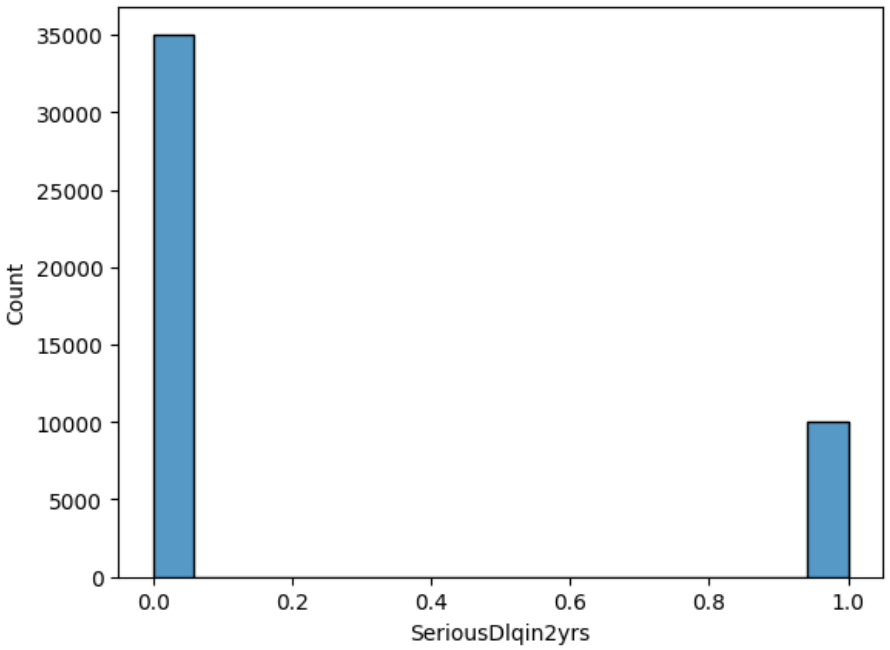
به نام خدا

۳.

پیش‌پردازش در این بخش با پیش‌پردازش بر روی داده‌ها هیستوگرام مربوط به ویژگی هدف را رسم می‌کنیم:



شکل ۱: نمودار هیستوگرام ویژگی هدف

همانطور که مشاهده می‌کنیم مقادیر صفر در داده‌های ما بیشتر وجود دارند.
پس از این داده‌هایی که مقدار نامعتبر NaN دارند به کمک کتابخانه sk-learn با میانه مربوط به همان ستون جایگزین می‌کنیم.

۱. به کمک قضیه حد مرکزی با انتخاب تعدادی نمونه از مشتریان بدحساب و محاسبه میانگین و و واریانس آنها بازه اطمینان ۹۰ ٪ را برای تمام مشتریان بدست می‌آوریم.

$$confidence_interval : \bar{X} \pm Z(90\%)\frac{\sigma}{\sqrt{n}}$$

با شبیه‌سازی این بازه به شکل زیر خواهد بود:

$$[45.87959, 46.01078]$$

۲. در این بخش به کمک طبقه‌بند GridSearchCV بهترین پارامتر برای درخت تصمیم را با اجرای همه حالت‌های آن محاسبه می‌کنیم.
اینکار همراه با روش stratfield 5-fold صورت می‌گیرد. به این معنی که در هر بار آموزش کلاس k دسته از داده‌ها کنار گذاشته می‌شود و مدل با k-1 داده آموزش داده می‌شود. اینکار را برای همه دسته‌ها و k بار تکرار می‌کنیم.
نتیجه بهترین پارامترها:

```
[{'max_depth': 15, 'max_features': 1, 'min_samples_leaf': 3},
 {'max_depth': 15, 'max_features': 2, 'min_samples_leaf': 3},
 {'max_depth': 15, 'max_features': 4, 'min_samples_leaf': 3},
 {'max_depth': 15, 'max_features': 4, 'min_samples_leaf': 3},
 {'max_depth': 15, 'max_features': 2, 'min_samples_leaf': 3}]
```

شکل ۲: بهترین پارامترها برای مدل random forest

حال به کمک پارامترهای بهینه یک random forest را آموزش می‌دهیم و معیار ROC AUC را برای آن بدست می‌آوریم.

```
ROC AUC 0 : 0.833518159509901
ROC AUC 1 : 0.8310143234949158
ROC AUC 2 : 0.8238320574170168
ROC AUC 3 : 0.8339836508343812
ROC AUC 4 : 0.8239447406885954
```

شکل ۳: معیار ROC AUC برای random forest

۳. معیار max features کمترین تاثیرگذاری را دارد. معیار max depth و min sample leaf از overfit شدن مدل به داده‌ها جلوگیری می‌کنند. همچنین تعداد درخت‌ها معیار مهمی در جنگل می‌باشد.

٤. مشابه بخش ۲ به دنبال پارامترهای بهینه برای آموزش مدل random forest هستیم. در این بخش تمامی حالات بررسی نمی‌شوند و به شکل تصادفی برخی حالات بررسی می‌شوند. اینکار باعث کاهش حجم محاسبات می‌شود.

در روش Bagging از تعدادی طبقه‌بند پایه استفاده می‌شود و به روش bootstrapping داده‌ها بین آنها پخش می‌شود تا مدل‌ها آموزش داده شوند.

پارامترهای بهینه بصورت زیر بدست می‌آیند:

```
[{'max_samples': 0.7, 'max_features': 4, 'estimator__C': 1},
 {'max_samples': 0.7, 'max_features': 4, 'estimator__C': 10},
 {'max_samples': 0.5, 'max_features': 4, 'estimator__C': 1},
 {'max_samples': 0.5, 'max_features': 4, 'estimator__C': 1},
 {'max_samples': 0.5, 'max_features': 4, 'estimator__C': 1}]
```

شکل ٤: پارامترهای بهینه برای مدل bagging

به کمک پارامترهای بهینه یک مدل bagging را آموزش می‌دهیم و مشابه بخش ۲ معیار ROC AUC را برای آن بدست می‌آوریم.

```
ROC AUC 0 : 0.7512569176374133
ROC AUC 1 : 0.7472883146016238
ROC AUC 2 : 0.7697435380457742
ROC AUC 3 : 0.772792864420937
ROC AUC 4 : 0.7450728822299895
```

شکل ۵: معیار ROC AUC برای bagging

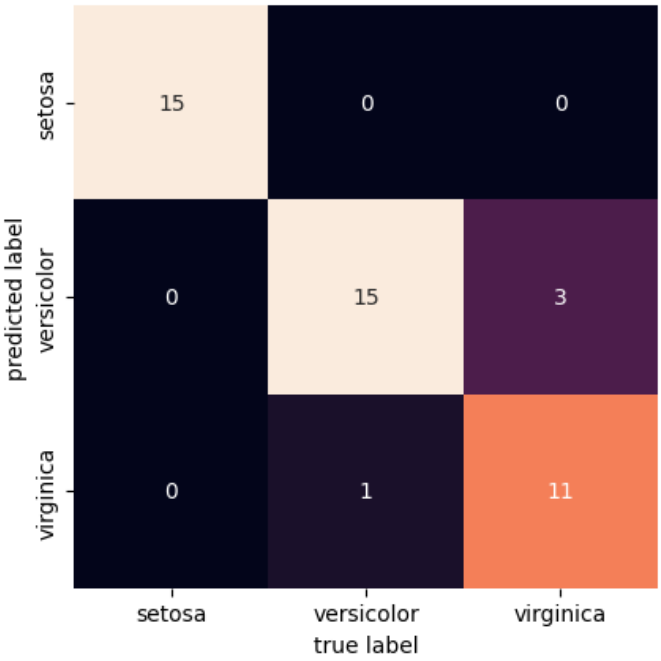
۵. پارامترهایی که در بخش قبل مقداردهی شدند هر کدام به نحوی در عملکرد موثر هستند. مثلا پارامتر C مربوط به logistic regression است که ضریب regularization است. هر قدر این مقدار بیشتر باشد رگولاریزیشن سخت گیرانه تر خواهد بود و میزان overfitting کمتر می‌شود.

معیار دیگر max features است که بر روی مدل bagging تاثیرگذار است. این معیار به طبقه‌بندها اجازه می‌دهد به مقداری از ویژگی‌ها انتخاب کرده و آموزش ببینند. محدود کردن تعداد ویژگی‌ها باعث ساده‌تر شدن مدل می‌شود. برای افزایش دقت مدل‌ها این معیار افزایش می‌یابد.

معیار مهم دیگر max samples است. این معیار مشخص می‌کند هر طبقه‌بند چه سهمی از کل داده‌ها را برای آموزش استفاده کند. اگر طبقه‌بندها از داده‌های بیشتری برای آموزش استفاده کنند دقت بالاتری خواهند داشت.

٤. در این بخش AdaBoost Classification را پیاده‌سازی خواهیم کرد. روش AdaBoost بر مبنای مجموع چند classifier است. بگونه‌ای که ابتدا یک طبقه‌بند را انتخاب کرده و داده‌های آموزشی توسط آن طبقه‌بندی می‌شود. داده‌ها در ابتدا وزن یکسانی دارند. پس از طبقه‌بندی وزن داده‌ها به‌روز می‌شود. بگونه‌ای که داده‌هایی که به اشتباه طبقه‌بندی شدند وزن بیشتر و داده‌هایی که به درستی طبقه‌بندی شده‌اند وزن کمتری خواهند داشت. در ادامه داده‌ها روی مجموع طبقه‌بند فعلی و یک طبقه‌بند دیگر train شده و وزن داده‌ها دوباره بر همان اساس به‌روز خواهد شد. هر طبقه‌بند با یک ضریبی در طبقه‌بند نهایی شرکت می‌کند که آن ضریب توسط وزن داده‌هایی که طبقه‌بندی کرده تعیین می‌شود. (روابط ریاضی مربوط در سوال اول این تمرین نوشته شده)

روابط مربوطه را در کد داده شده پیاده کرده و نتیجه به شکل ماتریس آشفتگی به شکل زیر خواهد بود:



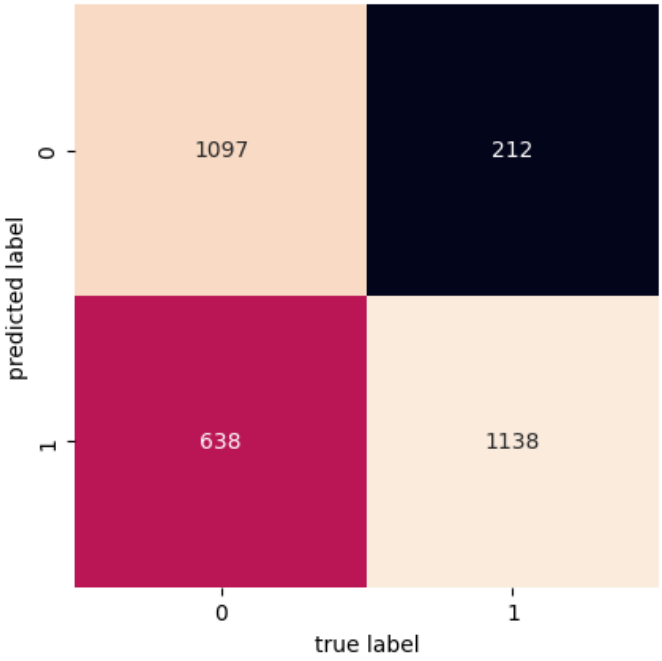
شکل ٦: ماتریس آشفتگی AdaBoost Classifier

با توجه به ماتریس دقت مدل در حدود ۹۱ ٪ است که دقت خوبی محسوب می‌شود.

۶. در این بخش یک درخت تصمیم و الگوریتم ID3 را پیاده‌سازی می‌کنیم.
الگوریتم ساخت درخت هم یک الگوریتم بازگشتی است. به این معنی یک ویژگی به عنوان ریشه (خروجی الگوریتم ID3) در نظر گرفته می‌شود سپس فرزند این ریشه به عنوان ریشه یک درخت در نظر گرفته می‌شود و الگوریتم ID3 با کنار گذاشتن ویژگی اول اعمال می‌شود و همین روند ادامه می‌یابد تا به برگ‌ها برسیم.

در زمان تست داده‌ها از ریشه شروع می‌کنند و با توجه به اینکه با کدام ویژگی درخت همخوانی بیشتری دارند در درخت پایین می‌روند تا به برگ‌ها برسند و نهایتا دارای برچسب شوند.

برای پیاده‌سازی درخت از یک کلاس استفاده می‌کنیم. این کلاس متدهای مختلفی برای بخش train و test دارد.
با آموزش درخت با ۲۰ ٪ داده‌های آموزشی و تست با ۸۰ ٪ داده‌ها ماتریس آشفتگی بصورت زیر بدست می‌آید:



شکل ۷: ماتریس آشفتگی درخت تصمیم

دقت مدل تقریبا ۷۲ ٪ است.