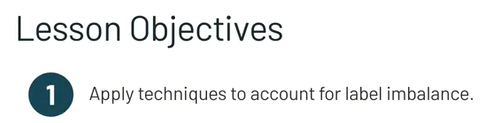
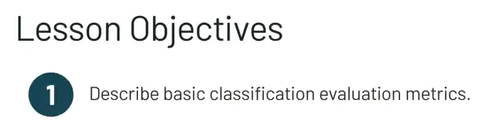
Class Imbalance for Classification

# Lesson Introduction

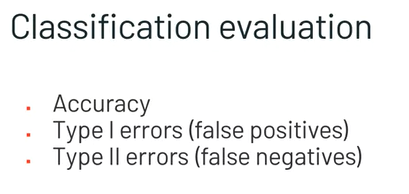


Welcome to lesson 3, label imbalance for classification. [Sınıflandırma için etiket dengesizliği olan 3. derse hoş geldiniz.] In this lesson, we'll be learning how to apply various techniques to account for a label or class imbalance in our data-sets. [Bu derste, veri kümelerimizdeki bir etiket veya sınıf dengesizliğini hesaba katmak için çeşitli teknikleri nasıl uygulayacağımızı öğreneceğiz.] We'll be framing this lesson inside of an imbalanced binary classification setting with random forests where we'll try to optimize for recall, meaning the minority class will be focused on in terms of the model's accuracy. [Bu dersi, geri çağırma için optimize etmeye çalışacağımız rastgele ormanlarla dengesiz bir ikili sınıflandırma ayarı içinde çerçeveleyeceğiz, yani azınlık sınıfı, modelin doğruluğu açısından odaklanacaktır.] This will lead us to making different kinds of decisions to balance out the classes in the label or target variable within the data-set. [Bu, veri kümesindeki etiket veya hedef değişkendeki sınıfları dengelemek için farklı türde kararlar vermemize yol açacaktır.] Before we move on to learning how to account for label imbalances, we'll do a review of classification evaluation metrics in the next video. [Etiket dengesizliklerini nasıl hesaplayacağımızı öğrenmeye geçmeden önce, bir sonraki videoda sınıflandırma değerlendirme metriklerini gözden geçireceğiz.]

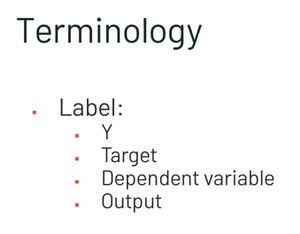
# A Review of Classification Evaluation Metrics



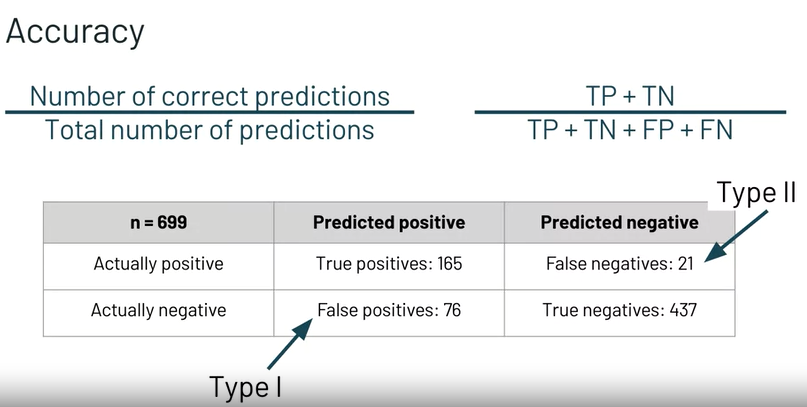
In this video, we'll be reviewing basic classification evaluation metrics that we previously learned about. [Bu videoda, daha önce öğrendiğimiz temel sınıflandırma değerlendirme metriklerini inceleyeceğiz.]



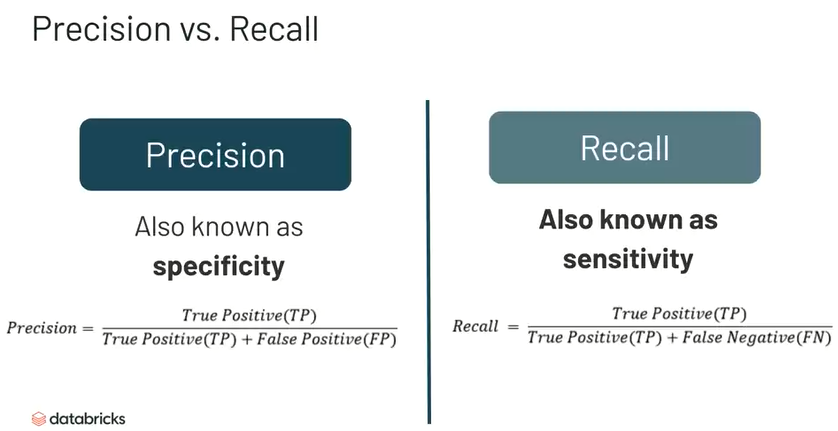
Recall that when we're talking about model accuracy, when it comes to evaluating how well a classification machine learning model performs, accuracy can be overly simplistic at best and misleading at worst. [Model doğruluğundan bahsederken, bir sınıflandırma makinesi öğrenme modelinin ne kadar iyi performans gösterdiğini değerlendirmeye gelince, doğruluğun en iyi ihtimalle aşırı basit ve en kötü ihtimalle yanıltıcı olabileceğini hatırlayın.] This is because simply reporting the number of points the model classified correctly or incorrectly doesn't capture all the information that we need to know and we're interested in. [Bunun nedeni, modelin doğru veya yanlış olarak sınıflandırdığı noktaların sayısını basitçe raporlamanın, bilmemiz gereken ve ilgilendiğimiz tüm bilgileri yakalamamasıdır.] We're usually more interested in whether a model makes more false positive or Type I errors or false negative, Type II errors. [Genellikle bir modelin daha fazla yanlış pozitif mi yoksa Tip I hatalar mı yoksa yanlış negatif Tip II hatalar mı yaptığıyla daha çok ilgileniriz.] These balances between the different types of errors can have serious real-world implications. [Farklı hata türleri arasındaki bu dengelerin gerçek dünyada ciddi sonuçları olabilir.] A classic example of this is with medical testing and diagnosis. [Bunun klasik bir örneği tıbbi test ve teşhistir.] If a healthy patient is classified as falsely positive for having a disease, we run the risk of having potentially costly or inconvenient, maybe uncomfortable and stressful follow-up testing for the patient, as well as the anxiety of having been identified as having a disease. [Sağlıklı bir hasta bir hastalığı olduğu için yanlış pozitif olarak sınıflandırılırsa, hasta için potansiyel olarak maliyetli veya uygunsuz, belki rahatsız edici ve stresli takip testleri ve ayrıca bir hastalığı olduğunun tespit edilmesi endişesi riskiyle karşı karşıya kalırız. .] On the other hand, if a diseased patient is classified as falsely negative for the disease, that could actually cause more health problems or even death. [Öte yandan, hastalıklı bir hasta, hastalık için yanlış negatif olarak sınıflandırılırsa, bu aslında daha fazla sağlık sorununa ve hatta ölüme neden olabilir.] In a case like that, we would want to avoid the false negatives as much as possible, even at the expense of false positive results because we would rather error on the side of caution and incorrectly classify the patient as having a disease and then dealing with the resulting costs of that versus potentially causing a death. [Böyle bir durumda, yanlış pozitif sonuçlar pahasına bile olsa, yanlış negatiflerden mümkün olduğunca kaçınmak isteriz, çünkü ihtiyatlı olmayı tercih ederiz ve hastayı yanlış bir şekilde bir hastalığı var olarak sınıflandırır ve sonra onunla uğraşırız. bunun sonucunda ortaya çıkan maliyetlere karşı potansiyel olarak bir ölüme neden olur.] In other applications like with business problems, the opposite might be true, where having a false positive might be even worse than having a false negative. [İş sorunları gibi diğer uygulamalarda, yanlış bir pozitifin yanlış bir negatiften daha kötü olabileceği durumlarda bunun tersi doğru olabilir.] It all depends on your use case and that's where it's important to have knowledge of the problem you're trying to solve, your objectives, and the domain knowledge that comes into it. [Her şey kullanım durumunuza bağlıdır ve çözmeye çalıştığınız sorun, hedefleriniz ve içine giren alan bilgisi hakkında bilgi sahibi olmanızın önemli olduğu yer burasıdır.]



Some terminology that you may have already heard us use throughout this course for referring to the label, we have used the terms Y, target, dependent variable, or output. [Etikete atıfta bulunmak için bu kurs boyunca kullandığımızı duymuş olabileceğiniz bazı terminolojiler, Y, hedef, bağımlı değişken veya çıktı terimlerini kullandık.] In the world of data science, you'll hear all of these terms used interchangeably. [Veri bilimi dünyasında, bu terimlerin hepsinin birbirinin yerine kullanıldığını duyacaksınız.] Before we go any further into reviewing accuracy and other classification metrics, it's important to remember that all of these terms are referring to the same thing, and in this lesson we'll try to use the term label just for consistency. [Doğruluk ve diğer sınıflandırma ölçütlerini incelemeye devam etmeden önce, tüm bu terimlerin aynı şeye atıfta bulunduğunu hatırlamak önemlidir ve bu derste etiket terimini yalnızca tutarlılık için kullanmaya çalışacağız.]



As you may recall, the definition of accuracy is the number of predictions that the model got right divided by the total number of predictions the model made. [Hatırlayacağınız gibi, doğruluk tanımı, modelin doğru yaptığı tahmin sayısının, modelin yaptığı toplam tahmin sayısına bölünmesidir.] That actually is the same thing as saying true positive plus true negatives divided by true positive plus true negative plus false positive plus false negative. [Bu aslında doğru pozitif artı gerçek negatif bölü gerçek pozitif artı gerçek negatif artı yanlış pozitif artı yanlış negatif demekle aynı şeydir.] When we look at the confusion metrics as shown on the side, we see in this example that there were 699 predictions made, and along the top we have the predicted values, and along the side or the column we have the actual values. [Yan tarafta gösterildiği gibi karışıklık metriklerine baktığımızda, bu örnekte 699 tahmin yapıldığını görüyoruz ve üst kısımda tahmin edilen değerler var ve yan veya sütun boyunca gerçek değerler var.] We see that the top right box of false negatives, where there's 21 false negatives that were predicted, those are the Type II errors, where the bottom left box, the 76 false positives are the Type I errors. [Sağ üstteki yanlış negatif kutusunun, tahmin edilen 21 yanlış negatifin olduğu yerde, bunların Tip II hatalar olduğunu, sol alt kutunun, 76 yanlış pozitifin Tip I hatalar olduğunu görüyoruz.]



You might also remember the ways that we defined precision versus recall. [Ayrıca kesinliğe karşı hatırlamayı nasıl tanımladığımızı da hatırlayabilirsiniz.] Precision is also known as specificity, and the way you calculate it is the number of true positives divided by the true positives and the false positives. [Kesinlik, aynı zamanda özgüllük olarak da bilinir ve bunu hesaplama şekliniz, gerçek pozitiflerin sayısının gerçek pozitiflere ve yanlış pozitiflere bölünmesiyle elde edilir.] Recall is also known as sensitivity, and it can be calculated as the number of true positives divided by the true positives and the false negatives. [Hatırlama, duyarlılık olarak da bilinir ve gerçek pozitiflerin sayısının gerçek pozitiflere ve yanlış negatiflere bölünmesiyle hesaplanabilir.]

# A Review of Assigning Classes

In the following demo video, we will do a review of assigning classes based on their predicted probabilities. [Aşağıdaki demo videoda, tahmin edilen olasılıklarına göre sınıfları atamanın bir incelemesini yapacağız.] The video, will review how to assign the classes and specifically we'll look at the threshold for a certain class and how we have a default value for the threshold but that each model threshold can be independently evaluated and chosen for a particular model. [Video, sınıfların nasıl atanacağını inceleyecek ve özellikle belirli bir sınıfın eşiğine ve eşik için nasıl varsayılan bir değere sahip olduğumuza, ancak her model eşiğinin belirli bir model için bağımsız olarak değerlendirilip seçilebileceğine bakacağız.] First, we will start by preparing our dataset and we'll be using the same user metrics lifestyle table that we've previously created. [İlk olarak, veri setimizi hazırlayarak başlayacağız ve daha önce oluşturduğumuz aynı kullanıcı metrikleri yaşam tarzı tablosunu kullanacağız.] This time in the demo will be attempting to predict which lifestyle category users fall into. [Demoda bu sefer, kullanıcıların hangi yaşam tarzı kategorisine girdiğini tahmin etmeye çalışacak.] We see that we have several lifestyles and we want to know whether each user is into either two of the four categories. [Birkaç yaşam tarzımız olduğunu görüyoruz ve her kullanıcının dört kategoriden ikisine girip girmediğini bilmek istiyoruz.] We'll group sedentary and cardio enthusiasts together, we'll also group athlete and weight trainer together. [Hareketsiz ve kardiyo meraklılarını birlikte gruplayacağız, ayrıca sporcu ve ağırlık antrenörlerini birlikte gruplayacağız.] We're doing this just with a quick list comprehension in Python and using the pandas.DataFrame. [Bunu Python'da hızlı bir liste kavrayışı ve pandas.DataFrame kullanarak yapıyoruz.] Next, we can check how many of each category there are in the dataset and we see that it is roughly balanced. [Ardından, veri setinde her kategoriden kaç tane olduğunu kontrol edebiliyoruz ve kabaca dengeli olduğunu görüyoruz.] We'll go ahead and start building our classification model. [Devam edeceğiz ve sınıflandırma modelimizi oluşturmaya başlayacağız.] Do our train test split after we select what we want our x and y to be and we said we're choosing the lifestyle and numerical as the y category. [X ve y'mizin ne olmasını istediğimizi seçtikten sonra tren testimizi bölüştürün ve y kategorisi olarak yaşam tarzını ve sayısalı seçtiğimizi söyledik.] Then we fit our random forest classifier and we'll examine the output of the model and in this case, we're not looking at the accuracy or how the model performs because we're just looking at the predicted classes. [Sonra rastgele orman sınıflandırıcımıza uyuyoruz ve modelin çıktısını inceleyeceğiz ve bu durumda, modelin doğruluğuna veya nasıl performans gösterdiğine bakmıyoruz çünkü sadece tahmin edilen sınıflara bakıyoruz.] We're going to skip the classification metrics and just look at the predicted classes from the model. [Sınıflandırma ölçütlerini atlayacağız ve sadece modelden tahmin edilen sınıflara bakacağız.] We'll get our DataFrame of the actual lifestyle and the lifestyle that the model predicted. [Modelin öngördüğü gerçek yaşam tarzına ve yaşam tarzına ilişkin DataFrame'imizi alacağız.] It's the two classes, 0 and 1 for lifestyle. [Yaşam tarzı için 0 ve 1 olmak üzere iki sınıf.] Then same thing for the second column, lifestyle predicted, which is what our model predicts for each row. [Sonra ikinci sütun için aynı şey, yaşam tarzı tahmini, modelimizin her satır için öngördüğü şey.] The third column is the predicted probabilities. [Üçüncü sütun, tahmin edilen olasılıklardır.] The way the model determines whether it's class 0 or 1, is if the default threshold of 0.5. [Modelin sınıf 0 mı yoksa 1 mi olduğunu belirleme şekli, varsayılan eşiğin 0,5 olup olmadığıdır.] If that particular sample has a probability of greater than or less than 0.5. [Söz konusu örneğin 0,5'ten büyük veya küçük bir olasılığı varsa.] We see in most of these cases we have 0 percent probability. [Bu vakaların çoğunda yüzde 0 olasılığımız olduğunu görüyoruz.] We have a bunch of 1 for the probabilities, so 100 percent. [Olasılıklar için bir demetimiz var, yani yüzde 100.] Then there's just that 1.95 for the probability. [O zaman olasılık için sadece 1,95 var.] This is pretty straight forward where the model easily determines which of the two classes most of these samples fall into. [Bu, modelin bu örneklerin çoğunun iki sınıftan hangisine girdiğini kolayca belirlediği oldukça basittir.] Again, that's using that default threshold of 0.5, so if it was above 0.5, it would predict class 1. [Yine, bu 0,5'lik varsayılan eşiği kullanıyor, yani 0,5'in üzerinde olsaydı, sınıf 1'i tahmin ederdi.] But remember that we can actually manually adjust that threshold. [Ancak bu eşiği gerçekten manuel olarak ayarlayabileceğimizi unutmayın.] If we want to change how we prioritize precision versus recall, and avoid mistakenly assigning someone to class 0. [Kesinliğe karşı hatırlamaya nasıl öncelik verdiğimizi değiştirmek ve birini yanlışlıkla 0 sınıfına atamaktan kaçınmak istiyorsak.] Even if that means we incorrectly assigned someone to class 1. [Bu, birini yanlışlıkla 1. sınıfa atadığımız anlamına gelse bile.] We can do this by adjusting the probability threshold to 0.7, and that means that in order to assign someone to class 0, we have to tell the model, or we have to evaluate the model's results and say, the probability has to be greater than 70 percent for us to conclude that that person belongs in that category or that class. [Bunu, olasılık eşiğini 0,7'ye ayarlayarak yapabiliriz ve bu, birini sınıf 0'a atamak için modeli söylememiz gerektiği veya modelin sonuçlarını değerlendirmemiz ve diyelim ki, olasılığın şundan büyük olması gerektiği anlamına gelir. O kişinin o kategoriye veya o sınıfa ait olduğu sonucuna varmamız için yüzde 70.] We do this here, again using a simple list comprehension and Python, where we set the threshold. [Bunu burada yine basit bir liste kavrayışı ve eşiği belirlediğimiz Python kullanarak yapıyoruz.] If x is greater than 0.7 for every x or row in the DataFrame with the predicted probabilities. [Tahmin edilen olasılıklarla DataFrame'deki her x veya satır için x 0,7'den büyükse.] Then we get this new DataFrame that has an additional column, lifestyle predicted, adjusted, and that's anywhere where the probability was greater than 0.7. [Ardından, ek bir sütunu olan, yaşam tarzının tahmin edildiği, düzeltildiği ve olasılığın 0,7'den büyük olduğu her yerde bu yeni DataFrame'i elde ederiz.] We will predict 0, and if it's less than 0.7, we predict 1. [0 tahmin edeceğiz ve 0,7'den küçükse 1 tahmin edeceğiz.]

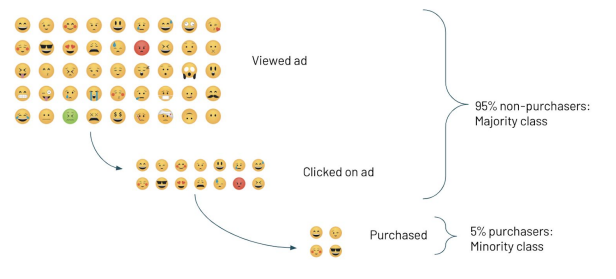
# Problems with Class Imbalance

Often, a dataset will have many more cases of one class label than the other. This is known as class imbalance, and is very common. An example of this is customers who complete a sale after viewing a social media ad. Most of the people who view the ad don’t click on the link, and of the few who click, most don’t end up actually purchasing the product. Therefore, if you are trying to build a model to classify which viewers will purchase the product, you will have to address this class imbalance issue.

Çoğu zaman, bir veri kümesinde bir sınıf etiketinin diğerinden çok daha fazla durumu bulunur. Bu, sınıf dengesizliği olarak bilinir ve çok yaygındır. Bunun bir örneği, bir sosyal medya reklamını görüntüledikten sonra bir satışı tamamlayan müşterilerdir. Reklamı görüntüleyen kişilerin çoğu bağlantıya tıklamıyor ve tıklayanların çoğu aslında ürünü satın almıyor. Bu nedenle, ürünü hangi izleyicilerin satın alacağını sınıflandırmak için bir model oluşturmaya çalışıyorsanız, bu sınıf dengesizliği sorununu çözmeniz gerekecektir.

To illustrate why this is a problem, imagine that 95% of viewers don’t purchase the product. If you built a model that classifies viewers into non purchasers and purchasers, and got 95% accuracy, you and your boss might think that sounds like a good model. But as a savvy data scientist, you know that if you didn’t even build a model and just predicted every viewer as a non purchaser, you would be accurate 95% of the time! Therefore your 95% accuracy just went from good to garbage.

Bunun neden bir sorun olduğunu göstermek için, izleyicilerin %95'inin ürünü satın almadığını hayal edin. İzleyicileri alıcı olmayanlar ve alıcılar olarak sınıflandıran bir model oluşturduysanız ve %95 doğruluk elde ettiyseniz, siz ve patronunuz bunun kulağa iyi bir model gibi geldiğini düşünebilirsiniz. Ancak bilgili bir veri bilimcisi olarak, bir model bile oluşturmasaydınız ve her izleyiciyi alıcı olmayan biri olarak tahmin etseydiniz, zamanın %95'inde doğru olacağınızı bilirsiniz! Bu nedenle, %95 doğruluğunuz iyiden çöpe gitti.



Accuracy is clearly not an adequate metric to evaluate a model with an imbalanced data set. In fact, precision and recall give us a better understanding of how a model performs on an imbalance data set. Looking at the confusion matrix can also help to understand the model’s predictions.

Doğruluk, dengesiz bir veri kümesine sahip bir modeli değerlendirmek için açıkça yeterli bir ölçü değildir. Aslında, kesinlik ve hatırlama, bir modelin bir dengesizlik veri setinde nasıl performans gösterdiğini daha iyi anlamamızı sağlar. Karışıklık matrisine bakmak, modelin tahminlerini anlamaya da yardımcı olabilir.

When you have a very imbalanced dataset, you may have very few samples of the minority class. This makes it hard for algorithms to learn patterns, or recognize and correctly classify new samples. It simply hasn’t seen enough of the minority class to understand when to predict it. Fortunately, there are techniques that address the label imbalance problem. We will learn about these in the next videos.

Çok dengesiz bir veri kümeniz olduğunda, azınlık sınıfından çok az örneğiniz olabilir. Bu, algoritmaların kalıpları öğrenmesini veya yeni örnekleri tanımasını ve doğru şekilde sınıflandırmasını zorlaştırır. Azınlık sınıfının bunu ne zaman tahmin edeceğini anlayacak kadarını görmedi. Neyse ki, etiket dengesizliği sorununu ele alan teknikler var. Bunları sonraki videolarda öğreneceğiz.

# Label-based Bootstrap Sampling

One important way to deal with a dataset that has imbalance classes is to use bootstrap sampling, like we talked about in the last lesson, to artificially enhance the dataset so that the classes are more balanced.

There are two main methods that can be used to do this:

You can increase instances from the minority class. This is called over-sampling (or sampling with replacement).

You can remove instances from the majority class, called under-sampling.

There are also methods that combine these two methods.

Over sampling works better if you have a smaller dataset and cannot afford to remove any samples, while undersampling can work if you have a large dataset.

One thing to note is that you don’t necessarily need to achieve a perfect 1 to 1 ratio between the classes. You can try out different methods and see what works to get better model performance.

Besides these sampling methods, there are also ways to generate synthetic data samples by randomly sampling the attributes from instances in the minority class. There are algorithms that are specifically designed to generate such synthetic samples. The most commonly used is called SMOTE or the Synthetic Minority Over-sampling Technique.

SMOTE works by oversampling, where it creates synthetic samples from the minority class instead of creating copies like traditional oversampling methods.

Dengesiz sınıflara sahip bir veri kümesiyle başa çıkmanın önemli bir yolu, son derste bahsettiğimiz gibi, sınıfların daha dengeli olması için veri kümesini yapay olarak geliştirmek için önyükleme örneklemesi kullanmaktır. Bunu yapmak için kullanılabilecek iki ana yöntem vardır: Azınlık sınıfından örnekleri artırabilirsiniz. Buna aşırı örnekleme (veya değiştirme ile örnekleme) denir. Yetersiz örnekleme adı verilen çoğunluk sınıfından örnekleri kaldırabilirsiniz. Bu iki yöntemi birleştiren yöntemler de vardır. Daha küçük bir veri kümeniz varsa ve herhangi bir örneği kaldırmayı göze alamıyorsanız, aşırı örnekleme daha iyi sonuç verirken, büyük bir veri kümeniz varsa yetersiz örnekleme işe yarayabilir. Unutulmaması gereken bir şey, sınıflar arasında mutlaka 1'e 1 oranında mükemmel bir oran elde etmenize gerek olmadığıdır. Farklı yöntemler deneyebilir ve daha iyi model performansı elde etmek için neyin işe yaradığını görebilirsiniz. Bu örnekleme yöntemlerinin yanı sıra, azınlık sınıfındaki örneklerden öznitelikleri rastgele örnekleyerek sentetik veri örnekleri üretmenin yolları da vardır. Bu tür sentetik örnekleri oluşturmak için özel olarak tasarlanmış algoritmalar vardır. En yaygın olarak kullanılanı SMOTE veya Sentetik Azınlık Aşırı Örnekleme Tekniği olarak adlandırılır. SMOTE, geleneksel aşırı örnekleme yöntemleri gibi kopyalar oluşturmak yerine azınlık sınıfından sentetik örnekler oluşturduğu aşırı örnekleme ile çalışır.

Oversampling and Undersampling Classes

In the following video, we'll demonstrate how to bootstrap records based on their label values. [Aşağıdaki videoda, kayıtların etiket değerlerine göre nasıl önyükleneceğini göstereceğiz.] And this is where we will be talking about having an imbalance data set between classes. [Ve burada sınıflar arasında bir dengesizlik veri setinden bahsedeceğiz.] And how we can use the bootstrap to oversample or undersample to make sure that we get a more balanced distribution of classes in our bootstrapped training data set. [Ve önyüklemeli eğitim veri kümemizde daha dengeli bir sınıf dağılımı elde ettiğimizden emin olmak için önyüklemeyi aşırı veya eksik örneklemek için nasıl kullanabiliriz.]

We'll start with using the same lifestyle metrics table that we've previously been using. [Daha önce kullandığımız yaşam tarzı metrikleri tablosunu kullanarak başlayacağız.] And we'll look at what the unique lifestyles that are in the data set. [Ve veri setindeki benzersiz yaşam tarzlarının neler olduğuna bakacağız.]

We see we have four categories or classes, and in this demo we want to know whether each user is sedentary or active. [Dört kategori veya sınıfımız olduğunu görüyoruz ve bu demoda her kullanıcının hareketsiz mi yoksa aktif mi olduğunu bilmek istiyoruz.] So sedentary means they're in the sedentary class, and active means they are in any of the other three lifestyle classes. [Yani hareketsiz, hareketsiz sınıfta oldukları anlamına gelir ve aktif, diğer üç yaşam tarzı sınıfından herhangi birinde oldukları anlamına gelir.] So we'll convert the four categories into two numeric classes, zero and one. [Bu yüzden dört kategoriyi sıfır ve bir olmak üzere iki sayısal sınıfa dönüştüreceğiz.] And we'll do this in command seven with a list comprehension and Python and the Panda's data frame. [Ve bunu yedinci komutta liste kavrayışı ve Python ve Panda'nın veri çerçevesi ile yapacağız.] Where we're just manually saying, if the sample is sedentary, assign it a zero in this new lifestyle numb column. [Sadece manuel olarak söylediğimiz yerde, örnek hareketsizse, bu yeni yaşam tarzı uyuşukluk sütununda ona sıfır atayın.] Otherwise, assign it a one for the value for that column. [Aksi takdirde, o sütunun değeri için bir tane atayın.] And then we see that we have an imbalanced set of 2,688 of the class one. [Ve sonra, birinci sınıf 2,688'lik dengesiz bir setimiz olduğunu görüyoruz.] So the non sedentary and 312 of class zero or sedentary. [Yani sedanter olmayan ve 312 sınıfı sıfır veya sedanter.] We'll separate these out into an X and Y. [Bunları X ve Y olarak ayıracağız.] And now that we have this data set with a slightly imbalanced target class, where Class one is almost ten times bigger than class zero. [Ve şimdi, birinci sınıfın sıfırdan neredeyse on kat daha büyük olduğu, biraz dengesiz bir hedef sınıfla bu veri setine sahibiz.]

Which isn't a very major imbalance relatively speaking, but will still attempt to balance this data set using the bootstrap method. [Bu göreceli olarak çok büyük bir dengesizlik değil, ancak yine de bu veri kümesini önyükleme yöntemini kullanarak dengelemeye çalışacak.]

We're going to perform random over sampling, which means we'll randomly duplicate examples from the minority class by sampling them with replacement. [Rastgele örnekleme üzerinden örnekleme yapacağız, bu da azınlık sınıfından örnekleri ikame ile örnekleyerek rasgele çoğaltacağımız anlamına geliyor.] In order to do this, we're going to install the Imbalanced Learn Library, which is compatible with psychic learn. [Bunu yapmak için, psişik öğrenme ile uyumlu olan Imbalanced Learn Library'yi kuracağız.] And we can do this on our data bricks workspace using the percent sign and pip install over the library. [Ve bunu, kütüphane üzerinden yüzde işareti ve pip kurulumunu kullanarak veri tuğlaları çalışma alanımızda yapabiliriz.] We're going the import the random over sampler function from the over sampling library. [Örnekleme kitaplığından rastgele örnekleyici işlevini içe aktarıyoruz.] And we will instantiate an instance of the random over sampler using a sampling strategy of minority. [Ve bir azınlık örnekleme stratejisi kullanarak rastgele örnekleyicinin bir örneğini başlatacağız.]

Now that we've done that, we can fit and apply the over sample transformation that we just built. [Artık bunu yaptığımıza göre, az önce inşa ettiğimiz over sample dönüşümünü sığdırabilir ve uygulayabiliriz.] And we're going to make a new X and Y data set called X over and Y over by doing a fit resample. [Ve uygun bir yeniden örnekleme yaparak X over ve Y over adında yeni bir X ve Y veri seti oluşturacağız.] And now that we've done that, we can look at the new value counts. [Ve şimdi bunu yaptığımıza göre, yeni değer sayılarına bakabiliriz.] And we see that it's exactly balanced between class zero and class one in our new over sampled data set. [Ve yeni aşırı örneklenmiş veri setimizde sıfırıncı sınıf ile birinci sınıf arasında tam olarak dengeli olduğunu görüyoruz.] Next, we will look at undersampling the majority class, which is another technique to use the bootstrap. [Daha sonra, önyüklemeyi kullanmak için başka bir teknik olan çoğunluk sınıfının alt örneklenmesine bakacağız.] And this operates by randomly deleting examples from the majority class. [Ve bu, çoğunluk sınıfından örnekleri rastgele silerek çalışır.] We set our sampling strategy to majority instead of minority with the random under sampler. [Rastgele örnekleyici ile örnekleme stratejimizi azınlık yerine çoğunluk olarak belirledik.] And this will make our majority class the same size as the minority class. [Ve bu, çoğunluk sınıfımızı azınlık sınıfıyla aynı boyutta yapacaktır.]

Obviously, if you had a very small data set, especially a small training data set, you wouldn't want to do this. [Açıkçası, çok küçük bir veri setiniz olsaydı, özellikle küçük bir eğitim veri setiniz olsaydı, bunu yapmak istemezdiniz.] And we see here that we now have 300 approximately samples in each of the two classes. [Ve burada, iki sınıfın her birinde yaklaşık 300 örneğimiz olduğunu görüyoruz.] But in this case, we would probably want to use the minority over sampler, so that we have a larger data set. [Ancak bu durumda, daha büyük bir veri setimiz olması için muhtemelen örnekleyici üzerinde azınlık kullanmak isteriz.]

We can also specify a ratio instead of just saying that we want the two classes to be the same size. [İki sınıfın aynı boyutta olmasını istediğimizi söylemek yerine bir oran da belirtebiliriz.] So for our sampling strategy, we could just put in a number. [Yani örnekleme stratejimiz için sadece bir sayı koyabiliriz.] In this case, we're going to do 0.75 to say that we want the majority class to be 0.75 of the number of minority class. [Bu durumda, çoğunluk sınıfının azınlık sınıfı sayısının 0.75'i olmasını istediğimizi söylemek için 0.75 yapacağız.] So when we look at, fitting that to our X and Y, we now see that we get these value counts where the majority class is or the minority class is 0.75% compared with the majority class. [Dolayısıyla, bunu X ve Y'mize uydurarak baktığımızda, çoğunluk sınıfının olduğu veya azınlık sınıfının çoğunluk sınıfına kıyasla %0,75 olduğu bu değer sayılarını elde ettiğimizi görüyoruz.] So now we have a slightly more balanced data set. [Şimdi biraz daha dengeli bir veri setimiz var.]

# Label-based Evaluation Weighting

One of the simplest ways to deal with a class imbalance in a dataset is to apply a weight to each class in the cost function. The minority class will be weighted more heavily than the majority class in the cost function, so that the model ends up placing equal emphasis on both classes while it is learning.

Bir veri kümesindeki sınıf dengesizliğiyle başa çıkmanın en basit yollarından biri, maliyet işlevinde her sınıfa bir ağırlık uygulamaktır. Azınlık sınıfı, maliyet fonksiyonunda çoğunluk sınıfından daha ağır bir şekilde ağırlıklandırılacaktır, böylece model öğrenirken her iki sınıfa da eşit önem vererek sona erer.

There are a few ways to accomplish this. The weightings can be calculated manually as part of the cost function itself, so that the minority class is given a higher relative weight and the majority class is given a lower relative weight.

Bunu başarmanın birkaç yolu vardır. Ağırlıklar, maliyet fonksiyonunun bir parçası olarak manuel olarak hesaplanabilir, böylece azınlık sınıfına daha yüksek bir göreli ağırlık verilir ve çoğunluk sınıfına daha düşük bir göreli ağırlık verilir.

In addition, some classification models have built-in methods for dealing with imbalanced classes

Ek olarak, bazı sınıflandırma modellerinde dengesiz sınıflarla başa çıkmak için yerleşik yöntemler bulunur.

In scikit-learn, many classification models have a class\_weight parameter that can be set to achieve balanced classes. The classes are automatically weighted inversely proportional to how frequently they appear in the data set

Scikit-learn'de birçok sınıflandırma modeli, dengeli sınıflar elde etmek için ayarlanabilen bir sınıf\_ağırlık parametresine sahiptir. Sınıflar, veri setinde ne sıklıkta göründükleri ile ters orantılı olarak otomatik olarak ağırlıklandırılır.

Wj = n/knj

where Wj is the weight to class j,

n is the number of observations,

nj is the number of observations in class j,

and k is the total number of classes.

Sklearn also has a built-in utility function that will calculate weights based on class frequencies: sklearn.utils.

class\_weight.compute\_class\_weight

There are a few parameters that can be passed to the compute class weight function. You can specify ‘balanced’, so that the function will calculate the class weights for you. You can also input a dictionary where the keys are the classes and the values are the desired class weights. If None is given, the class weights will be uniform.

Hesaplama sınıfı ağırlık işlevine geçirilebilecek birkaç parametre vardır. Fonksiyonun sizin için sınıf ağırlıklarını hesaplaması için 'dengeli' belirtebilirsiniz. Anahtarların sınıflar ve değerlerin istenen sınıf ağırlıkları olduğu bir sözlük de girebilirsiniz. Hiçbiri verilirse, sınıf ağırlıkları tek tip olacaktır.

# Weighting Classes in Random Forest

In this demo will be looking at how to perform record weighting based on the class distribution in our training data set. [Bu demoda, eğitim veri setimizde sınıf dağılımına dayalı olarak rekor ağırlıklandırmanın nasıl gerçekleştirileceğine bakılacaktır.] Were doing this in order to achieve equal waiting of the labeled classes when we're evaluating models. [Modelleri değerlendirirken etiketli sınıfların eşit beklemesini sağlamak için bunu yapıyorduk.] And we'll be doing this with a random forest model. [Ve bunu rastgele bir orman modeliyle yapacağız.] I'm going to skip quickly pass the data preparation step and we will get our lifestyle table that we were previously using, but this time we're also joining it to the HT\_userstable because in this example we want to look at what country our user is from, and so we get that data from that users table. [Hızlıca veri hazırlama adımını geçeceğim ve daha önce kullandığımız yaşam tarzı tablomuzu alacağız, ancak bu sefer onu HT\_userstable'a da katıyoruz çünkü bu örnekte kullanıcımızın hangi ülkede olduğuna bakmak istiyoruz. kimdendir ve bu nedenle bu verileri o kullanıcılar tablosundan alırız.] Okay, let's take a look at our data frame. [Tamam, veri çerçevemize bir göz atalım.] And we see it looks the same as before, but now we have the country column in the last column. [Ve eskisi gibi göründüğünü görüyoruz, ancak şimdi son sütunda ülke sütunumuz var.] Let's check and see how many of each of the country classes there are using value counts. [Değer sayımlarını kullanan ülke sınıflarının her birinin kaç tane olduğunu kontrol edelim ve görelim.] And we see that there is United States and Canada that's the only two countries represented in the data set and based on the value counts, it looks like a pretty imbalanced data set where the United States is over represented in our data and Canada is a much smaller sample. [Ve görüyoruz ki, veri setinde temsil edilen sadece iki ülke olan Amerika Birleşik Devletleri ve Kanada var ve değer sayılarına dayanarak, Amerika Birleşik Devletleri'nin verilerimizde fazla temsil edildiği ve Kanada'nın çok fazla temsil edildiği oldukça dengesiz bir veri seti gibi görünüyor. daha küçük örnek] Now we're going to encode the categories for the two classes of country, and I'm just taking a look at the head of the data frame to see which class label it's using for United States versus Canada. [Şimdi iki ülke sınıfı için kategorileri kodlayacağız ve Amerika Birleşik Devletleri ve Kanada için hangi sınıf etiketini kullandığını görmek için veri çerçevesinin başına bir göz atacağım.] So because we see one listed for United States, we know that United States has been encoded as one. [Amerika Birleşik Devletleri için listelenmiş bir tane gördüğümüz için, Amerika Birleşik Devletleri'nin bir olarak kodlandığını biliyoruz.] And we can assume that Canada has been encoded as zero. [Ve Kanada'nın sıfır olarak kodlandığını varsayabiliriz.] Okay, now will prepare our X and Y data, we just dropping some of the columns that we don't need as well as the target label columns of country and country category from our X. [Tamam, şimdi X ve Y verilerimizi hazırlayacağız, sadece ihtiyacımız olmayan bazı sütunları ve ayrıca ülke ve ülke kategorisinin hedef etiket sütunlarını X'imizden atıyoruz.] And then Y is going to be the categorical country column. [Ve sonra Y kategorik ülke sütunu olacak.] Now we'll train our random forest model and this time we're going to use class weights as we learned in one of the videos, scikit learn has a built in utility function that can calculate weights for classes based on the class frequencies in the data set. [Şimdi rasgele orman modelimizi eğiteceğiz ve bu sefer videolardan birinde öğrendiğimiz gibi sınıf ağırlıklarını kullanacağız, scikit öğrenme, sınıf frekanslarına göre sınıfların ağırlıklarını hesaplayabilen yerleşik bir yardımcı fonksiyona sahiptir. veri seti.] It does this by automatically weighting the classes inversely proportional to how frequently they appear in the data. [Bunu, verilerde ne sıklıkta göründükleri ile ters orantılı olarak sınıfları otomatik olarak ağırlıklandırarak yapar.] So that means that the minority class will be weighted more heavily than the majority class. [Bu, azınlık sınıfının çoğunluk sınıfından daha fazla ağırlıklı olacağı anlamına gelir.] We can use the class weight function as a parameter in a model, and there's several options that we can set, so the default is always none where the class weights will be uniform and won't be altered based on class frequencies. [Sınıf ağırlığı işlevini bir modelde parametre olarak kullanabiliriz ve ayarlayabileceğimiz birkaç seçenek vardır, bu nedenle sınıf ağırlıklarının tek tip olacağı ve sınıf frekanslarına göre değiştirilmediği durumlarda varsayılan her zaman hiçbiridir.] This is just how we always have instantiated our models before. [Bu, modellerimizi daha önce hep böyle somutlaştırmıştık.] We can also specify to set the class weights as balanced where this function will calculate the class weights automatically and incorporate those weights into the cost function for the model. [Bu fonksiyonun sınıf ağırlıklarını otomatik olarak hesaplayacağı ve bu ağırlıkları model için maliyet fonksiyonuna dahil edeceği yerde sınıf ağırlıklarını dengeli olarak ayarlamayı da belirtebiliriz.] There's also a balanced subsample option, which is pretty much the same thing as the balanced option, except that the weights are computed for each individual tree based on that bootstrap sample. [Dengeli bir alt örnek seçeneği de vardır; bu, ağırlıkların her bir ağaç için o önyükleme örneğine dayalı olarak hesaplanması dışında, dengeli seçenekle hemen hemen aynı şeydir.] So instead of computing them overall for the model, it does that at each individual tree. [Dolayısıyla, bunları model için genel olarak hesaplamak yerine, bunu her bir ağaçta yapar.] And then finally we can specify a dictionary with our own class weights that we determine where we set the keys to the class labels and the values are the class weights that we want each of those labels to have. [Ve son olarak, sınıf etiketlerinin anahtarlarını nereye ayarladığımızı belirlediğimiz kendi sınıf ağırlıklarımızla bir sözlük belirleyebiliriz ve değerler, bu etiketlerin her birinin sahip olmasını istediğimiz sınıf ağırlıklarıdır.] Alright, so we'll do our usual data preparation for modeling sliding into the training and test set, and we're going to look at the value counts for the Y training and test sets, and this gives us how many of each class there are in those sets. [Pekala, bu yüzden eğitim ve test setine kayan modelleme için olağan veri hazırlığımızı yapacağız ve Y eğitim ve test setleri için değer sayılarına bakacağız ve bu bize orada her sınıftan kaç tane olduğunu verir. bu setlerde var.] It's just a good way to check whether you actually got enough of the minority class in each of these splits, so it looks like we did get some of costero which is Canada in each of the sets, so we're good to go. [Bu, bu bölümlerin her birinde azınlık sınıfından yeterince alıp almadığınızı kontrol etmenin iyi bir yolu, bu yüzden her sette Kanada olan bir miktar costero aldık, bu yüzden gitmeye hazırız.] And now we're going to fit our first random forest classifier model with no class weight specified. [Ve şimdi, sınıf ağırlığı belirtilmeden ilk rastgele orman sınıflandırıcı modelimize uyacağız.] Ordinarily, we wouldn't bother to type in this class weight parameter, but since we're focusing on this aspect of class weights in the demo will put it in there just to make it really explicit. [Normalde, bu sınıf ağırlığı parametresini yazmaya zahmet etmeyiz, ancak sınıf ağırlıklarının bu yönüne odaklandığımız için demoda bunu gerçekten açık hale getirmek için oraya koyacağız.] Okay, and we see from the confusion matrix the top left number zero here is the number of samples that were actually. of class zero and were predicted a being class zero. [Tamam, ve karışıklık matrisinden görüyoruz ki, burada sol üstteki sıfır sayısı, gerçekte olan örneklerin sayısıdır. sıfır sınıfı ve sıfır sınıfı olduğu tahmin edildi.] So that's not very good. [Yani bu pek iyi değil.] And we see that all 16 of the samples that were Class Zero the model is predicting that are class one on. [Ve modelin öngördüğü Sınıf Sıfır olan 16 örneğin hepsinin birinci sınıf olduğunu görüyoruz.] That's what we would be expecting with this imbalance data set. [Bu dengesizlik veri setinden beklediğimiz şey bu.] And then the model is doing really well on the class. [Ve sonra model sınıfta gerçekten iyi gidiyor.] Won the majority class. [Çoğunluk sınıfını kazandı.] So we can say that the recall for the minority classes very poor because the model is not catching any of the minority class. [Dolayısıyla azınlık sınıfları için geri çağırmanın çok zayıf olduğunu söyleyebiliriz çünkü model azınlık sınıflarından hiçbirini yakalayamıyor.] It's not classifying any of them correctly. [Hiçbirini doğru bir şekilde sınıflandırmıyor.]

Next, weaken in Stan. [Ardından, Stan'de zayıflayın.] She ate a random forced classifier, setting the class wait to balanced and see how it does basically the exact same result. [Rastgele bir zorunlu sınıflandırıcı yedi, sınıf beklemesini dengeli hale getirdi ve temelde aynı sonucu nasıl yaptığını gördü.] So we're seeing that setting the class weight balance so far isn't really having much impact. [Dolayısıyla, şimdiye kadar sınıf ağırlık dengesini ayarlamanın pek bir etkisi olmadığını görüyoruz.] This might be because the data set is relatively small. [Bunun nedeni, veri setinin nispeten küçük olması olabilir.]

We'll also try the balanced sub sample option and exact same thing for that. [Ayrıca dengeli alt örnek seçeneğini deneyeceğiz ve bunun için tamamen aynı şeyi yapacağız.] Next, we'll try setting a dictionary of ratios. [Ardından, bir oranlar sözlüğü oluşturmayı deneyeceğiz.] We can calculate the exact ratio we want to use in order to evenly balanced the classes based on their frequency in the data set, and then we can use that in our class Wait Dictionary that we supply as an option to the class. [Veri setindeki sıklıklarına göre sınıfları eşit olarak dengelemek için kullanmak istediğimiz oranı tam olarak hesaplayabilir ve bunu sınıfa bir seçenek olarak sunduğumuz Bekle Sözlüğü'nde kullanabiliriz.] Wait parameter there is an SK learn you tills class weight function that will calculate that exactly show for us so we won't import the class weight function from SK learned you tills library and we're going to create this value or variable weights calling the class wait Compute class weight function where we say that we want the class Wait, it computes to be balanced We're giving it the labels of the classes and we're setting our why data set to be the why data set that we created. [Bekle parametresi var, bir SK öğreniyor sınıf ağırlığı fonksiyonu var, bunu tam olarak bizim için hesaplayacak, bu yüzden sınıf ağırlığı fonksiyonunu SK'den öğrenilenler kütüphanesinden içe aktarmayacağız ve bu değeri veya değişken ağırlıkları çağırarak yaratacağız. class wait sınıfı istediğimizi söylediğimiz sınıf ağırlığını hesapla fonksiyonu Bekle, dengelenecek hesaplıyor Sınıfların etiketlerini veriyoruz ve neden veri setimizi oluşturduğumuz neden veri seti olacak şekilde ayarlıyoruz.] So it's just started labels from the original data set and we can look at the weights that it calculated for each class. [Yani orijinal veri setinden etiketler yeni başladı ve her sınıf için hesapladığı ağırlıklara bakabiliriz.] So we see that class zero is waited at about 20 whereas class one is waited at about 0.5. [Yani sıfır sınıfının yaklaşık 20'de, birinci sınıfın ise yaklaşık 0,5'te beklendiğini görüyoruz.] So this is giving us a higher weight for the minority class and a smaller wait for the majority class. [Yani bu bize azınlık sınıfı için daha fazla ağırlık ve çoğunluk sınıfı için daha küçük bir bekleme süresi veriyor.]

Now that we have this weights array, we can create a dictionary of those weights. [Artık bu ağırlıklar dizisine sahip olduğumuza göre, bu ağırlıkların bir sözlüğünü oluşturabiliriz.] That will be labeled zero and one and enumerates is the function that will give us those zero and one, index values for each of the weights. [Bu, sıfır ve bir olarak etiketlenecek ve numaralandırılacak, bize bu sıfır ve bir, ağırlıkların her biri için indeks değerlerini verecek olan fonksiyondur.] And then we use the python function dicked thio, create a dictionary. [Ve sonra python işlevi thio dicked kullanırız, bir sözlük oluştururuz.]

So now we see it's taken thes weights and assigned each of them a numerical index. [Şimdi ağırlıkları aldığını ve her birine sayısal bir indeks atadığını görüyoruz.] So that corresponds to our class labels zero and one. [Yani bu, sıfır ve bir sınıf etiketlerimize karşılık gelir.]

Next, when we in Stan, she ate the random forced classifier. [Sonra, Stan'deyken rastgele zorunlu sınıflandırıcıyı yedi.] We'll set the class weight equal to this class weights dictionary. [Sınıf ağırlığını bu sınıf ağırlıkları sözlüğüne eşitleyeceğiz.] We could also used the actual dictionary itself, and we could just copy paste that in here instead of using the variable name. [Gerçek sözlüğün kendisini de kullanabilirdik ve değişken adını kullanmak yerine onu buraya kopyalayıp yapıştırabiliriz.] And unfortunately, we see that this method isn't working very well either. [Ve ne yazık ki bu yöntemin de pek işe yaramadığını görüyoruz.] We still have very low recall for the minority class. [Azınlık sınıfı için hala çok düşük bir hatırlama oranına sahibiz.] So now, just just out of curiosity, we can try setting really high in really low class weights just to see what happens. [Şimdi, sırf meraktan, ne olduğunu görmek için gerçekten düşük sınıf ağırlıklarında gerçekten yüksek ayarlamayı deneyebiliriz.] So in this case, I'm actually I manually typed in this dictionary where we have, 99,999 for the class, wait for zero and a very small number for the class. [Yani bu durumda, ben aslında bu sözlüğe manuel olarak yazdım, burada sınıf için 99,999, sınıf için sıfır bekleyin ve çok küçük bir sayı var.] Wait for one. [Birini bekleyin.] And now we see that we've actually gotten excellent recall on the minority class because out of the 16 samples of that class. [Ve şimdi, azınlık sınıfında, o sınıfın 16 örneğinden dolayı gerçekten mükemmel bir hatırlama elde ettiğimizi görüyoruz.] The model correctly predicted all of them. [Model hepsini doğru tahmin etti.] But the precision has gone down, so the number of positively identified of the majority class has gone down a lot. [Ancak kesinlik azaldı, bu nedenle çoğunluk sınıfının pozitif olarak tanımlanan sayısı çok düştü.] So the model predicted class zero for 293 instances of Class One, and we can try. [Böylece model, Sınıf Bir'in 293 örneği için sıfır sınıfını öngördü ve deneyebiliriz.] Maybe we'll lower these numbers a little bit, make them a little less extreme and see what happens. [Belki bu sayıları biraz düşürürüz, biraz daha az aşırı hale getiririz ve ne olacağını görürüz.] So now we're back kind of where we started so we could potentially play around with us a little bit. [Şimdi başladığımız yere geri döndük, böylece potansiyel olarak bizimle biraz oynayabiliriz.] But given that it's Seems that the balance between the precision and recall is tricky to figure out here with the classmates. [Ancak, kesinlik ve hatırlama arasındaki dengeyi burada sınıf arkadaşlarıyla çözmenin zor olduğu görülüyor.] We would probably then move on to trying different methods with our modeling. [Muhtemelen daha sonra modellememizle farklı yöntemler denemeye devam edeceğiz.] So now that you've seen some of the ways you can use the built in class weight function, we'll look at how a model would perform on this data set. [Artık yerleşik sınıf ağırlığı işlevini kullanmanın bazı yollarını gördüğünüze göre, bir modelin bu veri kümesinde nasıl performans göstereceğine bakacağız.] If we actually balance the classes for the data set before doing any modeling. [Herhangi bir modelleme yapmadan önce veri seti için sınıfları gerçekten dengelersek.] So instead of using the class weights, were going to go back to what we did in a previous demo. [Yani sınıf ağırlıklarını kullanmak yerine, önceki bir demoda yaptığımız şeye geri dönecektik.] Where we are going to oversample the minority class, using the imbalanced learn library and the random over sampler function. [Dengesiz öğrenme kitaplığını ve rastgele örnekleyici işlevini kullanarak azınlık sınıfını aşırı örnekleyeceğimiz yer.] And now when we examine the value counts for ry, we see that we now have a completely balanced data set of about almost 3000 samples for each class. [Ve şimdi ry için değer sayımlarını incelediğimizde, her sınıf için yaklaşık 3000 örnekten oluşan tamamen dengeli bir veri setimiz olduğunu görüyoruz.] And now if we train a random for this classifier on that data set, we see that we're actually getting really good results. [Ve şimdi, bu sınıflandırıcı için o veri setinde rastgele bir eğitirsek, gerçekten iyi sonuçlar aldığımızı görüyoruz.] Where all of the minority class was predicted correctly and almost all of the majority class was predicted correctly. [Azınlık sınıfının tamamının doğru tahmin edildiği ve çoğunluk sınıfının neredeyse tamamının doğru tahmin

# Label Imbalance Lab

Next, you'll complete a lab where you’ll apply label balancing techniques to a random forest to optimize a recall metric using Databricks.

To do so:

1. Log in to your Databricks Community Edition account.

2. Locate your ADSDA folder.

3. Open Module 4.

4. Click on 4.3.4 Lab - Label Imbalance to open your notebook.

Note: If you haven’t already imported the course DBC file, review instructions in Module 1's "Hands-on with Databricks" video.

Remember that if you need help with this lab, the solution to the lab has been included in the course materials file.