به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس داده کاوی پیشرفته

تمرین اول

نام و نام خانوادگی	عرفان شهابی
شماره دانشجویی	۸۱۰۱۰۳۱۶۶
تاریخ ارسال گزارش	۱۴۰۴.۰۳.۰۱

فهرست

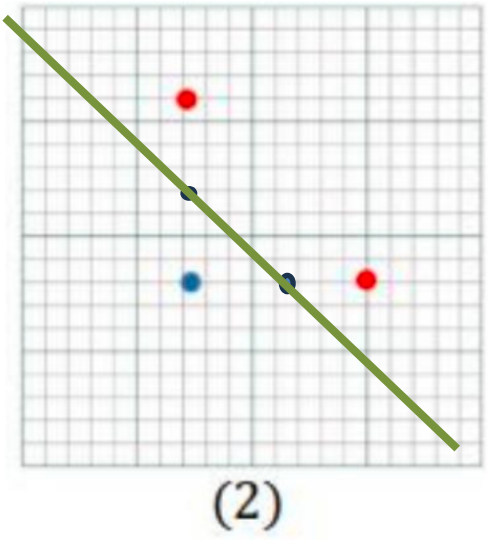
شکل‌ها

جدول‌ها

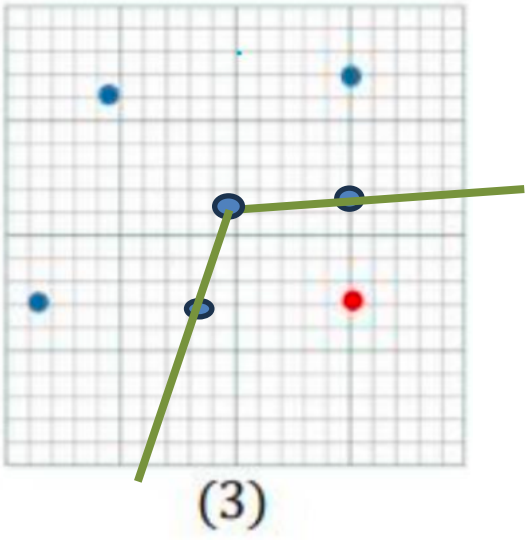
نمودارها

سوال ۱

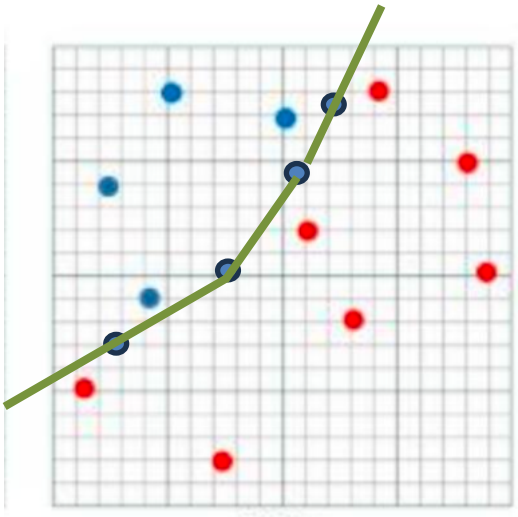
الف:



(2)



(3)



(4)

ب:

خیر، اگر چه در نگاه اول ذخیره مرز تصمیم به جای کل داده های آموزشی ممکن است حافظه کمتری اشغال کند، اما در الگوریتم NN-1 این مرز ها معمولا بسیار پیچیده و غیر خطی هستند.

توصیف و ذخیره سازی این مرز ها (مثل نمودارهای ورونوی در فضای n بعدی) میتواند حتی هزینه بیشتری نسبت به نگهداری داده ها داشته باشد.

علاوه بر این، چون NN-1 یک الگوریتم Lazy است، حذف داده های آموزشی باعث از دست رفتن عملکرد اصلی آن می شود. بنابراین در حالت کلی، ذخیره مرز تصمیم همیشه منجر به بهبود حافظه نخواهد شد.

ج:

خیر، الگوریتم KNN این مشکل را ندارد. درخت تصمیم پس از آموزش ساختاری ثابت پیدا می کند و اگر داده جدیدی برسد، باید کل درخت یا بخش از آن مجدد آموزش داده شود. اما الگوریتم KNN یک الگوریتم lazy است، یعنی هیچ مدل یا ساختاری از پیش یاد نمیگیرد و تنها داده ها را نگه میدارد. پس اگر داده حدیدی برسد، تنها کافی است آن را به مجموعه داده ها اضافه کنیم و نیازی به بازآموزی یا تغییر در ساختار مدل نیست.

در نتیجه KNN نسبت به ورود داده های جدید مقاوم تر و انعطاف پذیر تر است.

سوال ۲

الف:

هرس کردن درخت تصمیم ضروری است چرا که از overfitting جلوگیری میکند. درخت تصمیم در صورت عدم کنترل، ممکن است بسیار عمیق و پیچیده شود و به جای یادگیری الگو، نویز داده ها را یاد بگیرد. هرس کمک میکند درخت ساده تر شود و تعمیم پذیری بهتر شود و عملکرد روی داده های جدید بهبود پیدا کند.

مقایسه پیش هرس و پس هرس:

پیش هرس معمولا سریع تر و ساده تر است چرا که درخت کامل ساخته نمی شود. همچنین در این روش هنگام ساخت درخت، تقسیم ها را زودتر متوقف می کنیم. البته این کار ممکن است درخت را خیلی زود متوقف کند و دقت کاهش یابد و کنترل ساده ولی گاهی ناکارآمد می شود.

در صورتی که پس هرس معمولا کند تر و پیچیده تر است چرا که ابتدا درخت کامل ساخته می شود و تحلیل بعد از ساخت کامل درخت انجام می شود. این روش ابتدا درخت کامل ساخته می شود، سپس گره هایی که عملکرد مفیدی ندارند حذف میشوند و احتمال بهینه تر بودن ساختار نهایی بیشتر است و امکان کنترل دقیق تر روی تعمیم دهی وجود دارد.

ب:

برای گسترش درخت تصمیم T با استفاده از داده های جدید D’ و ساخت درخت جدید T’، می توان داده های درون D’ را از ریشه درخت T عبور داد و در هر گره آن ها را طبق شرط های درخت هدایت کرد. سپس در گره هایی که داده های جدید متمرکز می شوند و کیفیت تقسیم پایین است، می توان گره های جدید اضافه کرد یا دوباره تقسیم انجام شود. اما برای انجام این کار به اطلاعات آماری درون هر گره (مانند تعداد نمونه ها و توزیع کلاس ها) نیاز داریم تا تصمیم گیری برای ادامه تقسیم امکان پذیر باشد.

در نتیجه برای گسترش درخت T بدون ساخت مجدد از ابتدا، نیاز به حفظ داده های عبوری یا خلاصه های آماری هر گره داریم.

سوال ۳

شماره رکورد	سردرد	سرفه	تب	سرماخوردگی؟
۱	دارد	دارد	دارد	آری
۲	دارد	دارد	دارد	خیر
۳	دارد	دارد	دارد	آری
۴	دارد	دارد	ندارد	خیر
۵	ندارد	دارد	ندارد	آری
۶	ندارد	ندارد	ندارد	خیر
۷	ندارد	ندارد	ندارد	آری
۸	دارد	ندارد	ندارد	خیر
۹	ندارد	ندارد	دارد	خیر
۱۰	ندارد	دارد	دارد	آری

جدول 1

P(Yes | X) = P((yes) سر درد|Yes)×P(سرفه(No)|Yes)×P(تب(yes)|Yes)×P(سرماخوردگی(yes))

P(No | X) = P((yes) سر درد|No)×P(سرفه(No)|No)×P(تب(yes)|No)×P(سرماخوردگی(no))

P(Yes | X):

P((yes) سر درد|Yes) = 2/5

P(سرفه(No)|Yes) = 1/5

P(تب(yes)|Yes) = 3/5

P(سرماخوردگی(yes)) = 5/10

⇨ P(Yes | X) = 0.024

P(No | X):

P((yes) سر درد|No) = 3/5

P(سرفه(No)|No) = 3/5

P(تب(yes)|No) = 2/5

P(سرماخوردگی(no)) = 5/10

⇨ P(No | X) = 0.072

احتمال سرما خورده نبودن فرد با این علائم بیشتر است در نتیجه فرد سرما خوردگی ندارد.

سوال ۴

الف:

در این سناریو با یک عدم تعادل شدید مواجه هستیم به این صورت که:

- ٪۹۹ از تراکنش ها سالم هستند.
- تنها ۱٪ از تراکنش ها مشکوک به تقلب هستند.

در چنین حالتی، اگر مدل ما همیشه تراکنش سالم را پیش بینی کند، ۹۹٪ مواقع درست خواهد بود بدون این که مدل واقعا چیزی را یاد گرفته باشد.

بنابراین این داده ها به تنهایی برای آموزش مناسب نیستند، چون مدل احتمالا یاد میگیرد که همیشه کلاس غالب (یعنی کلاس سالم) را پیش بینی کند.

برای حل این مشکل میتوان از راهکار هایی مانند oversampling برای کلاس اقلیت، undersampling برای کلاس اکثریت، وزن گذاری کلاس ها و یا استفاده از معیارهای ارزیابی مناسب تر مانند F1-Score بجای Accuracy استفاده کرد.

ب:

در داده های imbalance و دارای عدم تعادل بالا، معیار Accuracy گمراه کننده است. مثلا اگر ٪۹۹ داده ها سالم باشند، مدلی که همیشه کلاس سالم را خروجی می دهد، دقتی نزدیک به ٪۹۹ خواهد داشت در صورتی که هیچ تراکنش مشکوکی را شناسایی نمی کند.

به همین دلیل میتوان از معیارهای زیر بجای Accuracy استفاده کرد:

Precision: از بین تراکنش های شناسایی شده به عنوان مشکوک، چند درصد واقعا مشکوک بوده اند؟

Recall (Sensitivity): از بین تمام تراکنش های مشکوک واقعی، چند درصد شناسایی شده اند؟

F1-Score: precision و recall . میانگین هارمونیک بین

ج:

در این کاربرد، هدف تشخیص دقیق چهره کاربر مجاز است و جلوگیری از باز شدن قفل توسط افراد غیر مجاز.

بنابراین دو حالت حیاتی داریم:

1. False Positive: چهره اشتباهی (مثلا یک دوست یا عکس) به اشتباه به عنوان صاحب گوشی تشخیص داده شود => در این صورت امنیت نقض می شود.
2. False Negative: خود کاربر اصلی به اشتباه شناسایی نشود => فقط کمی ناراحت کننده است ولی امنیت نقض نمی شود.

به همین دلیل بهترین معیار برای این کار استفاده از Precision است چرا که اولویت ما جلوگیری از پذیرش نادرست (FP) است.

د:

در کاربردهای پزشکی، به ویژه تشخیص سرطان دو موضوع زیر اهمیت زیادی دارند:

1 - موارد واقعا بیمار (True Positive) را از دست ندهیم (FN خطرناک است).
2 - حتی اگر برخی افراد سالم اشتباها بیمار تشخیص داده شوند (False Positive باشند)، معمولا مشکلی نیست چرا که تست های بعدی میتوانند این موضوع را مشخص کنند.

به همین دلیل در این بخش مناسب ترین معیار Recall (Sensitivity) است چرا که این معیار مشخص می کند از بین تمام بیماران واقعی، مدل چند درصد را به درستی شناسایی کرده است. اگر مدل Recall پایینی داشته باشد، بیماران واقعی را از دست می دهیم و این موضوع می تواند مرگبار باشد.

سوال ۵

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

جدول 2

الف:

در این قسمت ابتدا توزیع کلاس Evade را بررسی میکنیم:

Evade = Yes => 5, 8, 10

Evade = No => 1, 2, 3, 4, 6, 7, 9

P(Evade = Yes) = 0.3

P(Evade = No) = 0.7

$$H(x) = -\sum_i p(x_i)\log(x_i)$$

$$H(Evoid) = -0.3log0.3 - 0.7log0.7 = 0.88$$

ب:

در این قسمت بعد از محاسبه آنتروپی کل باید Info را به ازای هر ویژگی محاسبه کنیم و سپس gain ویژگی ها را نیز محاسبه کنیم و از بین آن ها بیشترین را انتخاب کنیم:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$Gain(A) = Info(D) - Info_A(D)$$

For Refund:

P(Refund(Yes)) = 0.3

P(Refund(No)) = 0.7

Refund(Yes), Evade(Yes): 0

Refund(Yes), Evade(No): 3

Refund(No), Evade(Yes): 3

Refund(No), Evade(No): 4

$$Info_{Refund} = \frac{3}{10} \times I(0,3) + \frac{7}{10} \times I(3,4)$$

$$Info_{Refund} = -\frac{3}{10} \times [0] + \frac{7}{10} \times \left[-\frac{3}{7}\log\left(\frac{3}{7}\right) - \frac{4}{7}\log\left(\frac{4}{7}\right)\right] = 0.69$$

$$Gain_{Refund} = 0.88 - 0.69 = 0.19$$

For Marital:

P(Marital(Single)) = 0.4

P(Marital(Married)) = 0.4

P(Marital(Divorced)) = 0.2

Marital(Single), Evade(Yes): 2

Marital(Single), Evade(No): 2

Marital(Married), Evade(Yes): 0

Marital(Married), Evade(No): 4

Marital(Divorced), Evade(Yes): 1

Marital(Divorced), Evade(No): 1

$$Info_{Married} = \frac{4}{10} \times I(2,2) + \frac{4}{10} \times I(0,4) + \frac{2}{10} \times I(1,1)$$

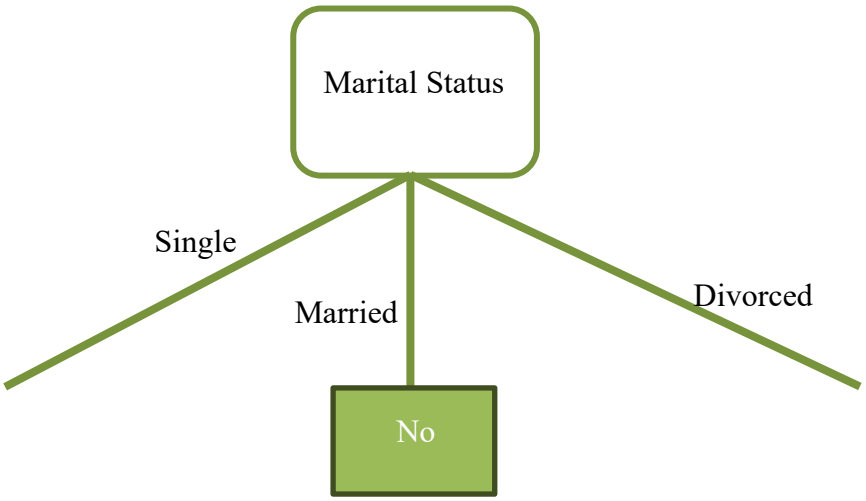
$$Info_{Refund} = \frac{4}{10} \times \left[-\frac{2}{4}\log\left(\frac{2}{4}\right) - \frac{2}{4}\log\left(\frac{2}{4}\right)\right] + \frac{4}{10} \times \left[-\frac{0}{4}\log\left(\frac{0}{4}\right) + \frac{4}{4}\log\left(\frac{4}{4}\right)\right] + \frac{2}{10} \times [-\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right) = 0.4 + 0 + 0.2 = 0.6$$

$$Gain_{Refund} = 0.88 - 0.6 = 0.28$$

که بر همین اساس و طبق محاسبات انجام شده، بیشترین Gain برای ستون Marital Status است که این ویژگی برای آزمون در درخت تصمیم استفاده می شود.

ج:

همانطور که در بخش قبل مشاهده شد ابتدا از ویژگی Marital Status برای آزمون درخت تصمیم استفاده می کنیم.



حالا برای ادامه ساخت کامل درخت مراحل قبلی را برای Single و Divorced تکرار می کنیم.

برای Single:

$$H(Evoid) = -\frac{3}{6}\log\left(\frac{3}{6}\right) - \frac{3}{6}\log\left(\frac{3}{6}\right) = 1$$

For Refund (Single):

P(Refund(Yes)) = 2/4

P(Refund(No)) = 2/4

Refund(Yes), Evade(Yes): 0

Refund(Yes), Evade(No): 1

Refund(No), Evade(Yes): 2

Refund(No), Evade(No): 1

$$Info_{Refund} = \frac{2}{4} \times I(0,1) + \frac{2}{4} \times I(2,1)$$

$$Info_{Refund} = -\frac{2}{4} \times [0] + \frac{2}{4} \times \left[-\frac{2}{3}\log\left(\frac{2}{3}\right) - \frac{1}{3}\log\left(\frac{1}{3}\right)\right] = 0.46$$

$$Gain_{Refund} = 1 - 0.46 = 0.54$$

برای Taxable Incom هم باید یک Treshhold در نظر بگیریم و براساس آن مقدار Gain را محاسبه کنیم.

با توجه به مقادیر مختلف من میزان Treshhold را 100K در نظر گرفتم. بر همین اساس:

For Tax (Single):

P(Tax (>100K)) = 1/4

P(Tax (<100K)) = 3/4

Tax (>100K), Evade(Yes): 0

Tax (>100K), Evade(No): 1

Tax (<100K), Evade(Yes): 2

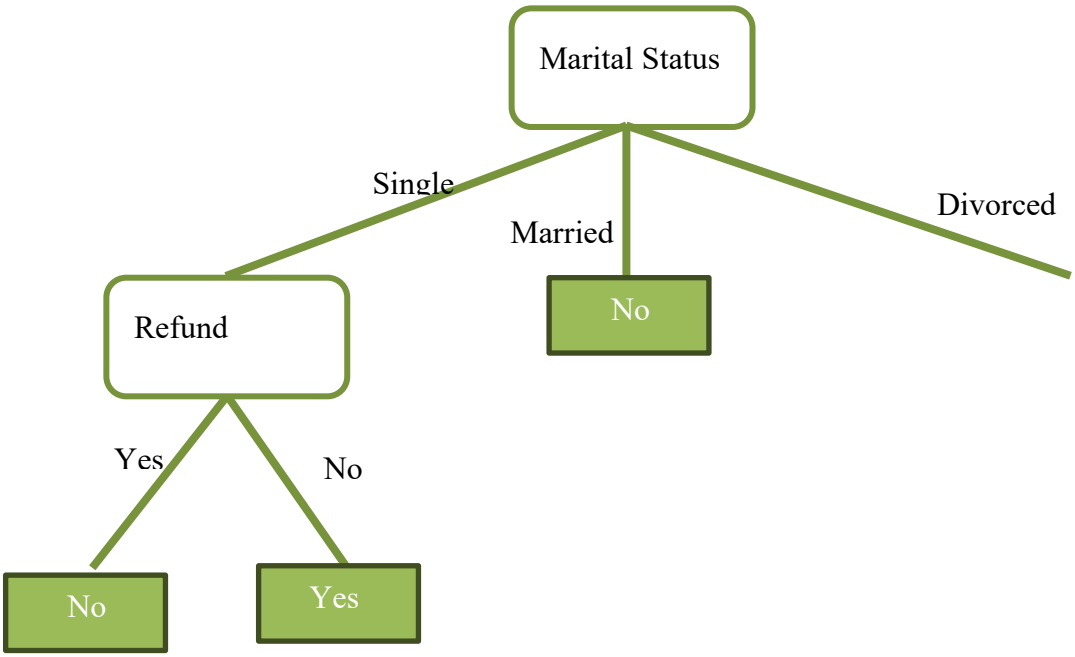
Tax (<100K), Evade(No): 1

$$Info_{Tax} = \frac{1}{4} \times I(0,1) + \frac{3}{4} \times I(2,1)$$

$$Info_{Tax} = -\frac{1}{4} \times [0] + \frac{3}{4} \times \left[-\frac{2}{3}\log\left(\frac{2}{3}\right) - \frac{1}{3}\log\left(\frac{1}{3}\right)\right] = 0.69$$

$$Gain_{Tax} = 1 - 0.69 = 0.31$$

همانطور که مشخص است، Gain(Tax) < Gain(Refund) است که نشان می دهد باید شکست بعد از Single را با Refund انجام بدهیم پس ادامه درخت به شکل زیر است:



برای Divorced:

$$H(Evoid) = -\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right) = 1$$

For Refund (Divorced):

P(Refund(Yes)) = 2/4

P(Refund(No)) = 2/4

Refund(Yes), Evade(Yes): 0

Refund(Yes), Evade(No): 1

Refund(No), Evade(Yes): 1

Refund(No), Evade(No): 0

$$Info_{Refund} = \frac{1}{2} \times I(0,1) + \frac{1}{2} \times I(0,1)$$

$$Info_{Refund} = -\frac{1}{2} \times [0] - \frac{1}{2} \times [0] = 0$$

$$Gain_{Refund} = 1 - 0 = 1$$

For Tax (Single):

P(Tax (>100K)) = 1/2

P(Tax (<100K)) = 1/2

Tax (>100K), Evade(Yes): 0

Tax (>100K), Evade(No): 1

Tax (<100K), Evade(Yes): 1

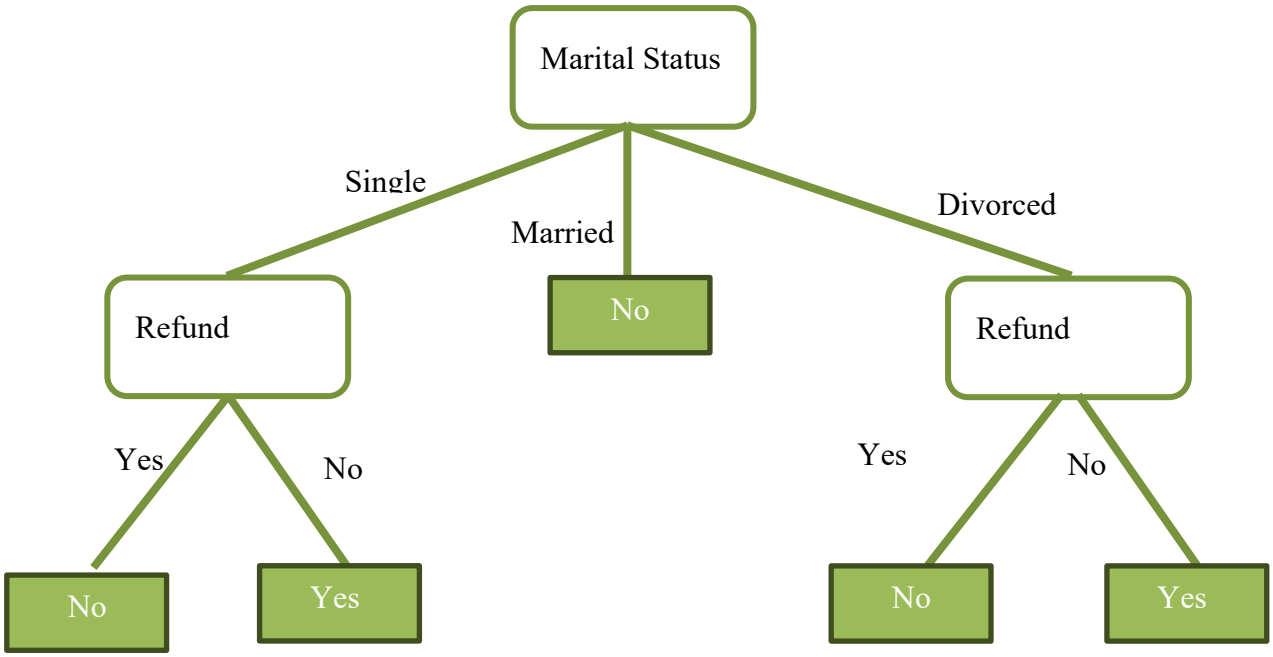
Tax (<100K), Evade(No): 0

$$Info_{Tax} = \frac{1}{2} \times I(0,1) + \frac{1}{2} \times I(0,1)$$

$$Info_{Tax} = -\frac{1}{2} \times [0] - \frac{1}{2} \times [0] = 0$$

$$Gain_{Tax} = 1 - 0 = 1$$

که به همین دلیل برای Divorced تفاوت نمیکند که شکست را با کدام ویژگی انجام دهیم به همین دلیل درخت تصمیم برای این سوال به شرح زیر است:

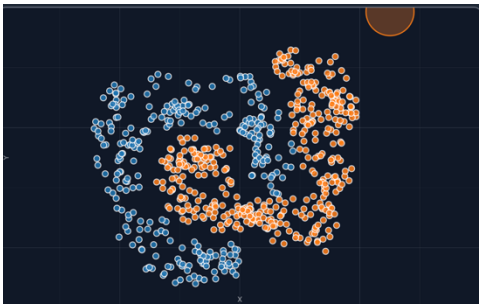


سوال عملی

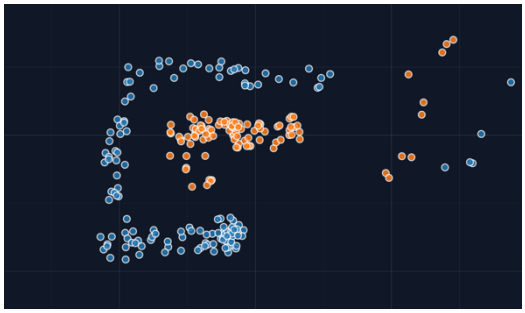
1 . مقایسه و تحلیل دسته بندی های مختلف

الف)

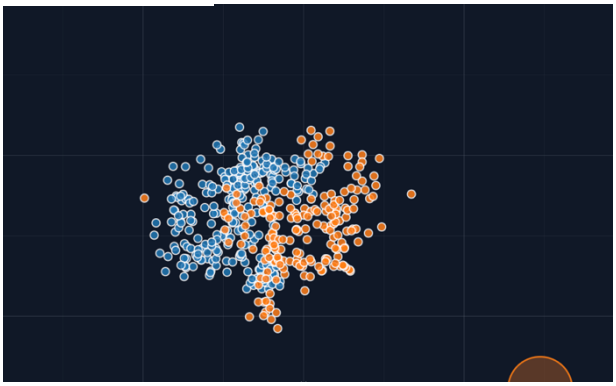
برای این قسمت از سوال من سه الگوی زیر را تولید کردم:



شکل 1



شکل 2



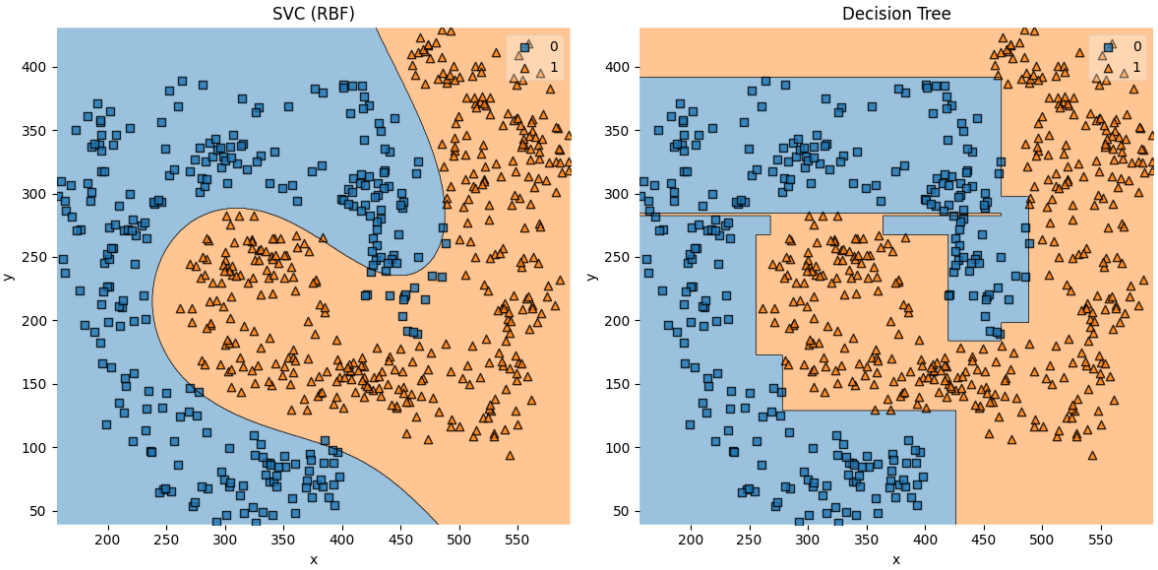
شکل 3

برای نمونه اول، سعی کردم نمونه ای را ترسیم کنم که ایجاد تقسیم کننده خطی بین داده ها مشکل باشد. برای نمونه دوم هم سعی کردم outlayer ها را در نظر بگیرم برای نمونه سوم سعی کردم داده ها را جوری ترسیم کنم که کلاس بندی نتواند به سادگی انجام بگیرد و مشخص نباشد.

ب)

برای این قسمت مقایسه ها به شرح زیر است:

برای مجموعه داده ۱:



شکل 4

در این قسمت ابتدا دو classifier درخت تصمیم و SVC (RBF) نشان داده شده است.

در SVC (RBF):

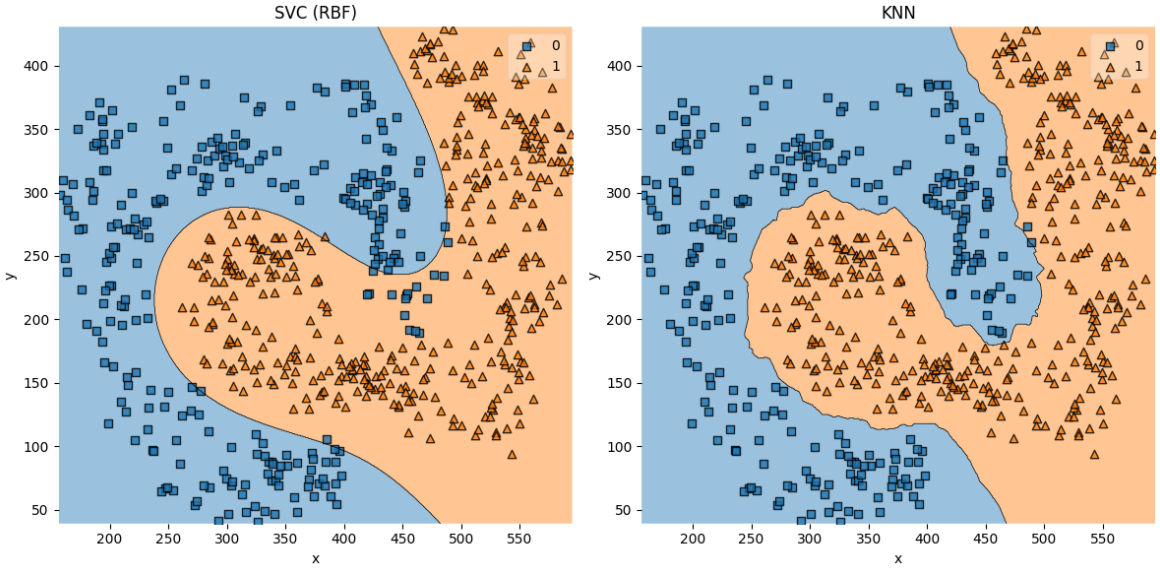
- مرز تصمیم پیوسته، نرم و منحنی دار است.
- شکل مرز نشان می دهد که SVC توانسته ساختار پیچیده و غیر خطی بین داده های دو کلاس را یاد بگیرد.
- این مدل تعمیم پذیری خوبی دارد و از Overfitting دوری کرده است. (خطوط تصمیم زیاد خرد نشده اند)
- الگوریتم به خوبی یک ناحیه بسته برای کلاس نارنجی در مرکز ایجاد کرده است که درخت تصمیم نتوانسته این کار را به خوبی انجام دهد.

در درخت تصمیم:

- مرز تصمیم به صورت پله ای و از خطوط افقی و عمودی تشکیل شده است.
- ساختار کاملا با ذات درخت تصمیم همخوانی دارد چرا که درخت تصمیم داده ها را با مقایسه روی ویژگی های تکی (مصلا x و یا y به تنهایی) تقسیم می کند.

- مشکل اینجاست که مرزها ساده و زاویه دار هستند و ممکن است به خوبی مرز پیچیده بین کلاس ها را مدل نکند.
- ساختار زیاد شاخه ای باعث شده که احتمال Overfitting زیاد باشد یعنی مدل به نویز یا جزئیات آموزش وابسته شده باشد.

در مقایسه دوم من SVC (RBF) را با KNN مقایسه کردم:

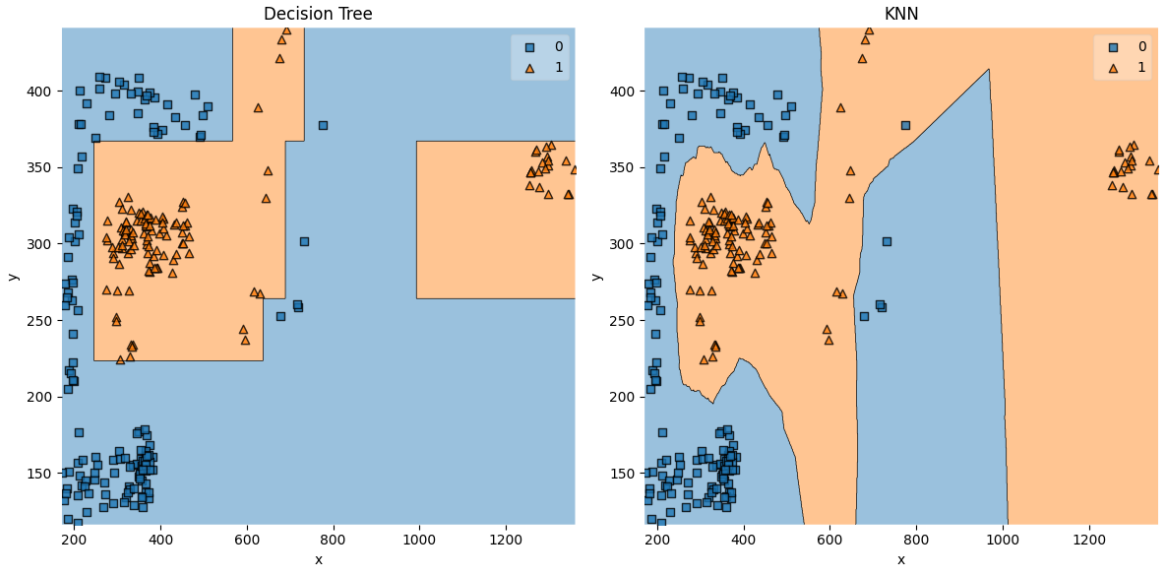


شکل 5

در قسمت قبل به توضیحات درباره SVC پرداختم و در این مقایسه برای KNN داریم:

- در این روش مرز تصمیم با توجه به نزدیک ترین همسایه شکل گرفته یعنی کلاس هر نقطه بر اساس k نمونه نزدیک آن تغییین می شود.
- مرز تصمیم بسیار انعطاف پذیر و محلی است و به شکل داده ها بسیار وابسته است.
- برای خلاف SVC این مرز به شدت به تراکم داده ها و انتخاب k حساس است.
- نقاطی که بین دو کلاس قرار دارند ممکن است باعث شکل گیری نواحی عجیب بشوند مثل ناحیه نارنجی داخل آبی.

برای مجموعه داده ۲:



شکل 6

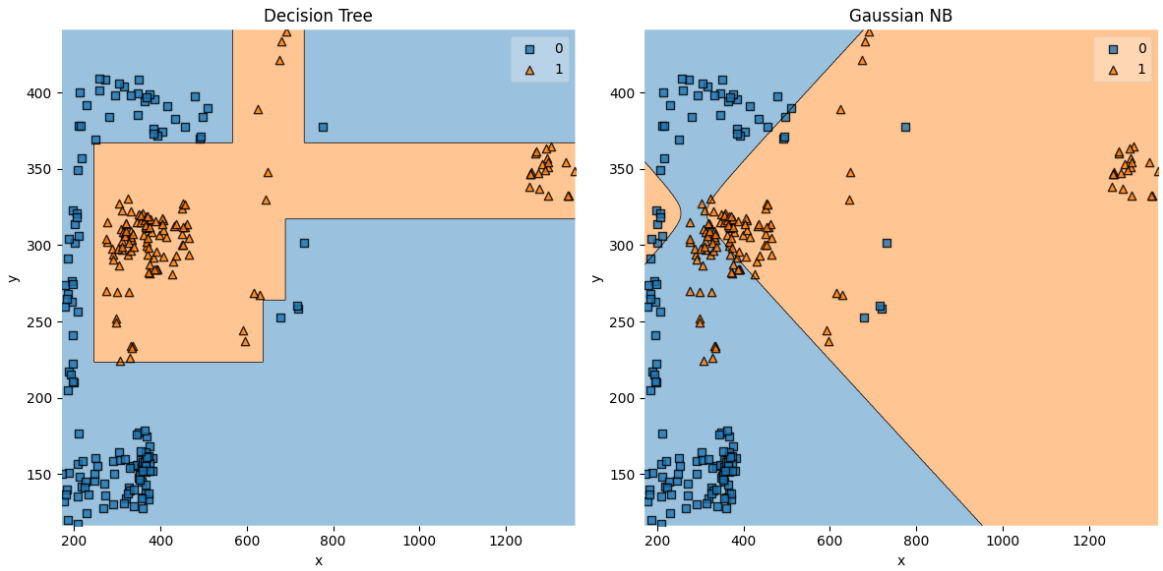
در این قسمت من دو مدل Gussian Naïve Bayes و درخت تصمیم را بررسی کردم.

برای مدل درخت تصمیم:

- مرز های تصمیم کاملا پله ای و مستطیلی هستند.
- نواحی تصمیم به صورت بلوک های مستطیلی بزرگ نمایش داده شده اند، که نشان میدهد مدل تا حد زیادی generalize شده است و یا درخت کم عمق است.
- در برخی نواحی مانند اطراف مرکز داده ها مرز های تصمیم خیلی دقیق نیست و به نظر می رسد underfitting اتفاق افتاده است.
- همچنین این مدل به داده های پرت بسیار حساس بوده و همانطور که مشخص است این داده ها را در نظر گرفته اما نه به اندازه KNN.

برای مدل Knn:

- مرز های تصمیم به صورت دندانه دار، ریز و دقیق است چرا که این مدل هر نقطه را با توجه به همسایگانش طبقه بندی می کند.
- مدل تلاش کرده تمام جزئیات محلی را یاد بگیرند حتی اگر نویز باشند که به معنی این است که احتمال overfitting بالاست.
- مدل به شدت به داده های پرت حساس است.



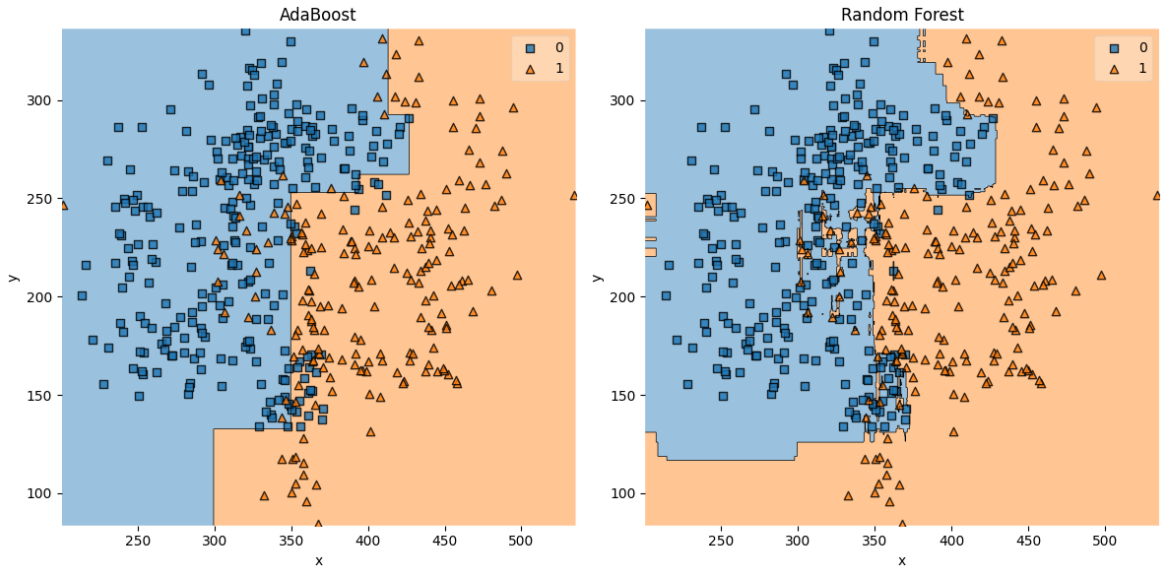
شکل 7

نمودار درخت تصمیم:

- همانطور که گفته شد، در این مدل مرز تصمیم به صورت پله ای است.
- ناحیه نارنجی حول تراکم اصلی نمونه های این کلاس ساخته شده اما به نظر می رسد که بسیار محدود و سفت است.
- مرز ها به شدت ساده و محدود هستند که احتمال underfitting را افزایش می دهد که معنای این است درخت به اندازه کافی عمیق نیست و یا pruning شدیدی انجام شده.

مدل Gaussian Naïve Bayes:

- در این شکل مرز تصمیم کاملا خطی و مورب است به دلیل اینکه این مدل فرض میکند که ویژگی ها از توزیع نرمال مستقل در هر کلاس پیروی می کنند.
- به جای تلاش برای دنبال کردن داده ها، این مدل بر اساس احتمال شرطی کلاس ها و میانگین و واریناس هر ویژگی مرز تصمیم ایجاد می کند.
- مرز مورب در اینجا ناشی از اختلاف در میانگین های کلاس ها در هر دو ویژگی است.
- این مدل به نویز و داده های پرت حساس است.



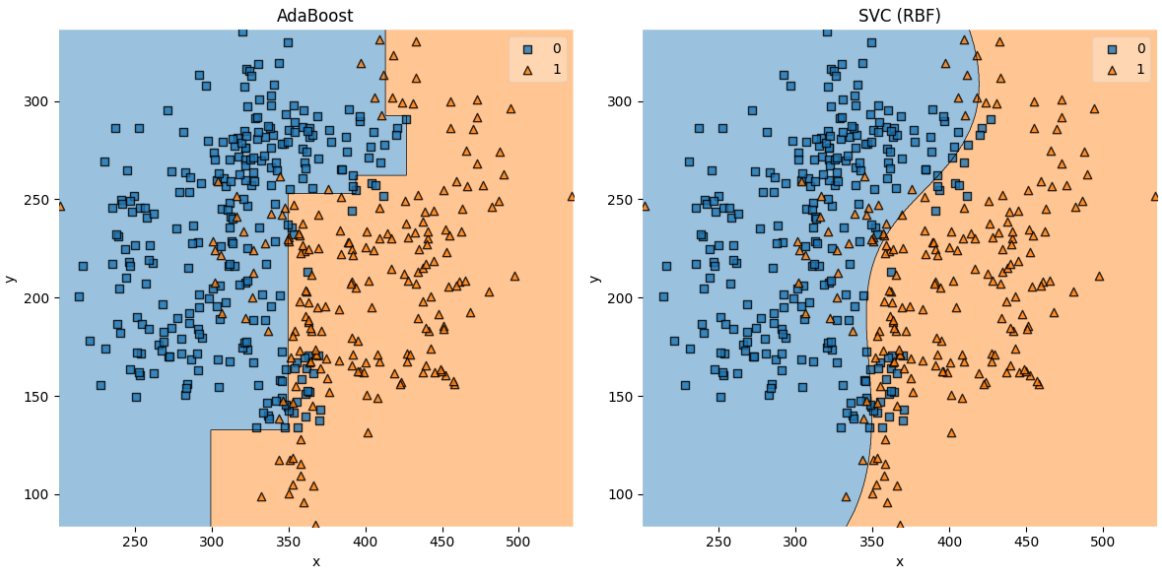
شکل 8

مدل AdaBoost:

- مرز تصمیم نسبتا صاف و گسترده است.
- AdaBoost با ترکیب مجموعه ای از مدل های ضعیف مثل درخت های کم عمق، سعی می کند به تدریج روی نمونه هاییی که اشتباه طبقه بندی شده اند تمرکز بیشتری داشته باشد.
- در این مدل مرز تصمیم تمایل دارد ساده و با تعمیم بالا باقی بماند.
- برخی جزئیات کوچک را از دست داده که نشان دهنده تمایل به کاهش واریانس است (یعنی از overfitting دوری می کند).

مدل Random Forest:

- این مدل از تعداد زیادی درخت تصمیم به صورت تصادفی و مستقل استفاده میکند.
- مرز تصمیم دارای جزئیات بسیار بیشتری است.
- برخی مناطق ریز و پیچیده در نواحی مرزی دیده می شود که نشان می دهد مدل سعی کرده تمام ساختار ها را پوشش دهد.
- به نظر می رسد در ناحیه مرزی مدل دچار overfitting خفیف شده است چرا که شکل مرز ها دندانه دار و ناپیوسته است.



شکل 9

برای مدل AdaBoost:

- در این مدل همانطور که گفته شد مرزها به صورت پله ای هستند چرا که اغلب در این مدل از درخت های تصمیم ساده به عنوان weak learner استفاده می شود.
- مرزها در برخی نقاط ناهموار و زاویه دار هستند و بعضی جزئیات در لبه ها را از دست داده اند.
- در نواحی مرکزی و پرتراکم، تصمیم نسبتا خوبی گرفته شده ولی در مرز ها ممکن است generalization ضعیف تر. باشد.
- مدل حساس به نقاط پرت است.

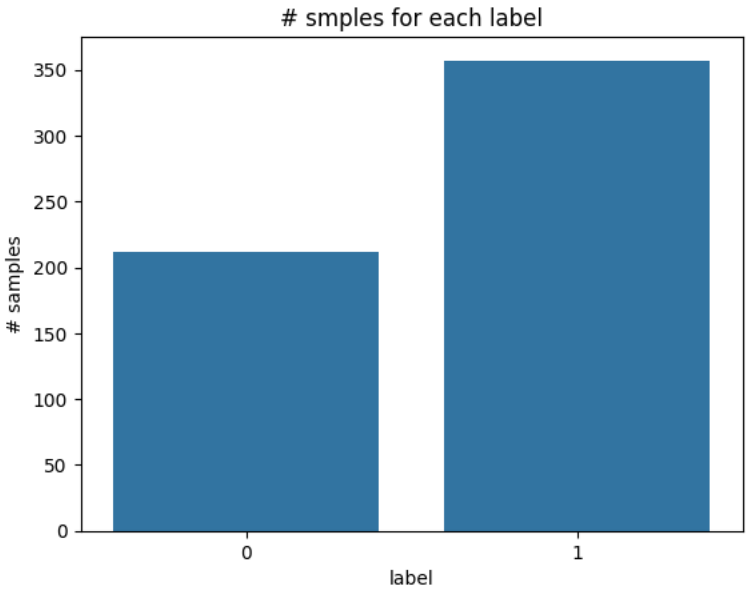
برای مدل SVC (RBF):

- مرزها کاملا نرم، منحنی و پیوسته هستند که ویژگی های کلیدی کرنل RBF است.
- مرز دقیقا از نواحی با بیشترین تغییر کلاس عبور می کند و سعی میکند داده ها را با بیشترین margin جدا کند.
- این مدل نسبت به AdaBoost دقیق تر و با قابلیت تفکیک بهتر بین کلاس ها در مرزهای پیچیده عمل می کند.
- در نواحی پر تراکم وسط نمودار، مرز طبیعی تری نسبت به AdaBoost رسم شده که نشان دهنده تناسب بهتر با توزیع داده است.

2 -آشنایی و تحلیل با مفاهیم Variance، Bias و نمودار های Learning Curve و Roc

ب) در این قسمت بعد از بارگزاری داده ها، تعداد نمونه ها برای هر یک از برچسب ها به دست آورده شده و نمودار متانسب به آن رسم شده که به شرح زیر است:

```
label
1    357
0    212
Name: count, dtype: int64
```



نمودار 1

ج) همانطور که از خروجی ها مشخص است داده ها کمی نامتوازن هستند و داده های مربوط به برچسب ۱ تعداد بیشتری دارند. برای متعادل سازی داده ها من از روش SMOTE استفاده کردم. این تکنیک برای Oversampling کلاس اقلیت طراحی شده و به جای تکرار ساده نمونه های کلاس اقلیت، نمونه های جدید مصنوعی بر اساس ترکیب خطی همسایگان نزدیک ایجاد می کند.

د) برای این قسمت من از متد df.isnull().sum استفاده کردم که خروجی این متد در پایین آورده شده است و همانطور که از خروجی مشخص است در این مجموعه داده، ما مقادیر گمشده نداریم.

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
```

```
concavity error             0
concave points error        0
symmetry error              0
fractal dimension error     0
worst radius                0
worst texture               0
worst perimeter             0
worst area                  0
worst smoothness            0
worst compactness           0
worst concavity             0
worst concave points        0
worst symmetry              0
worst fractal dimension     0
label                       0
dtype: int64
Does we have null values? False
```

ه) در این قسمت من از StandardScaler برای نرمالسازی داده ها استفاده کردم. این روش مقادیر هر ویژگی را طوری تغییر می دهد که در آن:

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

نتیجه این است که هر ویژگی دارای میانگین صفر و واریانس یک خواهد بود.

- اگر نرمالسازی انجام نشود چه اتفاقی می افتد؟

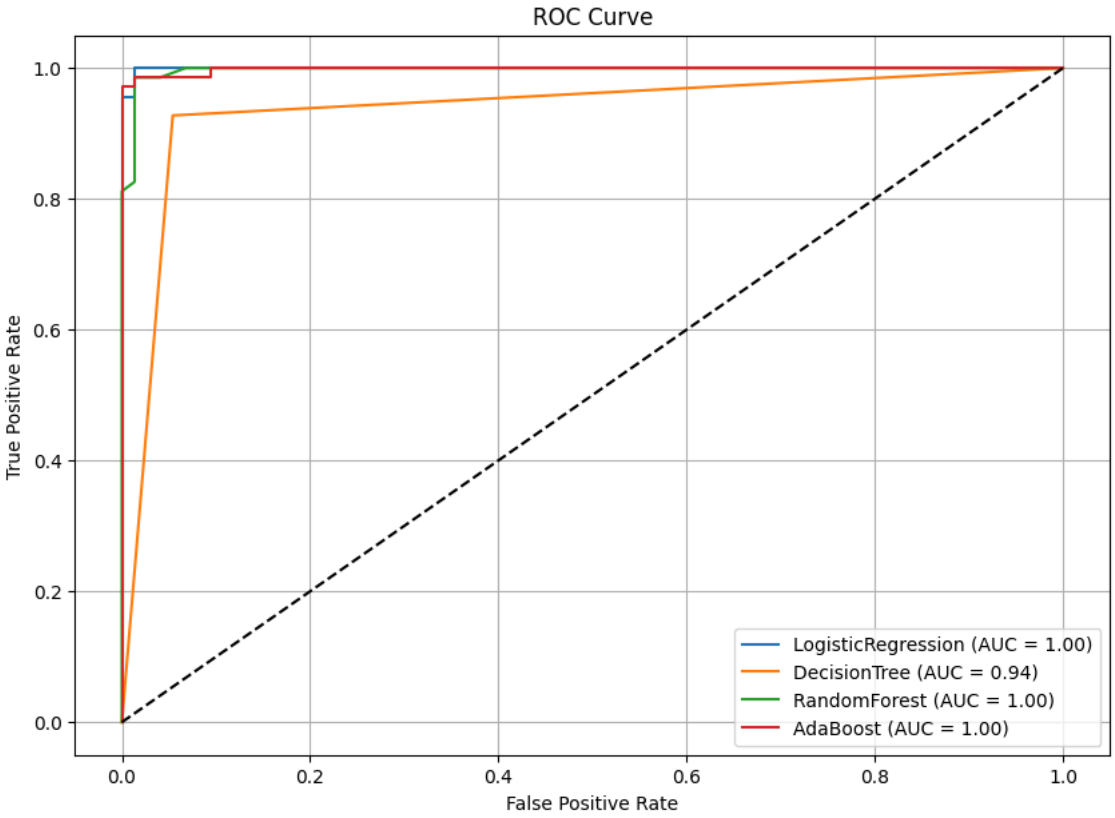
در مدل هایی که به مقیاس ویژگی ها حساس هستند، ویژگی هایی که مقادیر عددی بزرگتری دارند (مثلا قد به میلیمتر در مقابل درآمد به هزار تومان)، تاثیر بیشتری در تصمیم گیری مدل خواهند داست.

این باعث می شود که مدل منحرف شود و دقتش پایین بیاید. مثلا در KNN اگر یک ویژگی در مقیاس هزار باشد و دیگری بین ۰ تا ۱، فاصله اقلیدسی عمدتا توسط ویژگی بزرگتر تعیین می شود.

- آیا همه مدل ها به نرمالسازی نیاز دارند؟

در حالت کلی خیر، بعضی مدل ها مانند KNN و SVM نیاز به نرمالسازی دارند چرا که بر اساس فاصله یا وزن ویژگی تصمیم میگیرند و برخی مدل ها هم مانند Random Forest و Decision Tree نیاز به نرمال سازی ندارند و مقیاس ناپذیرند و صرفا ویژگی ها را split می کنند.

و) برای این قسمت با توجه به خواسته سوال نمودار ROC Curve برای مدل های خواسته شده رسم شد که در زیر قابل مشاهده است:



نمودار 2

در این نمودار محور افقی نرخ مثبت کاذب (False Positive Rate) و محور عمودی نرخ مثبت واقعی (True Positive Rate) را نشان میدهد. هر چه نمودار به گوشه بالا چپ نزدیک تر باشد عمل کرد مدل بهتر است. همچنین مساحت زیر منحنی (AUC) معیار اصلی مقایسه است.

Logistic Regression:

برای این مدل مساحت زیر نمودار برابر با ۱ است که عملکردی ایده آل است و منحنی به شکل کامل به گوشه بالا چپ چسبیده است که نشان دهنده تفکیک کامل دو کلاس است. احتمالا داده ها به خوبی توسط مدل خطی قابل تفکیک اند و یا داده های آموزشی نسبتا ساده هستند.

Random Forest:

برای این مدل هم مساحت زیر نمودار برابر با ۱ است که مانند مدل قبلی نشان دهنده عملکرد بی نقص است. این مدل قدرت بالایی در مدل سازی روابط پیچیده دارد و در اینجا داده ها را کاملا درست طبقه بندی کرده است.

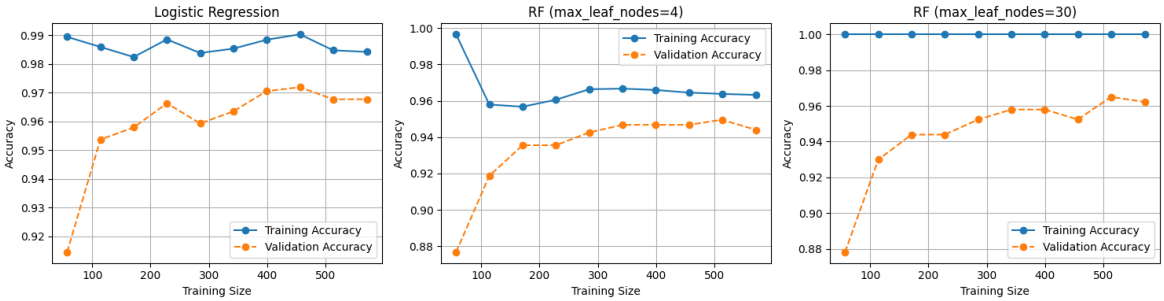
AdaBoost:

در این مدل هم مساحت زیر نمودار برابر با ۱ است که نشان دهنده عملکرد بی نقص مدل است. همچنین نمودار نشان میدهد AdaBoost توانسته به خوبی نقاط سخت را یاد بگیرد و هیچ خطای مثبت کاذبی ندارد.

Decision Tree:

مساحت زیر نمودار برای این مدل برابر با ۰.۹۴ بوده است که همچنان عملکرد خوب است اما به نقص نیست. منحنی در ابتدا سریع بالا می رود، اما سپس کمی به خط قطر نزدیک تر می شود، که نشان دهنده مقداری اشتباه در طبقه بندی است. احتمالا درخت ساده بوده و یا دچار Overfitting/Underfitting شده است.

ز) در این قسمت نیز بنا بر خواسته شده نمودار Learning Curve مدل های خواسته شده رسم شده که به شرح زیر است:



نمودار 3

Logistic Regression:

برای این مدل دقت آموزش حدود ۹۸٪ بوده است و نمودار ثابت و بدون افت خاص با افزایش داده هاست.

دقت Validation برای این مدل با افزایش حجم داده ها رشدی از حدود ۹۱٪ تا ۹۷٪ داشته است.

فاصله بین دو منحنی با افزایش داده ها نسبتا کم شده که نشان دهنده این است که مدل به خوبی generalized شده است.

این مدل bias متوسط و Variance کم دارد. یعنی نه مدل خیلی ساده است و نه خیلی پیچیده و به خوبی توانسته روی داده های جدید تعمیم پیدا کند.

واریانس: کم، بایاس: متوسط

رفتار یادگیری: تعادل خوب بین آموزش و تست و یادگیری پایدار

Random Forest (max_leaf_nodes=4):

دقت آموزش از ۱۰۰٪ به حدود ۹۶٪ کاهش یافته و تثبیت شده است.

دقت اعتبار سنجی از ۸۸٪ به حدود ۹۵٪ افزایش یافته است.

فاصله بین دو منحنی هنوز قابل توجه است.

در کل مدل دچار bias بالا و variance کم است. یعنی به دلیل محدودیت در عمق (max_leaf_nodes=4)، مدل توان یادگیری کامل داده را ندارد و overfitting دیده می شود.

واریانس: کم، بایاس: زیاد

رفتار یادگیری: ظرفیت یادگیری پایین است و underfitting دیده می شود.

Random Forest (max_leaf_nodes=30):

در این مدل دقت آموزش برابر ۱۰۰٪ و تقریبا کامل برای همه حجم های داده است.

دقت اعتبار سنجی از ۸۸٪ تا ۹۶٪ رشد داشته و در حال نزدیک تر شدن به آموزش است.

فاصله بین دو منحنب کاهش یافته ولی هنوز کمی وجود دارد.

مدل نسبتا bias کم و کمی با variance است. چون دقت اموزش بالا رفته ولی دقت اعتبار سنجی هنوز کمی عقب تر است احتمال overfitting خفیف وجود دارد ولی با افزایش داده عملکرد بهتر می شود.

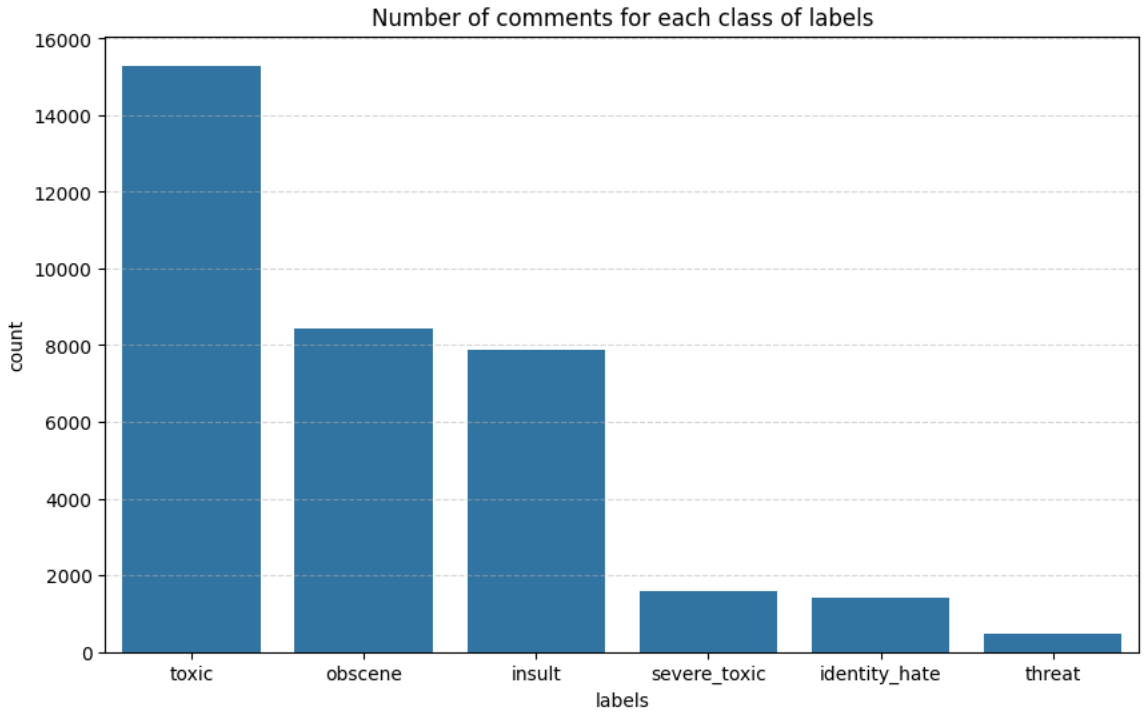
واریانس: متوسط، بایاس: کم

ظرفیت یادگیری: ظرفیت یادگیری بالاست اما کمی overfitting وجود دارد که احتمالا با افزایش داده ها حل شود.

3 -آشنایی با Multi-Label Classification:

الف) در طبقه‌بندی چندکلاسه (Multi-Class Classification)، هر نمونه فقط به یک کلاس از بین چند کلاس ممکن تعلق دارد و کلاس‌ها متقابلاً انحصاری هستند، مانند تشخیص نوع میوه (سیب، موز یا پرتقال). اما در طبقه‌بندی چندبرچسبی (Multi-Label Classification)، هر نمونه می‌تواند به چند کلاس به‌طور هم‌زمان تعلق داشته باشد، زیرا برچسب‌ها مستقل از یکدیگر هستند؛ برای مثال یک فیلم می‌تواند هم‌زمان کمدی و درام باشد. بنابراین تفاوت اصلی در تعداد برچسب‌های قابل انتساب به هر نمونه و استقلال بین برچسب‌هاست.

ج) در این بخش تعداد کامنت های هر دسته از برچسب ها شمارش شده و در نمودار زیر قابل مشاهده است:

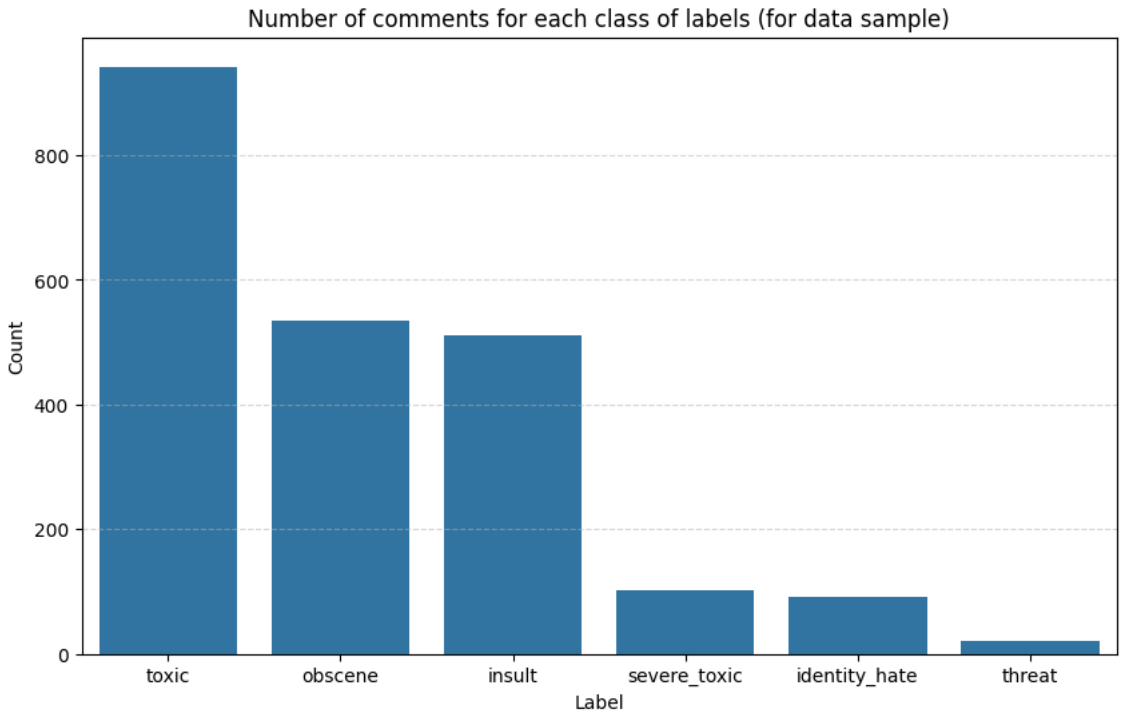


نمودار 4

بیشترین نظرات:

```
toxic            15294
obscene           8449
insult            7877
severe_toxic      1595
identity_hate     1405
threat             478
dtype: int64
```

د/ه) در این قسمت نیز یک نمونه تصادفی شامل ۱۰۰۰۰ کامنت از دیتافریم استخراج شده که در این نمونه تعداد کامنت ها برای هر دسته در زیر آمده است:



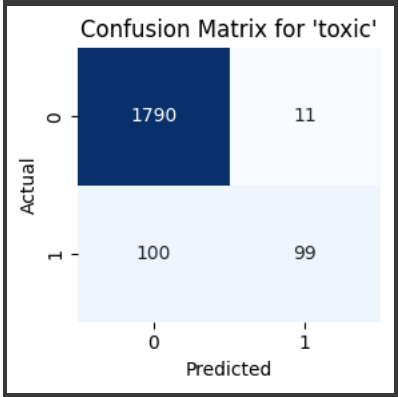
نمودار 5

```
toxic              942
obscene            535
insult             511
severe_toxic       102
identity_hate       91
threat              20
dtype: int64
```

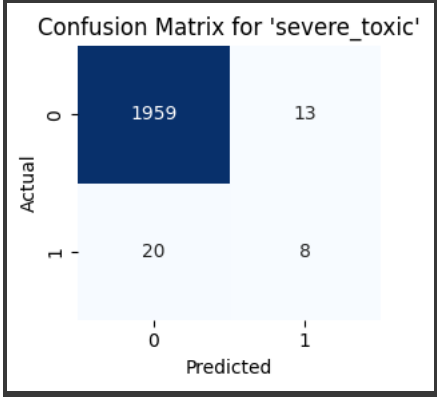
ح) برای پیاده‌سازی طبقه‌بندی چندبرچسبی (Multi-Label) با مدل Logistic Regression، می‌توان از استراتژی One-vs-Rest (OvR) استفاده کرد؛ در این روش برای هر برچسب (label) یک مدل مجزای Logistic Regression آموزش داده می‌شود که وظیفه دارد وجود یا عدم وجود آن برچسب را برای هر نمونه پیش‌بینی کند. بنابراین اگر مثلاً ۵ برچسب ممکن داشته باشیم، ۵ مدل لجستیک مستقل خواهیم داشت و هر مدل یک مسئله دوتایی (0 یا 1) را حل می‌کند. یکی دیگر از روش‌های رایج برای Multi-Label استفاده از Classifier Chains است که در آن پیش‌بینی هر برچسب به‌عنوان ویژگی به مدل بعدی داده می‌شود و وابستگی بین برچسب‌ها در نظر گرفته می‌شود.

ط) برای این قسمت از سوال من با استفاده از کتابخانه scikit learn مدل Logistic Regression را با استفاده از استراتژی OvR رو داده ها اعمال کردم که نتایج و ماتریکس آشفتگی برای هر دسته در زیر آورده شده است.

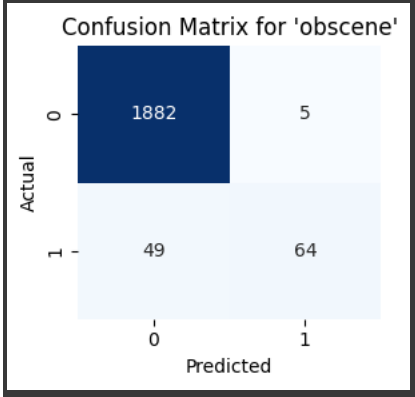
```
Label: toxic
Accuracy: 0.945
Recall: 0.497
```



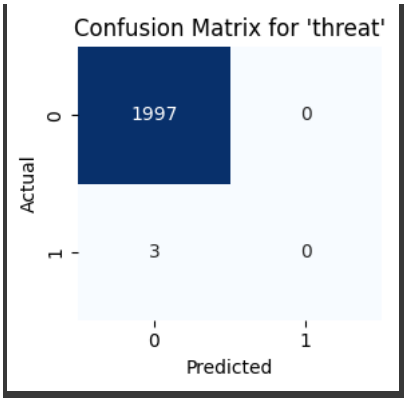

```
Label: severe_toxic
Accuracy: 0.984
Recall: 0.286
```



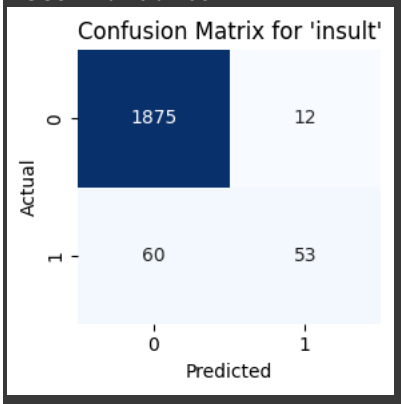

```
Label: obscene
Accuracy: 0.973
Recall: 0.566
```



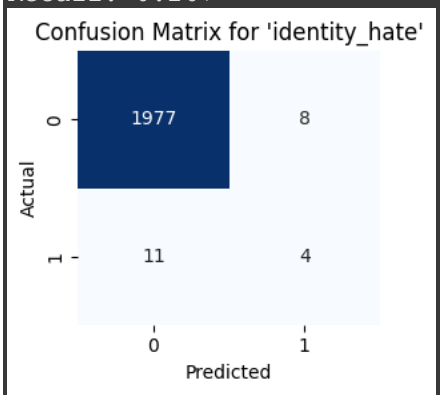

```
Label: threat
Accuracy: 0.999
Recall: 0.000
```




```
Label: insult
Accuracy: 0.964
Recall: 0.469
```




```
Label: identity_hate
Accuracy: 0.991
Recall: 0.267
```



در نگاه اول Accuracy بالا برای همه دسته ها بیننده را گمراه میکند که مدل به خوبی عملکرده است. اما با توجه به اینکه داده ها نامتوازن هستند و حتی با فعال کردن متد class_weight = ‘balanced’ در کد هم این مشکل حل نشده است.

همچنین Recall پایین در چند کلاس نشان می دهد که مدل در تشخیص بسیاری از نمونه های واقعی آن کلاس ناموفق بوده و یا کلاس اقلیت را نادیده گرفته است. در کل من فکر میکنم این عملکرد مدل به دلیل نامتوازن بودن داده ها بوده است.

اظهارنامه استفاده از هوش مصنوعی

تأیید می‌کنم که از ابزارهای هوش مصنوعی مطابق با دستورالعمل‌های بارگذاری شده در سامانه Elearn درس به طور مسئولانه استفاده کرده‌ام. تمام اجزای کار خود را درک می‌کنم و آماده بحث شفاهی درباره آنها هستم.