Karar Ağaçlarına Giriş

Kursa Başlamadan Önce

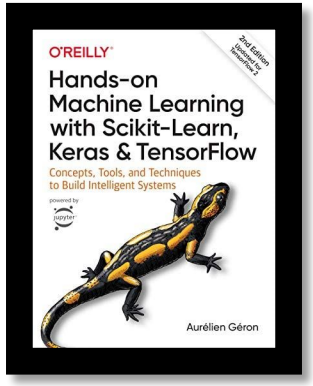
- Bu kurs tamamlandığı takdirde giriş düzeyi yapay zeka algoritmaları ve veri analizine temel oluşturabilecek genel bilgileri edinmiş olacaksınız
- Spesifik konular anlaşılması zor ve kişide ders esnasında mantığın oturması kolay olmayacağından bol bol bireysel pratik gerekmektedir
- Slaytlar yazılara boğulmadan görsellerle anlatılacaktır. Bu yüzden ders esnasında not tutulması **son derece** önemlidir
- Konu başlıkları temel düzey algoritmalar için yeterli olduğundan başlıklar araştırılmalı, bol bol uygulama ve teorik bilgiler içeren sitelerde araştırma yapılmalıdır

Ömer Cengiz

Github: omercengiz
Linkedin: omercengiz96
Twitter: omerrcengizz

Kaynaklar

Deep Learning Türkiye



Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Aurelien Geron

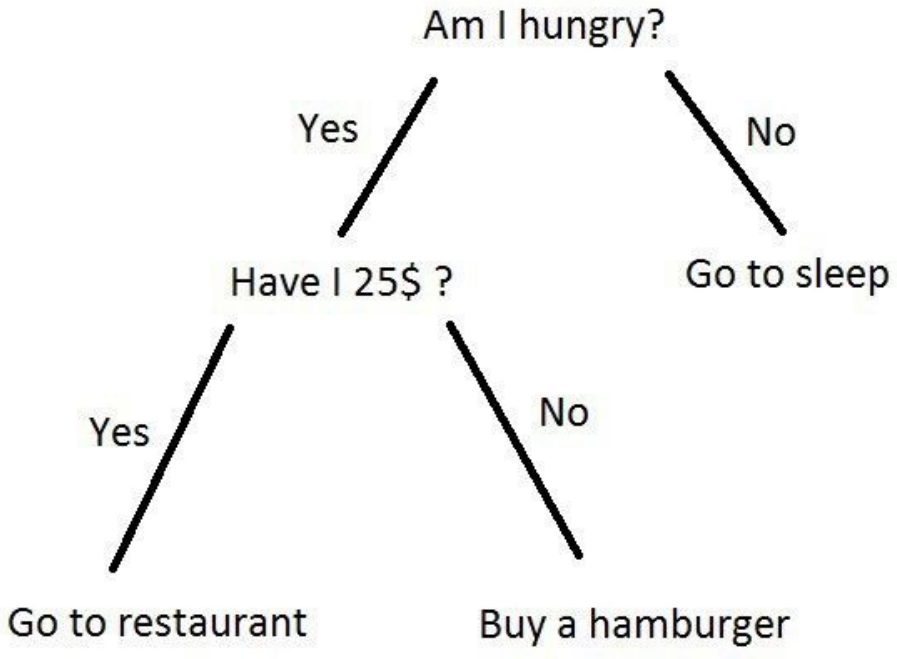
Stanford University
stanford.edu/~shervine/
Shervine Amidi

Karar Ağaçları

Bir karar ağacı, bir dizi olası karar yolunu ve her yol için bir sonucu temsil etmek için bir ağaç yapısı kullanır.

Karar ağacı, en sık kullanılan sınıflandırma tekniklerinden biridir.

Anlamak ve yorumlamak çok kolaydır ve bir tahmine ulaşma süreci tamamen şeffaftır.

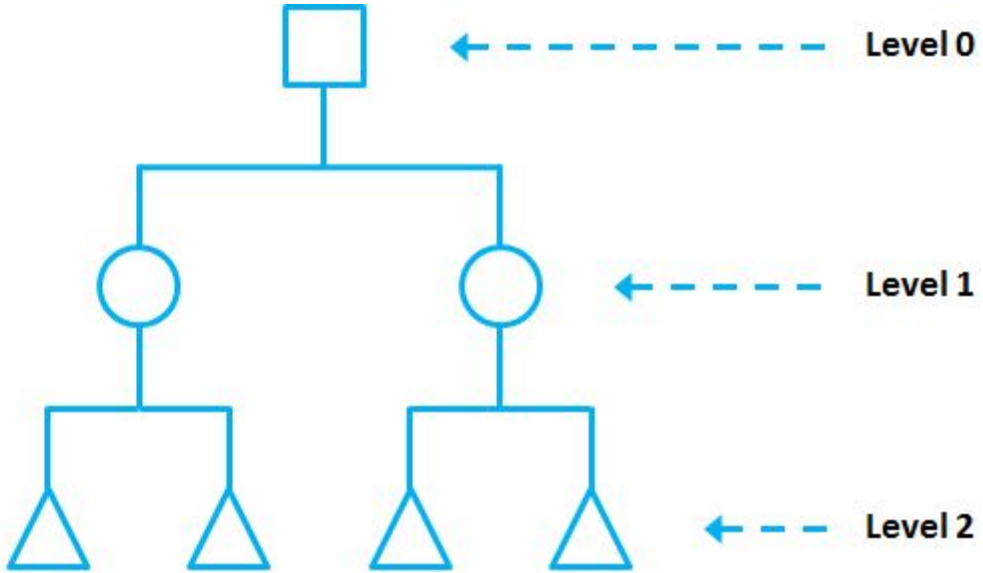


Karar Ağaçlarının Yapısı

Root: Karar ağaçlarının ilk hücrelerine kök (root veya root node) denir. Her bir gözlem kökteki koşula göre **“Evet”** veya **“Hayır”** olarak sınıflandırılır.

Node: Kök hücrelerinin altında düğümler (interval nodes veya nodes) bulunur. Her bir gözlem düğümler yardımıyla sınıflandırılır. Düğüm sayısı arttıkça modelin karmaşıklığı da artar.

Leaf: Karar ağacının en altında yapraklar (leaf nodes veya leaves) bulunur. Yapraklar, bize sonucu verir.



Karar Ağaçları Uygulaması

Karar Ağaçları Uygulaması

Dört sınıf etiketine sahip olan aşağıdaki iki boyutlu verileri inceleyelim

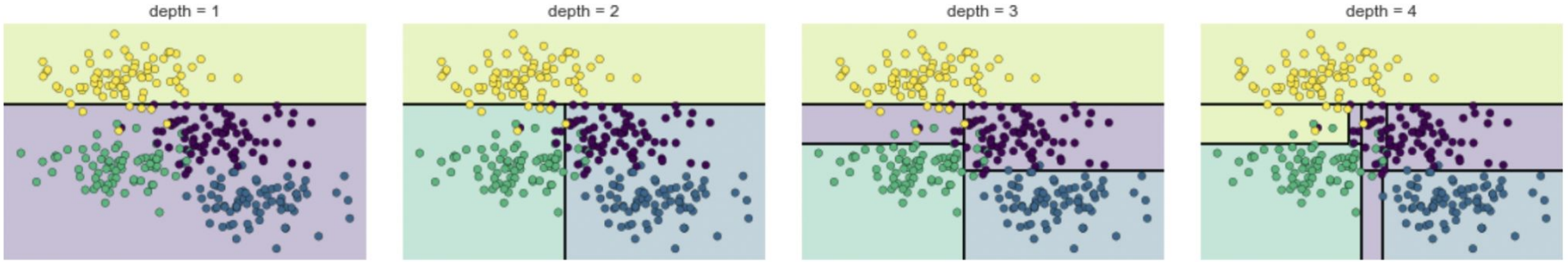
```
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=300, centers=4,
                  random_state=0, cluster_std=1.0)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='rainbow');
```

Karar Ağacı Uygulama

Bu veriler üzerine inşa edilen basit bir karar ağacı, verileri bazı nicel kriterlere göre bir veya diğer eksen boyunca yinelemeli olarak bölecek ve her düzeyde, içindeki puanların çoğunluk oyu ile yeni bölgenin etiketini atayacaktır.



İlk bölmeden sonra, üst daldaki her noktanın değişmeden kaldığına dikkat edin, bu nedenle bu dalı daha fazla alt bölümlere ayırmaya gerek yoktur. Bir rengin tamamını içeren düğümler dışında, her seviyede her bölge iki özellikten biri boyunca tekrar bölünür.

Karar Ağaçları Nasıl Hesaplanır

Gini Index

Pure, seçilen bir veri kümesi örneğindeki tüm verilerin aynı sınıfa ait olduğu anlamına gelir.

Impure, verilerin farklı sınıfların karışımı olduğu anlamına gelir.

Gini Impurity, yeni bir örnek, veri setinden sınıf etiketlerinin dağılımına göre rastgele sınıflandırılmışsa, rastgele bir değişkenin yeni bir örneğinin yanlış sınıflandırılma olasılığının bir ölçümüdür.

Veri kümemiz Pure ise, yanlış sınıflandırma olasılığı 0'dır. Örneğimiz farklı sınıfların karışımıysa, yanlış sınıflandırma olasılığı yüksek olacaktır.

$$Gini = 1 - \sum_j p_j^2$$

Gini = 1 - (Kırmızı Topun Seçilme Olasılığı^2 + Mavi Topun Seçilme Olasılığı^2)

2 Kırmızı - 2 Mavi
Gini = 1 - ((1/2)^2 + (1/2)^2)
Gini = 0.5

0 Kırmızı - 4 Mavi
Gini = 1 - ((0)^2 + (1)^2)
Gini = 0

3 Kırmızı - 1 Mavi
Gini = 1 - ((3/4)^2 + (1/4)^2)
Gini = 0.375

Entropi

Bir karar ağacı oluşturmak için hangi soruların hangi sırayla sorulacağına karar vermemiz gerekir. Ağacın her aşamasında elediğimiz bazı olasılıklar ve elemediğimiz bazı olasılıklar vardır.

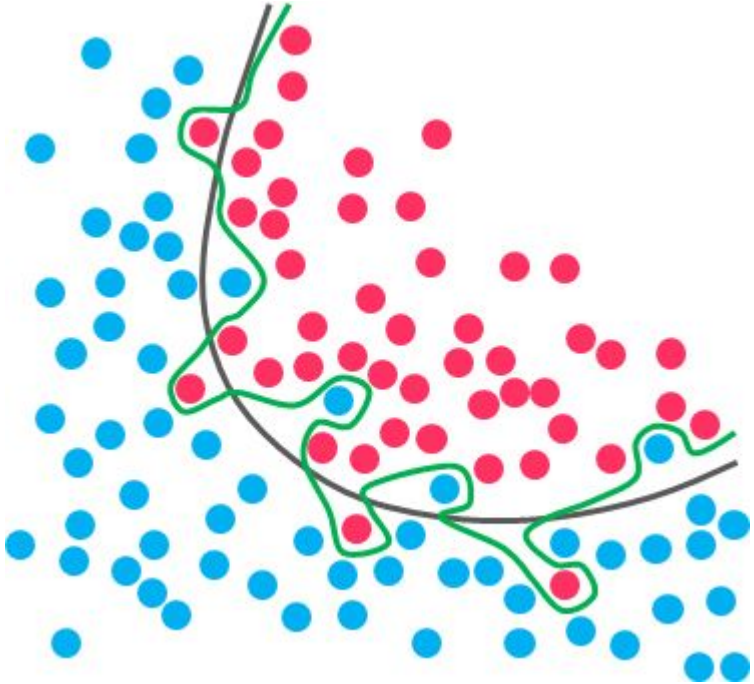
Örnek: *Bir hayvanın beşten fazla bacağı olmadığını öğrendikten sonra çekirge olma ihtimalini ortadan kaldırdık. Ördek olma ihtimalini ortadan kaldırmadık. Olası her soru, kalan olasılıkları cevaplarına göre bölümlere ayırır.*

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

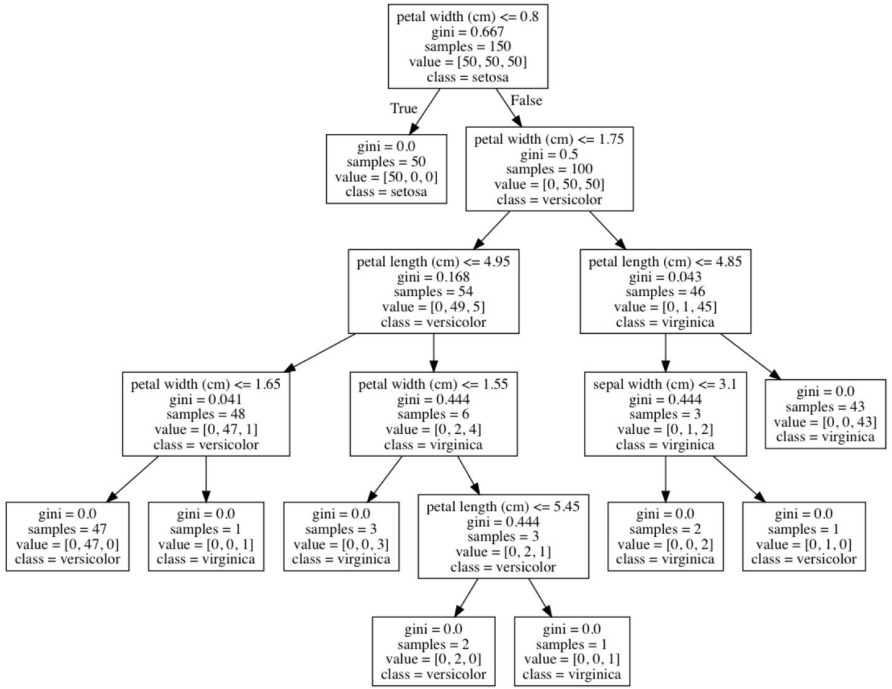
Entropi, rastgele bir değişkenin belirsizliğinin ölçüsüdür. Entropi ne kadar yüksek olursa elde edilen bilgi de o kadar fazla olur.

Karar Ağaçları Avantajları

- Hem sürekli hem de ayrık veriler ile çalışabilir
- Veri önişlemeye daha az ihtiyaç duyar. Outlier detection veya scaling gerektirmez
- Karar ağaçları kolaylıkla görselleştirilebilir ve sınıflandırma kuralları açıkça görülebilir bu yüzden anlaması ve yorumlaması kolaydır
- Çoklu çıktı için kullanılabilir



Karar Ağaçları Görselleştirmesi

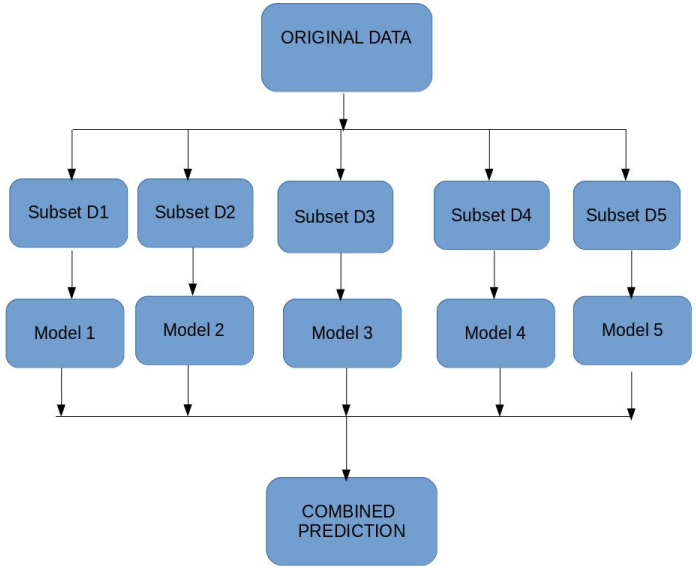


Ensemble Learning

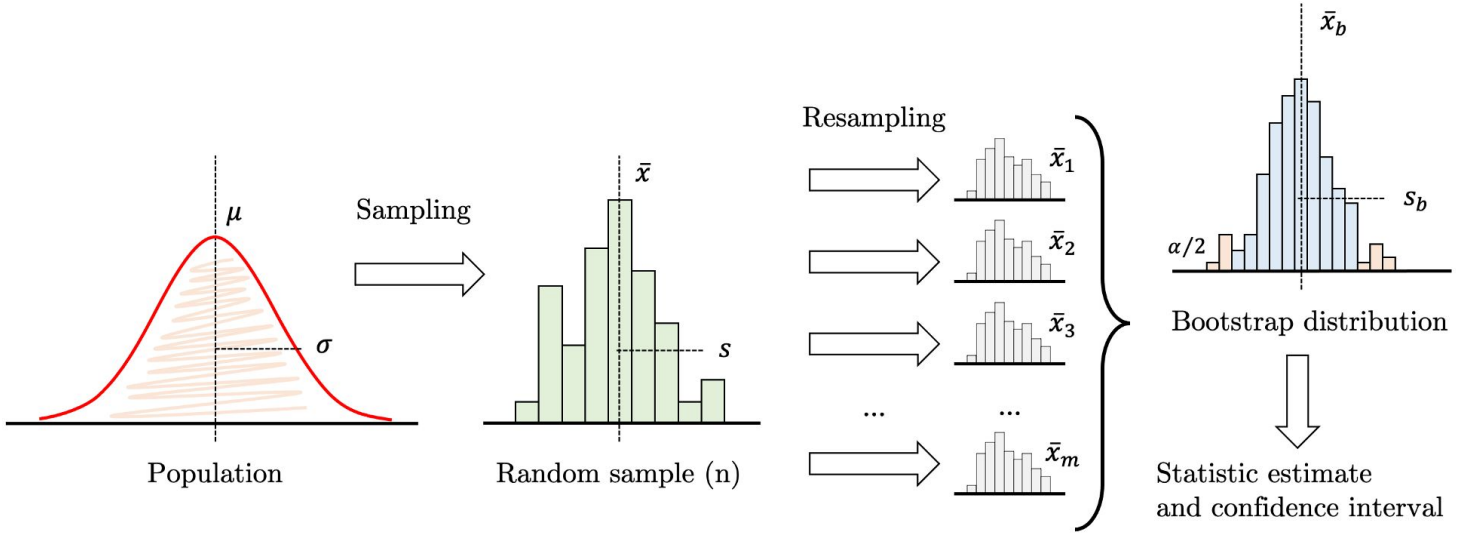
Birden fazla sınıflandırma, tahmin algoritmasının birlikte çalışması mantığıyla daha başarılı sonuçları vermesi olayına denir.

Örnek: Random forest, birden fazla decision tree algoritması aynı problem için bir kaç kere çizilmesi ve problemin çözümünde hep birlikte kullanılmasıdır.

- Maximum Oylama (Max Voting)
- Ortalama (Averaging)
- Ağırlıklı Ortalama (Weighted Averaging)
- Yığma (Stacking)
- Harmanlama (Blending)



Bootstrapping



$$low \leq \mu \leq high$$

$$\Pr[\bar{x}_b \leq low] = \alpha/2$$

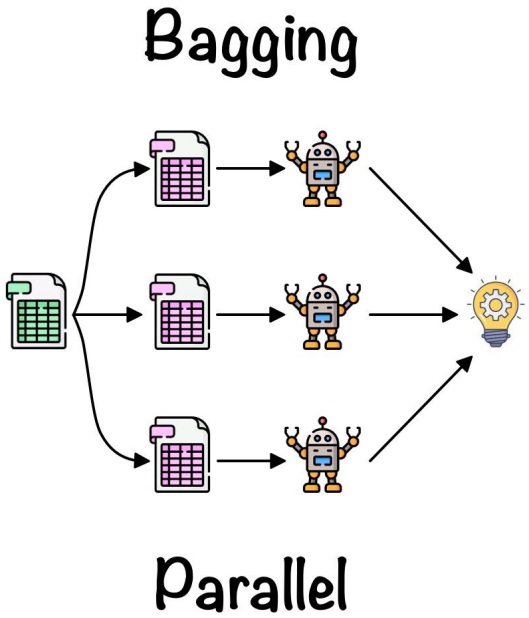
$$\Pr[\bar{x}_b \geq high] = \alpha/2$$

Bagging

Bootstrap Aggregation veya Bagging, gürültülü bir veri kümesindeki varyansı azaltmak için yaygın olarak kullanılan topluluk öğrenme yöntemidir.

Ana verisetinden rastgele veri altkumeleri oluşturulduktan sonra, birden fazla zayıf model bağımsız olarak eğitilir.

Rastgele orman algoritması, birbiriyle ilişkisiz karar ağaçlarından rastgele bir orman oluşturmak için hem baggingi hem de özellik rastgeleliğini (feature randomness) kullanan torbalama yönteminin bir uzantısıdır.

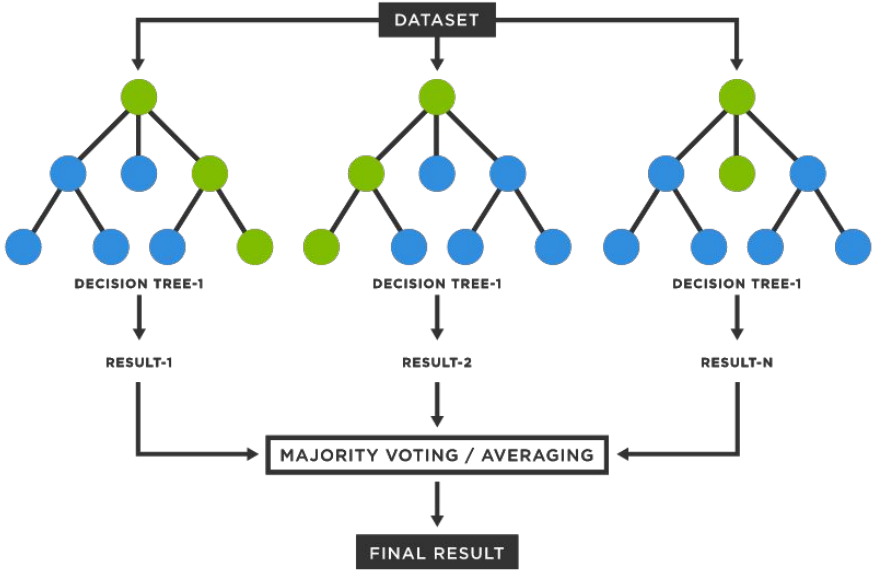


Random Forest

Bu kavram - overfitting etkisini azaltmak için birden fazla fazla uydurma tahmincisinin birleştirilebileceği bagging adı verilen bir topluluk yönteminin altında calisir.

Bagging, her biri verilere fazla uyan ve daha iyi bir sınıflandırma bulmak için sonuçların ortalamasını alan bir paralel tahmin ediciler topluluğu kullanır.

Rastgele karar ağaçları topluluğu, rastgele orman olarak bilinir.

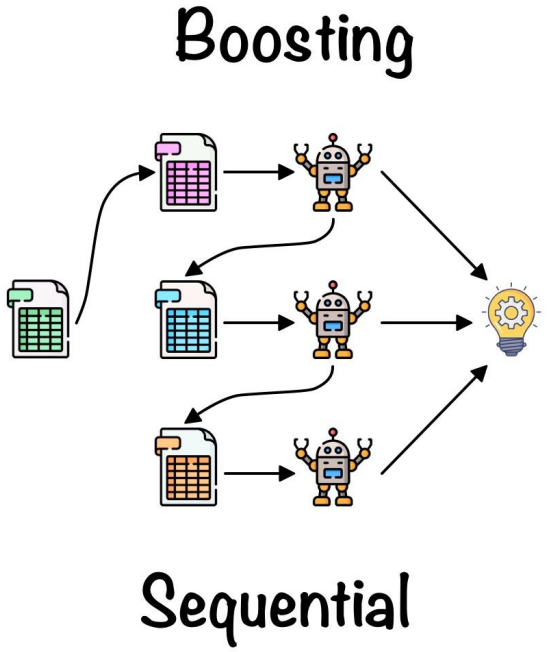


Boosting

Boosting, eğitim hatalarını en aza indirmek için bir dizi zayıf öğreniciyi güçlü bir öğrenicide birleştiren bir topluluk öğrenme yöntemidir.

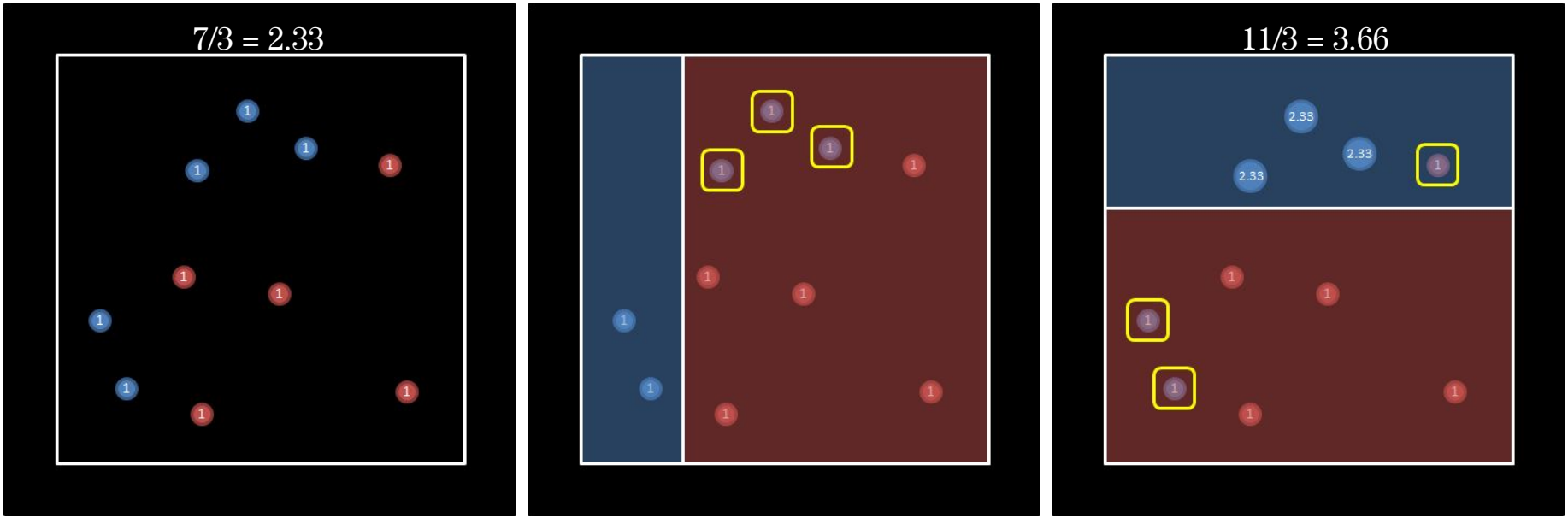
Güçlendirmede, rastgele bir veri örneği seçilir ve ardından sıralı olarak eğitilir - yani, her model bir öncekinin zayıflıklarını telafi etmeye çalışır.

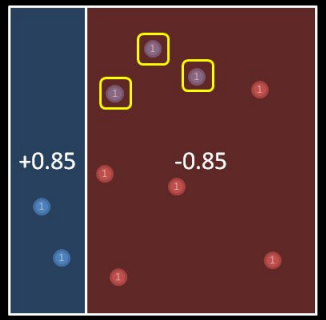
Her yinelemede, her bir sınıflandırıcıdan gelen zayıf kurallar, tek bir güçlü tahmin kuralı oluşturmak için birleştirilir.



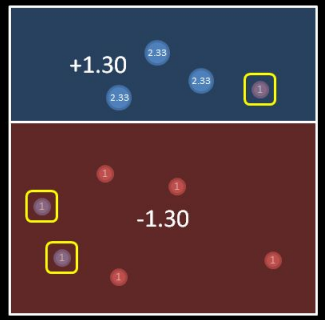
Boosting Uygulamasi

Kaynak:https://medium.com/@sertacozker/boosting-algoritmalar%C4%B1-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-edac1174e971

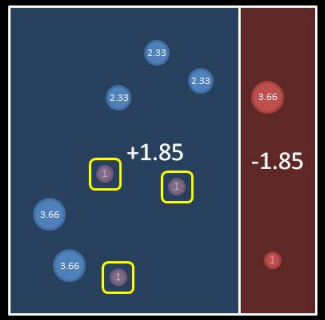




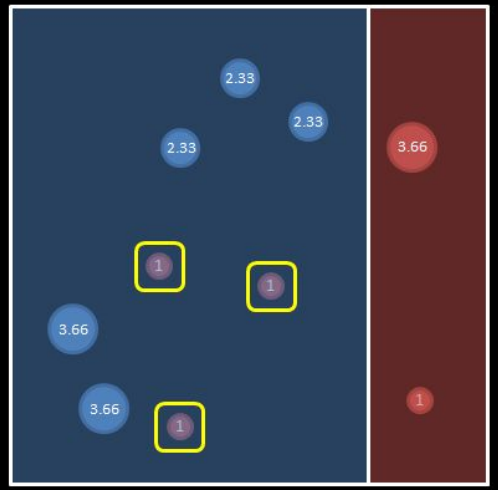
1. iterasyonun ağırlığı = ln(7/3) = 0.85



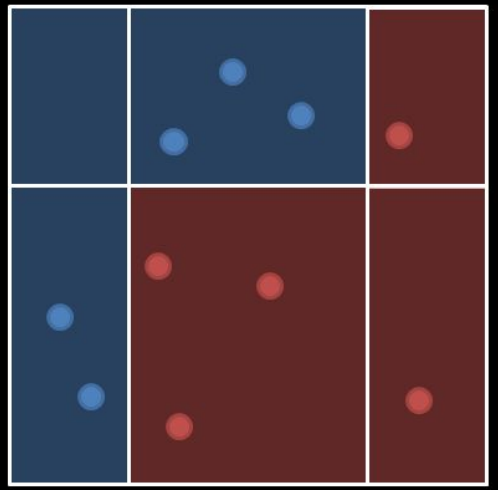
2. iterasyonun ağırlığı = ln(11/3) = 1.30



3. iterasyonun ağırlığı = ln(19/3) = 1.85



+0.85 +1.30 +1.85 **4.00**	-0.85 +1.30 +1.85 **2.30**	-0.85 +1.30 -1.85 **-1.40**
+0.85 -1.30 +1.85 **1.40**	-0.85 -1.30 +1.85 **-0.30**	-0.85 -1.30 -1.85 **-4.00**



XGBoost