به نام خدا

مرضیه علیدادی – ۸۱۰۱۰۱۲۳۶

تکلیف ۳ داده کاوی

سوال ۱:

آ)

- class P: Heart Disease = "yes"
- class N: Heart Disease = "no"

$$Info(D) = I(6.4) = -\sum_{i=1}^{m} p_i log_2(p_i) = -\sum_{i=1}^{m} \frac{|C_{i.D}|}{|D|} log_2 \frac{|C_{i.D}|}{|D|}$$

$$= -(\frac{6}{10} log_2 \frac{6}{10} + \frac{4}{10} log_2 \frac{4}{10})$$

$$= -\big(0.6 * (-0.737) + 0.4 * (-1.322)\big)$$

$$= -\big((-0.4422) + (-0.5288)\big) = 0.971$$

$$Info_{Age}(D) = \frac{5}{10} I(3.2) + \frac{5}{10} I(3.2)$$

$$= -\frac{5}{10}\Big(\frac{3}{5} log_2 \frac{3}{5} + \frac{2}{5} log_2 \frac{2}{5}\Big)$$

$$-\frac{5}{10}\Big(\frac{3}{5} log_2 \frac{3}{5} + \frac{2}{5} log_2 \frac{2}{5}\Big)$$

$$= -0.5\big(0.6(-0.737) + 0.4(-1.322)\big)$$

$$-0.5\big(0.6(-0.737) + 0.4(-1.322)\big)$$

$$= -0.5\big((-0.4422) + (-0.5288)\big)$$

$$-0.5\big((-0.4422) + (-0.5288)\big)$$

$$= -0.5(-0.971) - 0.5(-0.971)$$

$$= 0.4855 + 0.4855 = 0.971$$

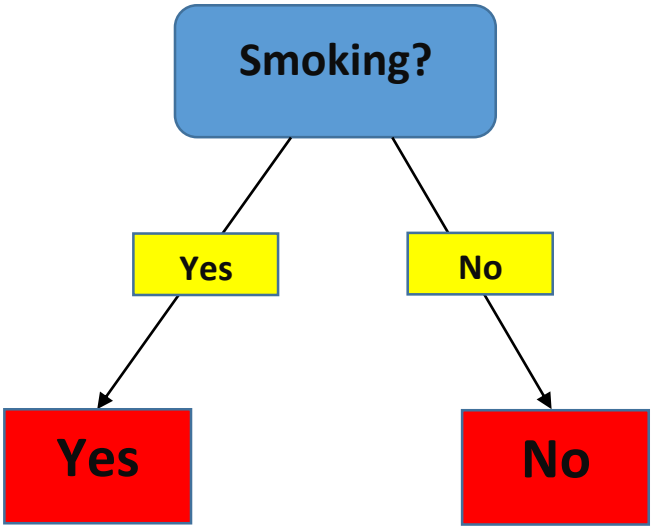
$$Info_{Gender}(D) = \frac{7}{10}I(4.3) + \frac{3}{10}I(2.1)$$
$$= -\frac{7}{10}\left(\frac{4}{7}log_2\frac{4}{7} + \frac{3}{7}log_2\frac{3}{7}\right)$$
$$-\frac{3}{10}\left(\frac{2}{3}log_2\frac{2}{3} + \frac{1}{3}log_2\frac{1}{3}\right)$$
$$= -0.7\big(0.57(-0.811) + 0.43(-1.2176)\big)$$
$$-0.3\big(0.67(-0.5778) + 0.33(-1.5995)\big)$$
$$= -0.7\big((-0.462) + (-0.523)\big)$$
$$-0.3\big((-0.387) + (-0.527)\big)$$
$$= -0.7(-0.985) - 0.3(-0.914)$$
$$= 0.69 + 0.27 = 0.96$$

$$Info_{Smoking}(D) = \frac{6}{10}I(6.0) + \frac{4}{10}I(4.0)$$
$$= -\frac{6}{10}\left(\frac{6}{6}log_2\frac{6}{6} + \frac{0}{6}log_2\frac{0}{6}\right)$$
$$-\frac{4}{10}\left(\frac{4}{4}log_2\frac{4}{4} + \frac{0}{4}log_2\frac{0}{4}\right) = -0.4(0 + 0) = 0$$

$$Info_{Exercise}(D) = \frac{6}{10}I(4.2) + \frac{4}{10}I(2.2)$$
$$= -\frac{6}{10}\left(\frac{4}{6}log_2\frac{4}{6} + \frac{2}{6}log_2\frac{2}{6}\right)$$
$$-\frac{4}{10}\left(\frac{2}{4}log_2\frac{2}{4} + \frac{2}{4}log_2\frac{2}{4}\right)$$
$$= -0.6\big(0.67(-0.5778) + 0.33(-1.5995)\big)$$
$$-0.4\big(0.5(-1) + 0.5(-1)\big)$$
$$= -0.6\big((-0.387) + (-0.527)\big)$$
$$-0.4(-0.5 - 0.5)$$
$$= 0.548 + 0.4 = 0.948$$

$$Gain(Age) = Info(D) - Info_{Age}(D) = 0.971 - 0.971 = 0$$
$$Gain(Gender) = Info(D) - Info_{Gender}(D) = 0.971 - 0.96 = 0.011$$
$$Gain(Smoking) = Info(D) - Info_{Smoking}(D) = 0.971 - 0 = 0.971$$
$$Gain(Exercise) = Info(D) - Info_{Exercise}(D)$$
$$= 0.971 - 0.948 = 0.023$$

با توجه به این‌که information gain مربوط به smoking از بقیه بیشتر است، این ویژگی برای split اولیه در ریشه‌ی درخت تصمیم انتخاب می‌شود:



با توجه به این‌که با بررسی همین یک ویژگی، تمام نمونه‌های مربوط به هر گره، متعلق به یک کلاس یکسان هستند، الگوریتم خاتمه می‌یابد و عمق درخت تصمیم برابر ۱ است.

ب) معمولاً زمانی که مدل، دقت بالایی روی داده‌های train دارد، امکان overfit بودن آن بر روی داده‌های train زیاد است. و ممکن است که این دقت بالا، در داده‌های تست حاصل نشود.

پارامتر تاثیرگذار دیگری که بر روی دقت مدل بر روی داده‌های جدید تاثیر می‌گذارد، پیچیده نبودن مدل است. مدل‌های پیچیده، ممکن است بر روی داده‌های train، overfit شده باشند، و همچنین قابلیت تفسیر پذیری پایینی دارند. که این موجب می‌شود که نتوان overfit بودن/نبودن آن‌ها بر روی داده‌های train را بررسی کرد.

در نتیجه، باید بین پیچیدگی مدل و دقت آن، تعادل برقرار کرد. تا در عین حال که سعی می‌شود مدل خیلی پیچیده و overfit نشود، دقت قابل قبولی داشته باشد.

برای بررسی دقت مدل, بهتر است داده‌ها را به دو بخش train set و test set تقسیم کرد. و دقت مدل ساخته شده به کمک train set را، بر روی test set ارزیابی کرد.

درخت تصمیم تولید شده در این سوال، دارای دقت ۱۰۰ درصد بر روی داده‌های train است. که این مسئله ممکن است به دلیل overfit بودن آن بر روی داده‌های train باشد. با توجه به این‌که عمق این درخت برابر ۱ است و پیچیده نیست، می‌توان آن را تفسیر کرد و منطقی بودن/نبودن نحوه‌ی تصمیم گیری آن را بررسی کرد. در این درخت، با توجه به مصرف دخانیات، ابتلا به بیماری قلبی پیشبینی شده است. که این تصمیم گیری منطقی به نظر می‌رسد و احتمال overfit بودن آن را کم می‌کند. همچنین، اگر ۱۰ داده‌ی موجود را به دو بخش تقسیم کنیم و از یک بخش برای ساخت درخت و از بخش دیگر برای تست آن استفاده کنیم، در هر دو بخش داده‌های تست و آموزش، به دقت ۱۰۰ درصد می‌رسیم. که با توجه به این نتایج و دلایل، به نظر می‌رسد که این درخت تصمیم تولید شده، توانایی تعمیم برای دادگانی که در مجموعه آموزش قرار ندارند را دارد.

ت) در این درخت تصمیم، از ویژگی smoking برای تعیین این‌که هر نمونه در کدام کلاس heart disease قرار می‌گیرد، استفاده می‌شود. با توجه به این‌که هر دو این نمونه‌ها دارای مقدار No برای smoking هستند، در کلاس No مربوط به heart disease قرار می‌گیرند. و پیشبینی می‌شود که با توجه به این‌که دخانیات مصرف نمی‌کنند، مبتلا به بیماری قلبی نیستند.

د) اگر این نمونه واقعاً متعلق به کلاس Yes باشند، با توجه به این‌که کلاس پیشبینی شده برای هر دوی آن‌ها No است، داریم:

Confusion Matrix:

Actual Class / Predicted Class	Yes	No	Total
Yes	0	2	2
No	0	0	0
Total	0	2	2

True Positive (TP): 0

False Positive (FP): 0

True Negative (TN): 0

False Negative (FN): 2

$$precision = \frac{TP}{TP + FP} = \frac{0}{0}$$

Precision این را بیان می‌کند که چند درصد از نمونه‌هایی که کلاس‌شان برابر Yes پیشبینی شده، واقعاً متعلق به کلاس Yes بوده‌اند. با توجه به این‌که برای هیچ یک از نمونه‌ها پیشبینی نشده که متعلق به کلاس Yes هستند، پس precision تعریف نشده‌است.

سوال ۲:

آ) در این مجموعه داده، ویژگی text برای دسته بندی داده‌ها استفاده خواهد شد. این ویژگی، یک سری کلمات را نشان می‌دهد، که وجود آن‌ها در متن ایمیل، با توجه به label ای که برای آن‌ها تعیین شده، برای تعیین spam بودن/نبودن ایمیل، استفاده خواهد شد. این کلمات (ویژگی‌ها) عبارتند از:

meet, today, ready, free, phone, ticket

برای استفاده از الگوریتم naïve bayes، باید فرض استقلال شرطی را در نظر گرفت. یعنی این فرض در نظر گرفته شود که ویژگی‌ها (کلمات) به صورت مستقل از یکدیگر و با احتمال ثابت، در کلاس‌های spam و non-spam توزیع شده‌اند. و فقط به کلاسی که به آن تعلق دارند، بستگی دارند.

ب) کلماتی که تنها در یکی از کلاس‌ها ظاهر شده‌اند، ممکن است به عنوان یک ویژگی ضعیف در مدل محسوب شوند. و قدرت دسته‌بندی را کاهش دهند. زیرا اطلاعاتی درباره‌ی کلاس دیگر شامل نمی‌شوند. و همچنین، ممکن است این کلمات نادر باشند و موجب overfit شدن مدل بر روی داده‌های train شوند. از همه مهم‌تر، مشکل zero-probability است. باید هر احتمال شرطی مقدارش غیر صفر باشد.

برای حل این مشکل احتمالی، می‌توان از روش‌هایی استفاده کرد:

- می‌توان کلماتی که به تعداد دفعات پایین ظاهر شده‌اند، را از مجموعه داده‌ها حذف کرد؛ تا بر قدرت مدل تاثیر منفی نگذارند.
- می‌توان کلماتی که به دفعات پایین ظاهر شده‌اند را با هم ترکیب کرد و تحت کلمات جدید با تعداد تکرار مطلوب به مجموعه داده اضافه کرد.
- می‌توان از روش Laplacian correction استفاده کرد. این روش، به این صورت عمل می‌کند که مثلا اگر کلمه‌ی w فقط در کلاس ۱ ظاهر شده است، به ازای هر یک از کلاس‌های ۱ و ۲، یک ایمیل شامل کلمه‌ی w به مجموعه داده‌ها اضافه می‌کند.

ج) استفاده از روش سوم، یعنی Laplacian correction:

چهار کلمه‌ی meet، free، phone و ready هر کدام فقط در یک کلاس ظاهر شده‌اند. پس مجموعه داده‌ها به شکل زیر به‌روز می‌شود:

Index	Text	Label
1	Today meet ready	Not-spam
2	Free phone today	Spam
3	free ticket	Spam
4	today ticket ready	Not-spam
5	free ticket free	Spam
6	meet	Not-spam
7	meet	Spam
8	free	Not-spam
9	free	Spam
10	phone	Not-spam
11	phone	Spam
12	ready	Not-spam
13	ready	Spam

احتمال هر یک از کلاس‌ها، به شرح زیر است:

$$\mathrm{P(Spam)} = \frac{7}{13}$$

$$\mathrm{P(Not-spam)} = \frac{6}{13}$$

احتمال تعلق هر یک از کلمات به هر یک از کلاس‌ها، به شرح زیر است:

$$\mathrm{P(meet|\ Spam)} = \frac{1}{7}$$

$$\mathrm{P(today|\ Spam)} = \frac{1}{7}$$

$$\mathrm{P(ready|\ Spam)} = \frac{1}{7}$$

$$\mathrm{P(free\ |\ Spam)} = \frac{4}{7}$$

$$\mathrm{P(phone|\ Spam)} = \frac{2}{7}$$

$$\mathrm{P(ticket|\ Spam)} = \frac{2}{7}$$

$$\mathrm{P(meet|\ Not-spam)} = \frac{2}{6}$$

$$\mathrm{P(today|\ Not-spam)} = \frac{2}{6}$$

$$\mathrm{P(ready|\ Not-spam)} = \frac{3}{6}$$

$$\mathrm{P(free\ |\ Not-spam)} = \frac{1}{6}$$

$$\mathrm{P(phone|\ Not-spam)} = \frac{1}{6}$$

$$\mathrm{P(ticket|\ Not-spam)} = \frac{1}{6}$$

این احتمال‌ها بدین شرح تفسیر می‌شوند: مثلا $\mathrm{P(free\ |\ Spam)}$ احتمال تعلق کلمه‌ی free به کلاس Spam را بیان می‌کند.

د) احتمال تعلق ایمیل شامل کلمات today phone ready به هر یک از دو کلاس:

$$P(\text{today phone ready}|\ \text{Spam}) = P(\text{today} \mid \text{Spam}) * P(\text{phone} \mid \text{Spam}) * P(\text{ready}|\ \text{Spam}) * P(\text{Spam}) = \frac{1}{7} * \frac{2}{7} * \frac{1}{7} * \frac{7}{13} = 0.003$$

$$P(\text{today ready}|\ \text{Not} - \text{spam}) = P(\text{today} \mid \text{Not} - \text{spam}) * P(\text{phone} \mid \text{Not} - \text{spam}) * P(\text{ready}|\ \text{Not} - \text{spam}) * P(\text{Not} - \text{spam}) = \frac{2}{6} * \frac{1}{6} * \frac{3}{6} * \frac{6}{13} = 0.012$$

با توجه به این‌که احتمال تعلق ایمیل شامل کلمات today phone ready به کلاس Not-spam بیشتر است، این ایمیل، برچسب Not-spam را دریافت می‌کند.

سوال ۳:

آ) درخت‌های تصمیم تولید شده با استفاده از این روش، دارای دقت تصادفی هستند و بسته به مجموعه داده‌ی مدنظر و ترتیب تصادفی انتخاب پارامترها، به دقت متفاوتی می‌رسند. این روش، گاهاً می‌تواند منجر به overfit شدن درخت تصمیم بر روی داده‌های train شود؛ زیرا داده‌ها را به طور تصادفی برای split کردن درخت استفاده می‌کند و با توجه با همین تصادفی بودن انتخاب‌ها، پیچیدگی غیرقابل تفسیر در درخت ایجاد می‌کند.

ب) این روش‌های هموار سازی، برای حل مشکلاتی از قبیل zero-probability در naïve bayes کاربرد دارند. اما باید از این روش‌ها به نحو مناسبی استفاده کرد؛ در غیر این صورت، ممکن است مشکلاتی از قبیل overfit یا underfit شدن مدل بر روی داده‌های train ایجاد شود. در صورتی که از یک روش هموارسازی به شکلی استفاده کنیم که تاکید و توجه زیادی بر روی رفتار یک سری از ویژگی‌ها باشد و یا هموار سازی شدیدی روی داده‌ها اعمال شود، احتمال پیچیده شدن مدل به صورت غیرمعقول و overfit شدن مدل بر روی داده‌های train و دقت ناکافی آن برای داده‌های جدید وجود دارد. از طرف دیگر، در صورتی که از هیچ یک از روش‌های هموار سازی استفاده نشود، یا در حد ساده

و ابتدایی از آن‌ها استفاده شود، احتمال ساده بودن مدل و underfit شدن آن وجود دارد. که در این صورت، دقت مدل هم برای داده‌های train و هم برای داده‌های جدید، کم و ناکافی خواهد بود.

در نتیجه، ضروری است تا از روش‌های هموار سازی، به نحو مناسب و با تنظیم مناسب پارامترهای آن‌ها استفاده کنیم؛ تا از افراط و تفریط در هموار سازی جلوگیری شود و بتوان از مزایای این روش‌ها، بدون overfit یا underfit شدن مدل، بهره برد.

ج) در روش bagging، تعدادی مدل مستقل از هم بر روی داده‌های train تولید می‌شود. و در نهایت، خروجی این مدل‌ها از طریق رای اکثریت و یا میانگین گیری، تجمیع می‌شود. این روش، از overfit شدن مدل نهایی و بالا بودن واریانس آن جلوگیری می‌کند.

در روش boosting، در هر مرحله، تعدادی مدل را بر روی مجموعه داده‌ها آموزش می‌دهیم. و این کار را به صورت iterative انجام می‌دهیم. در هر مرحله، به نمونه‌هایی که در مراحل قبلی به درستی دسته بندی نشدند، وزن بیشتری داده می‌شود تا مدل‌ها، در بررسی خود، دقت بیشتری صرف دسته بندی آن‌ها کنند. و در نهایت، خروجی این مدل‌ها از طریق رای اکثریت و یا میان گیری، به صورت وزن دار، تجمیع می‌شود. به این صورت که هر مدل که در طی مراحل دسته بندی دقت بیشتری داشته، در تجمیع نتیجه‌ها، وزن بیشتری به آن در رای گیری یا میانگین گیری تعلق می‌گیرد. در این روش، با توجه به این‌که توجه بیشتری به نمونه‌هایی که سخت تر دسته بندی می‌شوند می‌شود، از بایاس شدن نتیجه نهایی جلوگیری می‌شود.

پس تفاوت این دو روش ensemble در ترکیب خروجی‌های مدل‌ها، در این است که boosting بر خلاف bagging، ترکیب خروجی‌های مدل‌ها را به صورت وزن دار انجام می‌دهد. و این موجب می‌شود تا دقت بیشتری داشته باشد؛ اما گاهاً نسبت به نمونه‌هایی که سخت‌تر دسته بندی می‌شوند، overfit می‌شود.

د) منحنی ROC ابزار مهمی برای ارزیابی مدل‌های طبقه‌بندی باینری است. برای استفاده از این ابزار برای طبقه بندی‌های چند کلاسه، باید از راهکارهایی استفاده کنیم: OvR و OvO

روش OvR (one vs rest)، هر بار یکی از کلاس‌ها را در مقابل بقیه‌ی کلاس‌ها در نظر می‌گیرد. به این ترتیب، که یکی از کلاس‌ها را به عنوان کلاس positive در نظر می‌گیرد و همه‌ی بقیه‌ی کلاس‌ها را به کلاس negative در نظر می‌گیرد. در نتیجه، گویی با یک طبقه بندی باینری مواجه هستیم. این

کار برای تک کلاس‌ها انجام می‌شود. در نتیجه، اگر در ابتدا یک طبقه بندی n کلاسه داشتیم، در نهایت n طبقه بندی باینری خواهیم داشت. و باید n نمودار ROC رسم کنیم.

روش OvO (one vs one)، هر کلاس را هر بار در مقابل یکی از کلاس‌های دیگر در نظر می‌گیرد. در واقع تمام زوج کلاس‌های ممکن را در مقابل یکدیگر در نظر می‌گیرد و یک بار، یکی را به عنوان کلاس positive و دیگری را به عنوان کلاس negative در نظر می‌گیرد و بار دیگر، جای کلاس‌های positive و negative را عوض می‌کند و آن‌ها را در مقابل هم به عنوان یه طبقه بندی باینری در نظر می‌گیرد. در نتیجه، اگر در ابتدا یک طبقه بندی n کلاسه داشتیم، در نهایت n(n-1) طبقه بندی باینری خواهیم داشت. و باید n(n-1) نمودار ROC رسم کنیم.

۵) این توابع فعال‌سازی، به شبکه‌های عصبی برای غیرخطی بودن کمک می‌کنند. اگر از این توابع فعال‌سازی غیرخطی استفاده نکنیم، شبکه عصبی قادر به حل مسائل پیچیده واقعی مانند پردازش تصویر، ویدیو، صوت، گفتار و متن، پردازش زبان طبیعی و غیره نخواهد بود؛ زیرا شبکه عصبی خطی خواهد بود و مدل‌های خطی نمی‌توانند مسائل پیچیده واقعی را حل کنند. اگر چه مدل‌های خطی ساده هستند، اما محاسباتی ضعیف دارند و قادر به پردازش مسائل پیچیده نیستند. بنابراین، اگر از توابع فعال‌سازی استفاده نکنیم، صرفاً با استفاده از تعداد لایه‌های پنهان بیشتر در شبکه عصبی، هنوز هم مدل خطی و دارای کارایی پایین خواهد بود. تابع فعال‌سازی در یک شبکه عصبی، یک عملیات ریاضی است که بر روی خروجی هر گره (نورون) در شبکه اعمال می‌شود و تعیین می‌کند که آیا نورون باید فعال شود یا خیر. هدف از استفاده از توابع فعال‌سازی، معرفی غیرخطی بودن به خروجی نورون است که این کار باعث می‌شود شبکه بتواند رابطه‌های پیچیده‌تری بین ورودی‌ها و خروجی‌ها یاد بگیرد. برخی از توابع فعال‌سازی محبوب عبارتند از: sigmoid، ReLU و tanh.

- sigmoid:

مزایا: این تابع، ورودی‌ها را به بازه ۰ تا ۱ نگاشت می‌کند؛ که این خاصیت، آن را برای مسائل دسته‌بندی باینری مناسب می‌کند. همچنین، قابل مشتق گرفتن است، که باعث می‌شود برای استفاده در الگوریتم backpropagation مناسب باشد.

معایب: این تابع، برای ورودی‌های بزرگ، اشباع می‌شود؛ که باعث می‌شود گرادیان آن خیلی کوچک شود و مشکل Vanishing Gradient رخ دهد. همچنین، خروجی آن، مرکزش صفر نیست؛ که این می‌تواند منجر به همگرایی کند الگوریتم‌های بهینه‌سازی مانند gradient descent شود.

- ReLU:

مزایا: این تابع، به دلیل این‌که تنها با مقدار آستانه ورودی درگیر می‌شود، محاسبات کارآمدی دارد. همچنین، این تابع باعث جلوگیری از مشکل Vanishing Gradient که در تابع فعال‌سازی sigmoid رخ می‌دهد، می‌شود.

معایب: این تابع، ممکن است نورون‌های مرده ایجاد کند؛ به این معنی که برخی از نورون‌ها ممکن است برای همه ورودی‌ها خروجی صفر تولید کنند و در نتیجه، اطلاعاتی از بین بروند. همچنین، این تابع، در مقدار صفر قابل مشتق‌گیری نیست؛ که ممکن است چالش‌هایی را برای الگوریتم‌های بهینه‌سازی مانند gradient descent ایجاد کند.

- tanh:

مزایا: این تابع، خروجی با میانگین صفر و واریانس یک تولید می‌کند؛ که می‌تواند عملکرد شبکه را بهبود بخشد. این تابع قابلیت مشتق‌پذیری دارد، که آن را برای استفاده در الگوریتم backpropagation مناسب می‌کند.

معایب: این تابع، برای ورودی‌های بزرگ، اشباع می‌شود؛ که باعث می‌شود گرادیان آن خیلی کوچک شود و مشکل Vanishing Gradient رخ دهد.

و) این تکنیک‌ها، برای تولید نمونه‌های جدید با استفاده از اعمال تغییر شکل‌های مختلف بر روی داده‌های موجود به کار می‌روند. این روش‌ها می‌توانند با افزایش تعداد نمونه‌های کلاس‌های اقلیتی در مجموعه داده‌ها، به حل مشکل کلاس‌های نامتوازن کمک کنند. به عنوان مثال، اگر یک مجموعه داده فقط چند نمونه از یک بیماری نادر داشته باشد، ممکن است مدل به سمت کلاس اکثریت تمایل پیدا کرده و عملکرد ضعیفی روی کلاس اقلیتی ارائه دهد. در این صورت، می‌توان با استفاده از این تکنیک‌ها، نمونه‌های اضافی با چرخش، مقیاس‌بندی یا وارون‌کردن تصاویر اصلی بسازیم. با انجام این کار، می‌توانیم تنوع بیشتری به مجموعه داده بدهیم و برای مدل، نمونه‌های بیشتری برای یادگیری فراهم کنیم. در کل، این تکنیک‌ها با ایجاد نمونه‌های آموزشی متنوع و تعداد بیشتر برای مدل، می‌توانند باعث بهبود عملکرد مدل‌های آموزش داده شده روی مجموعه داده‌های نامتوازن شوند.

این تکنیک‌ها با چالش‌ها و محدودیت‌هایی همراه هستند:

۱. بعضی از انواع داده‌ها به راحتی قابل افزایش نیستند. به عنوان مثال، استفاده از تکنیک‌های سنتی مانند چرخش یا وارون‌کردن در داده‌های متنی بسیار دشوار یا غیرمنطقی است.

۲. استفاده نامناسب از تکنیک‌های افزایش داده، می‌تواند منجر به overfit شدن مدل بر روی داده‌های آموزش شود. در این حالت، مدل بر داده‌های آموزش خیلی تمرکز می‌کند و عملکرد ضعیفی روی داده‌های تست ارائه می‌دهد.

۳. تولید تعداد زیادی نمونه، ممکن است باعث افزایش هزینه محاسباتی برای آموزش مدل شود؛ که در مجموعه داده‌های بزرگ، این محدودیت محسوس است. به همین دلیل، انتخاب و استفاده از تکنیک‌های مناسب برای نوع داده و مسئله مورد نظر بسیار مهم است.

ز)

در الگوریتم KNN ، مقدار K تعداد همسایگانی را نشان می‌دهد که در هنگام پیش‌بینی مورد بررسی قرار می‌گیرند.

یکی از روش‌های معمول برای انتخاب بهترین مقدار K در الگوریتم KNN، استفاده از cross validation است. این روش، ابتدا داده‌ها را به چند بخش تقسیم می‌کند. سپس هر بار از یکی از این بخش‌ها به عنوان مجموعه داده‌ی تست استفاده می‌کند و مدل را روی بخش‌های باقی‌مانده آموزش می‌دهد. سپس، عملکرد مدل ارزیابی شده و این فرآیند با استفاده از مقادیر مختلف K تکرار می‌شود تا بهترین مقدار پیدا شود.

اگر به K مقدار خیلی کوچکی داده شود، الگوریتم به سرعت به نویز یا داده‌های پرت واکنش نشان می‌دهد و این باعث overfit شدن مدل و عدم توانایی تعمیم مدل برای پیش‌بینی در داده‌های جدید، و کاهش عملکرد پیش‌بینی در داده‌های آزمایشی می‌شود. از طرف دیگر، اگر به K مقدار خیلی بزرگی داده شود، الگوریتم قادر به درک الگوهای مهم در داده‌ها نخواهد بود و موجب کاهش عملکرد پیش‌بینی هم در داده‌های آموزشی و هم در داده‌های تست می‌شود. این موضوع باعث underfit شدن مدل، و کاهش دقت پیش‌بینی مدل می‌شود. بنابراین، انتخاب یک مقدار مناسب برای K، برای به دست آوردن عملکرد بهینه در الگوریتم KNN، بسیار حائز اهمیت است.