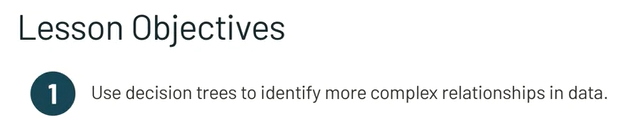
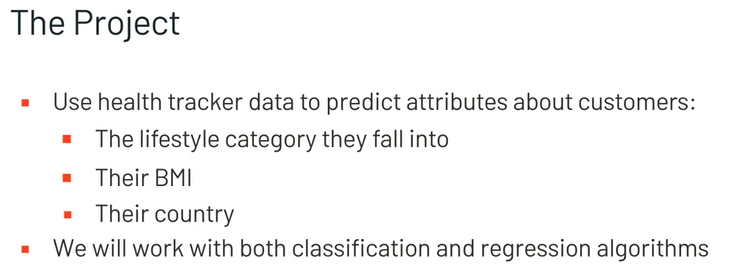
Applied Decision Trees

Lesson Introduction

Welcome to Lesson one of applied tree-based models. [Uygulamalı ağaç tabanlı modellerden biri olan Ders'e hoş geldiniz.]



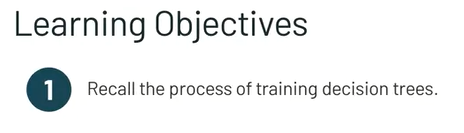
Throughout this lesson, we will learn how to use decision trees to identify complex relationships in data. [Bu ders boyunca, verilerdeki karmaşık ilişkileri tanımlamak için karar ağaçlarının nasıl kullanılacağını öğreneceğiz.]



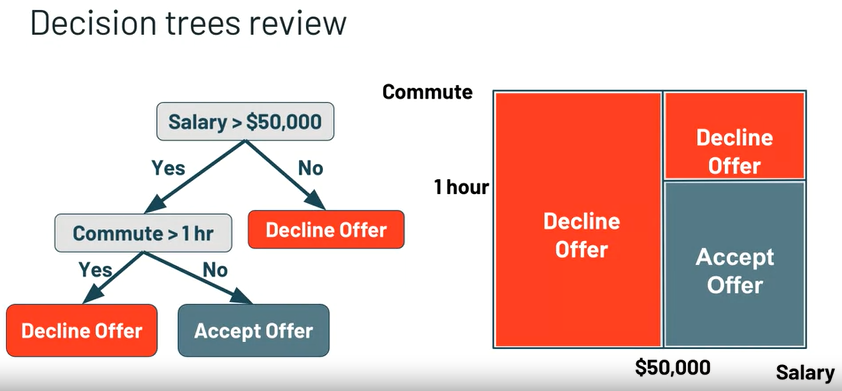
We'll be referring to an example data science project that you'll use decision trees to solve. [Çözmek için karar ağaçlarını kullanacağınız örnek bir veri bilimi projesine atıfta bulunacağız.] The problem we'll be working on involves the healthcare tracker device data that we've used previously in this course. [Üzerinde çalışacağımız sorun, bu kursta daha önce kullandığımız sağlık bakım cihazı verileriyle ilgilidir.] In this example, we'll be using this data to try to predict certain things about customers, like which lifestyle category they fall into, their BMI, or their country. [Bu örnekte, müşteriler hakkında hangi yaşam tarzı kategorisine girdikleri, BMI'ları veya ülkeleri gibi belirli şeyleri tahmin etmeye çalışmak için bu verileri kullanacağız.]

As you can tell, this means that will be working on both classification and regression problems. [Anlayabileceğiniz gibi, bu hem sınıflandırma hem de regresyon problemleri üzerinde çalışılacağı anlamına gelir.] In this lesson, we'll learn more about how an algorithm called decision trees can be used for both of these. [Bu derste, karar ağaçları adı verilen bir algoritmanın bunların her ikisi için nasıl kullanılabileceği hakkında daha fazla şey öğreneceğiz.]

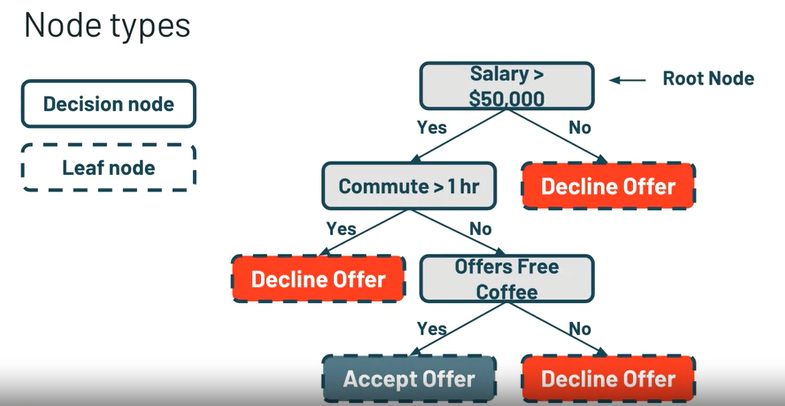
# A Review of Decision Trees



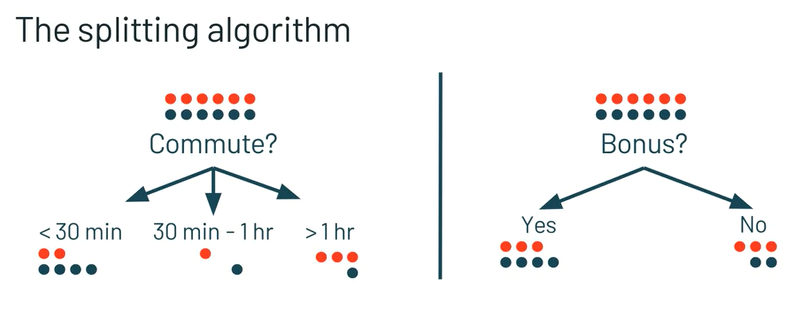
In this video, we'll be going over the process of training decision trees that we previously learned about. [Bu videoda, daha önce öğrendiğimiz karar ağaçlarının eğitim sürecini ele alacağız.]



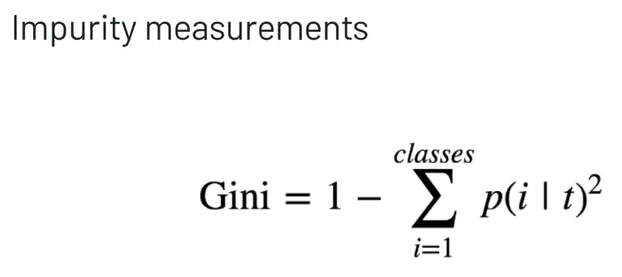
Remember that decision trees mirror a typical human decision-making process. [Karar ağaçlarının tipik bir insan karar verme sürecini yansıttığını unutmayın.] They follow a tree-like structure where we start with a decision node that asks a question in form of a yes or no question and then we proceed down the tree based on the answer to that question. [Evet veya hayır şeklinde bir soru soran bir karar düğümü ile başladığımız ağaç benzeri bir yapıyı takip ediyorlar ve sonra bu sorunun cevabına göre ağaçtan aşağı doğru ilerliyoruz.] We continue going to whatever depth we have specified as a hyperparameter or until every terminal leaf node is pure and we've gotten to one answer per node. [Hiperparametre olarak belirlediğimiz derinliğe veya her terminal yaprak düğümü saf olana ve düğüm başına bir yanıt elde edene kadar devam ediyoruz.] This layer decision nodes allow us to represent complex relationships in data that linear models often struggle with. [Bu katman karar düğümleri, doğrusal modellerin sıklıkla mücadele ettiği verilerdeki karmaşık ilişkileri temsil etmemizi sağlar.] In the picture on the right, we're showing how the decision-making process for a decision tree creates boundaries rather than a straight line like in a linear model. [Sağdaki resimde, bir karar ağacı için karar verme sürecinin lineer bir modelde olduğu gibi düz bir çizgi yerine nasıl sınırlar oluşturduğunu gösteriyoruz.] The boundaries are between different regions where we have commute on the, I guess you could y-axis here and salary on the x axis. [Sınırlar, gidip geldiğimiz farklı bölgeler arasında, sanırım burada y ekseni ve x ekseninde maaş alabilirsiniz.] So each decision we make splits the space into two. [Yani verdiğimiz her karar, alanı ikiye böler.] We start off with splitting the salary into 50,000 as the first question and that divides our space into two sections. [İlk soru olarak maaşı 50.000'e bölerek başlıyoruz ve bu da alanımızı iki bölüme ayırıyor.] Then within the next box, we can either decline or accept the offer based on if the commute is one hour. [Ardından, bir sonraki kutuda, işe gidip gelme süresinin bir saat olup olmadığına bağlı olarak teklifi reddedebilir veya kabul edebiliriz.] We've seen how this process of decision-making that decision trees use can be used for classification or regression problems. [Karar ağaçlarının kullandığı bu karar verme sürecinin sınıflandırma veya regresyon problemlerinde nasıl kullanılabileceğini gördük.]



We have a few key types of nodes in a decision tree. [Bir karar ağacında birkaç anahtar düğüm tipimiz var.] The root node is the top of the decision tree and represents the first decision. [Kök düğüm, karar ağacının en üst noktasıdır ve ilk kararı temsil eder.] Decision node is a general term that refers to any node at which a decision is made and leaf nodes which are also called terminal nodes are where the decision tree comes to a final or terminal prediction. [Karar düğümü, bir kararın verildiği herhangi bir düğümü ifade eden genel bir terimdir ve uç düğümler olarak da adlandırılan yaprak düğümler, karar ağacının nihai veya nihai bir tahmine geldiği yerdir.]



The decision tree will consider all the possible splits and choose the best one at each node. [Karar ağacı tüm olası bölünmeleri dikkate alacak ve her düğümde en iyisini seçecektir.] The best one is based on which question has the highest information gain and the results of each branch should be as homogenous or pure as possible by finding the best split for each feature in the data set and then comparing the best split for each feature. [En iyisi, hangi sorunun en yüksek bilgi kazancına sahip olduğuna dayanır ve veri setindeki her bir özellik için en iyi bölmeyi bularak ve ardından her bir özellik için en iyi bölmeyi karşılaştırarak her dalın sonuçlarının mümkün olduğunca homojen veya saf olması gerekir.] Finally, selecting the feature that leads to the best overall split. [Son olarak, en iyi genel bölünmeye yol açan özelliğin seçilmesi.] The tree decides how it will make each split. [Ağaç, her bir bölünmeyi nasıl yapacağına karar verir.] Each split point is selected greedily. [Her bölme noktası açgözlülükle seçilir.] Rather than taking into account what might be better later on, it's only looking at what is best at this stage or this particular decision point. [Daha sonra neyin daha iyi olabileceğini hesaba katmak yerine, yalnızca bu aşamada veya bu belirli karar noktasında neyin en iyi olduğuna bakmaktır.]

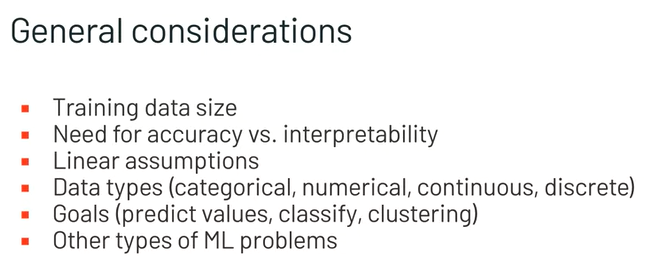


We have certain ways of measuring how to determine the quality of each split. [Her bir bölünmenin kalitesini nasıl belirleyeceğimizi ölçmek için belirli yollarımız var.] The most commonly used is the Gini impurity which is a measure of how pure or similar the classes in a given node are. [En yaygın olarak kullanılan, belirli bir düğümdeki sınıfların ne kadar saf veya benzer olduğunun bir ölçüsü olan Gini safsızlığıdır.] A Gini score of zero means that that node is completely pure and all the samples in that node are of the same class. [Gini puanının sıfır olması, o düğümün tamamen saf olduğu ve o düğümdeki tüm örneklerin aynı sınıftan olduğu anlamına gelir.] The impurity is calculated for a parent node which is the node before the split and the child node or the node after the split to determine how much information is gained from the parent to the child by making that split. [Safsızlık, bölmeden önceki düğüm ve alt düğüm veya bölmeden sonraki düğüm olan bir üst düğüm için bu bölmeyi yaparak ebeveynden çocuğa ne kadar bilgi kazanıldığını belirlemek için hesaplanır.]

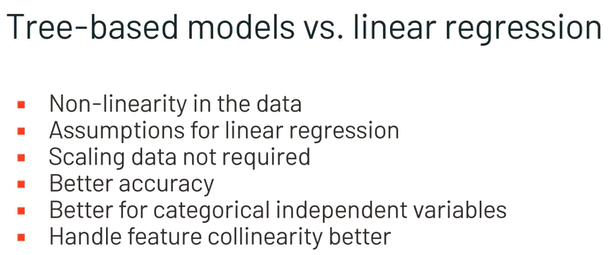
# Algorithm Selection



In this video, we'll describe how a big part of designing machine learning solutions is all about identifying which algorithm to use for a particular problem. [Bu videoda, makine öğrenimi çözümleri tasarlamanın büyük bir bölümünün belirli bir problem için hangi algoritmanın kullanılacağını belirlemek olduğunu açıklayacağız.] We'll review how we go about selecting the best algorithm and some of the reasons why we might want to use the tree-based model instead of a linear regression model. [En iyi algoritmayı nasıl seçeceğimizi ve doğrusal bir regresyon modeli yerine ağaç tabanlı modeli kullanmak isteyebileceğimizin bazı nedenlerini gözden geçireceğiz.]



Some of the considerations for selecting which machine learning model to use for a particular problem are; the size of your training data set, your need for accuracy versus the need for interpretability in terms of where your priorities lie, whether the data set meets the key assumptions for a linear model, what data types are in your data set, for example, categorical, and numerical, continuous or discrete, what your goals are for the particular machine learning problem: are you trying to predict some values or classify something, or are you trying to cluster similar items together? [Belirli bir problem için hangi makine öğrenimi modelinin kullanılacağını seçmeye yönelik bazı hususlar şunlardır; eğitim veri kümenizin boyutu, önceliklerinizin nerede olduğu, veri kümesinin doğrusal bir model için temel varsayımları karşılayıp karşılamadığı, veri kümenizde hangi veri türlerinin bulunduğu açısından doğruluk gereksiniminize karşı yorumlanabilirlik gereksiniminiz, örneğin, kategorik ve sayısal, sürekli veya ayrık, belirli makine öğrenimi problemi için hedefleriniz nelerdir: bazı değerleri tahmin etmeye veya bir şeyi sınıflandırmaya mı çalışıyorsunuz yoksa benzer öğeleri bir araya getirmeye mi çalışıyorsunuz?] There can be other types of machine learning problems and other considerations which we haven't talked about in this course, and we won't get into. [Bu kursta bahsetmediğimiz ve girmeyeceğimiz başka türde makine öğrenimi sorunları ve diğer hususlar olabilir.] But you may be familiar with some of these like, you may have heard of recommendation engines which generate recommendations like Netflix, or algorithms that will classify images or sounds, or certain types of machinery models that can detect anomalies in data. [Ancak bunlardan bazılarına aşina olabilirsiniz, Netflix gibi öneriler üreten öneri motorlarını veya görüntüleri veya sesleri sınıflandıracak algoritmaları veya verilerdeki anormallikleri algılayabilen belirli türdeki makine modellerini duymuş olabilirsiniz.]



Some of the reasons why you would choose to use a tree-based model instead of a linear model like linear regression, is that linear regression only works with data where there is a linear relationship. [Doğrusal regresyon gibi doğrusal bir model yerine ağaç tabanlı bir model kullanmayı seçmenizin nedenlerinden bazıları, doğrusal gerilemenin yalnızca doğrusal bir ilişkinin olduğu verilerle çalışmasıdır.] The tree-based models do a better job at capturing non-linearity in the data. [Ağaç tabanlı modeller, verilerdeki doğrusal olmayanlığı yakalamada daha iyi bir iş çıkarır.] Additionally, if your data doesn't meet the other assumptions for linear regression, you would use the tree-based model. [Ek olarak, verileriniz doğrusal regresyon için diğer varsayımları karşılamıyorsa, ağaç tabanlı modeli kullanırsınız.] Benefit of tree-based models is that you don't have to scale your data first, and generally, they get better accuracy than linear regression. [Ağaç tabanlı modellerin avantajı, önce verilerinizi ölçeklendirmeniz gerekmemesi ve genellikle doğrusal regresyondan daha iyi doğruluk elde etmeleridir.] The tree-based models also work better for categorical independent variables and they tend to handle feature collinearity better. [Ağaç tabanlı modeller ayrıca kategorik bağımsız değişkenler için daha iyi çalışır ve özellik eşdoğrusallığını daha iyi ele alma eğilimindedir.]

# Feature Engineering in Decision Trees

Unlike linear models, which require feature scaling as part of the data preprocessing step, decision trees are invariant to feature scaling. This means that their performance is not affected by scaling or not scaling feature variables. They are not sensitive to outliers or the variance in the data. The splitting algorithm itself is indifferent to scale. Recall that it uses information gain and impurity measures to determine the optimal splits to make. These decisions are based on comparing a particular feature to some value, and it doesn’t matter what scale the feature is on.

Veri ön işleme adımının bir parçası olarak özellik ölçeklendirmeyi gerektiren doğrusal modellerin aksine, karar ağaçları özellik ölçeklendirmede değişmezdir. Bu, performanslarının özellik değişkenlerinin ölçeklenmesinden veya ölçeklendirilmemesinden etkilenmediği anlamına gelir. Aykırı değerlere veya verilerdeki varyansa duyarlı değildirler. Bölme algoritmasının kendisi ölçeklendirmeye kayıtsızdır. Yapılacak en uygun bölmeleri belirlemek için bilgi kazancı ve safsızlık ölçütlerini kullandığını hatırlayın. Bu kararlar, belirli bir özelliği bir değerle karşılaştırmaya dayanır ve özelliğin hangi ölçekte olduğu önemli değildir.

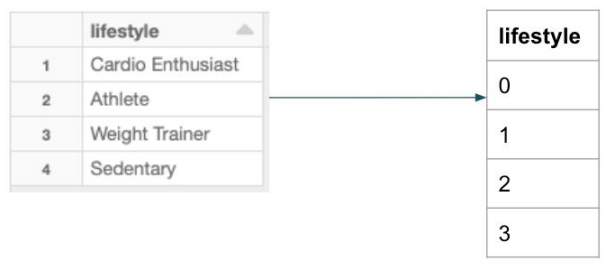
Categorical variables are handled differently in different models and different programming languages. With scikit-learn, categorical variables have to be encoded for ML algorithms. The problem is that, for decision trees, one-hot encoding categorical features is very inefficient, because it creates a sparse matrix in the dataset where there are many more columns full of zeros.

Kategorik değişkenler, farklı modellerde ve farklı programlama dillerinde farklı şekilde ele alınır. scikit-learn ile ML algoritmaları için kategorik değişkenlerin kodlanması gerekir. Sorun şu ki, karar ağaçları için tek-sıcak kodlama kategorik özellikleri çok verimsizdir, çünkü veri kümesinde sıfırlarla dolu çok daha fazla sütunun olduğu seyrek bir matris oluşturur.

Another approach is label encoding:

This approach converts categorical variables to integers. For example, it might convert the lifestyle column to numbers, but this implies a ranking order where “weight trainer” is greater than or more important than “athlete” or “cardio enthusiast”, and this is not intended. If categorical data is not truly ordinal the tree will end up making splits that do not make sense, because the model will interpret these numbers as ranked.

Bu yaklaşım, kategorik değişkenleri tam sayılara dönüştürür. Örneğin, yaşam tarzı sütununu sayılara dönüştürebilir, ancak bu, 'ağırlık antrenörü'nün 'atlet' veya 'kardiyo meraklısı'ndan daha büyük veya daha önemli olduğu bir sıralama düzeni anlamına gelir ve bu amaçlanmamıştır. Kategorik veriler gerçekten sıralı değilse, ağaç mantıklı olmayan bölmeler yaparak sona erecektir, çünkü model bu sayıları dereceli olarak yorumlayacaktır.



# String Indexing Categorical Features

In this video, we'll be demonstrating how to do one-hot encoding of categorical features, as well as indexing categorical features. [Bu videoda, kategorik özelliklerin tek-sıcak kodlamasının ve ayrıca kategorik özelliklerin indekslenmesinin nasıl yapıldığını göstereceğiz.] Then we'll compare the results from each set of features with these two different methods by building and evaluating a decision tree. [Daha sonra, bir karar ağacı oluşturup değerlendirerek her bir özellik kümesinin sonuçlarını bu iki farklı yöntemle karşılaştıracağız.]

Our first step is to prepare the data where we need to aggregate the user level table. [İlk adımımız, kullanıcı düzeyi tablosunu toplamamız gereken verileri hazırlamaktır.] Remember that one of our project objectives for this lesson is to predict a customer's BMI or body mass index based on their other recorded metrics. [Bu ders için proje hedeflerimizden birinin, bir müşterinin kayıtlı diğer ölçümlerine dayalı olarak BMI veya vücut kitle indeksini tahmin etmek olduğunu unutmayın.] Therefore, we need to do a user-level aggregation of the data since the data is coming in more frequently every hour for each user, so we want to aggregate that into averages. [Bu nedenle, veriler her kullanıcı için her saat başı daha sık geldiğinden, verileri kullanıcı düzeyinde toplamamız gerekiyor, bu nedenle bunu ortalamalarda toplamak istiyoruz.] To prepare the data set to do this, we'll use our ht daily metrics table, and we'll take it to the user level by grouping on device ID. [Bunu yapmak için veri setini hazırlamak için ht günlük metrik tablomuzu kullanacağız ve cihaz ID'si üzerinde gruplandırarak kullanıcı seviyesine çıkaracağız.]

Then if we want to view what that table looks like now. [O zaman bu tablonun şimdi nasıl göründüğünü görmek istiyorsak.] Okay. [Peki.] We see that we have our table grouped by device ID. [Cihaz kimliğine göre gruplandırılmış tablomuz olduğunu görüyoruz.] For each device ID, which represents each user, we have the average resting heart rate, active heart rate, the BMI, V02, workout minutes, steps, and lifestyle. [Her kullanıcıyı temsil eden her cihaz kimliği için ortalama dinlenme kalp atış hızı, aktif kalp atış hızı, BMI, V02, egzersiz dakikaları, adımlar ve yaşam tarzına sahibiz.] You'll notice that all of these categories, aside from device ID, which we can ignore because we're not going to use it, but all of these features are numerical except for lifestyle, which is categorical. [Kullanmayacağımız için göz ardı edebileceğimiz cihaz kimliği dışında tüm bu kategorilerin, ancak kategorik olan yaşam tarzı dışında tüm bu özelliklerin sayısal olduğunu fark edeceksiniz.] Before we deal with that, we're going to convert this Spark DataFrame to a Pandas DataFrame so that we can easily work with it using scikitlearn. [Bununla ilgilenmeden önce, bu Spark DataFrame'i bir Pandas DataFrame'e dönüştüreceğiz, böylece onunla scikitlearn kullanarak kolayca çalışabiliriz.] All right, now let's look at the values in the lifestyle column. [Pekala, şimdi yaşam tarzı sütunundaki değerlere bakalım.] We see that we have sedentary weight trainer, athlete, and cardio enthusiast. [Hareketsiz bir ağırlık antrenörü, sporcu ve kardiyo tutkunumuz olduğunu görüyoruz.] We need to convert these to numerical in order for machine learning algorithms to be able to deal with them. [Makine öğrenmesi algoritmalarının bunlarla başa çıkabilmesi için bunları sayısala çevirmemiz gerekiyor.] The first method we'll try is using the label encoder from scikitlearn. [Deneyeceğimiz ilk yöntem, scikitlearn'den etiket kodlayıcıyı kullanmaktır.] This is how we import it, then we instantiate a label encoder object. [Bu şekilde içe aktarırız, ardından bir etiket kodlayıcı nesnesini başlatırız.] Then we apply it to our lifestyle column in our data frame by using fit transform on the column. [Daha sonra sütun üzerinde fit dönüşümünü kullanarak veri çerçevemizdeki yaşam tarzı sütunumuza uygularız.] By doing that, we're going to make a new column called lifestyle cat in the same data frame. [Bunu yaparak, aynı veri çerçevesinde yaşam tarzı kedisi adlı yeni bir sütun oluşturacağız.] If we run this cell and we look at the first five lines of the data frame, we see that we now have this additional column over here, lifestyle cat. [Bu hücreyi çalıştırırsak ve veri çerçevesinin ilk beş satırına bakarsak, şimdi burada bu ek sütunun olduğunu görüyoruz, yaşam tarzı kedisi.] It has encoded the categories as numbers. [Kategorileri sayı olarak kodlamıştır.] Now, if I look at the unique values for that, I see that we have 0-3. [Şimdi, bunun için benzersiz değerlere bakarsam, 0-3 olduğunu görüyorum.]

Now, we're going to proceed with our typical machine learning process where we will split our data set into our X and our Y or target and remember that we're trying to predict BMI, so that's what we'll use for the Y. [Şimdi, veri setimizi X ve Y veya hedefimize böleceğimiz tipik makine öğrenme sürecimize devam edeceğiz ve BMI'yi tahmin etmeye çalıştığımızı hatırlayacağız, bu yüzden Y için kullanacağız. .] Let's just look at the shape of each. [Şimdi her birinin şekline bakalım.] We see that we have 3,000 rows and six columns for the features. [Özellikler için 3.000 satır ve altı sütunumuz olduğunu görüyoruz.] Then the target data set is, as expected, 3,000 rows, one column. [Ardından hedef veri seti, beklendiği gibi, 3.000 satır, bir sütundur.] We'll split our data set into training and test set. [Veri setimizi eğitim ve test seti olarak ayıracağız.] Now, we're just going to fit a very simple decision tree. [Şimdi, çok basit bir karar ağacına sığdıracağız.] We see that we import it from sklearn.tree and we can import the decision tree regressor since we're specifically doing regression. [Bunu sklearn.tree'den içe aktardığımızı görüyoruz ve özellikle regresyon yaptığımız için karar ağacı regresörünü içe aktarabiliyoruz.] We instantiate the decision tree object. [Karar ağacı nesnesini somutlaştırıyoruz.] We're going to call it DT. [Biz buna DT diyeceğiz.] Then we fit it on the X and Y training data set. [Sonra onu X ve Y eğitim veri setine sığdırıyoruz.]

We see that we get this printout that shows us all of the parameters that are specified for the decision tree regressor. [Karar ağacı regresörü için belirtilen tüm parametreleri bize gösteren bu çıktıyı aldığımızı görüyoruz.] We didn't do too many of these obviously, so these are all just going to be the default values. [Bunların çoğunu açıkçası yapmadık, bu yüzden bunların hepsi sadece varsayılan değerler olacak.] Then we'll evaluate the decision tree by getting the R-squared value. [Daha sonra R-kare değerini alarak karar ağacını değerlendireceğiz.] We see that just in this first pass, it did pretty well. [Sadece bu ilk geçişte, oldukça iyi gittiğini görüyoruz.] It got 100 percent on the training set and 92 percent on the test set. [Eğitim setinde yüzde 100 ve test setinde yüzde 92 aldı.] Obviously, this is slightly over fit on the training set because we could probably get those values a little bit closer to each other. [Açıkçası, bu eğitim setine biraz fazla uyuyor çünkü muhtemelen bu değerleri birbirine biraz daha yakınlaştırabiliriz.] But we'll ignore that for now since we're not trying to focus on the accuracy right now. [Ancak şu anda doğruluğa odaklanmaya çalışmadığımız için şimdilik bunu görmezden geleceğiz.] Our second method that we'll try to deal with the categorical lifestyle column is using a Pandas built-in method called get dummies. [Kategorik yaşam tarzı sütunuyla uğraşmaya çalışacağımız ikinci yöntemimiz, get dummy adlı Pandas yerleşik yöntemini kullanmaktır.] What this does is it one-hot encodes categorical values. [Bunun yaptığı şey, kategorik değerleri tek seferde kodlamasıdır.] Scikit-learn also has a one-hot encoder class that does the same thing. [Scikit-learn ayrıca aynı şeyi yapan bir sıcak kodlayıcı sınıfına sahiptir.] But using the built-in pandas method is a little more straightforward in our case, and it just tends to be a little easier to use. [Ancak yerleşik pandalar yöntemini kullanmak bizim durumumuzda biraz daha basittir ve kullanımı biraz daha kolay olma eğilimindedir.] Now we're going to make a new version of the table just so that we don't impact the original version. [Şimdi orijinal versiyonu etkilememek için tablonun yeni bir versiyonunu yapacağız.] Next, we import Pandas as pd and we make a new data frame where we call the Pandas get dummies method on our data frame. [Ardından, Pandaları pd olarak içe aktarıyoruz ve veri çerçevemizde Pandas get dummy yöntemini çağırdığımız yeni bir veri çerçevesi oluşturuyoruz.] We give it the prefix equals lifestyle and specify which column we're using to get dummies. [Buna yaşam tarzına eşittir önekini veriyoruz ve mankenleri almak için hangi sütunu kullandığımızı belirliyoruz.] You'll see what that prefix does when we run this and then display the new data frame. [Bunu çalıştırdığımızda bu önekin ne yaptığını göreceksiniz ve ardından yeni veri çerçevesini görüntüleyeceksiniz.] We see that we have this data frame where we have these new columns, these four columns on the right, and they're filled with ones and zeros. [Bu yeni sütunlara sahip olduğumuz bu veri çerçevesine sahip olduğumuzu görüyoruz, bu dört sütun sağda ve bunlar birler ve sıfırlarla dolu.] Each column has this prefix lifestyle and then underscore and then whatever the value for that category was. [Her sütun bu önek yaşam tarzına sahiptir ve ardından alt çizgi ve ardından o kategorinin değeri ne olursa olsun.] What it does is it pulls out where previously the column had athlete or cardio enthusiasts as its value. [Yaptığı şey, daha önce sütunun değeri olarak atlet veya kardiyo meraklılarının olduğu yeri ortaya çıkarmaktır.] It pulls that into the name of the column and then it fills in zeros and ones depending on which one of these four categories that data sample or row fits into. [Bunu sütun adına çeker ve ardından veri örneğinin veya satırın bu dört kategoriden hangisine uyduğuna bağlı olarak sıfırları ve birleri doldurur.] This first row is the sedentary lifestyle. [Bu ilk sıra, yerleşik yaşam tarzıdır.] We get a one for the lifestyle sedentary and a zero for the other three categories. [Hareketsiz yaşam tarzı için bir tane ve diğer üç kategori için bir sıfır alıyoruz.]

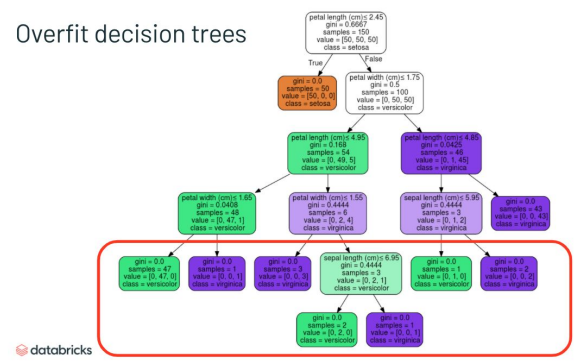
We will split our data frame into the X and Y, so our features and our target. [Veri çerçevemizi X ve Y olarak ayıracağız, bu nedenle özelliklerimiz ve hedefimiz.] We're going to drop our target from our feature X data set. [Hedefimizi özellik X veri setimizden çıkaracağız.] We're also going to drop the device ID because we're not using that for this model. [Bu model için kullanmadığımız için cihaz kimliğini de bırakacağız.] The model would have a problem with using this string value. [Model, bu dize değerini kullanmakta sorun yaşar.] We'll run that. [Bunu çalıştıracağız.] Here we see that instead of having six columns as before, we have nine because we added the four for the dummy variables, and then we dropped the device ID column. [Burada, daha önce olduğu gibi altı sütuna sahip olmak yerine dokuz sütunumuz olduğunu görüyoruz çünkü kukla değişkenler için dördü ekledik ve ardından cihaz kimliği sütununu düşürdük.] Then we do our train test split. [Ardından tren testi bölüşümümüzü yapıyoruz.] We can now train our decision tree on our training data set. [Artık karar ağacımızı eğitim veri setimiz üzerinde eğitebiliriz.]

Then we will evaluate how it did. [Sonra nasıl yapıldığını değerlendireceğiz.] It looks like it's slightly better this time than the previous round with the string indexing method where we got 100 percent on the training set again but 93.1 percent on the test set. [Yine eğitim setinde yüzde 100, test setinde ise yüzde 93,1 aldığımız dizi indeksleme yöntemiyle bu sefer önceki turdan biraz daha iyi gibi görünüyor.] I think it was 92 percent last time. [Geçen sefer yüzde 92 idi sanırım.] It did slightly better, but this could just be a fluke. [Biraz daha iyi oldu, ama bu sadece bir tesadüf olabilir.] We can't really say that it did a lot better from the one-hot encoding. [Tek-sıcak kodlamadan çok daha iyi sonuç verdiğini gerçekten söyleyemeyiz.] But now you know how to do string indexing of categorical features using two different methods. [Ama artık iki farklı yöntem kullanarak kategorik özelliklerin dizi indekslemesinin nasıl yapıldığını biliyorsunuz.]

# Preventing Overfitting

If we allow a decision tree to run its course without tuning any hyperparameters, it will continue through the entire training dataset until every data point is correctly classified and the tree has 100% accuracy on the training data. This usually results in a very overfit tree.

Bir karar ağacının herhangi bir hiperparametreyi ayarlamadan rotasını yürütmesine izin verirsek, her veri noktası doğru bir şekilde sınıflandırılana ve ağaç eğitim verilerinde %100 doğruluğa sahip olana kadar tüm eğitim veri kümesi boyunca devam eder. Bu genellikle çok fazla takılan bir ağaçla sonuçlanır.



Data scientists can correct this through a process called “pruning”, which just means adjusting the hyperparameters of the tree and preventing it from classifying every single data point. Trees can be pre-pruned, where we set limits before building the tree, or post-pruned, where we trim the tree back after it has been built by removing or collapsing nodes that don’t contain much information. Technically, when we talk about pruning decision trees, we’re talking only about post-pruning, but we’ll use the term here to refer to both methods.

Pre-pruning criteria can include:

● Maximum tree depth

○ limiting how deep the tree grows (how many levels of splitting)

● Minimum node size

○ requiring that each node have a minimum number of data points in order to split it further

● Minimum leaf size

○ Requiring at least a certain number of data points in each leaf

● Maximum features

○ maximum number of features to consider at each split

○ Introduces randomness

Veri bilimcileri bunu, ağacın hiperparametrelerini ayarlamak ve her bir veri noktasını sınıflandırmasını engellemek anlamına gelen 'budama' adı verilen bir işlemle düzeltebilir. Ağaçlar önceden budanabilir, burada ağacı oluşturmadan önce sınırlar koyarız veya sonradan budanabilir, burada ağaç oluşturulduktan sonra fazla bilgi içermeyen düğümleri kaldırarak veya daraltarak yeniden budanabiliriz. Teknik olarak, karar ağaçlarını budama hakkında konuştuğumuzda, sadece budama sonrası hakkında konuşuyoruz, ancak burada terimi her iki yönteme de atıfta bulunmak için kullanacağız.

Ön budama kriterleri şunları içerebilir:

● Maksimum ağaç derinliği

○ ağacın ne kadar derin büyüdüğünü (kaç seviye bölme) sınırlama

● Minimum düğüm boyutu

○ daha fazla bölmek için her düğümün minimum sayıda veri noktasına sahip olmasını gerektirme

● Minimum yaprak boyut

○ Her yaprakta en az belirli sayıda veri noktası gerektirmesi

● Maksimum özellikler

○ Her bölmede dikkate alınması gereken maksimum özellik sayısı

○ Rastgelelik sunar

# Decision Tree Pruning

In this demo video, we're going to walk through the process of tuning hyperparameters in order to prune decision trees using scikit learn. [Bu demo videoda, scikit öğrenmeyi kullanarak karar ağaçlarını budamak için hiperparametreleri ayarlama sürecinden geçeceğiz.] So remember that one of our project objectives is to predict a customers daily average number of steps based on their other recorded metrics. [Bu nedenle, proje hedeflerimizden birinin, bir müşterinin kayıtlı diğer ölçümlerine dayalı olarak günlük ortalama adım sayısını tahmin etmek olduğunu unutmayın.] Therefore, we're interested in another user level aggregation, and we're going to do the same thing with the user metrics lifestyle table that we created in the previous demo. [Bu nedenle, başka bir kullanıcı düzeyinde toplamayla ilgileniyoruz ve aynı şeyi önceki demoda oluşturduğumuz kullanıcı metrikleri yaşam tarzı tablosuyla yapacağız.] So let's just take a quick look at that table, and this looks pretty familiar from before. [Şimdi şu tabloya hızlıca bir göz atalım ve bu daha önceden oldukça tanıdık geliyor.] Now we're going to convert it to a pandas dataframe so that we can work with it more easily in scikit learn. [Şimdi onu bir panda veri çerçevesine dönüştüreceğiz, böylece onunla scikit öğrenmede daha kolay çalışabiliriz.] And then we will set up R, X and Y. [Sonra R, X ve Y'yi kuracağız.] And do we train, test, split? [Ve eğitiyor muyuz, test ediyor muyuz, ayırıyor muyuz?] Now we're going to start with fitting a base decision tree. [Şimdi bir temel karar ağacı uydurmaya başlayacağız.] So this is just a baseline model where we don't do any hyperparameter tuning. [Yani bu sadece herhangi bir hiperparametre ayarlaması yapmadığımız bir temel model.] And we're calling our decision tree regressor and fitting it on our training data. [Ve karar ağacı regresörümüzü çağırıyoruz ve onu eğitim verilerimize uyduruyoruz.] And then we will look at the results using the R squared. [Daha sonra R karesini kullanarak sonuçlara bakacağız.] So we see that while we get 100% accuracy on the training set, we only got 84.5% accuracy on the test set. [Dolayısıyla eğitim setinde %100 doğruluk elde ederken, test setinde sadece %84,5 doğruluk elde ettiğimizi görüyoruz.] That's a pretty big difference, and since the training score is so high, we can see clearly that there is over fitting and fairly high variance, since the test set its core isn't very good. [Bu oldukça büyük bir fark ve eğitim puanı çok yüksek olduğundan, testin çekirdeği çok iyi olmadığı için aşırı uydurma ve oldukça yüksek varyans olduğunu açıkça görebiliyoruz.] And the reason this is happening is because the decision tree is unpruned and we didn't adjust any hyper parameters or prune it at all. [Bunun olmasının nedeni, karar ağacının budanmamış olması ve herhangi bir hiper parametreyi ayarlamamış ya da hiç budamamış olmamızdır.] So it's fitting 100% perfectly onto the training set, but then it doesn't generalize very well when it comes to the test set. [Yani eğitim setine %100 uyuyor, ancak test seti söz konusu olduğunda çok iyi bir genelleme yapmıyor.] So remember, we talked about some of the hyperparameters that we can tune in a decision tree to prevent overfitting on our training set. [Bu nedenle, eğitim setimizde fazla takılmayı önlemek için bir karar ağacında ayarlayabileceğimiz bazı hiperparametrelerden bahsettiğimizi unutmayın.] One of these is the maximum tree depth. [Bunlardan biri maksimum ağaç derinliğidir.] Which limits how deep your tree grows, or how many levels of splitting that it goes through. [Bu, ağacınızın ne kadar derin büyüdüğünü veya kaç seviyeden geçtiğini sınırlar.] So we're going to instantiate a new decision tree regressor model, and this time we're setting our Max depth hyperparameter to four. [Bu yüzden yeni bir karar ağacı regresör modeli başlatacağız ve bu sefer Max derinlik hiperparametremizi dörde ayarlıyoruz.] There are different ways to go about choosing hyperparameters, and that is a whole another lesson. [Hiperparametreleri seçmenin farklı yolları vardır ve bu tamamen başka bir derstir.] So for here will just randomly choose the max depth and not worry too much about it. [Yani burada maksimum derinliği rastgele seçecek ve bu konuda çok fazla endişelenmeyecektir.] Then will fit this model onto our train set. [Daha sonra bu modeli tren setimize sığdıracağız.] And then we will look at the score. [Sonra skora bakacağız.] Okay, so we see now that there's not so much of a difference between the scores. [Tamam, şimdi puanlar arasında çok fazla bir fark olmadığını görüyoruz.] The models no longer overfitting to our training data, and it got slightly better on the test set. [Modeller artık eğitim verilerimize fazla uymuyor ve test setinde biraz daha iyi oldu.] But now it just has high bias, which means it's not actually learning the training set very well, and that makes sense because we're restricting the depth. [Ama şimdi sadece yüksek önyargıya sahip, bu da aslında eğitim setini çok iyi öğrenemediği anlamına geliyor ve bu mantıklı çünkü derinliği kısıtlıyoruz.] This is a pretty shallow tree, so it doesn't have a chance to really learn that training set very well. [Bu oldukça sığ bir ağaç, bu yüzden eğitim setini gerçekten çok iyi öğrenme şansı yok.] And this is not ideal because neither of these scores are very good, so we'll try tuning another hyperparameter. [Ve bu ideal değil çünkü bu puanların hiçbiri çok iyi değil, bu yüzden başka bir hiperparametreyi ayarlamayı deneyeceğiz.] This time will look at the minimum node size an remember that this sets your requirement of each node has to have a minimum number of data points in order to continue and split it further. [Bu sefer minimum düğüm boyutuna bakacak ve bunun, devam etmek ve daha fazla bölmek için her düğüm için minimum sayıda veri noktasına sahip olmanız gerektiğini belirlediğini unutmayın.] So if you're node reaches the minimum number of samples it needs in order to split it, will your tree will just stop, and it's fun. [Yani, düğümünüz onu bölmek için ihtiyaç duyduğu minimum örnek sayısına ulaşırsa, ağacınız duracak mı ve bu eğlenceli.] And this time we will call our new model DT node and you'll notice that we're continuing to use next step. [Ve bu sefer yeni model DT düğümümüzü arayacağız ve bir sonraki adımı kullanmaya devam ettiğimizi fark edeceksiniz.] But this time we changed it from four to six just to give the tree a little bit more of an advantage to learn that training data set. [Ancak bu sefer, ağaca bu eğitim veri setini öğrenmesi için biraz daha avantaj sağlamak için onu dörtten altıya değiştirdik.] And we're just going to kind of randomly choose three for the minimum samples per split. [Ve biz sadece bölme başına minimum numuneler için rastgele üç tane seçeceğiz.] And so we will go ahead and train the model and see how it does. [Ve böylece devam edip modeli eğiteceğiz ve nasıl olduğunu göreceğiz.] Okay, well, it's getting better, so we have less bias of 91.5 score on the training set. [Tamam, iyiye gidiyor, yani eğitim setinde daha az önyargımız 91.5 puan.] So it's learning the training set a little bit better, probably because we increase the max depth. [Bu yüzden, muhtemelen maksimum derinliği arttırdığımız için eğitim setini biraz daha iyi öğreniyor.] But we see that the test set, while it is a little bit better, it went from 87.6 to 89.2. [Ama görüyoruz ki test seti biraz daha iyi olsa da 87.6'dan 89.2'ye çıktı.] The bias and variance are still a little high. 'cause these scores aren't great. [Önyargı ve varyans hala biraz yüksek. Çünkü bu puanlar harika değil.] And the performance on the test set is still pretty far below the training performance. [Ve test setindeki performans hala eğitim performansının oldukça altında.] There a little too far apart from each other. [Birbirinden biraz fazla uzakta.] So then we want to try something else. [O zaman başka bir şey denemek istiyoruz.] This time will look at studying the minimum leaf size. [Bu sefer minimum yaprak boyutunu incelemeye bakacağız.] And this is the requirement that at least a certain number of data points are in each leaf in order to create that split on those leaves. [Ve bu, o yapraklarda o bölünmeyi oluşturmak için her yaprakta en az belirli sayıda veri noktasının olması gerekliliğidir.] Okay, so we're going to keep next steps, but this time will increase it a little bit more to 8. [Tamam, sonraki adımlara devam edeceğiz, ancak bu sefer 8'e biraz daha artıracağız.] We're going to have min samples per split, and we're going to decrease that to two, and then we'll kind of randomly try 3 for the minimum samples per leaf. [Bölünme başına minimum numunemiz olacak ve bunu ikiye indireceğiz ve sonra yaprak başına minimum numune için rastgele 3 deneyeceğiz.] Okay, very slight improvement, but it didn't really help much, so will try one more hyperparameter we can adjust for decision trees. [Tamam, çok küçük bir gelişme, ama pek yardımcı olmadı, bu yüzden karar ağaçları için ayarlayabileceğimiz bir hiperparametre daha deneyeceğiz.] And this time we will look at Max features and again we're keeping the previous three hyper parameters. [Ve bu sefer Max özelliklerine bakacağız ve yine önceki üç hiper parametreyi koruyoruz.] But this time will add Max features and say 3 for the maximum number of features that the tree can consider at each split. [Ancak bu sefer Maksimum özellikler ekleyecek ve ağacın her bölmede göz önünde bulundurabileceği maksimum özellik sayısı için 3 diyecek.] And this introduces randomness because at each split that remakes. [Ve bu, rastgeleliği ortaya çıkarır çünkü yeniden yapılan her bölünmede.] It's randomly choosing 3 features from which it can decide how to make the split instead of the entire data set and all of the features. [Tüm veri seti ve tüm özellikler yerine bölmenin nasıl yapılacağına karar verebileceği 3 özelliği rastgele seçiyor.] All right, we train our model and not really a big change. [Pekala, modelimizi eğitiyoruz ve gerçekten büyük bir değişiklik değil.] The training score 92.9 is still not great on the test score 89.9. [Eğitim puanı 92.9, test puanı 89.9'da hala harika değil.] It's not terrible, but there are ways that we can improve and do better with our predictions. [Korkunç değil, ancak tahminlerimizi geliştirmenin ve daha iyisini yapmanın yolları var.] We've seen that even with tuning the hyperparameters it can be difficult to get a decision tree that doesn't have high variance even though we've restricted. [Hiperparametreleri ayarlasak bile, kısıtladığımız halde yüksek varyansa sahip olmayan bir karar ağacı elde etmenin zor olabileceğini gördük.] How well it fits how much overfitting it does to the training set? [Eğitim setine ne kadar fazla uyum sağladığına ne kadar uyuyor?]

# Applied Decision Trees Lab

Next, you'll complete a lab where you will build and evaluate a decision tree to solve a regression problem, using Databricks.

To do so:

1. Log in to your Databricks Community Edition account.

2. Locate your ADSDA folder.

3. Open the Module 4, Lesson 4 folder.

4. Click on 4.1.3 Lab - Applied Decision Trees 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.