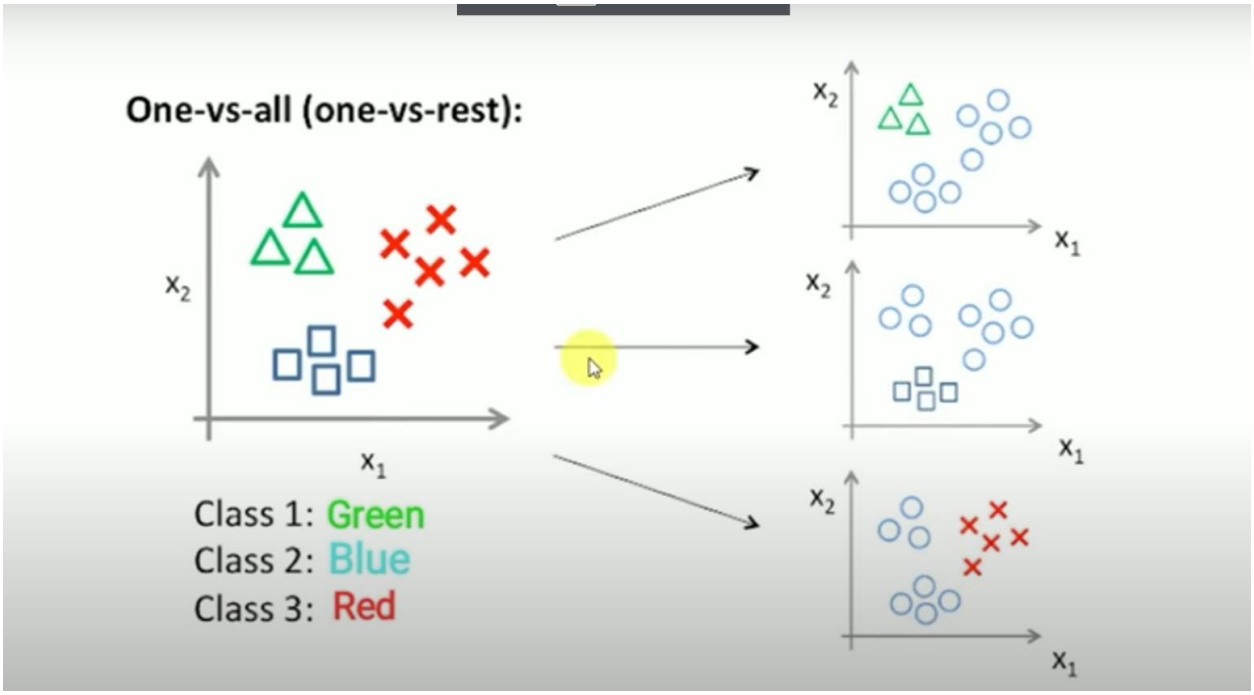
Mahan Veisi – 400243081 – ML HW2

-1 یکی از ساده ترین راه هایی که به ذهن میرسد این است که در ابتدا داده خود را به دو بخش اصلی train و test تقسیم کنیم. و پس از اموزش مدل خود روی داده های آموزشی، با داده های تست آن را بسنجیم. البته قابل ذکر است که بهتر است برای hyperparameterهای مناسب، بخشی از داده ها را نیز به validation اختصاص بدهیم. با اینکار متوجه میشویم که ایا مدل ما روی داده هایی که دیده است (داده های اموزشی) overfit شده است یا میتواند روی سایر داده های موجود خوب عمل کند.
یکی از کار های مهم بررسی مدل خود روی داده های outlier میباشد:
- برای رسیدگی به این نوع داده ها کارهای متفاوتی میتوان کرد. یکی از آن ها استاندار سازی دیتا بوده، همچنین استفاده ازlog میتواند تاثیر داده های بسیار بزرگ را کمتر کند.
- یکی دیگر از راه ها حذف آنها یا جابجایی آن ها با داده های مرکز گرا تر میباشد یا یک threshold برای آن در نظر بگیریم که برای داده های بیرون از این threshold، بتوانیم عملیات تشخیص را راحتتر کنیم.

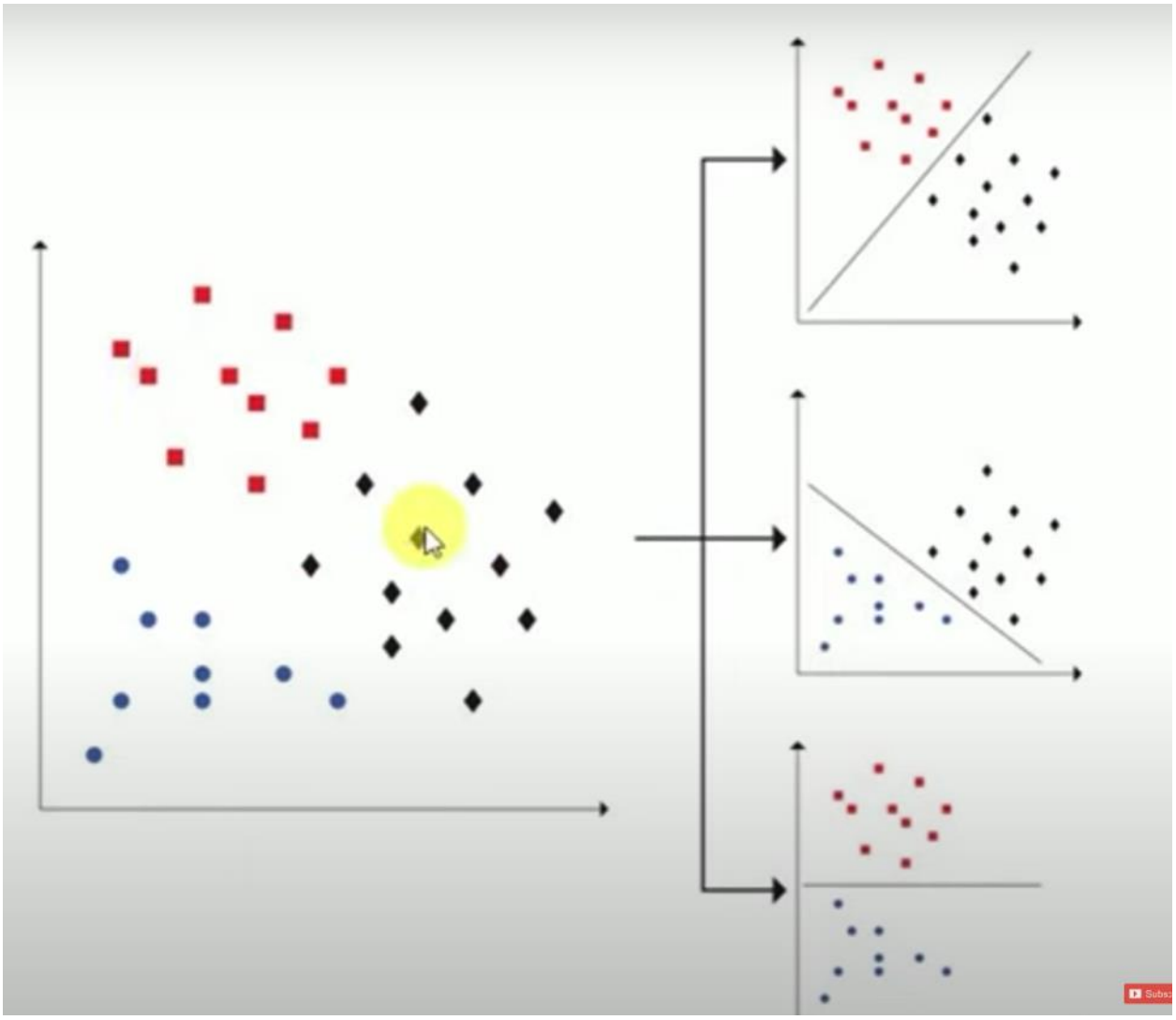
-2

برای زمانی که بیشتر از دو نوع لیبل داریم، در مسائل classification، روش های متفاوتی برای ارائه نشان دادن لیبل ها وجود دارد. از جمله:

one-vs-rest (one-vs-all): در این روش برای هر کلاس، یک طبقه‌بند دودویی جداگانه آموزش داده می‌شود تا نمونه‌های آن کلاس را از همه‌ی کلاس‌های دیگر تشخیص دهد. اگر تعداد N کلاس وجود داشته باشد، N طبقه‌بند دودویی آموزش داده می‌شود. در هنگام پیش‌بینی، کلاس مرتبط با طبقه‌بندی که بیشترین اطمینان را ایجاد کرده است، اختصاص داده می‌شود.



one-vs-one: برای N کلاس، N*(N-1) / 2 طبقه‌بند دودویی آموزش داده می‌شود، هرکدام با یک جفت از کلاس‌ها متفاوت. در هنگام پیش‌بینی، هر طبقه‌بند برای یک کلاس رأی می‌دهد و کلاسی که بیشترین رأی را برای خود جلب کند، اختصاص داده می‌شود.



انتخاب هر یک از این دو نوع دسته بندی به موضوع و میزان داده های ما و میزان اهمیت ما به هزینه های محاسباتی یا دقت مدل وابسته است.

برای مثال one-vs-all مشکلی که دارد این است که در هر بخش، سایر دیتا هارا در یک دسته قرار میدهد که میتواند مشکلاتی برای تشخیص را ایجاد کند. اما one-vs-one که تا حدی میتواند این مشکل را حل کند، ممکن است در زمانی که فیچر های زیادی داریم، بار محاسباتی و فضایی زیادی را ایجاد کند.

3- در مواردی که دیتا همانند شکل داده شده است، الگوریتم linear regression با این داده ها که به‌طور خطی قابل تفکیک نیستند، به مشکل خواهد خورد. زیرا در linear regression سعی میشود که یک خط را به گونه ای تشکل دهد که دو کلاس در آن جدا شوند. همانطور که در این شکل نیز مشخص است، یک linear decision نمیتواند کار خاصی کند و نیاز به یک خط تصمیم غیرخطی مانند دایره یا بیضی هستیم. که البته برای مدیریت آن الگوریتم های متفاوتی وجود دارند از جمله:

***یکم بیشترش کن

- **Kernelized Support Vector Machines (SVM):** که در این تمرین ازآن حسابی استفاده خواهد شد. که البته قابل ذکر است خود این الگوریتم در باطنش از یک خط تصمیم خطی استفاده میکند اما ترفند جالبی که استفاده میشود این است که این خط را در بعدی دیگر (معمولا یک بعد اضافه یا کم میشود) تشکل میشود.!
- **K-Nearest Neighbors (KNN):** در تمرین قبلی با آن اشنا شدیم و میتواند ایده تا حدی قابل قبول برای این مسئله باشد
- و ...

4- یکی از مشکلات رایج در این حوزه مواجه شدن با داده هایی میباشد که یک کلاس از داده ها به میزان بسیار زیادی، بیشتر از سایرین است که اتفاقا در همین تمرین و سوال یک کدی با آن و مواجه شدیم. این نوع دیتاست ها میتوانند باعث این شوند که روی کلاسی که داده اکثریت را دارد به خوبی عمل کند اما روی دیگری بسیار ضعیف باشد و اگر برنامه نویس به آن دقت نکرده باشد، ممکن است پس از بررسی accuracy، احساس کند که دقت بسیار بالایی داشته و شرایط ایده آل است در حالی که خلاف امر میباشد. در این شرایط راهکار هایی وجود دارد:

- resampling:

در این روش سه نوع کار میتوان انجام داد: Undersampling – Oversampling و SMOTE (Synthetic Minority Over-sampling

Technique)

که در روش undersampling به صورت تصادفی نمونه‌ها را از کلاس اکثریت حذف میکنیم تا توازن توزیع کلاس‌ها ایجاد شود.
در oversampling باید نمونه‌های مصنوعی برای کلاس اقلیت تولید یا تکرار شوند تا نمایندگی آن افزایش یابد.

در SMOTE که یه نسخه از oversampling است نمونه‌های مصنوعی برای کلاس اقلیت بر اساس شباهت ویژگی تولید میشوند.
در این تمرین کدی از undersampling استفاده شده است.

روش های دیگری نیز وجود دارند از جمله اینکه به هر کلاس وزن‌های مختلفی اختصاص دهیم تا الگوریتم در طول آموزش به کلاس اقلیت حساس‌تر شود. یکی دیگر از راه ها تنظیم threshold میباشد که البته در این تمرین استفاده شده و آستانه ای را تصمیم الگوریتم را تنظیم میکنیم تا بین دقت و recall توازن ایجاد شود. در حالت‌های نامتوازن، ممکن است مهم باشد که به recall بیشتر توجه کنیم و میزان آن را با کاهش دقت افزایش دهیم.

همچنین میتوان به Feature Engineering، Anomaly Detection و ... اشاره کرد.