Selecting Data for Modeling

Veri setiniz, kavramak veya hatta düzgün bir şekilde yazdırmak için **çok fazla değişkene** sahipti. Bu bunaltıcı miktardaki veriyi anlayabileceğiniz bir şeye nasıl dönüştürebilirsiniz?

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

Değişkenleri/sütunları seçmek için, veri setindeki tüm sütunların bir listesini görmemiz gerekecek. Bu, DataFrame'in **columns (sütunlar) özelliği** ile yapılır (aşağıdaki kodun en alt satırı).

import pandas as pd

melbourne\_file\_path = '../input/melbourne-housing-snapshot/melb\_data.csv'

melbourne\_data = pd.read\_csv(melbourne\_file\_path)

melbourne\_data.columns

Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',

'Date', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',

'Landsize', 'BuildingArea', 'YearBuilt', 'CouncilArea', 'Lattitude',

'Longtitude', 'Regionname', 'Propertycount'],

dtype='object')

*# The Melbourne data has some missing values (some houses for which some variables weren't recorded.)*

*# We'll learn to handle missing values in a later tutorial.*

*# Your Iowa data doesn't have missing values in the columns you use.*

*# So we will take the simplest option for now, and drop houses from our data.*

*# Don't worry about this much for now, though the code is:*

*# dropna drops missing values (think of na as "not available")*

melbourne\_data = melbourne\_data.dropna(axis=0)

Verilerinizin bir alt kümesini seçmenin birçok yolu vardır. **Pandas kursu** bunları daha derinlemesine ele alır, ancak şimdilik iki yaklaşıma odaklanacağız:

* **Nokta notasyonu**: "Tahmin hedefi"ni seçmek için kullandığımız yöntem.
* **Sütun listesi ile seçim**: "Özellikleri" seçmek için kullandığımız yöntem.

## **Selecting The Prediction Target**

Bir değişkeni **nokta notasyonu** ile çekebilirsiniz. Bu tek sütun, geniş anlamda tek bir veri sütununa sahip bir DataFrame'e benzeyen bir **Series**'de depolanır.

Nokta notasyonunu, tahmin etmek istediğimiz sütunu, yani **tahmin hedefini** seçmek için kullanacağız. Geleneksel olarak, tahmin hedefine **y** denir. Bu nedenle, Melbourne verilerindeki ev fiyatlarını kaydetmek için ihtiyacımız olan kod:

y = melbourne\_data.Price

# Choosing "Features"

Modelimize girdi olarak verilen (ve daha sonra tahmin yapmak için kullanılan) sütunlara **"özellikler"** denir. Bizim durumumuzda bunlar, ev fiyatını belirlemek için kullanılan sütunlar olacaktır. Bazen hedef sütun dışındaki tüm sütunları özellik olarak kullanırsınız. Diğer zamanlarda ise daha az özellikle daha iyi sonuçlar alabilirsiniz.

Şimdilik, yalnızca birkaç özellikle bir model oluşturacağız. Daha sonra, farklı özelliklerle oluşturulan modelleri nasıl yineleyeceğinizi ve karşılaştıracağınızı göreceksiniz.

Birden fazla özelliği, köşeli parantez içinde bir sütun adı **listesi** sağlayarak seçeriz. Bu listedeki her öğe bir **dize** (tırnak içinde) olmalıdır.

melbourne\_features = ['Rooms', 'Bathroom', 'Landsize', 'Lattitude', 'Longtitude']

Geleneksel olarak bu veriye X adı verilir.

X = melbourne\_data[melbourne\_features]

Ev fiyatlarını describe metodu ve en üstteki birkaç satırı gösteren head metodunu kullanarak tahmin etmek için kullanacağımız verilere hızlıca bir göz atalım.

X.describe()

| Rooms | Bathroom | Landsize | Lattitude | Longtitude |
| --- | --- | --- | --- | --- |
| count | 6196.000000 | 6196.000000 | 6196.000000 | 6196.000000 | 6196.000000 |
| mean | 2.931407 | 1.576340 | 471.006940 | -37.807904 | 144.990201 |
| std | 0.971079 | 0.711362 | 897.449881 | 0.075850 | 0.099165 |
| min | 1.000000 | 1.000000 | 0.000000 | -38.164920 | 144.542370 |
| 25% | 2.000000 | 1.000000 | 152.000000 | -37.855438 | 144.926198 |
| 50% | 3.000000 | 1.000000 | 373.000000 | -37.802250 | 144.995800 |
| 75% | 4.000000 | 2.000000 | 628.000000 | -37.758200 | 145.052700 |
| max | 8.000000 | 8.000000 | 37000.000000 | -37.457090 | 145.526350 |

X.head()

| Rooms | Bathroom | Landsize | Lattitude | Longtitude |
| --- | --- | --- | --- | --- |
| 1 | 2 | 1.0 | 156.0 | -37.8079 | 144.9934 |
| 2 | 3 | 2.0 | 134.0 | -37.8093 | 144.9944 |
| 4 | 4 | 1.0 | 120.0 | -37.8072 | 144.9941 |
| 6 | 3 | 2.0 | 245.0 | -37.8024 | 144.9993 |
| 7 | 2 | 1.0 | 256.0 | -37.8060 | 144.9954 |

Verilerinizi bu komutlarla **görsel olarak kontrol etmek**, bir veri bilimcisinin işinin önemli bir parçasıdır. Veri setinde sık sık daha fazla incelemeyi hak eden **sürprizlerle** karşılaşırsınız.

# Building Your Model[¶](https://www.kaggle.com/code/dansbecker/your-first-machine-learning-model#Building-Your-Model)

Modellerinizi oluşturmak için **scikit-learn** kütüphanesini kullanacaksınız. Kod yazarken, örnek kodda göreceğiniz gibi bu kütüphane **sklearn** olarak yazılır. Scikit-learn, genellikle DataFrame'lerde depolanan veri türlerini modellemek için açık ara en popüler kütüphanedir.

Model oluşturma ve kullanma adımları şunlardır:

* **Tanımla (Define):** Modelin türü ne olacak? Bir karar ağacı mı? Başka bir model türü mü? Model türünün bazı diğer parametreleri de bu adımda belirtilir.
* **Uydur (Fit):** Sağlanan verilerden kalıpları yakalar. Bu, modellemenin kalbidir.
* **Tahmin Et (Predict):** Adından da anlaşılacağı gibi, tahmin yapar.
* **Değerlendir (Evaluate):** Modelin tahminlerinin ne kadar doğru olduğunu belirler.

İşte scikit-learn ile bir karar ağacı modelini tanımlama ve onu özellikler ve hedef değişkenle uydurma örneği:

from sklearn.tree import DecisionTreeRegressor

*# Define model. Specify a number for random\_state to ensure same results each run*

melbourne\_model = DecisionTreeRegressor(random\_state=1)

*# Fit model*

melbourne\_model.fit(X, y)

DecisionTreeRegressor(random\_state=1)

Birçok makine öğrenimi modeli, model eğitiminde bir miktar **rastgeleliğe** 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, seçtiğiniz değerden önemli ölçüde etkilenmez.

Artık tahmin yapmak için kullanabileceğimiz **uygun bir modelimiz** var.

Uygulamada, fiyatlarını zaten bildiğimiz evler yerine piyasaya çıkan **yeni evler için tahminler** yapmak isteyeceksiniz. Ancak, predict fonksiyonunun nasıl çalıştığını görmek için eğitim verilerinin ilk birkaç satırı için tahminler yapacağız.

print("Making predictions for the following 5 houses:")

print(X.head())

print("The predictions are")

print(melbourne\_model.predict(X.head()))

Making predictions for the following 5 houses:

Rooms Bathroom Landsize Lattitude Longtitude

1 2 1.0 156.0 -37.8079 144.9934

2 3 2.0 134.0 -37.8093 144.9944

4 4 1.0 120.0 -37.8072 144.9941

6 3 2.0 245.0 -37.8024 144.9993

7 2 1.0 256.0 -37.8060 144.9954

The predictions are

[1035000. 1465000. 1600000. 1876000. 1636000.]