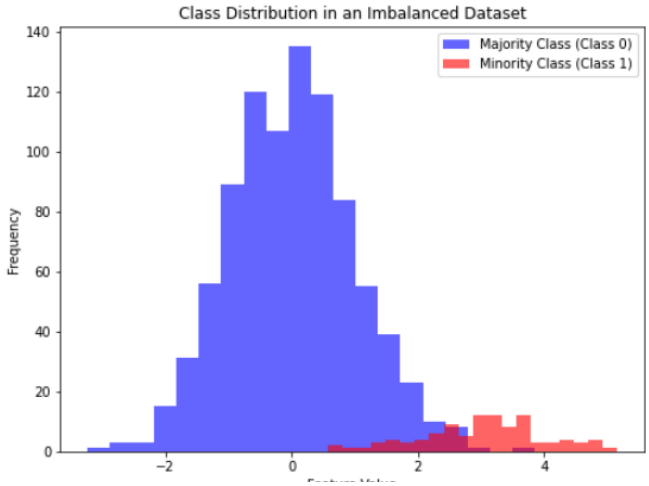
Dengesiz Veri Setleri ile Makine Öğrenmesi Modelleme

Makine öğrenmesi modeli geliştirme ile yolu en az 1 kere kesişen herkesin çok iyi bildiği bir şey vardır ki modelin eğitileceği veri, modelin kendisinden ve algoritmalarından çok daha kritik bir role sahiptir. Modelimizi bir araç olarak düşünürsek veri setimizi bu aracın çalışmasını sağlayan bir çeşit yakıt olarak düşünebiliriz. Makine öğrenmesi modelleri bebek gibi de düşünülebilir. Bir bebeği yetiştirirken yani hayata hazırlarken verdiğimiz verilerin yapılandırılmış ve doğru olmasıyla birlikte veriliş biçimine (yani algoritmalarına) nasıl ve neden dikkat ediyorsak, bir modelin eğitim sürecinde kullanılan verilerin amacı çok da farklı değildir. Bu nedenle model eğitirken en çok dikkat edilmesi gerekenler arasında verinin temizliği, türü, yapılandırılması, kullanılma biçimi gibi çeşitli faktörler yer almaktadır. Verinin yapılandırılması, özellik mühendisliği gibi konular artık yaygınlaştığından ötürü bu yazıda, çok dikkate alınmayan ama aslında yine model eğitiminde kritik bir role sahip olan **dengesiz veri setleri ile makine öğrenmesi modelleme** konusunu ele alacağız.

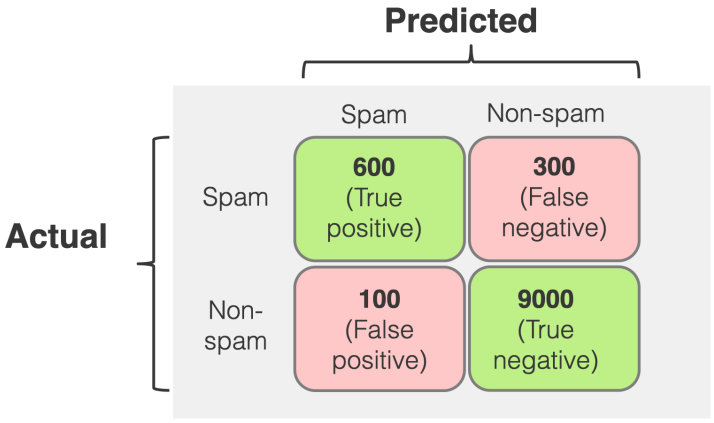
Öncelikle, bir dengesiz veri seti nedir?

Dengesiz veri seti, sınıflandırma problemlerinde karşımıza çıkan, hedef değişken sınıflarının farklı sayılarda verilere sahip olduğu durumlarda modelin yanlış öğrenerek yanlış tahminler yapmasına yol açan, veriler arasındaki can sıkıcı dengesizlik durumudur. Ama bugün bu konuyu can sıkıcı olmaktan kurtararak, nasıl baş edeceğimizi öğreneceğiz. Histogram üzerinde örnek bir dengesiz veri setini inceleyelim:



Figür 1

Bir sınıflandırma probleminde iki sınıfımız olduğunu düşünürsek (mesela gerçek dünya örneği olarak spam email sınıflandırma gibi) çoğunluğu spam olmayan, azınlığı spam olan olarak düşünebiliriz. Bu da veri setinde modelin performansını yüksek derecede etkileyen bir dengesizliğe neden olur. Sınıflar arası fark az olduğu zaman ve büyük veri setlerindeki dengesizlik her zaman sorun çıkartmaz, ancak küçük veri setlerindeki en ufak bir dengesizlik bile çok büyük performans kayıplarına yol açabileceğinden veri mühendisliği konusunda dikkatli olmamız gerekecektir.



Figür 2 - Spam Email Sınıflandırma Karışıklık Matrisi

Yukarıdaki karmaşıklık matrisinde de görmüş olduğumuz üzere True Negative ve False Positive değerlerinde büyük bir açıklık mevcut. Peki bu bir sınıflandırma modelinin performans metriklerinde nasıl sonuçlara yol açıyor? Hesaplayıp yorumlayalım.

- Accuracy: Modelin tüm tahminleri içinde ne kadar doğru sonuç verdiğini gösterir. Genel başarı oranıdır; ancak dengesiz veri setlerinde yanıltıcı olabilir çünkü model yalnızca çoğunluk sınıfını tahmin ederek yüksek skor alabilir.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

- Precision: Modelin "pozitif" dediği örneklerin ne kadarının gerçekten pozitif olduğunu gösterir. Yanlış alarm verme oranını ölçer ve özellikle yanlış pozitiflerin maliyetli olduğu durumlarda kritiktir.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: Gerçek pozitif örneklerin ne kadarını doğru tahmin ettiğini gösterir. Kaçırılan pozitif örnekleri (false negatives) dikkate alır; dolayısıyla önemli olan pozitifleri yakalamaktır.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- F1-Score: Precision ve Recall'un **harmonik** ortalamasıdır. Özellikle precision ve recall arasında denge kurmak gerektiğinde ve veri dengesiz olduğunda daha güvenilir bir genel performans ölçütü sağlar.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Figür 2'deki örneğin içerdiği metriklerin arasındaki fark bize burada bir şeylerin yanlış olduğunu söylüyor, bu da veri setindeki sınıfların dağılımının dengesizliği olarak yorumlanabilir. Dengesiz veri setlerinin bize açmış olduğu bu sorunları çözmek için öncelikle bu sınıflandırma ölçümlerini yorumlamayı iyice öğrenerek başlamalıyız, ardından birkaç farklı yola başvurabiliriz. Yazının devamında bazı potansiyel çözüm alternatiflerini inceleyeceğiz.

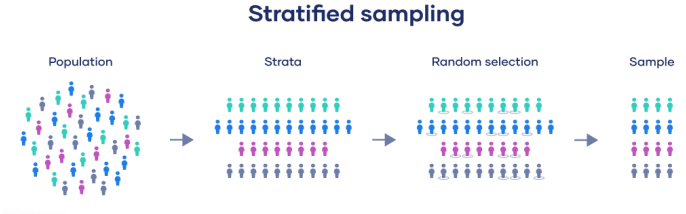
Potansiyel Çözüm Önerileri

1. **Stratified Sampling (Tabakalı Örnekleme)**

Temel fikir, verideki her bir sınıfın (örneğin "spam" ve "spam değil") oranını koruyarak eğitim ve test setlerine dağıtmak.

Neden?

Rastgele bölünce (mesela %80 eğitim, %20 test) azınlık sınıfındaki örnekler bazen tamamıyla test setine, bazen de tamamıyla eğitim setine gidebilir. Böylece model, test sırasında hiç görmediği bir sınıfla karşılaşabilir veya eğitim sırasında yeterince örnek göremez. Stratified sampling Figür 3'te gösterilen dengeli ayrıştırmayı yapmamıza olanak sağlar.



Figür 3

1.1. Stratified Random Sampling (Tabakalı Rastgele Örnekleme)

- **Nasıl çalışır?**
- Veri setindeki her sınıf için örnekleri ayrı bir torbaya koyarız.
- Her torbadan, istenen oranda (örneğin %80) rastgele çekerek eğitim setini oluştururuz; kalan %20'yi de test setine ekleriz.
- **Örnek:**
- Elinizde 100 e-posta var: 90 "spam değil", 10 "spam".
- Stratified Random Sampling ile %80 eğitim seti: "Spam değil"den 72 örnek, "spam"den 8 örnek
- %20 test seti: "Spam değil"den 18, "spam"den 2 örnek

1.2. Stratified K Fold (Tabakalı K Katlama)

- **Nasıl çalışır?**
- Veriyi K parçaya (*fold*) bölerken her parçanın içinde de sınıf oranları korunur.
- Örneğin K=5 ise; her fold'ta "spam" ve "spam değil" oranı orijinal veridekiyle aynı kalır.
- Modeli 5 kez eğitir ve her seferinde bir fold'u test, kalan 4 fold'u eğitim olarak kullanmış olursunuz.
- Avantajı:
- Küçük veri setlerinde bile her fold'da dengeli sınıf dağılımı olur.
- Modelin genel performansını daha güvenilir ölçeriz.

1.3. Stratified Shuffle Split (Tabakalı Karışık Ayırma)

- Nasıl çalışır?
- Tıpkı Stratified Random Sampling gibi sınıf oranlarını korur, ama birden fazla kez (örneğin 10 farklı rastgele bölünme) eğitim-test seti oluşturmanıza izin verir.
- Her bölünmede veriler karıştırılır (shuffle) ve sonra tabakalı örnekleme yapılır.
- Ne zaman tercih edilir?
- Model performansının, farklı rastgele bölünmelerde ne kadar değiştiğini görmek istediğinizde.
- K-Fold'dan daha esnek: istediğiniz sayıda "deneme" (split) yapabilirsiniz.

2. **SMOTE (Synthetic Minority Over-sampling Technique)**

Temel fikir, azınlık sınıfına ait "yeni" sentetik (yapay) örnekler oluşturarak dengeyi sağlamaktır. **Nasıl çalışır?:** Azınlık sınıfındaki her örnek için, en yakın komşularını (aynı sınıftan) bulur. Rastgele bir komşu seçer ve bu iki örnek arasındaki çizgi üzerinde yeni bir nokta (örnek) üretir.

Örnek:

a. "Spam" sınıfında 10 örnek var, "spam değil"de 90.
b. SMOTE ile "spam" sınıfını 90'a tamamlamak için 80 sentetik örnek üretebilirsiniz.

Dikkat edilmesi gerekenler:

c. Aşırı yapay örnekler, orijinal verinin yapısını bozabilir.
d. SMOTE'u genellikle eğitim verisine uygulayıp, test verisini olduğu gibi bırakmak en iyisidir.

3. **Adjusting Class Weights (Sınıf Ağırlıklarını Ayarlama)**

Temel fikir, modeli azınlık sınıfı hataları için "daha fazla ceza" vermeye zorlamaktır.

Nasıl çalışır?: Modelin kayıp (loss) fonksiyonuna, her sınıf için ayrı bir ağırlık verirsiniz. Azınlık sınıfına daha yüksek ağırlık vererek, modelin o sınıfı yanlış sınıflandırmasını daha "pahalı" hale getirirsiniz.

Örnek:

- "Spam" sınıfı %10, "spam değil" %90 ise: "Spam" sınıfına ağırlık 9, "spam değil"e ağırlık 1 verebilirsiniz. Böylece her bir "spam" hatası, 9 "spam değil" hatasına eşdeğer kayıp yaratır.

- Scikit-learn'de çoğu sınıflandırma algoritması **class_weight="balanced"** parametresi ile otomatik olarak ayarlar. Ya da kendi ağırlıklarınızı **class_weight={0:1, 1:9}** gibi bir sözlükle verebilirsiniz.

Sonuç ve Çıkarımlar

Bu yazıda dengesiz veri setinin sınıflandırma problemlerinde sınıf sayılarının eşit olmama durumunun getirdiği bir sorun olduğunu işledik. Neden ve nasıl problemlere yol açtığını da inceledikten sonra Stratified Sampling, SMOTE ve ağırlıkları ayarlama gibi potansiyel çözüm yollarını gözden geçirdik. Bu yazının çıkarımlarını daha temiz ve yapılandırılmış bir şekilde tabloya döküp görselleştirerek sonuçlandıracak olursak:

Durum	***Öneri***
Veri setiniz çok küçükse	*Stratified K-Fold + Class Weights*
Azınlık sınıfına daha fazla örnek eklemek istiyorsanız	*SMOTE + Stratified Split*
Hızlıca dengeyi korumak istiyorsanız	*Stratified Random Sampling/Shuffle Split*
Modeliniz zaten class_weight destekliyorsa	*Önce ağırlıkları ayarlayıp test edin*

Özetle, dengesiz veri setleriyle mücadele ederken:

1. Önce **sınıf dağılımını gözlemleyin**.
2. **Stratified** yöntemlerle eğitim-test bölünmesini yapın.
3. Gerekirse **SMOTE** ile sentetik örnek ekleyin.
4. **Class weights** ile modelinizi azınlık sınıfa karşı daha duyarlı hale getirin.

Bu adımları izleyerek, dengesizlikten kaynaklanan performans kayıplarını büyük ölçüde azaltabilirsiniz. Kendinize iyi bakın ve dengeyi korumayı unutmayın, kolay gelsin!

Referanslar

- https://medium.com/code-like-a-girl/good-train-test-split-an-approach-to-better-accuracy-91427584b614
- https://miuul.com/blog/dengesiz-veri-seti-ne-zaman-problem-olur
- https://medium.com/@sfkulgun/dengesiz-veri-seti-nedir-84bc4553ecab
- https://www.veribilimiokulu.com/dengesiz-veri-setlerinde-modelleme/
- https://simgeerek.medium.com/dengesiz-veri-setiyle-nas%C4%B1l-ba%C5%9Fa-%C3%A7%C4%B1k%C4%B1l%C4%B1r-dd1f8b5ff99d
- https://www.kaggle.com/code/tugrulyilmaz/dengesiz-veri-setinde-oversampling-y-ntemleri
- https://www.qualtrics.com/experience-management/research/stratified-random-sampling/
- https://www.baeldung.com/cs/ml-stratified-sampling
- Figür 1: https://semaphore.io/blog/imbalanced-data-machine-learning-python
- Figür 2: https://www.evidentlyai.com/classification-metrics/confusion-matrix
- Figür 3: https://www.scribbr.com/methodology/stratified-sampling/