# İçindekiler

[(Intro to Machine Learning)Makine Öğrenimine Giriş 3](#_Toc44807462)

[How Models Work (Modeller Nasıl Çalışır): 3](#_Toc44807463)

[Giriş: 3](#_Toc44807464)

[Decision Tree’nin Geliştirilmesi 4](#_Toc44807465)

[Basic Data Exploration (Temel Veri Keşfi) 5](#_Toc44807466)

[Using Pandas to Get Familiar With Your Data ( Verilerinizi Öğrenmek için Pandas kullanma): 5](#_Toc44807467)

[Interpreting Data Description (Veri Açıklamalarını Yorumlama): 6](#_Toc44807468)

[Excercise: Explore Your Data 6](#_Toc44807469)

[Your First Machine Learning Model: 8](#_Toc44807470)

[Selecting Data for Modeling (Modelleme için Veri Seçmek): 8](#_Toc44807471)

[Selecting The Prediction Target (Tahmin Hedefini Seçme) 11](#_Toc44807472)

[Choosing "Features" (Özellik Seçimi): 11](#_Toc44807473)

[Building Your Model (Model Oluşturma): 13](#_Toc44807474)

[Exercises: Your First Machine Learning Model 15](#_Toc44807475)

[Model Validation(Model geçerliliği): 18](#_Toc44807476)

[What is Model Validation: (Model Validation Nedir) 18](#_Toc44807477)

[The Problem with "In-Sample" Scores("In-Sample(Örnek İçi)" Puanlarla İlgili Sorun) 20](#_Toc44807478)

[Coding It 20](#_Toc44807479)

[Wow! 21](#_Toc44807480)

[Exercises: Model Validation 21](#_Toc44807481)

[Underfitting and Overfitting 23](#_Toc44807482)

[Experimenting With Different Models 23](#_Toc44807483)

[Examples: 25](#_Toc44807484)

[Sonuç: 26](#_Toc44807485)

[Exercise: Underfitting and Overfitting 27](#_Toc44807486)

[Random Forests: 29](#_Toc44807487)

[Giriş: 29](#_Toc44807488)

[Example 29](#_Toc44807489)

[Sonuç: 30](#_Toc44807490)

[Exercises: Random Forest 30](#_Toc44807491)

[Exercises: Machine Learning Competitions 32](#_Toc44807492)

[Introduction 32](#_Toc44807493)

[Creating a Model For the Competition 33](#_Toc44807494)

[Make Predictions 33](#_Toc44807495)

[KAYNAKLAR 35](#_Toc44807496)

# (Intro to Machine Learning)Makine Öğrenimine Giriş

Makine öğrenmesindeki temel fikirleri öğrenin ve ilk modellerinizi oluşturun.

## How Models Work (Modeller Nasıl Çalışır):

### Giriş:

Makine öğrenimi modellerinin nasıl çalıştığına ve nasıl kullanıldıklarına genel bir bakışla başlayacağız. Daha önce istatistiksel modelleme veya makine öğrenimi yaptıysanız bu temel görünebilir. Endişelenmeyin, yakında güçlü modeller oluşturmaya devam edeceğiz.

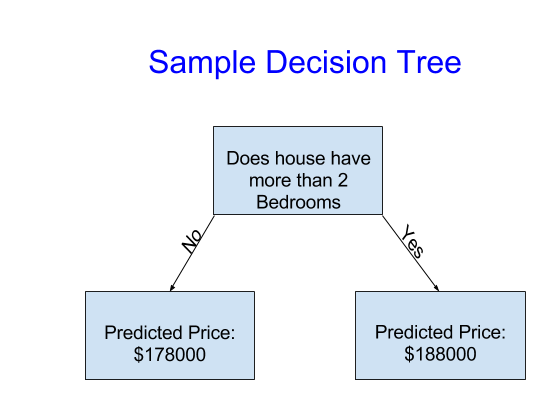
Bu mikro kurs, aşağıdaki senaryodan geçerken modeller oluşturmanızı sağlayacaktır:

Kuzeniniz gayrimenkul konusunda spekülasyonlar milyonlarca dolar kazandı. Veri bilimine gösterdiğiniz ilgi nedeniyle sizinle iş ortağı olmayı teklif etti. Parayı tedarik edecek ve çeşitli evlerin ne kadar değerli olduğunu tahmin eden modeller sunacaksınız.

Kuzeninize geçmişte gayrimenkul değerlerini nasıl tahmin ettiğini soruyorsunuz. Ve bunun sadece sezgi olduğunu söylüyor. Ancak daha fazla sorgulama, geçmişte gördüğü evlerden fiyat örüntülerini belirlediğini ve bu kalıpları düşündüğü yeni evler için tahminler yapmak için kullandığını ortaya koyuyor.

Makine öğrenimi de aynı şekilde çalışır. Karar Ağacı **(Decision Tree)** adlı bir modelle başlayacağız. Daha doğru tahminler veren meraklı modeller var. Ancak karar ağaçları’nın anlaşılması kolaydır ve bunlar veri bilimindeki en iyi modellerin bazıları için temel yapı taşıdır.

Basitlik için, mümkün olan en basit karar ağacıyla başlayacağız.



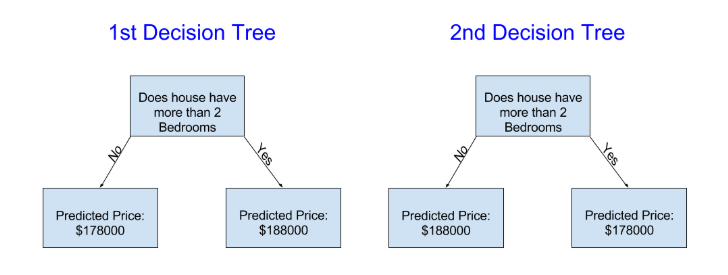
Evleri sadece iki kategoriye ayırır. Dikkate alınan herhangi bir ev için tahmini fiyat, aynı kategorideki evlerin tarihsel ortalama fiyatıdır.

Verileri, evlerin iki gruba nasıl ayrılacağına karar vermek için ve sonra her grupta öngörülen fiyatı belirlemek için kullanıyoruz. Verilerden pattern(desen) yakalamanın bu adımına**, modelin fit edilmesi (fitting)** veya **train edilmesi(training)** denir.

Modelin fit edilmesi için kullanılan verilere **training data** denir. Modelin nasıl fit edildiğine dair ayrıntılar (örneğin, verilerin nasıl bölüneceği) daha sonra kullanmak üzere kayıt edeceğimiz kadar karmaşıktır. Model fit edildikten sonra, yeni evlerin fiyatlarını **predict(tahmin)** edebilmek için yeni verilere uygulayabilirsiniz.

### Decision Tree’nin Geliştirilmesi

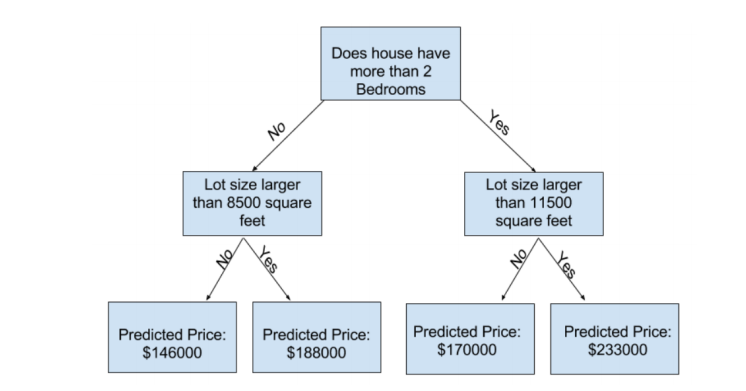
Aşağıdaki iki karardan hangisinin gayrimenkul eğitim verilerinin fit edilmesinden kaynaklanması daha olasıdır?



Soldaki karar ağacı (Karar Ağacı 1) muhtemelen daha mantıklıdır, çünkü daha fazla yatak odası olan evlerin daha az yatak odası olan evlerden daha yüksek fiyatlarla satılma eğiliminde olduğu gerçeğini yakalar.

Bu modelin en büyük eksikliği, banyo sayısı, lot büyüklüğü, yer vb. gibi ev fiyatını etkileyen çoğu faktörü yakalamamasıdır.

Daha fazla **"splits(bölme)"** olan bir ağaç kullanarak daha fazla faktör yakalayabilirsiniz. Bunlara **"deeper(daha derin)" ağaçlar** denir. Her evin toplam lot büyüklüğünü de dikkate alan bir karar ağacı şöyle görünebilir:



Herhangi bir evin fiyatını karar ağacından takip ederek, her zaman o evin özelliklerine karşılık gelen yolu seçerek tahmin edersiniz.

Ev için tahmini fiyat ağacın altındadır.

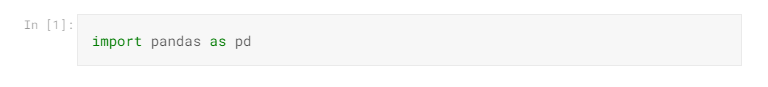
Altta tahmin yaptığımız noktaya **leaf(yaprak)** denir.

Yapraklardaki **splits(bölünmeler)** ve **values(değerler)** veriler tarafından belirlenecektir, bu nedenle çalışacağınız verileri kontrol etmenin zamanı geldi.

## Basic Data Exploration (Temel Veri Keşfi)

### Using Pandas to Get Familiar With Your Data ( Verilerinizi Öğrenmek için Pandas kullanma):

Herhangi bir makine öğrenimi projesinin ilk adımı, verileri tanımaktır. Bunun için Pandas kütüphanesini kullanacaksınız. **Pandas**, bilim insanlarının verileri keşfetmek ve işlemek için kullandığı temel araçtır. Çoğu kişi **Pandas** kodlarında **pd** olarak kısaltılır. Bunu şu komutla yapıyoruz:



Panda kütüphanesinin en önemli kısmı DataFrame'dir. Bir DataFrame, tablo olarak düşünebileceğiniz veri türünü tutar. Bu, Excel'deki bir sayfaya veya SQL veritabanındaki bir tabloya benzer.

Pandas, bu tür verilerle yapmak isteyeceğiniz birçok şey için güçlü yöntemlere sahiptir.

Örnek olarak, Avustralya, Melbourne'daki ev fiyatları hakkındaki verilere bakacağız( <https://www.kaggle.com/dansbecker/melbourne-housing-snapshot>).

Uygulamalı alıştırmalarda, aynı işlemleri Iowa'da ev fiyatları olan yeni bir veri kümesine uygulayacaksınız.

Örnek (Melbourne) verileri ../input/melbourne-housing-snapshot/melb\_data.csv dosya yolundadır.

Verileri aşağıdaki komutlarla yükler ve inceleriz:

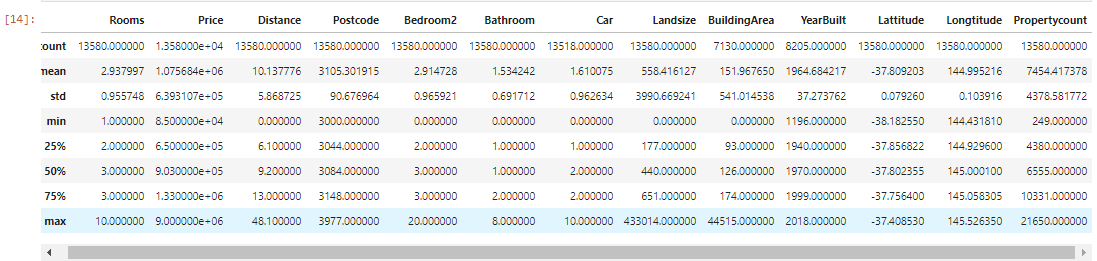
* # kolay erişim için dosya yolunu değişkene kaydet



* # verileri okuyun ve DataFrame'de melbourne \_data başlıklı verileri depolayın







### Interpreting Data Description (Veri Açıklamalarını Yorumlama):

Sonuçlar, orijinal veri kümenizdeki her sütun için 8 sayı gösterir. İlk sayı, sayı, kaç satırın eksik olmayan değerleri olduğunu gösterir.

Eksik değerler birçok nedenden dolayı ortaya çıkar. Örneğin, 1 yatak odalı bir ev araştırılırken 2. yatak odasının boyutu toplanmaz. Eksik veriler konusuna geri döneceğiz.

İkinci değer, ortalama olan **mean'dir**. Bunun altında **std**, değerlerin sayısal olarak ne kadar yayıldığını ölçen standart sapmadır.

**Min,% 25**,**% 50**,**% 75** ve maksimum değerleri yorumlamak için, her sütunu en düşükten en yüksek değere doğru sıraladığınızı düşünün.

İlk (en küçük) değer **min**. Liste'nin dörtte birini incelerseniz, değerlerin **% 25**'inden daha büyük ve değerlerin **% 75**'inden daha küçük bir sayı bulacaksınız.

Bu% 25 değerdir ("25. **percentile**" olarak telaffuz edilir). 50. ve 75. yüzdelikler benzer şekilde tanımlanır ve **max**. En büyük sayıdır.

### Excercise: Explore Your Data

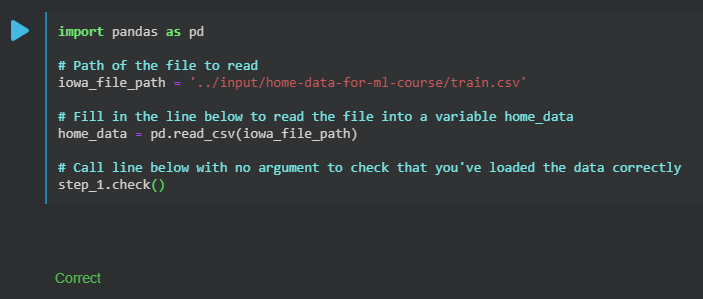
Bu alıştırma, bir veri dosyasını okuma ve verilerle ilgili istatistikleri anlama yeteneğinizi test edecektir.

Daha sonraki alıştırmalarda, verileri filtrelemek, bir makine öğrenme modeli oluşturmak ve modelinizi yinelemeli olarak geliştirmek için teknikler uygulayacaksınız.

Kurs örnekleri Melbourne'den gelen verileri kullanır. Bu teknikleri kendi başınıza uygulayabilmeniz için, bunları yeni bir veri kümesine (Iowa'dan konut fiyatları) uygulamanız gerekecektir.

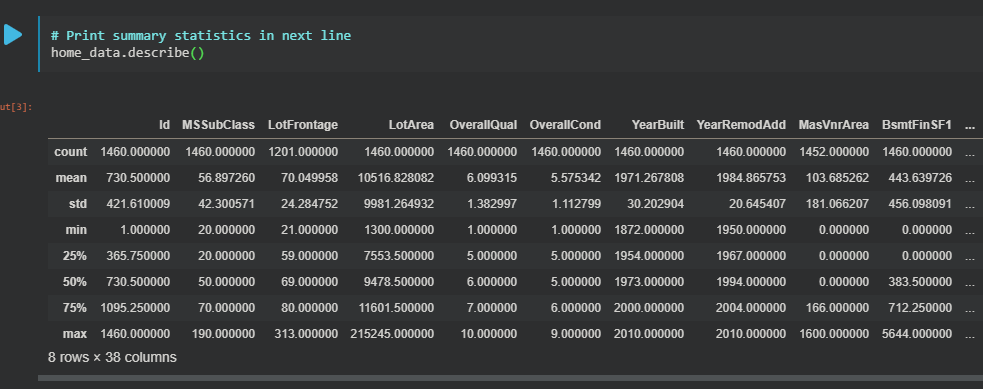
#### **Step 1: Loading Data (Veri Yükleme)**

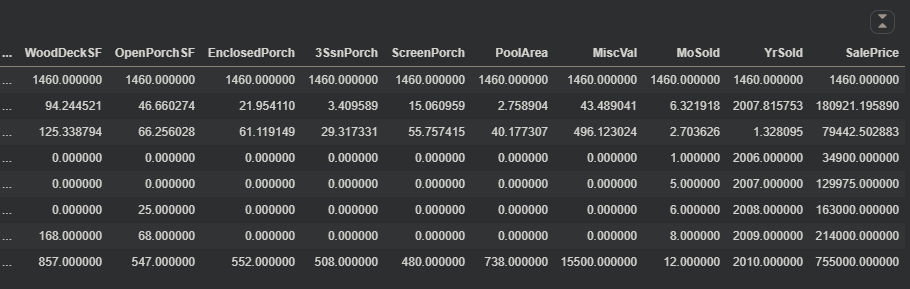
Iowa veri dosyasını home\_data adlı bir Pandas DataFrame'de okuyun.

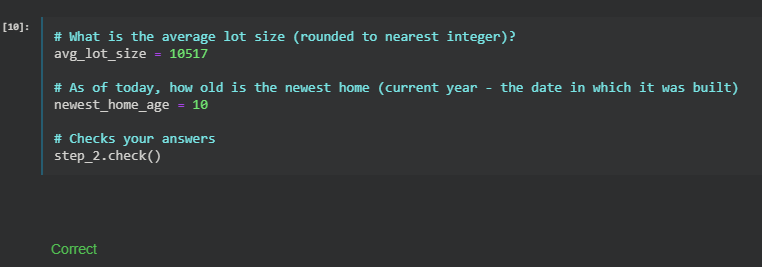


#### **Step 2: Review The Data (Verileri Gözden Geçirme)**

Verilerin özet istatistiklerini görüntülemek için öğrendiğiniz komutu kullanın. Ardından aşağıdaki soruları cevaplamak için değişkenleri doldurun







#### **Verilerinizi Düşünün**

Verilerinizdeki en yeni ev o kadar yeni değil. Bunun için birkaç potansiyel açıklama:

1- Bu verilerin toplandığı yeni evler inşa etmediler.

2- Veriler uzun zaman önce toplanmıştır. Veri yayımından sonra inşa edilen evler görünmezdi.

Nedeni yukarıdaki 1. açıklama ise, bu, bu verilerle oluşturduğunuz modele olan güveninizi etkiler mi? 2. neden ise ne olur?

Hangi açıklamanın daha mantıklı olduğunu görmek için verileri nasıl inceleyebilirsiniz?

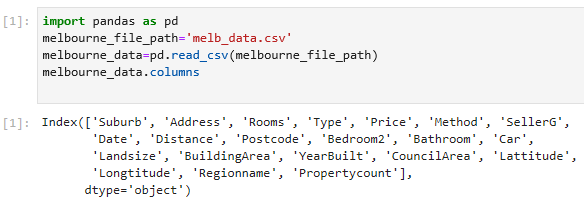
## Your First Machine Learning Model:

### Selecting Data for Modeling (Modelleme için Veri Seçmek):

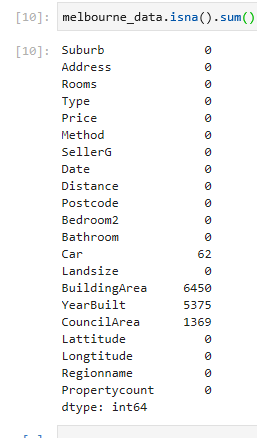
Veri kümenizin, kafanızda canlanması veya güzelce ekrana yazdırmak için çok fazla değişkeni vardı. Bu başa çıkılamaz veri miktarını anlayabileceğiniz bir şeye nasıl ayırabilirsiniz?

Sezgimizi kullanarak birkaç değişken seçerek başlayacağız. Daha sonraki kurslar, değişkenleri otomatik olarak önceliklendirmek için istatistiksel teknikleri gösterecektir.

Değişkenleri / sütunları seçmek için veri kümesindeki tüm sütunların bir listesini görmemiz gerekir. Bu, DataFrame’in **columns** özelliği ile yapılır. (Aşağıdaki kodun alt satırı.)



# Melbourne verilerinin bazı eksik değerleri vardır (bazı değişkenlerin kaydedilmediği bazı evler.)



# Daha sonraki bir derste eksik değerleri ele almayı öğreneceğiz.

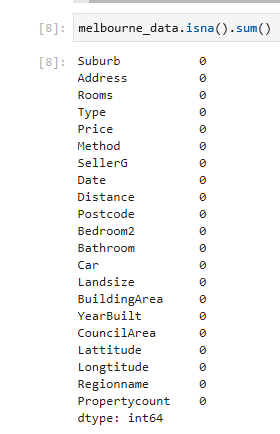
# Iowa verileriniz, kullandığınız sütunlarda eksik değerlere sahip değildi.

# Şimdilik en basit seçeneği alacağız ve verilerimizden eksik değere sahip evleri düşüreceğiz.

# dropna eksik değerleri düşürüyor (na'yı "mevcut değil" olarak düşünün)



# Column’ların içinde kaçar tane eksik veri var ona baktık.



Verilerinizin bir alt kümesini seçmenin birçok yolu vardır. Pandas Micro-Course (<https://www.kaggle.com/learn/pandas>) bunları daha derinlemesine ele alıyor, ancak şimdilik iki yaklaşıma odaklanacağız.

1. "Prediction Target(Tahmin hedefi)"’ni seçmek için kullandığımız nokta gösterimi(dot notation)
2. "Features(Özellikleri)" seçmek için kullandığımız bir sütun listesiyle seçim yapma

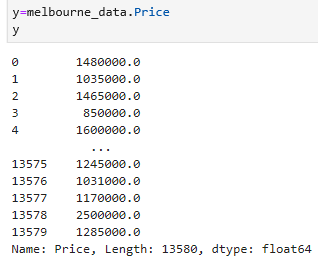
### Selecting The Prediction Target (Tahmin Hedefini Seçme)

**dot-notation** ile bir değişkeni(column) veri setinden çekebilirsiniz. Bu tek sütun, genel olarak yalnızca tek bir column’a sahip DataFrame benzeri bir **Seri’**de depolanır.

Tahmin etmek istediğimiz column’u seçmek için dot-notation kullanacağız, buna **prediction target** (tahmin hedefi) denir.

Kural olarak, prediction target (tahmin hedefi) **y** olarak adlandırılır.

Melbourne'deki ev fiyatlarını (price) kaydetmek için gereken kod.



### Choosing "Features" (Özellik Seçimi):

Modelimize girilen sütunlara (ve daha sonra tahminlerde kullanılan sütunlara) "features (özellikler)" denir.

Bizim durumumuzda, bunlar ev fiyatını belirlemek için kullanılan sütunlar olacaktır.

Bazen, target(hedef) hariç tüm sütunları feature(özellik) olarak kullanırsınız. Diğer zamanlarda daha az özellik ile daha iyi olacaksınız.

Şimdilik, sadece birkaç özelliğe sahip bir model oluşturacağız.

Daha sonra, farklı özelliklerle oluşturulan modellerin nasıl tekrarlanacağını ve karşılaştırılacağını göreceksiniz.

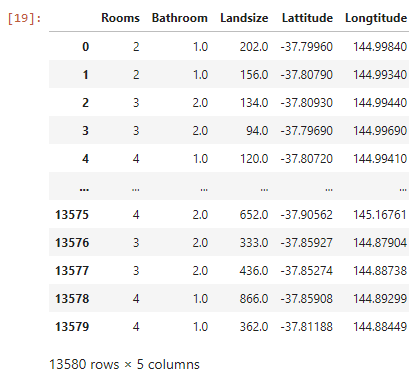
Köşeli parantez içine sütun adlarının listesini yazarak birden fazla özellik seçiyoruz. Bu listedeki her öğe bir string (tırnak işaretli) olmalıdır.

Here is an example:

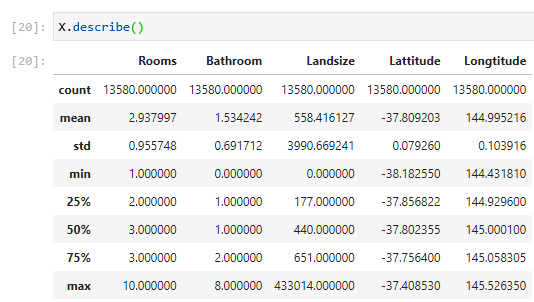


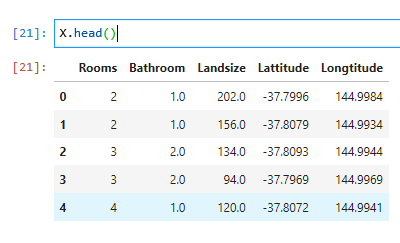
Kural olarak, bu verilere X denir.





En üstteki birkaç satırı gösteren **head** yöntemini ve **describe** yöntemini kullanarak konut fiyatlarını tahmin etmek için kullanacağımız verileri hızlı bir şekilde inceleyelim.





Verilerinizi bu komutlarla görsel olarak kontrol etmek, bir veri bilim insanının işinin önemli bir parçasıdır. Veri kümesinde sıklıkla daha fazla incelemeyi hak eden sürprizler bulacaksınız.

### Building Your Model (Model Oluşturma):

Modellerinizi oluşturmak için **scikit-learn** kütüphanesini kullanacaksınız.

Kodlama yaparken, bu kütüphane örnek kodda göreceğiniz gibi **sklearn** olarak yazılır.

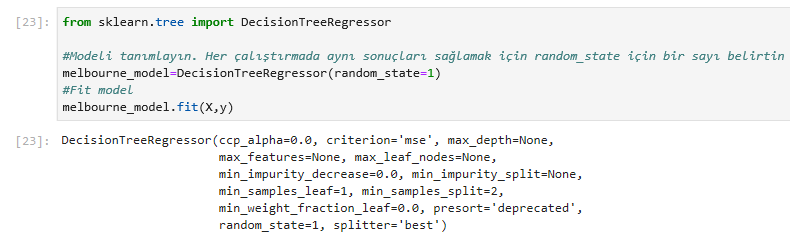
Scikit-learn, tipik olarak DataFrames'da depolanan veri türlerini modellemek için en popüler kütüphanedir.

**Bir model oluşturma ve kullanma adımları:**

* **define :** Ne tür bir model olacak? Karar ağacı mı? Başka bir model mi? Model tipinin diğer bazı parametreleri de belirtilir.
* **fit :** Sağlanan verilerden pattern(desen) yakalayın. Bu modellemenin kalbidir.
* **predict :** Tahmin
* **evaluate :** Modelin tahminlerinin ne kadar doğru olduğu belirleyin.

İşte **scikit-learn** ile bir **Decision Tree**(Karar Ağaçları)modelini tanımlama ve modeli feature’lara ve target değişkene **fit** etme örneği.

* Modeli tanımlayın. Her çalıştırmada aynı sonuçları sağlamak için random\_state için bir sayı belirtin



**random\_state:** Kodu her çalıştırdığımızda aynı çıktıyı alabilmek için girdiğimiz bir ifade. Örneğin, validation ve training olarak datayı ayırırken Python her seferinde datayı farklı yerlerinden böler, bir random state değeri belirlediğimizde de her çalıştırdığımızda aynı şekilde bölmüş olur ve aynı sonucu vermiş olur. Farklı değerler verdiğinde farklı sonuçlar aldığını göreceksin.

En iyi karar ağacını bulma problemi NP-Complete olarak sınıflandırılan problemlerdendir. Bu tip problemlerin çözümlerinde sezgisel algoritmalar kullanılır. Sezgisel algoritmalarda her kullanıldıklarında en iyi çözümü bulabileceklerini garanti etmezler ve her seferinde farklı sonuçlar üretirler. Dolayısıyla her ağaç inşa ettiğinde ağaç yapısı değişiklik gösterecektir. Modeli her çalıştırdığında aynı ağacı elde etmek istersen **random\_state** parametresini bir tamsayıya eşitlemen gerekir. Hangi tamsayıya eşitlediğinin bir önemi yok .

Birçok makine öğrenimi modeli, model eğitiminde bazı rasgeleliklere izin verir.

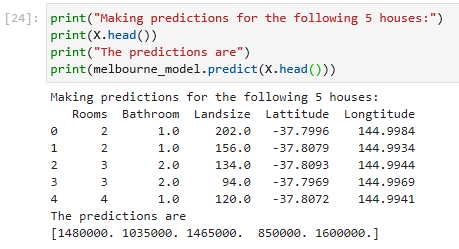
**Random\_state** için bir sayı belirtmek, her çalıştırmada aynı sonuçları almanızı sağlar. Bu iyi bir uygulama olarak kabul edilir.

Herhangi bir sayı kullanabilirsiniz ve model kalitesi tam olarak hangi değeri seçtiğinize bağlı olmayacaktır.

Herhangi bir sayı kullanabilirsiniz ve model kalitesi tam olarak hangi değeri seçtiğinize bağlı olmayacaktır.

Uygulamada, halihazırda fiyatlarımız olan evler yerine piyasaya çıkan yeni evler için tahminler yapmak isteyeceksiniz.

Ancak, tahmin işlevinin nasıl çalıştığını görmek için egzersiz verilerinin ilk birkaç satırı için tahminler yapacağız.

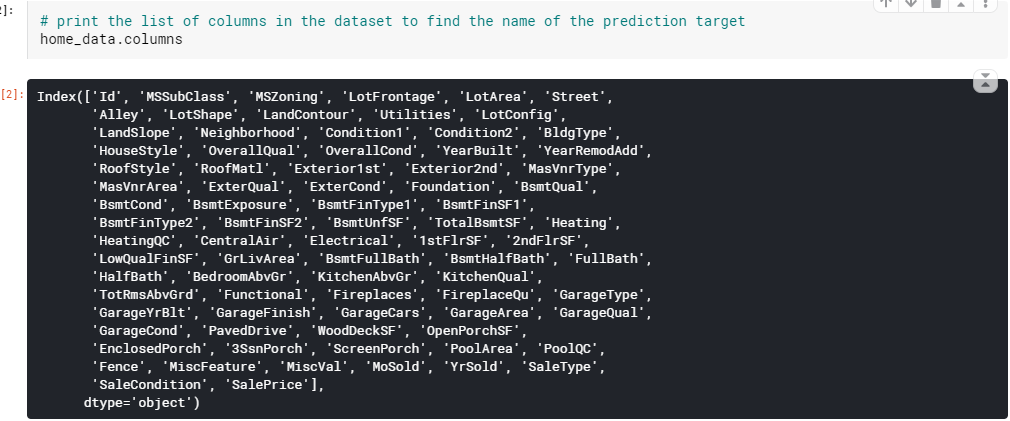


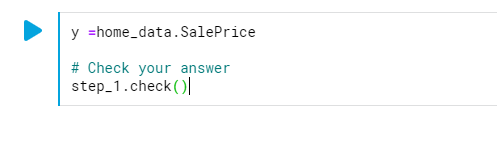
### Exercises: Your First Machine Learning Model

#### **Step 1: Specify Prediction Target: (Tahmin Hedefi Belirtme)**

"Satış fiyatı(sales price)" na karşılık gelen hedef değişkeni seçin. Bunu "y" adlı yeni bir değişkene kaydedin. İhtiyacınız olan sütunun adını bulmak için sütunların bir listesini yazdırmanız gerekir.

# tahmin hedefinin adını bulmak için veri kümesindeki sütunların listesini yazdır





# Aşağıdaki satırlar size bir ipucu veya çözüm gösterecektir.

# step\_1.hint()

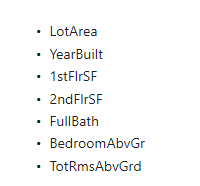
# step\_1.solution()

#### **Step 2: Create X**

Şimdi, predictive feature’ları (tahmin özelliklerini) tutan X adında bir DataFrame oluşturacaksınız.

Orijinal verilerden yalnızca bazı sütunlar istediğiniz için, önce X'de istediğiniz sütunların adlarını içeren bir liste oluşturacaksınız.

Listede yalnızca aşağıdaki sütunları kullanacaksınız :

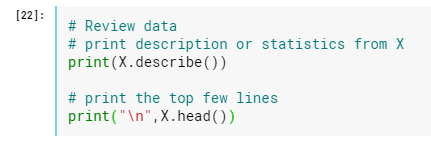


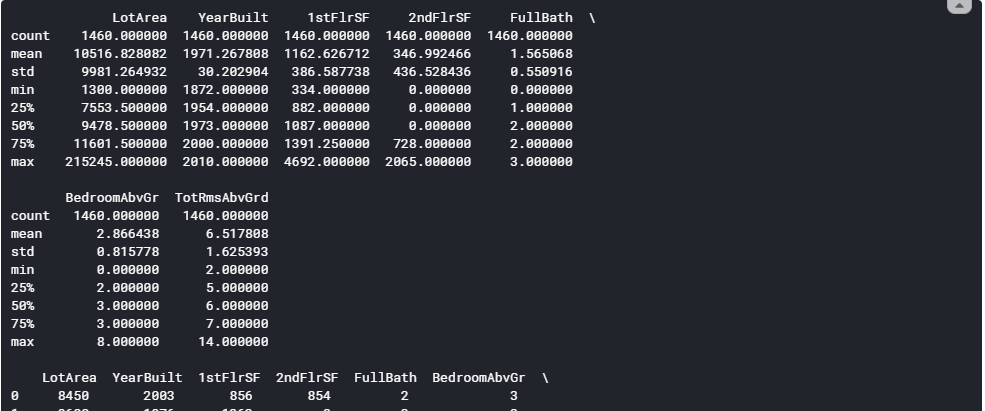
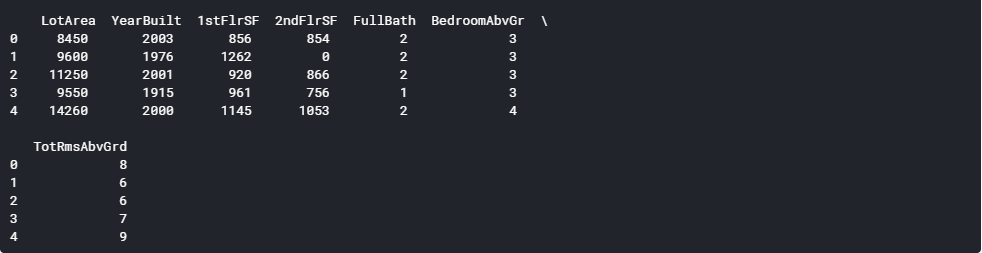
Bu özellik listesini oluşturduktan sonra, modeli fit etmek için kullanacağınız DataFrame'i oluşturmak için kullanın.



#### **Review Data**

Bir model oluşturmadan önce, mantıklı göründüğünü doğrulamak için X 'e hızlı bir göz atın



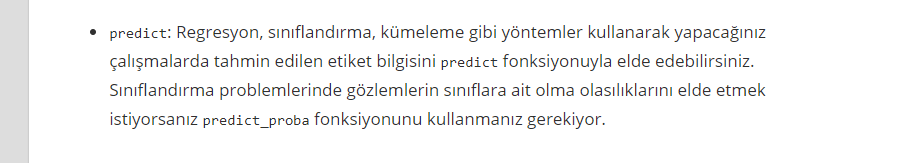
#### **Step 3: Specify and Fit Model:**

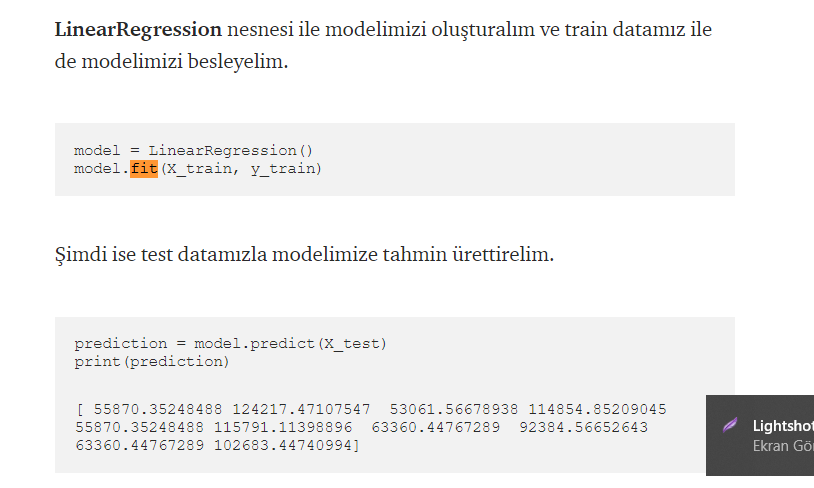
#### 

#### 

#### **Step 4: Make Predictions: (Tahmin Yapma)**

**Notlar:**





## Model Validation(Model geçerliliği):

Bir model oluşturdunuz. Ama ne kadar iyi?

Bu derste, modelinizin kalitesini ölçmek için model doğrulamayı kullanmayı öğreneceksiniz.

Model kalitesini ölçmek, modellerinizi tekrar tekrar geliştirmenin anahtarıdır.

### What is Model Validation: (Model Validation Nedir)

Oluşturduğunuz hemen hemen her modeli değerlendirmek isteyeceksiniz. Çoğu (hepsi olmasa da) uygulamada, model kalitesinin ilgili ölçüsü tahmini doğruluktur. Başka bir deyişle, modelin tahminleri gerçekte olanlara yakın olacak mı?

Birçok kişi tahmini doğruluğu ölçerken büyük bir hata yapar. "Training data" ile tahminler yaparlar ve bu tahminleri "Tranining data" daki hedef değerlerle karşılaştırırlar.

Bu yaklaşımla ilgili sorunu ve bir anda nasıl çözüleceğini göreceksiniz, ancak önce bunu nasıl yapacağımızı düşünelim.

Önce model kalitesini anlaşılabilir bir şekilde özetlemeniz gerekir. 10.000 ev için tahmini ve gerçek ev değerlerini karşılaştırırsanız, muhtemelen iyi ve kötü tahminlerin bir karışımını bulacaksınız. 10.000 tahmini ve gerçek değerin listesine bakmak anlamsız olacaktır. Bunu tek bir metrik olarak özetlemeliyiz.

Model kalitesini özetlemek için birçok metrik vardır, ancak **Mean Absolute Error**(ortalama mutlak hata) **(MAE** olarak da adlandırılır) olarak adlandırılan biriyle başlayacağız.

Bu metriği son kelimeyle başlayarak parçalayalım, hata.

Her ev için tahmin hatası:

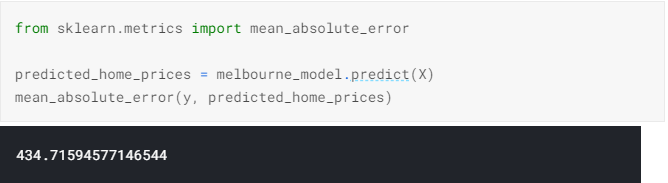


Yani, eğer bir ev 150.000 dolara mal olursa ve bunun 100.000 dolara mal olacağını tahmin ettiyseniz, hata 50.000 dolar. MAE metriği ile, her hatanın mutlak değerini alırız. Bu, her hatayı pozitif bir sayıya dönüştürür. Daha sonra bu mutlak hataların ortalamasını alırız. Bu bizim model kalitesinin ölçüsüdür. Düz İngilizce olarak, şu şekilde söylenebilir:

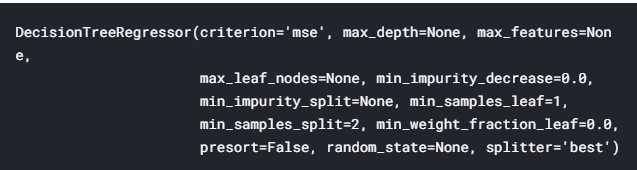
Ortalama olarak, tahminlerimiz yaklaşık X civarında.( On average, our predictions are off by about X.)

MAE'Yİ hesaplamak için önce bir modele ihtiyacımız var. Bu, code düğmesine tıklayarak inceleyebileceğiniz aşağıdaki gizli bir hücreye yerleştirilmiştir.

Bir modelimiz olduğunda, ortalama mutlak hatayı nasıl hesaplarız:







### The Problem with "In-Sample" Scores("In-Sample(Örnek İçi)" Puanlarla İlgili Sorun)

Sadece hesapladığımız ölçü "örnek" puanı olarak adlandırılabilir. Hem modeli oluşturmak hem de değerlendirmek için tek bir "örnek" ev kullandık. İşte bu yüzden kötü.

Büyük emlak piyasasında, kapı rengi ev fiyatı ilgisiz olduğunu düşünün. Ancak, modeli oluşturmak için kullandığınız veri örneğinde, yeşil Kapılı tüm evler çok pahalıydı. Modelin işi, ev fiyatlarını tahmin eden desenleri(patterns) bulmaktır, bu yüzden bu deseni görecektir ve her zaman yeşil Kapılı evler için yüksek fiyatları tahmin edecektir.

Bu model eğitim verilerinden türetildiğinden, model eğitim verilerinde doğru görünecektir.

Ancak, model yeni veriler gördüğünde bu örüntü tutmazsa, model pratikte kullanıldığında çok yanlış olur.

Modellerin pratik değeri yeni veriler üzerinde tahmin yapmaktan geldiğinden, modeli oluşturmak için kullanılmayan veriler üzerindeki performansı ölçüyoruz.

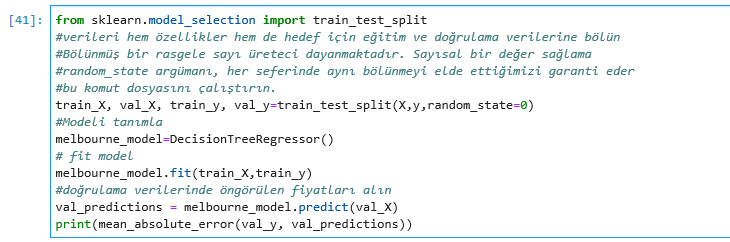
Bunu yapmanın en basit yolu, bazı verileri model oluşturma sürecinden dışlamak ve daha önce görmediği veriler üzerindeki modelin doğruluğunu test etmek için bunları kullanmaktır.Bu verilere **validation data** denir.

### Coding It

Scikit-learn Kütüphanesi, verileri iki parçaya bölmek için train\_test\_split işlevine sahiptir.

Bu verilerin bir kısmını modele uyacak şekilde eğitim verileri olarak kullanacağız ve diğer verileri mean\_absolute\_error hesaplamak için doğrulama verileri olarak kullanacağız.

Here is the code:





### Wow!

Örnek içi veriler için ortalama mutlak hatanız yaklaşık 500 dolardı. Örnek dışı 250.000 dolardan fazla.

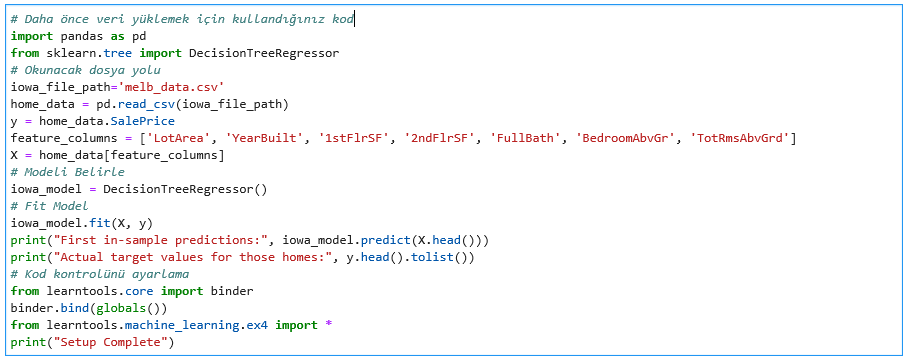
Bu, neredeyse tamamen doğru olan bir model ile en pratik amaçlar için kullanılamayan bir model arasındaki farktır. Bir referans noktası olarak, doğrulama verilerindeki ortalama ev değeri 1,1 milyon dolar. Yani yeni verilerdeki hata ortalama ev değerinin dörtte biri kadardır.

Daha iyi özellikler veya farklı model türleri bulmak için deneme yapmak gibi bu modeli geliştirmenin birçok yolu vardır.

### Exercises: Model Validation

Bir model yaptın. Bu alıştırmada modelinizin ne kadar iyi olduğunu test edeceksiniz.

Önceki alıştırmanın kaldığı kodlama ortamınızı ayarlamak için aşağıdaki hücreyi çalıştırın.



#### **Step 1: Split Your Data(Verilerinizi Bölün)**

Verilerinizi bölmek için train\_test\_split işlevini kullanın.

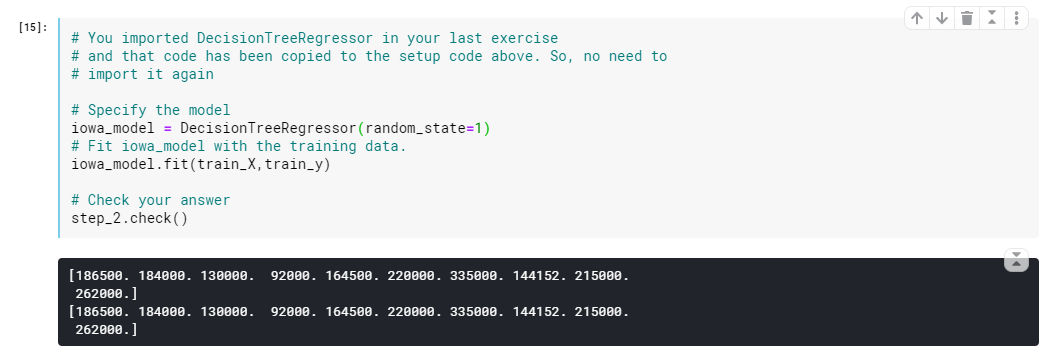
Random\_state = 1 argümanını verin, böylece kontrol fonksiyonları kodunuzu doğrularken ne bekleyeceğini bilir.

Geri çağırma, Özellikleri Veri Çerçevesi x yüklenir ve hedef yüklenir senin.

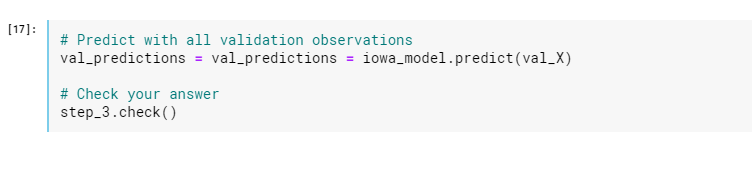


#### **Step 2: Specify and Fit the Model**

Bir DecisionTreeRegressor modeli oluşturun ve ilgili verilere uydurun. Modeli oluştururken random\_state öğesini tekrar 1 olarak ayarlayın.



#### **Step 3: Make Predictions with Validation data(Doğrulama verileriyle tahminler yapın)**



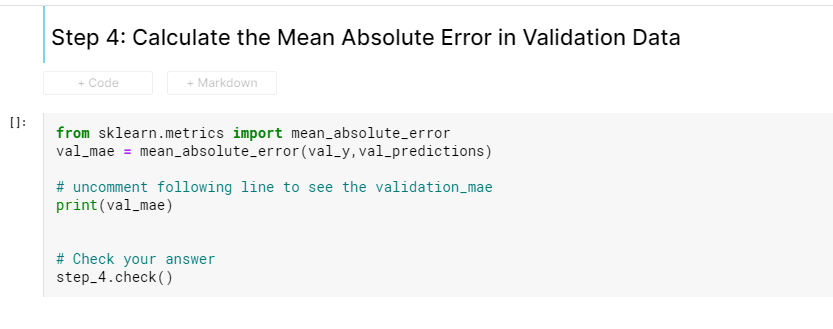
Doğrulama verilerinden tahminlerinizi ve gerçek değerlerinizi inceleyin.



İçinde gördüğünüzden farklı olan ne fark edersiniz-örnek tahminler (bu sayfadaki en üst kod hücresinden sonra yazdırılır).

Doğrulama tahminlerinin neden örnek içi (veya eğitim) tahminlerinden farklı olduğunu hatırlıyor musunuz? Bu son dersten önemli bir fikir.

#### **Step 4: Calculate the Mean Absolute Error in Validation Data(doğrulama verilerinde ortalama mutlak hatayı hesaplayın)**



MAE sonucu iyi mi? Uygulamalar arasında geçerli olan değerlerin genel bir kuralı yoktur. Ancak bir sonraki adımda bu sayının nasıl kullanılacağını (ve geliştirileceğini) göreceksiniz.

## Underfitting and Overfitting

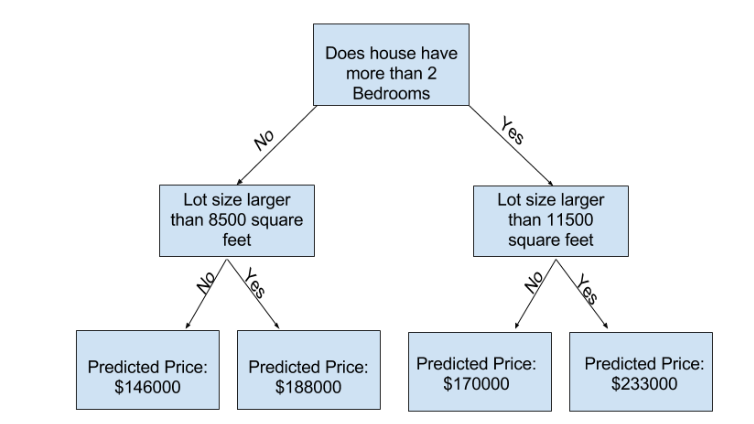
Bu adımın sonunda, uygun olmayan ve fazla uygunluk kavramlarını anlayacak ve modellerinizi daha doğru hale getirmek için bu fikirleri uygulayabileceksiniz.

### Experimenting With Different Models

Artık model doğruluğunu ölçmenin güvenilir bir yoluna sahip olduğunuza göre, alternatif modelleri deneyebilir ve hangisinin en iyi tahminleri verdiğini görebilirsiniz. Peki modeller için hangi alternatifleriniz var?

Scikit-learn documentation'da(<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>)  
karar ağacı modelinin birçok seçeneğe sahip olduğunu görebilirsiniz(uzun süre isteyeceğinizden veya ihtiyaç duyacağınızdan daha fazla).

En önemli seçenekler ağacın derinliğini belirler. Bu mikro kurstaki ilk dersten, bir ağacın derinliğinin, bir tahmine gelmeden önce kaç bölmenin yaptığının bir ölçüsü olduğunu hatırlayın. Bu nispeten derin olmayan bir ağaçtır.



Uygulamada, bir ağacın üst seviye (tüm evler) ve bir yaprak arasında 10 bölünmesi nadir değildir.

Uygulamada, bir ağacın üst seviye (tüm evler) ve bir yaprak arasında 10 bölünmesi nadir değildir. Ağaç derinleştikçe, veri kümesi daha az ev ile yapraklara dilimlenir. Bir ağacın yalnızca 1 bölünmesi varsa, verileri 2 gruba böler. Her grup tekrar bölünürse, 4 grup ev alırız. Bunların her birini tekrar bölmek 8 grup oluşturacaktır. Her seviyede daha fazla bölme ekleyerek grup sayısını ikiye katlamaya devam edersek, 10. seviyeye geldiğimizde 210 grup evimiz olacak. 1024 yaprak.

When we divide the houses amongst many leaves, we also have fewer houses in each leaf. Leaves with very few houses will make predictions that are quite close to those homes' actual values, but they may make very unreliable predictions for new data (because each prediction is based on only a few houses).

Bu, bir modelin eğitim verileriyle neredeyse mükemmel bir şekilde eşleştiği, ancak doğrulama ve diğer yeni verilerde zayıf olduğu overfitting adlı bir olgudur.

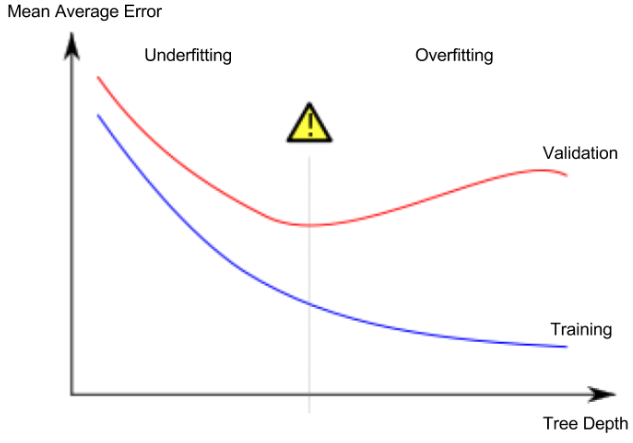
Ters tarafta, eğer ağacımızı çok yüzeysel yaparsak, evleri çok farklı gruplara ayırmaz.

Aşırı derecede, eğer bir ağaç evleri sadece 2 veya 4'e bölerse, her grup hala çok çeşitli evlere sahiptir.

Ortaya çıkan tahminler, eğitim verilerinde bile çoğu ev için çok uzak olabilir (ve aynı nedenden dolayı doğrulamada da kötü olacaktır). Bir model verilerde önemli ayrımlar(import distinctions) ve desenler(patterns) yakalamak için başarısız olduğunda, bu yüzden bile eğitim verilerinde(training data) kötü performans, bu underfitting denir.

Doğrulama verilerimizden tahmin ettiğimiz yeni veriler üzerindeki doğruluğu önemsediğimizden, underfitting ve overfitting arasındaki en etkili noktayı(sweet spot) bulmak istiyoruz.

Görsel olarak, (kırmızı) doğrulama eğrisinin düşük noktasını istiyoruz



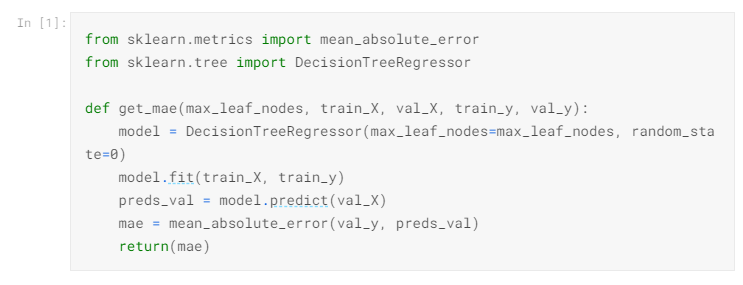
### Examples:

Ağaç derinliğini kontrol etmek için birkaç alternatif vardır ve birçoğu ağaçtaki bazı rotaların diğer rotalardan daha fazla derinliğe sahip olmasına izin verir.

Ancak max\_leaf\_nodes argümanı overfitting vs underfitting kontrol etmek için çok mantıklı bir yol sağlar.

Modelin yapmasına izin verdiğimiz daha fazla yaprak, yukarıdaki grafikteki underfitting alanından overfitting alanına daha fazla hareket ederiz.

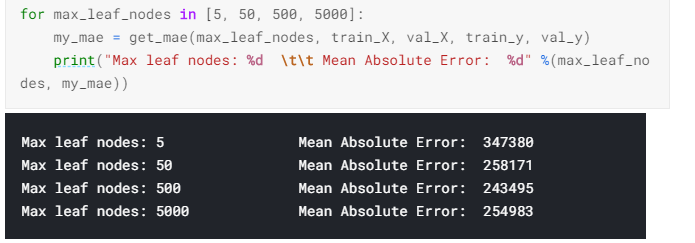
Max\_leaf\_nodes için farklı değerlerden Mae puanlarını karşılaştırmaya yardımcı olmak için bir yardımcı program işlevi kullanabiliriz:



Veriler, daha önce gördüğünüz (ve daha önce yazdığınız) kodu kullanarak train\_X, val\_X, train\_y ve val\_y içine yüklenir.

Max\_leaf\_nodes için farklı değerlerle oluşturulmuş modellerin doğruluğunu karşılaştırmak için bir for-loop kullanabiliriz.

# mae'yi farklı max\_leaf\_nodes değerleriyle karşılaştırın



Listelenen seçeneklerden 500, en uygun yaprak sayısıdır.

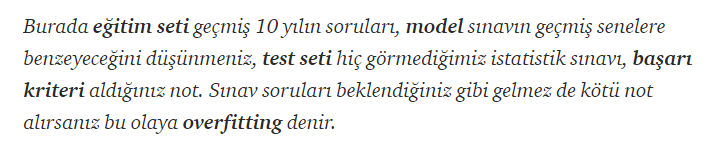
### Sonuç:

Modeller şunlardan herhangi birine sahip olabilir.

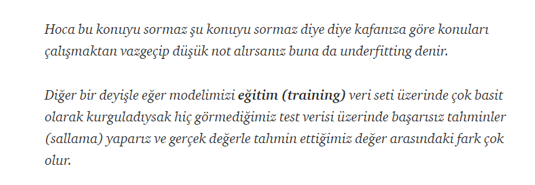
* Overfitting: gelecekte tekrarlamayacak sahte pattern(desen)leri yakalamak , daha az doğru tahminlere yol açmak veya
* Underfittin: alakalı pattern’leri yakalayamama, yine daha az doğru tahminlere yol açma.

Bir aday modelin doğruluğunu(accuracy) ölçmek için model eğitiminde(train) kullanılmayan doğrulama(validation) verilerini kullanıyoruz. Bu, birçok aday modeli denememize ve en iyisini elde etmemizi sağlar.

**Overfitting:**



**Underfitting:**



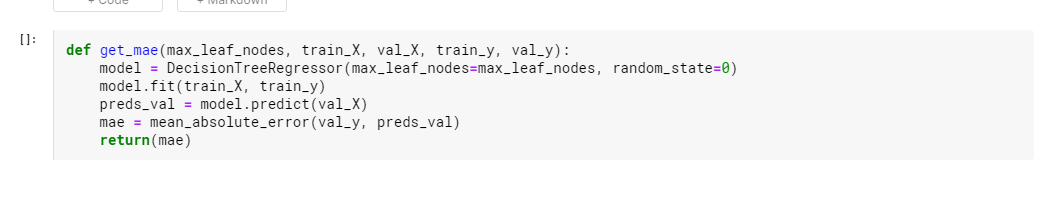
### Exercise: Underfitting and Overfitting

#### **Tekrarlamak:**

İlk modelinizi oluşturdunuz ve şimdi daha iyi tahminler yapmak için ağacın boyutunu optimize etme zamanı. Önceki adımı bıraktığınız yerde kodlama ortamınızı ayarlamak için bu hücreyi çalıştırın.



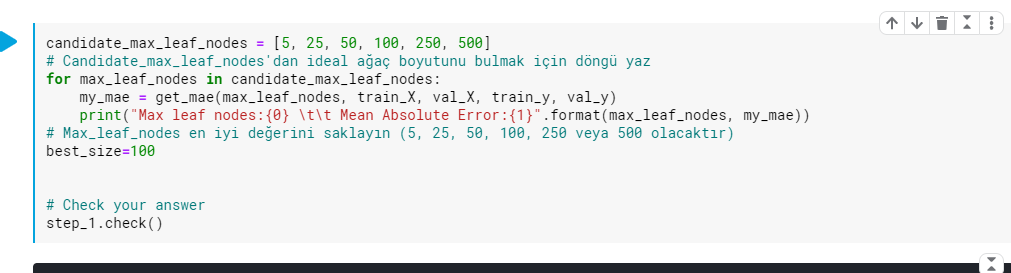
Get\_mae fonksiyonunu kendiniz yazabilirsiniz. Şimdilik tedarik edeceğiz. Bu, bir önceki derste okuduğunuz işlevle aynıdır. Aşağıdaki hücreyi çalıştırmanız yeterlidir.

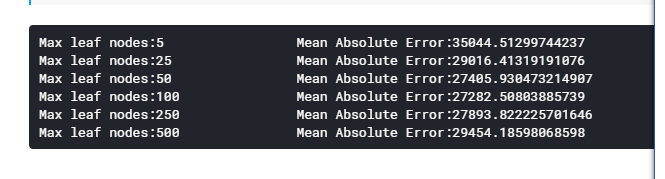


#### **Step 1: Compare Different Tree Sizes**

Bir dizi olası değerden max\_leaf\_nodes için aşağıdaki değerleri çalıştıran bir döngü yazın.

Her max\_leaf\_nodes değerinde get\_mae işlevini çağırın. Çıktıyı, verilerinizde en doğru modeli veren max\_leaf\_nodes değerini seçmenize izin verecek şekilde saklayın.

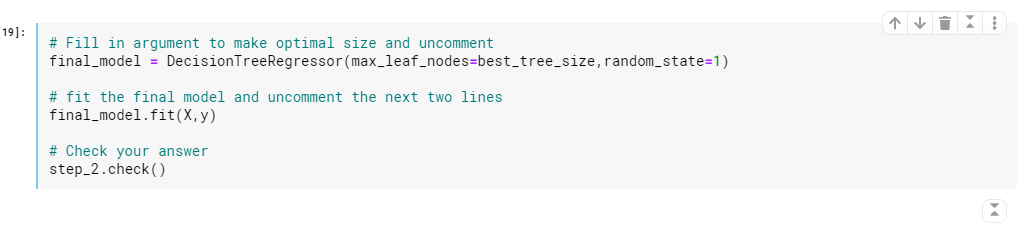




#### **Step 2: Fit Model Using All Data(Modeli tüm verileri kullanarak sığdır)**

En iyi ağaç boyutunu biliyorsun. Bu modeli pratikte deploy edecek olsaydınız, tüm verileri kullanarak ve bu ağaç boyutunu koruyarak daha da doğru hale getirirsiniz.

Yani, tüm modelleme kararlarınızı verdiğiniz için doğrulama verilerini saklamanız gerekmez.



Bu modeli ayarladınız ve sonuçlarınızı geliştirdiniz. Ancak hala modern makine öğrenimi standartlarına göre çok karmaşık olmayan Decision Tree modellerini kullanıyoruz. Bir sonraki adımda, modellerinizi daha da geliştirmek için Random Forest kullanmayı öğreneceksiniz.

## Random Forests:

### Giriş:

Decision Tree sizi zor bir kararla baş başa bırakır. Çok sayıda yapraklı derin bir ağaç, her tahmin, yaprağındaki sadece birkaç evden gelen tarihsel verilerden geldiğinden fazla olacaktır. Ancak, az yapraklı sığ bir ağaç kötü performans gösterecektir, çünkü ham verilerdeki birçok farklılığı yakalayamaz.

Günümüzün en sofistike modelleme teknikleri bile, underfitting ve overfitting arasındaki bu gerilim ile karşı karşıyadır.

Ancak, birçok model daha iyi performans sağlayabilecek akıllı fikirlere sahiptir. Örnek olarak **Random Forest’a** bakacağız.

Random Forest birçok ağaç kullanır ve her bileşen ağacının tahminlerini ortalayarak bir tahmin yapar.

Genellikle tek bir karar ağacından çok daha iyi tahmin doğruluğu(predictive accuracy) vardır ve varsayılan parametrelerle iyi çalışır.

Modellemeye devam ederseniz, daha iyi performansa sahip daha fazla model öğrenebilirsiniz, ancak bunların çoğu doğru parametreleri almaya duyarlıdır.

### Example

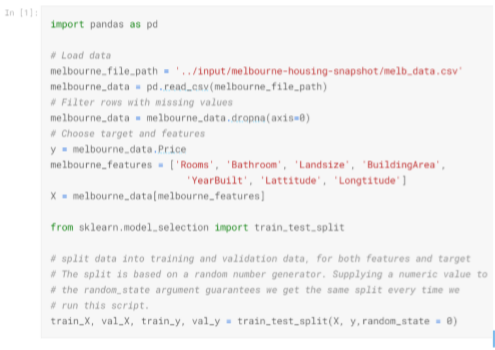
Verileri yüklemek için gereken kodu zaten birkaç kez gördünüz. Veri yüklemenin sonunda aşağıdaki değişkenler bulunur:

• train\_X

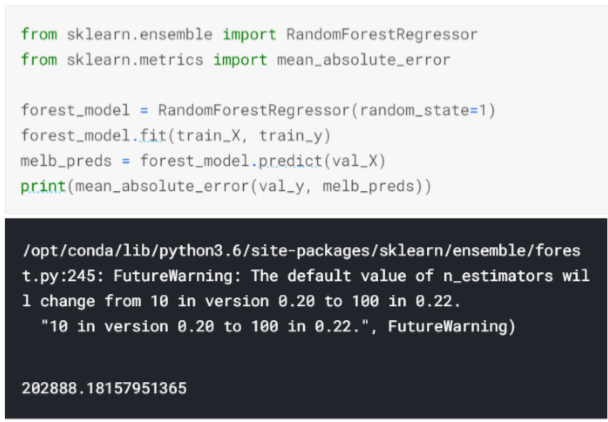
• val\_X

• train\_y

• val\_y



scikit-learn kütüphanesinde decision tree modeli oluşturduğumuz gibi bu kez random forest modeli oluşturacağız. – **DecisionTreeRegressor** yerine **RandomTreeRegressor** kullanacağız.



### Sonuç:

Daha da iyileştirilmesi muhtemeldir, ancak bu 250.000 olan en iyi karar ağacı hatası üzerinde büyük bir gelişmedir.

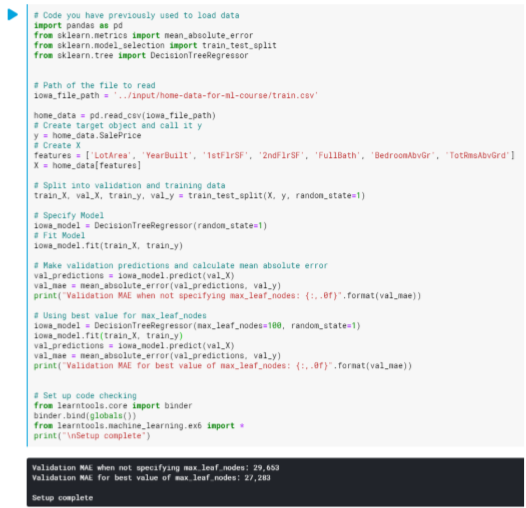
Single decision tree’nin maksimum derinliğini değiştirdiğimiz gibi Random Forest'ın da performansını değiştirmenize izin veren parametreler var.

Ancak Random Forest modellerinin en iyi özelliklerinden biri, bu ayarlama olmadan bile genellikle makul bir şekilde çalışmasıdır.

Yakında, doğru parametrelerle iyi ayarlandığında daha iyi performans sağlayan (ancak doğru model parametrelerini elde etmek için biraz beceri gerektiren) XGBoost modelini öğreneceksiniz.

### Exercises: Random Forest

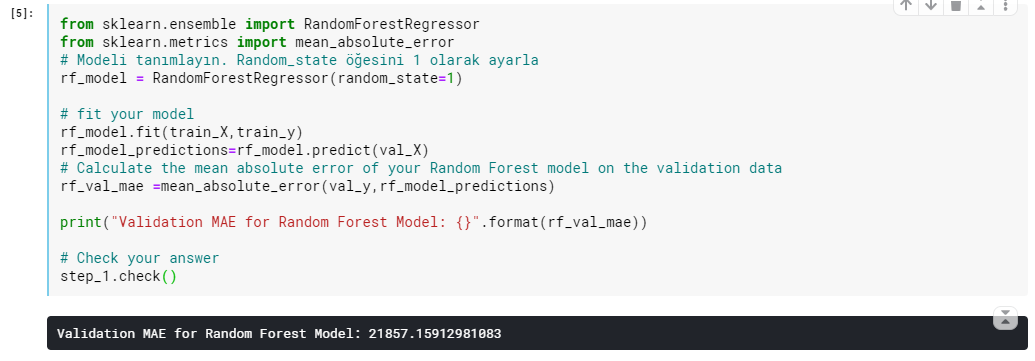
Şimdiye kadar yazdığımız kod:



#### **Exercises**

Veri bilimi her zaman bu kadar kolay değildir. Ancak Decision Tree’yi Random Forest ile değiştirmek kolay bir kazanç olacaktır.

#### **Step 1: Use a Random Forest**



Şimdiye kadar, projenizin her adımında belirli talimatları izlediniz. Bu, temel fikirleri öğrenmeye ve ilk modelinizi oluşturmaya yardımcı oldu, ancak şimdi işleri kendi başınıza denemek için yeterince bilgi sahibisiniz.

Machine Learning yarışmaları, bağımsız olarak bir machine learning projesinde gezinirken kendi fikirlerinizi denemek ve daha fazla bilgi edinmek için harika bir yoldur.

## Exercises: Machine Learning Competitions

### Introduction

Makine öğrenimi yarışmaları, veri bilimi becerilerinizi geliştirmenin ve ilerlemenizi ölçmenin harika bir yoludur.

Bu alıştırmada, bir Kaggle yarışması için tahminler oluşturacak ve sunacaksınız.

Bu notebook’daki adımlar:

• Tüm verilerinizle Random Forest modeli oluşturun. (X ve y)

• Target(hedef) içermeyen “test” verilini okuyun. Random Forest modelinizle test verilerindeki ev fiyatlarını tahmin edin.

• Bu tahminleri yarışmaya gönderin ve puanınızı görün.

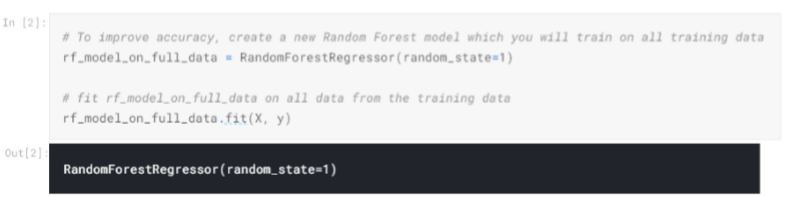
• İsteğe bağlı olarak, feature’lar ekleyerek veya modelinizi değiştirerek modelinizi geliştirip geliştiremeyeceğinizi görmek için tekrar deneyin. Daha sonra bunun rekabet lider panosunda nasıl etkilediğini görmek için yeniden gönderebilirsiniz.

Şimdiye kadar yazdığımız kod:



### Creating a Model For the Competition

Random Forest modeli oluşturun ve tüm X ve y ile modeli eğitin.



### Make Predictions

"Test" verileri dosyasını okuyun. Tahmin yapmak için modelinizi uygulayın.



Modelinizi geliştirmenin birçok yolu vardır ve deneme yapmak bu noktada öğrenmenin harika bir yoludur. Modelinizi geliştirmenin en iyi yolu özellikler eklemektir. Sütun listesine bakın ve konut fiyatlarını nelerin etkileyebileceğini düşünün. Bazı özellikler, eksik değerler veya sayısal olmayan veri türleri gibi sorunlar nedeniyle hatalara neden olur.

## 

# KAYNAKLAR

* Kaggle – Intro to Machine Learnin Course

<https://www.kaggle.com/learn/intro-to-machine-learning>

* <https://medium.com/data-science-tr/overfitting-underfitting-cross-validation-b47dfda0cf4e>
* <http://www.veridefteri.com/2017/11/23/scikit-learn-ile-veri-analitigine-giris/>

#### 