در نتیجه، روش پیشنهادی GenSVM پیشرفت قابل توجهی در زمینه طبقه‌بندی چندکلاسه با هاSVM ارائه می‌دهد، دقت و کارایی بهتری را به ارمغان می‌آورد، اگرچه با افزایش پیچیدگی و نیازهای منابع همراه است.

۲ سوال ۳

در ادامه حل این سوال مجموعه داده دارو درنظر گرفته شده است و الگوریتم‌ها به روی این مجموعه داده اجرا شده‌اند. در ادامه تحلیل آماری مجموعه داده دارو انجام شده است. همانطور که در شکل ۱ مشخص شده است این دیتاست تا حدی imbalance است و فراوانی داروی x و y بیشتر از باقی داروها می‌باشد اعداد دقیق مربوط به فراوانی این دیتاست در شکل ۲ نشان‌داده‌شده است.



شکل ۱: نمودار توزیع داده

```
The number of unique values in the column "Drug" is 5
drugY    91
drugX    54
drugA    23
drugC    16
drugB    16
Name: Drug, dtype: int64
```

جدول ۱ نشان‌دهنده چند سطر از مجموعه داده دارو می‌باشد که مقادیر توصیفی ستون‌های Sex، BP و Cholesterol به شرح زیر می‌باشد:

```
Values of Sex column =  ['F' 'M']
Values of BP column =  ['HIGH' 'LOW' 'NORMAL']
Values of Cholesterol column ['HIGH' 'NORMAL']
```

	Drug	Na_to_K	Cholesterol	BP	Sex	Age
۰	drugY	۳۵۵.۲۵	HIGH	HIGH	F	۲۳
۱	drugC	۰۹۳.۱۳	HIGH	LOW	M	۴۷
۲	drugC	۱۱۴.۱۰	HIGH	LOW	M	۴۷
۳	drugX	۷۹۸.۷	HIGH	NORMAL	F	۲۸
۴	drugY	۰۴۳.۱۸	HIGH	LOW	F	۶۱

جدول ۱: Data Sample

۱.۲ بخش اول

سوال:

توضیح دهید که از چه روشی برای انتخاب بخشی از داده استفاده کرده‌اید. آیا روش بهتری برای این کار می‌شناسید؟

سوال:

به منظور پاسخ داده به این سوال ابتدا باید کدی که به وسیله آن دیتاست تقسیم شده است مورد بررسی قرار بگیرد. کد زیر به منظور تقسیم دیتاست موجود به دو زیرمجموعه آموزش و ارزیابی است که در نتیجه آن ۴۰٪ از این دیتاست با توزیع یکنواخت به مجموعه ارزیابی تخصیص داده می‌شود.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=53)

```

Code : ۱ Data Selection

نکته حائز اهمیت این است که این دیتاست از تفکیک پذیری بالایی برخوردار است و نیازی نیست که از روش‌های پیشرفته‌تر به منظور تقسیم آن استفاده کنیم اما با توجه به imbalance بودن این دیتاست می‌توان از روش stratify به منظور حفظ توزیع داده‌ها و یا روش K-Fold-cross-validation استفاده کرد تا بهترین نتیجه ممکن را بدست آورد و از عملکرد مدل مطمئن شد.

در ادامه با استفاده از کد زیر و Configuration پیش‌فرض کتابخانه sklearn یک درخت تصمیم ایجاد شده است و روی دیتاست آموزش، آموزش دیده است. نتایج بدست آمده از این آموزش در شکل ۲ نشان داده شده است.

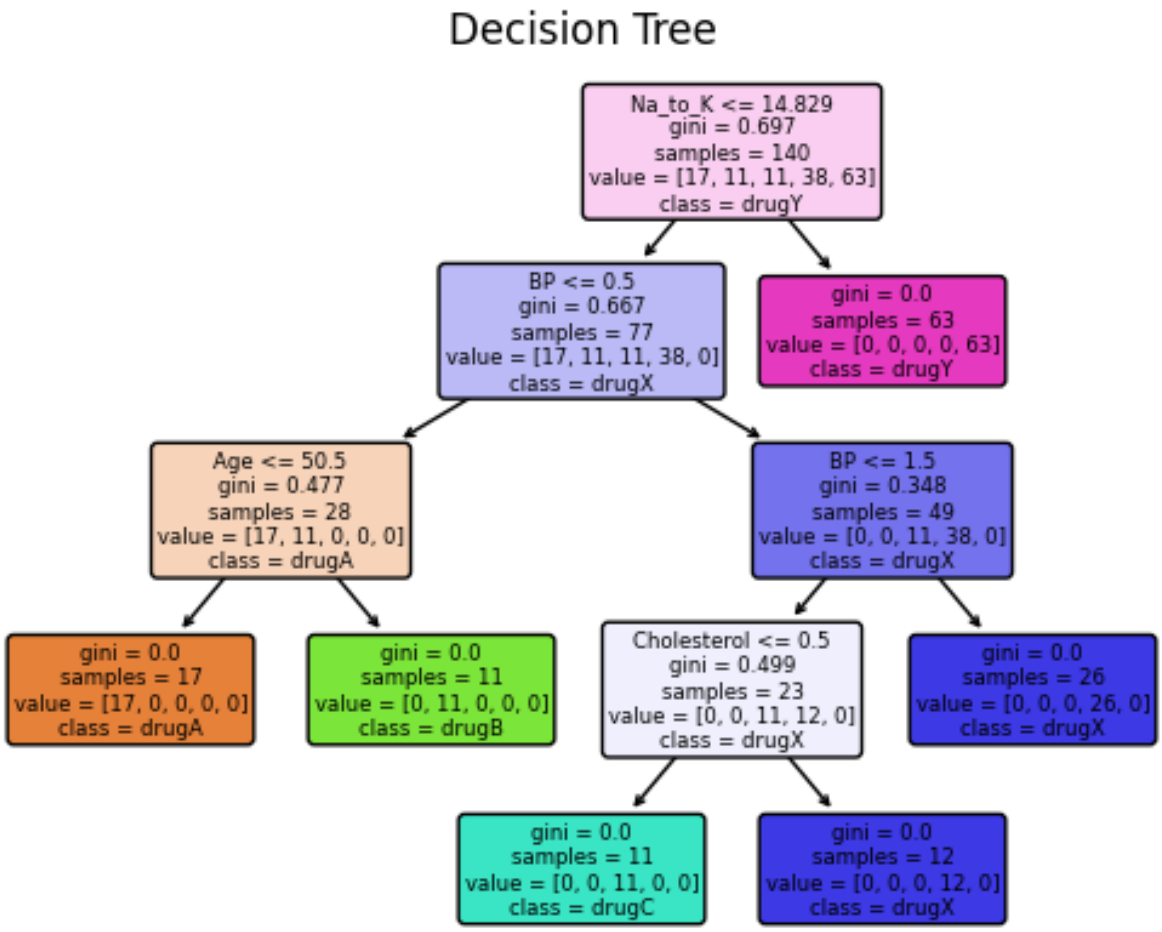
```
# Create a Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=53)

# Train the classifier on the training data
clf.fit(X_train, y_train)

plot_tree(clf, filled=True, feature_names=X.columns, class_names=clf.classes_, rounded=True)
plt.title('Decision Tree')
plt.show()
```

```
۱۰
```

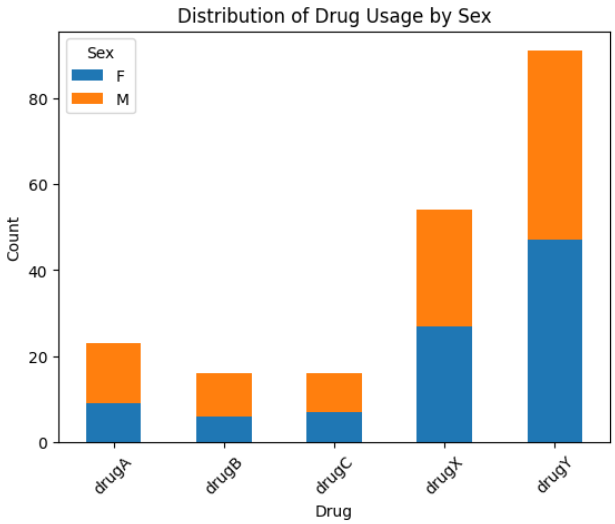
Code : ۲ Train Decision Tree



شکل ۲

۲.۲ بخش دوم

با توجه به شکل ۲ می‌توان مشخص کرد که از داده‌های جنسیت به منظور تعیین کلاس هر نمونه استفاده نشده است. به منظور تحلیل این مسئله لازم است تا پراکندگی جنسیت، نسبت به هر نوع دارو رسم شود. این نمودار در شکل ۳ رسم شده و است و همانطور که مشخص است میزان مردان نسبت به زنان همواره از entropy بالایی برخوردار بوده که به همین دلیل نیز جنسیت در نوع دارو تاثیر چندانی نگذاشته است، به بیان دیگر این بیماری با جنسیت وابستگی پایینی دارد.



شکل ۳: Drug over Sex

در ادامه متریک‌های موردنیاز به منظور بررسی نتایج مدل محاسبه شده است. همانطور که مشخص است برای دو زیرمجموعه آموزش و ارزیابی تمامی متریک‌های و ماتریس درهم‌ریختگی در شکل ۴آ مقداری برابر ۱ دارند که این به معنی عملکرد بی‌عیب مدل ما است.

```
training metrics:
Classification Report:
              precision    recall  f1-score   support

       drugA       00.1      00.1      00.1        17
       drugB       00.1      00.1      00.1        11
       drugC       00.1      00.1      00.1        11
       drugX       00.1      00.1      00.1        38
       drugY       00.1      00.1      00.1        63

    accuracy                           00.1       140
   macro avg       00.1      00.1      00.1       140
weighted avg       00.1      00.1      00.1       140

test metrics:

Classification Report:
              precision    recall  f1-score   support
```

drugA	00.1	00.1	00.1	6
drugB	00.1	00.1	00.1	5
drugC	00.1	00.1	00.1	5
drugX	00.1	00.1	00.1	16
drugY	00.1	00.1	00.1	28
accuracy			00.1	60
macro avg	00.1	00.1	00.1	60
weighted avg	00.1	00.1	00.1	60

Matrix Confusion (آ)

حال با استفاده از GridSearchCV هایپرپارامترهای متفاوت را با استفاده از متد ۵-fold-corss-validation بررسی می‌کنیم تا بهترین ساختار مدل را پیدا کنیم. در این جستجو مقادیر max_depth، min_samples_split و min_samples_leaf را در یک بازه متناسب با دیتاستی که در اختیار داریم تغییر می‌دهیم. کد زیر به منظور آموزش مدل‌های مختلف استفاده شده است و در انتها بهترین هایپرپارامترها را خروجی می‌دهد.

```
param_grid = {
    'max_depth': [2, 3, 4, 5, 10, 15 ],
    'min_samples_split':  [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 4, 7, 10]
}
scoring = {
    'precision': make_scorer(precision_score, average='macro'),
    'recall': make_scorer(recall_score, average='macro')
}
# Perform grid search with cross-validation
grid_search = GridSearchCV(estimator=clf, param_grid=param_grid,
                           scoring=scoring, refit='precision', return_train_score=True)
grid_search.fit(X_train, y_train)

```

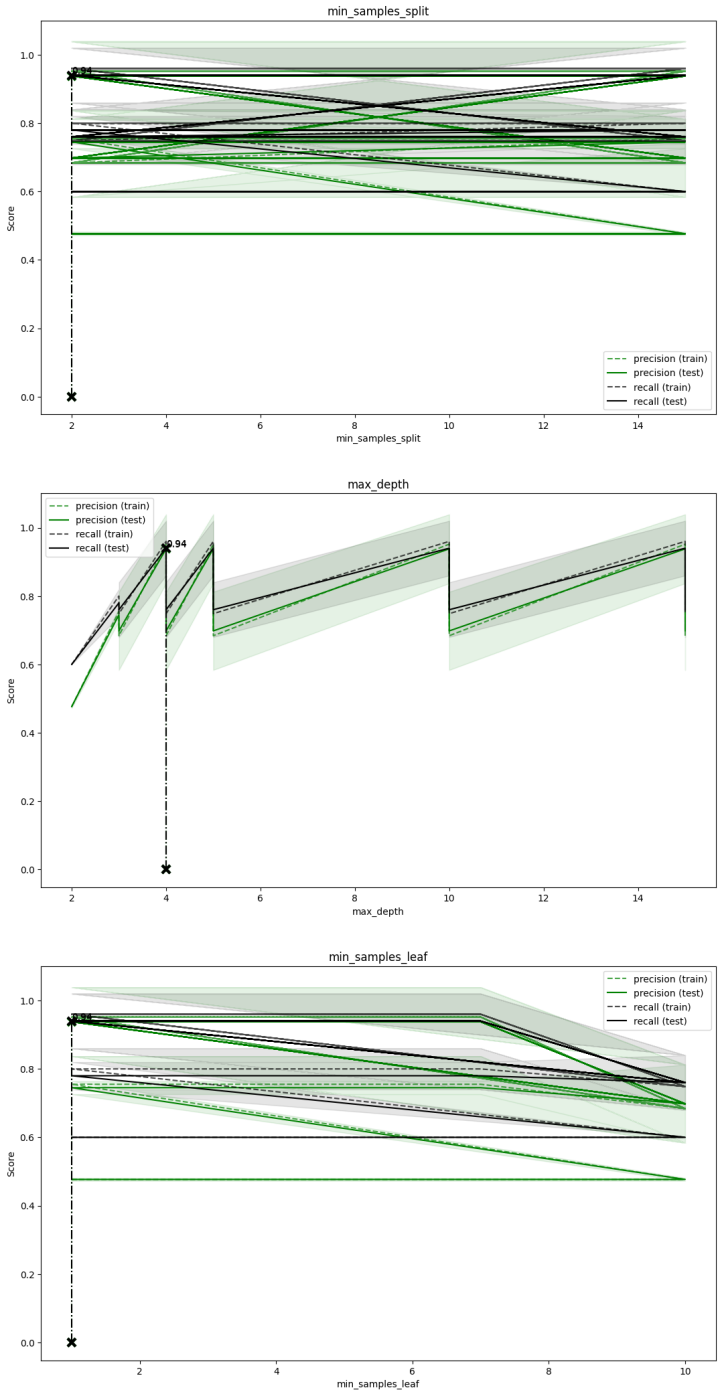
Code :۳ Grid search

نتایج بدست آمده از این گروه آموزش‌ها نشان می‌دهد که مدل بهینه هایپرپارامترهای زیر را دارد.

```
Best Parameters: {'max_depth': 4, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

نمودارهای موجود در شکل ۵ به ازای تغییرات هر هایپرپارامتر کشیده شده است. همانطور که مشخص است، متغییر min_samples_split با افزایش مقدار، گاهی موجب بهبود و گاهی موجب افت عملکرد مدل می‌شود که این بهبود یا افت به مقدار دو متغییر دیگر بستگی دارد بنابراین نمیتوان به صورت کلی نظر قطعی درمورد این متغییر داد. متغییر max_depth یک رفتار تکرار شونده را نشان می‌دهد که به غیر از مقادیر خاص،

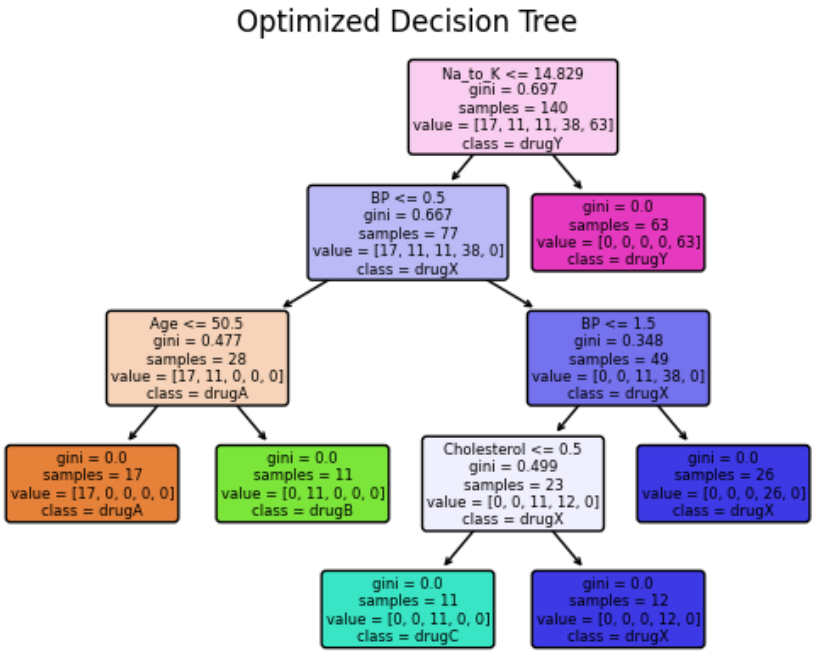
نمودار آن افزایشی می‌باشد بنابراین افزایش مقدار این متغییر می‌تواند به بهبود مدل کمک کند. در انتها متغییر `min_samples_leaf` و نمودار مربوط به آن نشان می‌دهد که با افزایش مقدار این متغییر عملکرد مدل در این مسئله به خصوص کاهش پیدا می‌کند بنابراین کمترین مقدار ممکن را برای این متغییر درنظر می‌گیریم.



شکل ۵: Graph Hyper-parameters

حال به سراغ بررسی نتایج بدست آمده از بهترین مدل می‌رویم، همانطور که در شکل ۶ مشخص است،

تفاوتی با درخت موجود در شکل ۲ نداشته است، از طرف دیگر با توجه به متریک‌ها و ماتریس درهم‌ریختگی در شکل ۷آ همچنان عملکرد مدل در بهترین حالت خود می‌باشد و تمام نمونه‌ها را به درستی تشخیص داده است



شکل ۶: Tree Best

```
training metrics:
Classification Report:
              precision    recall  f1-score   support

       drugA       00.1      00.1      00.1        17
       drugB       00.1      00.1      00.1        11
       drugC       00.1      00.1      00.1        11
       drugX       00.1      00.1      00.1        38
       drugY       00.1      00.1      00.1        63

    accuracy                           00.1       140
   macro avg       00.1      00.1      00.1       140
weighted avg       00.1      00.1      00.1       140

test metrics:
```

```
Classification Report:
              precision    recall  f1-score   support

       drugA       00.1      00.1      00.1         6
       drugB       00.1      00.1      00.1         5
       drugC       00.1      00.1      00.1         5
       drugX       00.1      00.1      00.1        16
       drugY       00.1      00.1      00.1        28

    accuracy                           00.1        60
   macro avg       00.1      00.1      00.1        60
weighted avg       00.1      00.1      00.1        60
```

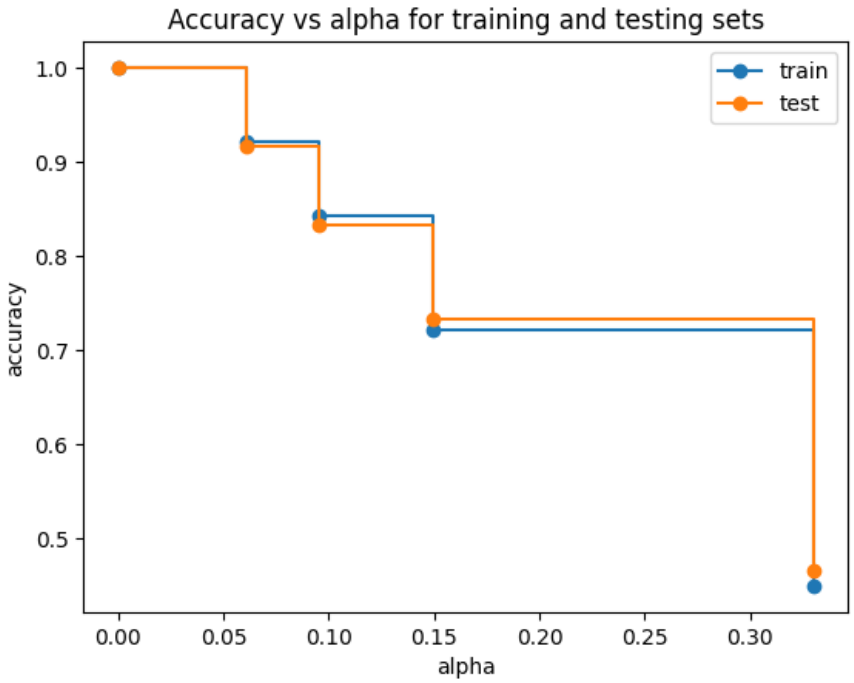
Matrix Confusion (آ)

در انتها به سراغ post-pruning می‌رویم و با استفاده از کد زیر چند α متفاوت را تست می‌کنیم که نتیجه آن در شکل ۸ قابل مشاهده است. در نتیجه post-pruning روی درخت به وجود آمده توست مجموعه داده دارو تاثیر منفی خواهد داشت.

```
clf = DecisionTreeClassifier(random_state=53)
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
clfs = []
for ccp_alpha in ccp_alphas:
    clf = DecisionTreeClassifier(random_state=53, ccp_alpha=ccp_alpha)
    clf.fit(X_train, y_train)
    clfs.append(clf)

```

Code :۴ best alpha



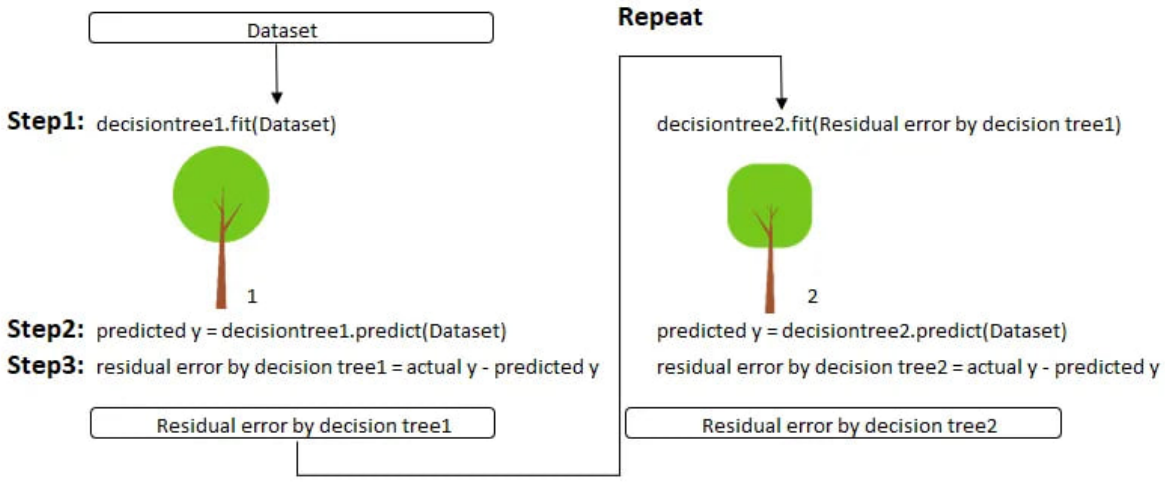
شکل ۸

۳.۲ بخش سوم

AdaBoost و RandomForest دو تکنیک یادگیری جمعی هستند که می‌توانند عملکرد درخت تصمیم را به طور قابل توجهی بهبود بخشند. این تکنیک‌ها به شرح زیر عمل می‌کنند.

AdaBoost: AdaBoost یک تکنیک تقویت است که بر بهبود دقت پیش‌بینی‌ها با ترکیب چندین درخت تصمیم تکیه دارد. ایده کلیدی پشت AdaBoost تنظیم وزن داده‌های آموزشی بر اساس عملکرد درخت قبلی است.

طی فرایند آموزش یک مدل AdaBoost ابتدا، وزن تمام نمونه‌ها برابر یکدیگر قرار داده می‌شود. سپس یک درخت تصمیم‌گیری با استفاده از این داده‌هاآموزش داده می‌شود. در ادامه، وزن نمونه‌هایی که نادرست طبقه‌بندی‌شده، افزایش می‌یابد تا درخت بعدی بر روی آن‌ها تمرکز کند. این فرایند برای تعداد مشخصی از تکرارها یا تا رسیدن به دقت مطلوب تکرار می‌شود. مدل نهایی یک جمع وزن‌دار از مجموعه درخت‌ها است، که در آن درخت‌های دقیق‌تر وزن بیشتری دارند. فرایند آموزش یک مدل AdaBoost در شکل ۹ آورده شده است.



شکل ۹: نحوه عملکرد و آموزش مدل AdaBoost [۲]

RandomForest: RandomForest یک است که با یک مجموعه از درختان تصمیم، یک جنگل تصمیم می‌سازد که هر کدام از درختان این جنگل بر روی یک زیرمجموعه تصادفی از داده‌ها و ویژگی‌ها آموزش داده می‌شوند.

در این روش، ابتدا زیرمجموعه‌های تصادفی از داده‌های آموزشی انتخاب می‌شوند. سپس یک درخت تصمیم‌گیری بر روی هر زیرمجموعه آموزش داده می‌شود. هر درخت به طور مستقل یک پیش‌بینی انجام می‌دهد و پیش‌بینی نهایی با رأی اکثریت بین تمامی درختان انجام می‌شود. این روش مؤثر است زیرا درختان را با آموزش آن‌ها بر روی نمونه‌ها و زیرمجموعه‌های ویژگی‌های مختلف، از هم جدا می‌کند. تجمیع پیش‌بینی‌ها از چندین درخت به طور کلی به عملکرد بهتری نسبت به هر درخت منفرد منجر می‌شود.

هر دو روش AdaBoost و RandomForest عملکرد درخت تصمیم‌گیری را بهبود می‌بخشند. در حالی که AdaBoost بر بهبود متوالی و تنظیم وزن‌ها تمرکز دارد، RandomForest مجموعه‌ای متنوع از درختان را می‌سازد و نتایج آن‌ها را میانگین می‌گیرد. این دو تکنیک می‌توانند مدلی بهتر ایجاد کنند. [۱]

حال با استفاده از کد زیر یک مدل RandomForest را آموزش می‌دهیم.

```
# Define the parameter grid for the Random Forest
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_features': ['auto', 'sqrt'],
    'max_depth': [4, 6, 8, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Initialize the Random Forest classifier
rf_clf = RandomForestClassifier(random_state=53)

```

```
# Initialize GridSearchCV
grid_search_rf = GridSearchCV(estimator=rf_clf, param_grid=param_grid_rf, cv=5, n_jobs=-1, )

# Fit GridSearchCV
grid_search_rf.fit(X_train, y_train)
```

Code :۵ Random Forest Training

نتایج بدست آمده از مدل فوق به شرح زیر می‌باشد که تمام متریک‌ها برابر ۱ شده‌اند و از روی ماتریس درهم‌ریختگی شکل ۱۰آ مشخص است که تمام نمونه‌ها به درستی تشخیص داده‌شده‌اند.

```
training metrics:
Classification Report:
              precision    recall  f1-score   support

       drugA       00.1      00.1      00.1        17
       drugB       00.1      00.1      00.1        11
       drugC       00.1      00.1      00.1        11
       drugX       00.1      00.1      00.1        38
       drugY       00.1      00.1      00.1        63

    accuracy                           00.1       140
   macro avg       00.1      00.1      00.1       140
weighted avg       00.1      00.1      00.1       140

test metrics:

Classification Report:
              precision    recall  f1-score   support

       drugA       00.1      00.1      00.1         6
       drugB       00.1      00.1      00.1         5
       drugC       00.1      00.1      00.1         5
       drugX       00.1      00.1      00.1        16
       drugY       00.1      00.1      00.1        28

    accuracy                           00.1        60
   macro avg       00.1      00.1      00.1        60
weighted avg       00.1      00.1      00.1        60
```

Matrix Confusion (آ)

۳ سوال ۴

روش Bayse با تکیه بر توضیح احتمالاتی مسئله و با فرض اینکه رگرسورها از یکدیگر مستقل هستند، از قانون Bayse استفاده می‌کند تا یک مدل طبقه‌بندی را ایجاد کند. با توجه به اینکه الگوریتم به توزیع احتمالاتی وابسته است نرمال کردن دیتاست تاثیری در نتیجه نخواهد داشت بنابراین پیش‌پردازش خاصی لازم نیست. همانطور که در ؟؟ مشخص است این دیتاست ۱۳ ویژگی را اندازه‌گیری کرده است تا به وسیله آنها پیشبینی انجام شود.

Target	Thal	CA	Slope	Oldpeak	Exang	Thalach	Restecg	FBS	Chol	Trestbps	CP
۰	۳	۲	۲	۰.۱	۰	۱۶۸	۱	۰	۲۱۲	۱۲۵	۰
۰	۳	۰	۰	۱.۳	۱	۱۵۵	۰	۱	۲۰۳	۱۴۰	۰
۰	۳	۰	۰	۶.۲	۱	۱۲۵	۱	۰	۱۷۴	۱۴۵	۰
۰	۳	۱	۲	۰.۰	۰	۱۶۱	۱	۰	۲۰۳	۱۴۸	۰
۰	۲	۳	۱	۹.۱	۰	۱۰۶	۱	۱	۲۹۴	۱۳۸	۰

جدول ۲: Data Sample

در ادامه برای آموزش مدل مبتنی بر Bayse مجموعه داده موجود با استفاده از کد زیر به دو قسمت آموزش و ارزیابی تقسیم می‌شود و پس از آن مدل به روی دیتاست مذکور تنظیم می‌شود.

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=53)

# Initialize and train the Gaussian Naive Bayes classifier
model = GaussianNB()X_train, y_train)

# Train model
model.fit(X_train, y_train)
```

Code :۶ best alpha

نتایج بدست آمده از مدل فوق روی مجموعه داده ارزیابی به شرح زیر می‌باشد. با توجه به متریک Recall می‌توانیم به این نتیجه برسیم که مدل در پیدا کردن نمونه‌های کلاس ۰، نسبت به نمونه‌های کلاس ۱ ضعیف‌تر عمل کرده است و با اینکه تعداد نمونه‌ها در هر دو کلاس یکسان بوده اما دقت مدل در پیدا کردن کلاس ۰، ۱۰% ضعیف‌تر بوده اما از سوی دیگر اگر مدل درمورد یک نمونه از کلاس ۰ پیشبینی کرده، احتمال اینکه این پیشبینی درست باشد بیشتر است و این مسئله از متریک Precision قابل نتیجه می‌باشد. به صورت کلی دقت مدل در پیدا کردن و میزان درست بودن پیشبینی‌هایش از رابطه‌ای بین Precision و Recall نتیجه می‌شود که به آن Score f-۱ می‌گویند. Score f-۱ برای هردو کلاس بالای ۸۰% است و اختلاف کمی دارد که نشان

می‌دهد مدل علاوه‌براینکه از دقت مناسبی برخوردار است، به کلاس خاصی متمایل نشده است. این نتایج در ماتریس درهم‌ریختگی موجود در شکل ۱۱آ قابل ملاحضه است.

```
Classification Report:
              precision    recall  f1-score   support

     Class 0       86.0      77.0      81.0       102
     Class 1       80.0      87.0      83.0       103

    accuracy                           82.0       205
   macro avg       83.0      82.0      82.0       205
weighted avg       83.0      82.0      82.0       205
```

Matrix Confusion (آ)

در کتابخانه Scikit-learn، اصطلاحات micro و macro به روش‌های مختلف میانگین‌گیری معیارها مانند Precision Recall، و Score f-۱ در مسائل طبقه‌بندی چاشاره دارند. روش micro متریک‌ها را به صورت کلی در شرایطی هر پیش‌بینی ارزش یکسانی دارد محاسبه می‌کند، به عبارت دیگر این روش تمام true positives، false negatives و false positives را برای محاسبه متریک موردنظر حساب می‌کند که می‌تواند تحت تأثیر imbalance بودن کلاس‌ها قرار گیرد اما عملکرد کلی جامعی را ارائه می‌دهد. در مقابل، میانگین‌گیری macro معیارها را برای هر کلاس به طور مستقل محاسبه و سپس میانگین می‌گیرد، و به هر کلاس بدون توجه به فراوانی آن وزن مساوی می‌دهد، که آن را نسبت به imbalance کلاس‌ها کمتر حساس می‌کند. انتخاب بین میانگین‌گیری micro و macro بستگی به این دارد که آیا می‌خواهید بر عملکرد کلی تأکید کنید یا عملکرد متعادل در تمام کلاس‌ها مدنظر می‌باشد.

در انتها به ارزیابی مدل روی ۵ نمونه از داده‌ها می‌پردازیم. با استفاده از کد زیر ۵ نمونه به صورت تصادفی از مجموعه داده استخراج می‌شود و سپس مدل روی این داده پیشبینی انجام می‌دهد و در گام آخر پیشبینی به همراه کلاس واقعی نمایش داده می‌شود که این نتایج در جدول ۳ قابل مشاهده می‌باشد. این مدل تنها در یک مورد از ۵ مورد اشتباه کرده است که کلاس را ۱ تشخیص داده اما این درحالی است که کلاس واقعی ۰ بوده است.

```
test_data = data.sample(n=5, random_state=53)
pred = model.predict(test_data.drop("target", axis='columns'))
test_data['predictions'] = pred
test_data.head()
```

Predictions	Target	Thal	CA	Slope	Oldpeak	Exang	Thalach	Restecg	FBS	Chol	T
۱	۱	۲	۱	۲	۰.۰	۰	۱۵۹	۰	۰	۳۰۳	
۱	۰	۳	۱	۱	۴.۰	۰	۱۵۰	۰	۰	۲۲۹	
۱	۱	۳	۰	۱	۴.۰	۰	۱۴۷	۰	۰	۲۵۸	
۰	۰	۳	۱	۲	۰.۰	۰	۱۹۵	۰	۰	۲۸۳	
۰	۰	۲	۱	۱	۲.۱	۱	۱۴۴	۰	۱	۲۴۹	

جدول ۳: Data Test

مراجع

[۱] .GeeksforGeeks adaboost. and forest random between Differences GeeksforGeeks، .۲۰۲۳ Accessed: .۲۰-۰۵-۲۰۲۴

[۲] gradi- adaboost، forest، Random learning: ensemble Basic Science. Data Towards Accessed: .۲۰۲۰ ،*Science Data Towards* explained. step by step — boosting ent .۲۰-۰۵-۲۰۲۴

[۳] Ac- n.d. Functions، Activation Cheatsheet: Learning Machine Yuan. Avinash cessed: May ۱۸، ۲۰۲۴.