Introduction

Bu ders ve bir sonraki ders, **gözetimsiz öğrenme (unsupervised learning)** algoritmaları olarak bilinenlerden faydalanacaktır. Gözetimsiz algoritmalar bir hedef kullanmaz; bunun yerine, amaçları verinin bir özelliğini öğrenmek, özelliklerin yapısını belirli bir şekilde temsil etmektir. Tahmin için **özellik mühendisliği (feature engineering)** bağlamında, gözetimsiz bir algoritmayı bir "**özellik keşfi**" tekniği olarak düşünebilirsiniz.

**Kümeleme (Clustering)**, basitçe veri noktalarını birbirine ne kadar benzer olduklarına göre gruplara ayırma anlamına gelir. Bir kümeleme algoritması, tabiri caizse, "**aynı tüye sahip kuşları bir araya getirir.**"

Özellik mühendisliği için kullanıldığında, örneğin bir pazar segmentini temsil eden müşteri gruplarını veya benzer hava koşullarını paylaşan coğrafi bölgeleri keşfetmeye çalışabiliriz. Küme etiketlerinden oluşan bir özellik eklemek, makine öğrenimi modellerinin karmaşık mekansal veya yakınlık ilişkilerini çözmesine yardımcı olabilir.

# Cluster Labels as a Feature[¶](https://www.kaggle.com/code/ryanholbrook/clustering-with-k-means" \l "Cluster-Labels-as-a-Feature" \t "_self)

Tek bir gerçek değerli özelliğe uygulandığında, kümeleme, geleneksel bir "**bölme (binning)**" veya "**ayrıklaştırma (discretization)**" dönüşümü gibi davranır. Birden fazla özellikte ise, "**çok boyutlu bölme (multi-dimensional binning)**" (bazen **vektör niceleme (vector quantization)** olarak da adlandırılır) gibidir.

Bir veri çerçevesine eklendiğinde, küme etiketlerinin bir özelliği şu şekilde görünebilir:

| Longitude | Latitude | Cluster |
| --- | --- | --- |
| -93.619 | 42.054 | 3 |
| -93.619 | 42.053 | 3 |
| -93.638 | 42.060 | 1 |
| -93.602 | 41.988 | 0 |

Şunu hatırlamak önemlidir: bu **Küme (Cluster)** özelliği kategoriktir. Burada, tipik bir kümeleme algoritmasının üreteceği şekilde bir etiket kodlaması (yani, bir tam sayı dizisi olarak) ile gösterilmiştir; modelinize bağlı olarak, bir **tek-sıcak kodlama (one-hot encoding)** daha uygun olabilir.

Küme etiketlerini eklemenin temel fikri, kümelerin özellikler arasındaki karmaşık ilişkileri daha basit parçalara ayırmasıdır. Böylece modelimiz, karmaşık bütünü bir kerede öğrenmek yerine, daha basit parçaları tek tek öğrenebilir. Bu bir "**böl ve fethet (divide and conquer)**" stratejisidir.

Şekil, kümelemenin basit bir doğrusal modeli nasıl iyileştirebileceğini gösteriyor. **"Yapım Yılı" (YearBuilt)** ve **"Satış Fiyatı" (SalePrice)** arasındaki eğrisel ilişki, bu tür bir model için çok karmaşık, bu yüzden model **yetersiz uyum (underfits)** gösteriyor. Ancak, daha küçük parçalar üzerinde ilişki **neredeyse** doğrusaldır ve bu, modelin kolayca öğrenebileceği bir durumdur.

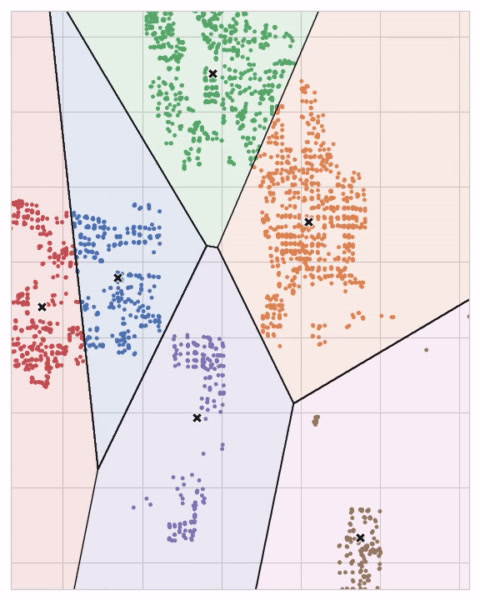
# k-Means Clustering

Pek çok kümeleme algoritması vardır. Bunlar, temel olarak "benzerliği" veya "yakınlığı" nasıl ölçtüklerine ve hangi tür özelliklerle çalıştıklarına göre farklılık gösterir. Kullanacağımız **k-ortalamalar (k-means)** algoritması, sezgiseldir ve özellik mühendisliği bağlamında uygulanması kolaydır. Uygulamanıza bağlı olarak, başka bir algoritma daha uygun olabilir.

K-ortalamalar kümelemesi, benzerliği sıradan düz çizgi mesafesini (başka bir deyişle, **Öklid mesafesini**) kullanarak ölçer. Özellik uzayının içine **merkez noktaları (centroids)** adı verilen bir dizi nokta yerleştirerek kümeler oluşturur. Veri kümesindeki her bir nokta, kendisine en yakın olan merkez noktasının kümesine atanır. "K-ortalamalar"daki "k", oluşturduğu merkez noktası (yani küme) sayısıdır. K'yi siz belirlersiniz.

Her merkez noktasının, bir dizi ışınsal çember aracılığıyla noktaları yakaladığını hayal edebilirsiniz. Rakip merkez noktalarından gelen çember kümeleri üst üste geldiğinde bir çizgi oluştururlar. Bunun sonucunda **Voronoi döşemesi (Voronoi tessallation)** adı verilen bir desen ortaya çıkar. Döşeme, gelecekteki verilerin hangi kümelere atanacağını gösterir; döşeme, esasen k-ortalamaların eğitim verilerinden öğrendiği şeydir.

Yukarıdaki Ames veri kümesindeki kümeleme bir k-ortalamalar kümelemesidir. İşte aynı şeklin döşeme ve merkez noktaları gösterilmiş hali.



# K-Means Algoritmasının Öğrenme Süreci

Bu bölümde, **k-ortalamalar (k-means)** algoritmasının kümeleri nasıl öğrendiğini ve bunun **özellik mühendisliği (feature engineering)** için ne anlama geldiğini inceleyeceğiz. **Scikit-learn**'ün uygulamasındaki üç parametreye odaklanacağız: n\_clusters, max\_iter ve n\_init.

### Algoritmanın İşleyişi

Bu basit, iki adımlı bir süreçtir. Algoritma, önceden tanımlanmış bir sayıda (n\_clusters) merkez noktasını rastgele başlatarak başlar. Ardından bu iki işlem üzerinde yineleme yapar:

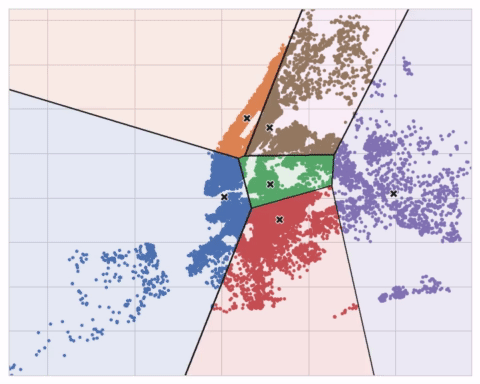
1. Noktaları en yakın küme merkez noktasına atar.
2. Her bir merkez noktasını, noktalarına olan mesafeyi en aza indirecek şekilde hareket ettirir.

Merkez noktaları artık hareket etmeyene veya maksimum bir yineleme sayısı (max\_iter) geçene kadar bu iki adımı tekrar eder.

### Tekrarlı Başlatmanın Önemi

Çoğu zaman, merkez noktalarının başlangıçtaki rastgele konumu kötü bir kümelemeyle sonuçlanır. Bu nedenle, algoritma belirli bir sayıda (n\_init) kez tekrarlanır ve her bir nokta ile merkez noktası arasındaki toplam mesafenin en az olduğu kümelemeyi, yani **en uygun kümelemeyi** döndürür.

Aşağıdaki animasyon, algoritmanın işleyişini gösteriyor. Sonucun başlangıç merkez noktalarına olan bağımlılığını ve yakınsayana kadar yinelemenin önemini ortaya koymaktadır.



Büyük sayıda küme için max\_iter'i veya karmaşık bir veri kümesi için n\_init'i artırmanız gerekebilir. Ancak, genellikle kendinizin seçmesi gereken tek parametre n\_clusters'dır (yani k). Bir özellik kümesi için en iyi bölme, kullandığınız modele ve neyi tahmin etmeye çalıştığınıza bağlıdır; bu nedenle, onu herhangi bir hiperparametre gibi ayarlamak en iyisidir (örneğin, çapraz doğrulama yoluyla).

# Example - California Housing

Mekansal özellikler olarak, **California Konut**'unun **'Enlem' (Latitude)** ve **'Boylam' (Longitude)**'ı k-ortalamalar kümelemesi için doğal adaylardır. Bu örnekte, Kaliforniya'nın farklı bölgelerinde ekonomik segmentler oluşturmak için bunları **'MedInc' (medyan gelir)** ile kümeleyeceğiz.

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from sklearn.cluster import KMeans

plt.style.use("seaborn-whitegrid")

plt.rc("figure", autolayout=True)

plt.rc(

"axes",

labelweight="bold",

labelsize="large",

titleweight="bold",

titlesize=14,

titlepad=10,

)

df = pd.read\_csv("../input/fe-course-data/housing.csv")

X = df.loc[:, ["MedInc", "Latitude", "Longitude"]]

X.head()

/opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np\_minversion} and <{np\_maxversion}"

/tmp/ipykernel\_20/577165241.py:6: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecated since 3.6, as they no longer correspond to the styles shipped by seaborn. However, they will remain available as 'seaborn-v0\_8-<style>'. Alternatively, directly use the seaborn API instead.

plt.style.use("seaborn-whitegrid")

| MedInc | Latitude | Longitude |
| --- | --- | --- |
| 0 | 8.3252 | 37.88 | -122.23 |
| 1 | 8.3014 | 37.86 | -122.22 |
| 2 | 7.2574 | 37.85 | -122.24 |
| 3 | 5.6431 | 37.85 | -122.25 |
| 4 | 3.8462 | 37.85 | -122.25 |

K-ortalama kümeleme ölçeğe duyarlı olduğundan, uç değerlere sahip verileri yeniden ölçeklendirmek veya normalleştirmek iyi bir fikir olabilir. Özelliklerimiz zaten aşağı yukarı aynı ölçekte olduğundan, olduğu gibi bırakacağız.

*# Create cluster feature*

kmeans = KMeans(n\_clusters=6)

X["Cluster"] = kmeans.fit\_predict(X)

X["Cluster"] = X["Cluster"].astype("category")

X.head()

/opt/conda/lib/python3.10/site-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

| MedInc | Latitude | Longitude | Cluster |
| --- | --- | --- | --- |
| 0 | 8.3252 | 37.88 | -122.23 | 5 |
| 1 | 8.3014 | 37.86 | -122.22 | 5 |
| 2 | 7.2574 | 37.85 | -122.24 | 5 |
| 3 | 5.6431 | 37.85 | -122.25 | 5 |
| 4 | 3.8462 | 37.85 | -122.25 | 2 |

Şimdi bunun ne kadar etkili olduğunu görmek için birkaç grafiğe bakalım. İlk olarak, kümelerin coğrafi dağılımını gösteren bir dağılım grafiği. Görünüşe göre algoritma, kıyılardaki yüksek gelirli bölgeler için ayrı segmentler oluşturmuş.

sns.relplot(

x="Longitude", y="Latitude", hue="Cluster", data=X, height=6,

);

/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight

self.\_figure.tight\_layout(\*args, \*\*kwargs)

Bu veri setindeki hedef MedHouseVal'dir (ortanca ev değeri). Bu kutu grafikleri, hedefin her küme içindeki dağılımını göstermektedir. Kümeleme bilgilendiriciyse, bu dağılımlar çoğunlukla MedHouseVal genelinde ayrılmalıdır; ki gördüğümüz de budur.

X["MedHouseVal"] = df["MedHouseVal"]

sns.catplot(x="MedHouseVal", y="Cluster", data=X, kind="boxen", height=6);

/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight

self.\_figure.tight\_layout(\*args, \*\*kwargs)