PROJE ODEVİ

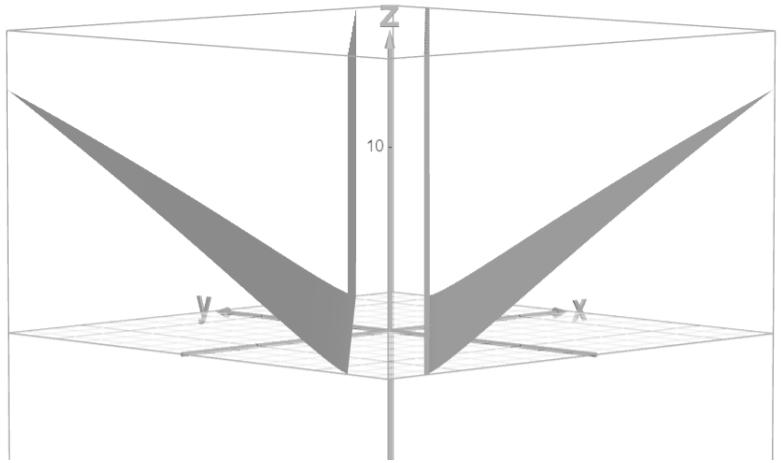
Eren Kara

B211210031

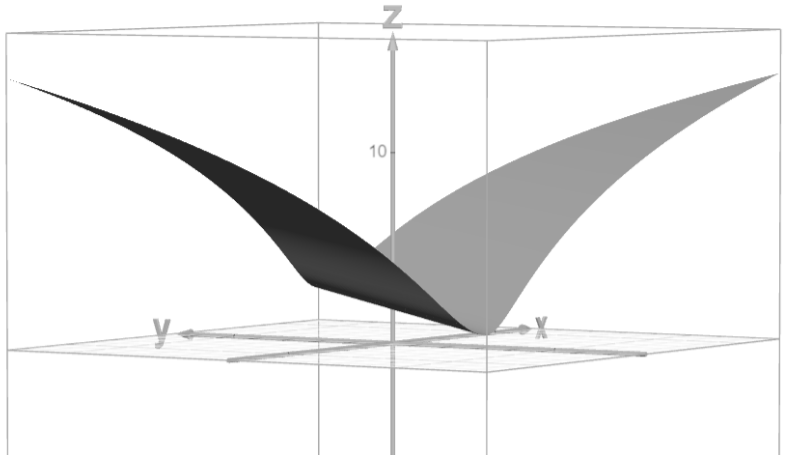
Destek Vektör Regresyon

$a\log\left(\alpha\left(|x_i - x_j|^2\right) + k\right)^c$ formülünü Kernel fonksiyonu olarak kullanabiliriz. Bu fonksiyonu kernel fonksiyonlarını inceleyerek ben oluşturdum.

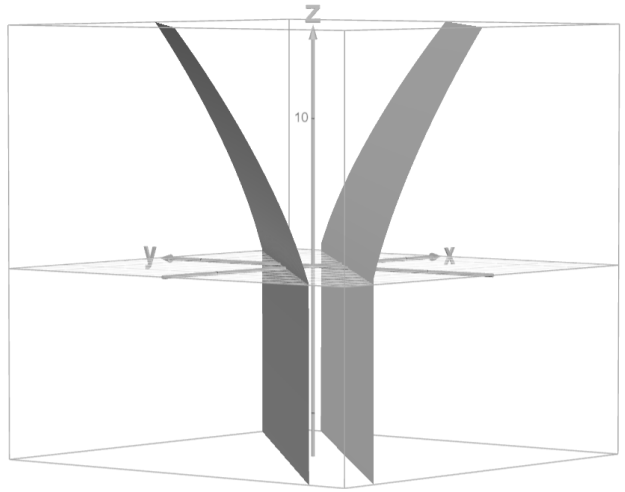
Bu fonksiyonu kullanıyor olmamın amacı parabolik eğri düzlemi oluşturabiliyor olması ve ayırıcı etken olarak kullanılmasının nispeten daha kolay olması. Yani bu fonksiyon ile ayrım yapmak daha kolay.



k=-20 c=4 b=0.1 alpha=3



k=-20 c=2 b=1 alpha=5

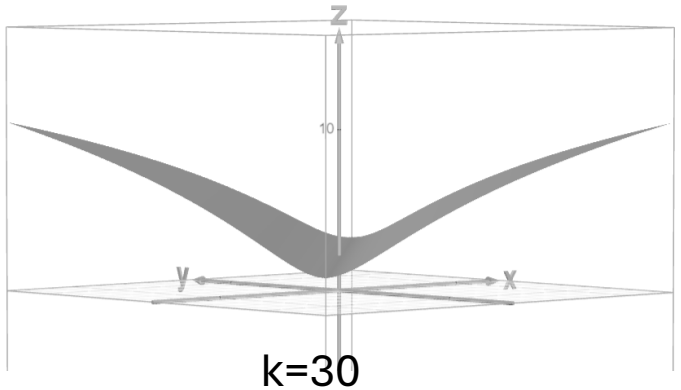


k=-40 c=3 b=0.6 alpha=5

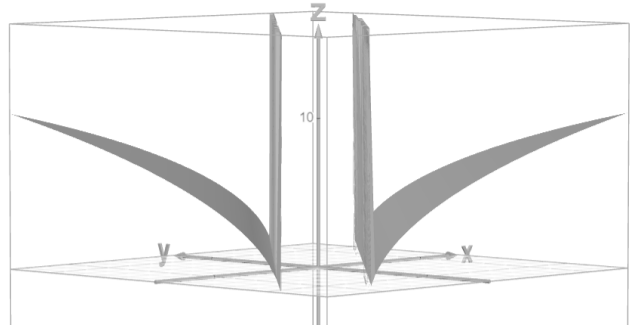
Şekillerde de görüldüğü üzere örnek uzayında dağılmış olan verileri bu fonksiyonu kullanarak ayırmak oldukça kolay. Doğru parametreleri cross validation ile buluyoruz ve hyper düzlemi istediğimiz gibi oluşturabiliyoruz.

EREN KARA

k parametresi yukarıdaki şekilde de görülebileceği gibi 1 veya 2 oluşmasında diğer bir deyişle bu iki düzlemin birbirine olan uzaklığı ile ilgili rol oynuyor.

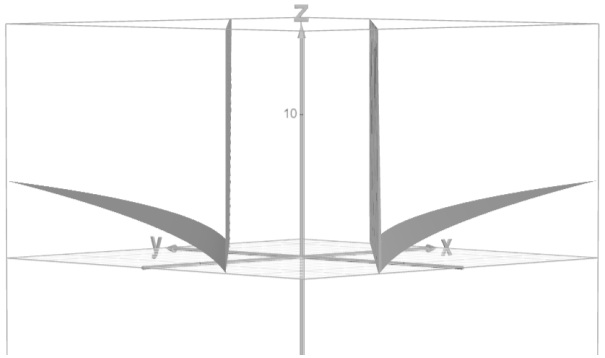


k=30

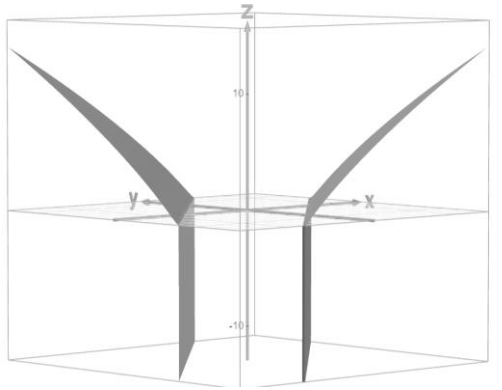


k=-30

c parametresi boyutlarla ilgili. Düzlemin kollarının sadece eksenin yukarısında mı aşağısında mı olacağını veya dar mı geniş mi olacağını gibi etkenleri belirliyor.

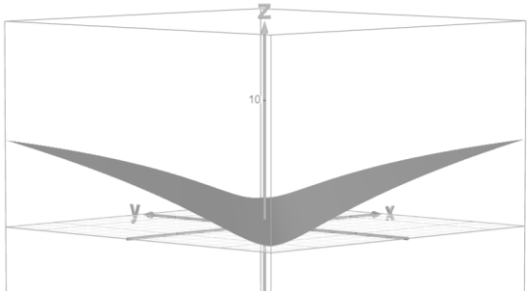


c=2

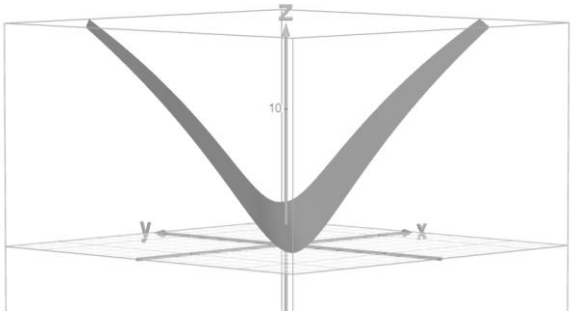


c=3

Alpha, a ve nispeten yukarda bahsettiğimiz k'da kolların açıklığını belirliyor.



b=1



b=3

EREN KARA

Lojistik Regression

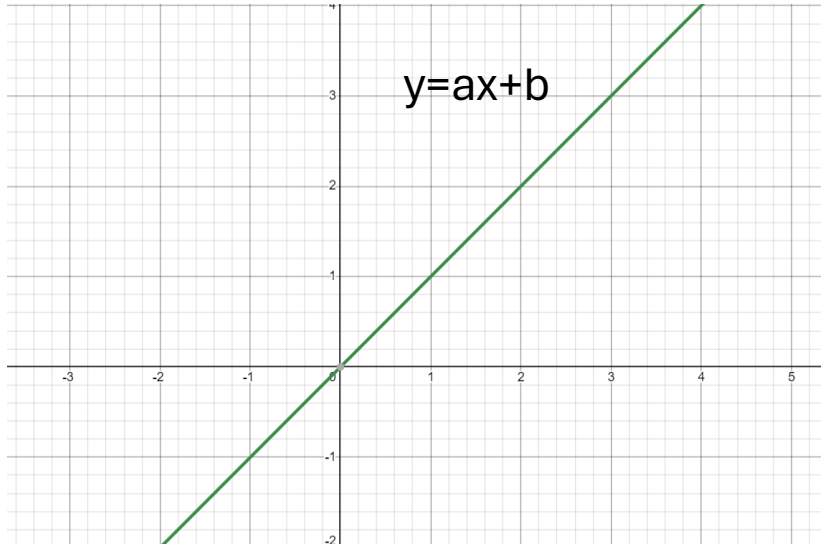
$\frac{1}{1+e^{-z}}$

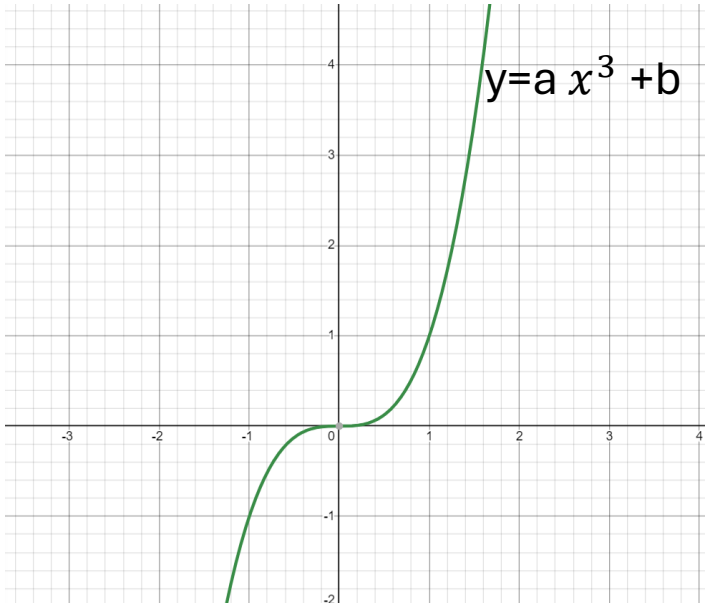
$$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w}\cdot\vec{x}+b}_{z}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$$

Lojistik Regression'daki doğrusal formül yerine ($wx + b$) yazılabilecek ve barışımı arttırıcı etki gösterecek fonksiyon kübik ($wx^3 + b$) fonksiyon olabilir. Bunu düşünme sebeplerimden biri kübik fonksiyonun değerlere karşı hassasiyeti linear fonksiyon'dan oldukça daha fazla olacak olması. Söylemek istediğim şey şu :

Herhangi bir x parametresindeki ufak bir artış y parametresindeki değeri çok büyük ölçüde değiştirecek ve bu da sigmoid fonksiyonun girişi olan z parametresini etkileyecek.

Yani diğer bir deyişle ise fonksiyonun girdisi bir tarafa hafif bir yakınlık göstermeye başladığında kübik bir işleme tabi tutulacağından hafif bir yakınsamada dahi fonksiyonun sonucunun yakınsadığı tarafa olan ilişkisi kübik ilişki ile büyüyecek.





Aynı değerler için kübik fonksiyon ihtimali çok daha fazla arttırıyor diyebiliriz. Burda temel kübik ve temel linear fonksiyonlarını baz aldım. Ancak katsayılar değiştikçe dediğim şey daha da farkedilebilir olacak.

x=4 için y=4 olurken sigmoide girmesi halinde sonuç sigmoid=0.982013790037933 2 olacak

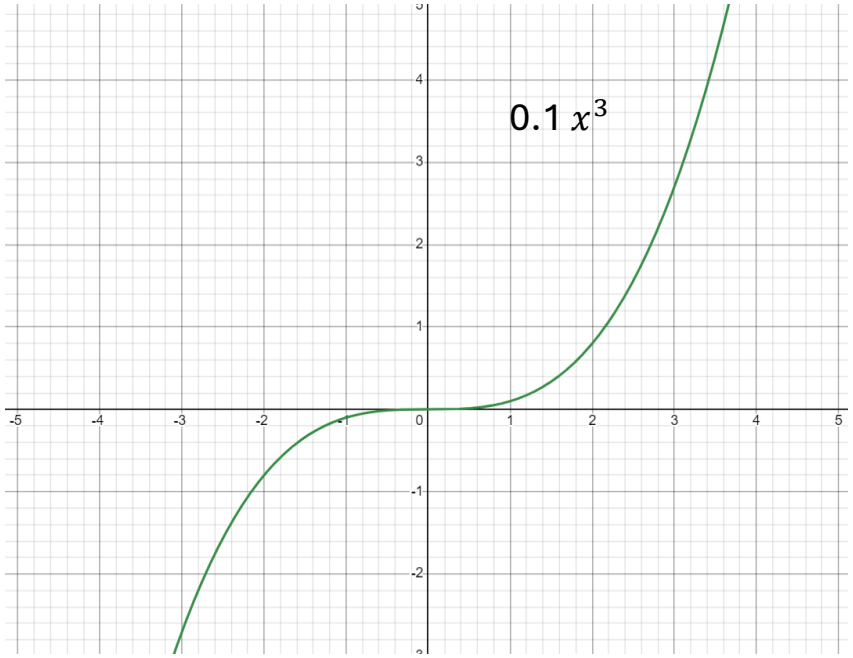
X=4 için y=16 olacak ve sigmoid=0.9999998874648 379

EREN KARA

Ayrıca sigmoid fonksiyonun'da yapısına uygun olarak kübik fonksiyon , fonksiyonun orta değerlerinde bir tarafa doğru hızlıca artmak veya azalmak yerine nerdeyse sabit ilerliyor. Bu da sınıflandırma yaparken benzer nitelik taşıyan değerlerin yani olasılıksal olarak yakın diyebileceğimiz sınıflandırmada ayırt edemeyeceğimiz değerlerin daha az bir tarafa yönelmesini sağlıyor. Yani olasılık oldukça uzun bir aralık boyunca istenilen değerlerde sabit kalacak ve bizler sigmoid functionun vertical olarak yapmış olduğu işlemi horizantal olarak benzer bir şekilde tekrar yapıyor olacağız.

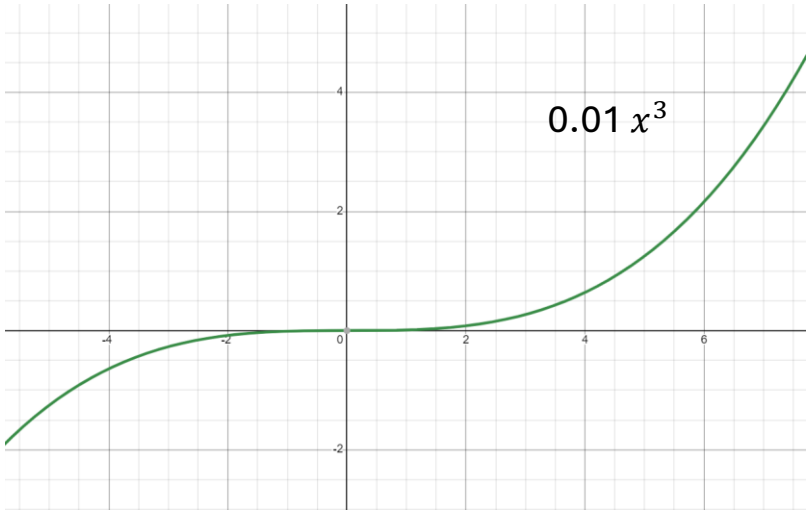
Diğer bir şekilde ifade edecek olursam, bu fonksiyon sayesinde girdilere artmayacağı ve azalmayacağı bir alan tanıyarak yakın x değerlerine sahip olan girdilerin çıkışta ortak olarak benzer bir y değerinde gruplanmasını sağlıyorum. Diğer dediğim özelliğin zıttı gibi dursa da burada anlatmak istediğim küçük değişikliklere fonksiyon genelinde hassas tepki veriyor ancak bir nokta hariç. Bu nokta da ise ayırt edilmesi güç benzer nitelikli girdiler gruplanacak. Bu sonuç ise sigmoid'e verildiğinde çok daha iyi olasılıksal tahminler elde edeceğiz.

Kısacası bence kübik fonksiyon sigmoidin yapısına daha uygun.

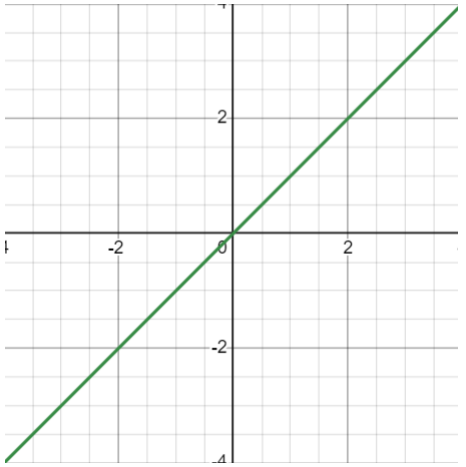


x=1 için y=0.1 olacak Sigmoid = 0.524

x=-1 için y=-0.1 olacak Sigmoid = 0.475



Yakın değerler veriyorlar



x=1 için y=1 olacak sigmoid=0.731

x=-1 için y=-1 olacak sigmoid=0.268

EREN KARA

Karar Ağacı Sınıflandırma Algoritması

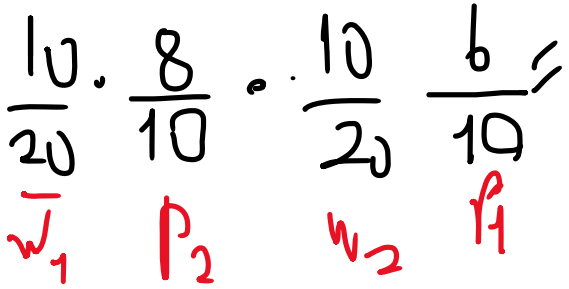
Amacımız leaf node'larda mümkün olduğunca tek bir sınıftan eleman bırakmak olduğundan bunu en hızlı şekilde yapan soruyu seçme gibi bir yöntem kullanabiliriz. ID3'ten farklı birkaç yapıya sahip ancak amacımız ayrıştırma olduğundan ve sınıflandırma temelinde olasılık barındırdığından önceki kullanılan algoritmalardan bağımsız bir algoritma yapmak imkansız bir hal alıyor. Formüldeki 4 sayısının sebebi ölçeklendirme için.

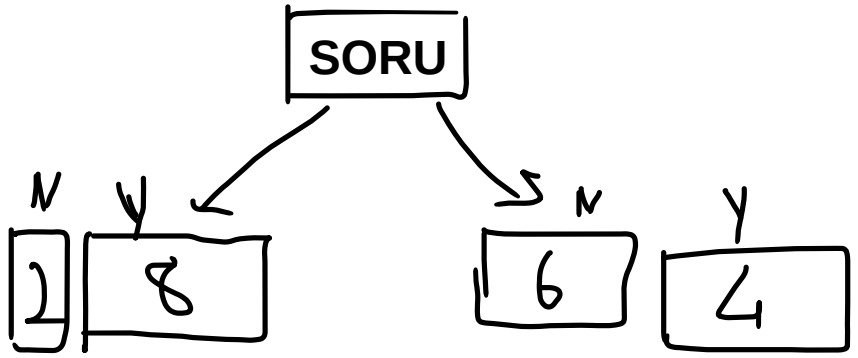
W = Weight , P = Probability

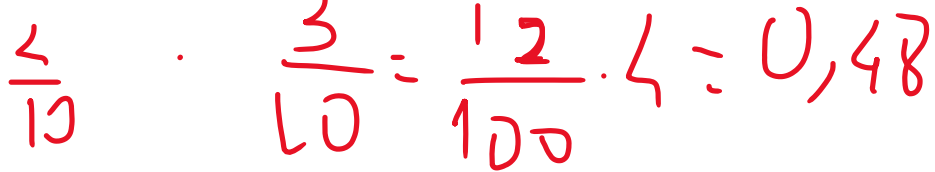
$$Fonksiyon = 4\ [W_1 \mathrm{Max}(P_1, P_2)\ \ W_2 \mathrm{Max}(P_1, P_2)]$$

Amacımız ayırdığımız yapraklara ağırlık vererek (bunu yapmamızın sebebi ağacın her kolunun aynı oranda dağılmasını bekliyoruz örneğin 3 kolu varsa ağacı 3 ortak parçaya bölmeli her bir dalda eşit sayıda eleman olmalı) , yaprakların sahip olduğu Yes veya No gibi sınıflandırmadan maximum olanı baz alarak dağılımın nasıl olduğunu ölçüyoruz. Burda maximumunu almamızın sebebi dağılımın olasılıksal olarak tek bir kategoride toplanmasını istiyor oluşumuz. Maximumunu alarak hangi kategoride en çok toplanmış onu elde ediyoruz.

Örnek: VERİ = 20 adet

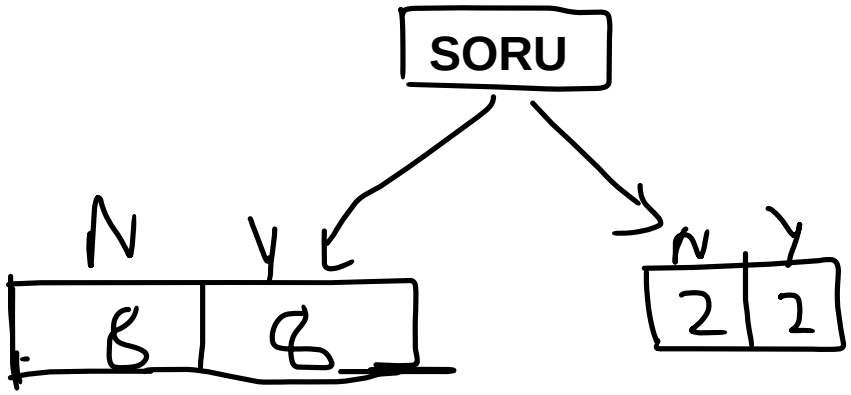






Burada sonuç 0.48 çıktı oldukça iyi bir rakam verilerin ağacın dalları arasında yeterince iyi dağıldığını ve olasılıksal olarak dağılımın dağılan dallarda bir tarafa toplandığını ifade ediyor.

EREN KARA



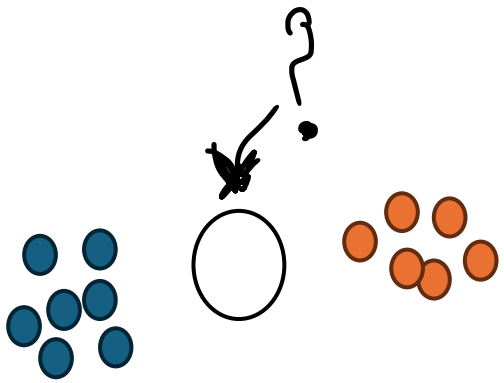
$$4\left[\left(\frac{16}{20}\cdot\frac{8}{16}\right)\cdot\left(\frac{4}{20}\cdot\frac{2}{4}\right)\right]=0.16$$

Yukarıdaki örnekte ağacın sol tarafındaki veri sayısı sağ tarafındakine göre daha fazla olduğu için ve olasılıksal olarak YES ve NO arasında eşit denilecek düzeyde ayırım olduğundan fonksiyon sonucu değerimiz 0.16 çıktı oldukça kötü bir sonuç. Bu soru ağacın ayırımı için mümkünse kullanılmamalı, sona bırakılmalı diğer sorular denenmeli.

EREN KARA

KNN Sınıflandırma Algoritması

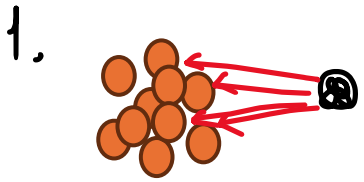
Algoritmada sadece mesafe metriği kullanmak yerine açıyı da ekleyerek yeni bir seçim metriği oluşturmak istedim. Açıyı kümeleme yapmak için kullanacağız. Ne demek istiyorum:

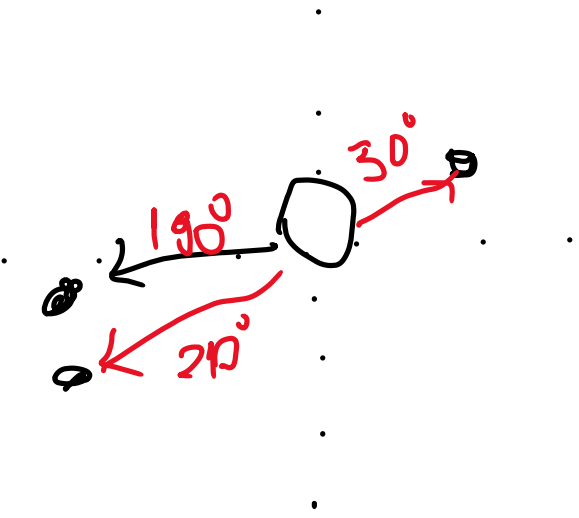


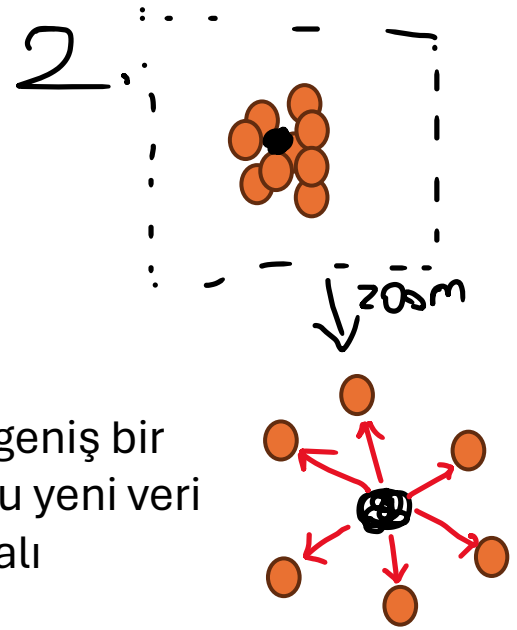
Burada bakmamız gereken metriğin mesafe olduğu kadar açı da olduğunu düşünüyorum. Yandaki gibi bir örneği gruplamak zor olacağından açıyı da dahil etmek işe yarayabilir.

Burada mesafelerini ölçtüğümüz gruplar ne kadar açısal olarak geniş bir alana yayılmışsa, bu yeni örneğin de o gruba dahil olma ihtimalinin fazla olması anlamına gelir.

1.Eğerki en yakın veri örnekleri birbirine yakın açılarda bulunuyorsa bu yeni örneğin o gruba dahil olma ihtimalini düşürmeli.







2. Ancak en yakın veri örnekleri açısal olarak çok geniş bir yelpazede ve tek bir örnek üzerinde birleşiyorsa bu yeni veri örneğinin de o gruba dahil olma ihtimalini arttırmalı

EREN KARA

Rastgele orman ağacı sınıflandırma algoritması

Bu algoritmayı optimize etmek için şöyle bir yöntem düşündüm. Rastgele orman algoritmasını inşaa ederken oluşturduğumuz her bir ağacın görmediği veriler bulumakta, bu verileri tekrar ağacı sınamak için kullanarak güven oyu veya güven oranı şeklinde adlandırabilecek bir sistem kurmak ağacın performansını iyileştirme de kullanılabilir.
Her bir ağaca daha önce görmediği örnek (ancak bu örneğin cevabının ne olduğunu biliyoruz) üzerinde yaptığı tahminin doğruluğuna göre bir güven oyu atanacak. Ağaç tutarlı tahmin yaptığı sürece ağacın güven oyu yükselecek. Yüksek güven oyuna veya diğer bir deyişle güven oranına sahip ağaçlara daha çok güvenilecek ve yaptığı tahminlerin sonuca etkisi daha yüksek boyutlarda olacak.

$\theta = Güven\ Oyu\ (Trust\ enhancement)$ $\theta = \frac{Doğru\ Tahmin}{Tahminler}$

$p = Predict\ ,\ Yes \rightarrow 1\ ,No\ \rightarrow -1$

$Sigmoid\ için\ threshold = 0.5$

$$\text{sigmoid}(\sum_{i=0}^{n} \theta_i p_i)$$

KISACASI: %30 luk veriyi güven oyunu elde etmek için kullanacağız. Güven oyu θ'dır. Bu güven oyu bir kez hesaplandıktan sonra sabit kalacak. Tahminlerde ağaca güvenimizi ifade edecek. Doğru tahmin oranı bize θ'yı verecek.

EREN KARA

Rastgele orman ağacı regresyon algoritması

Rastgele orman regresyon algoritması için sınıflandırma veya regresyon farketmeksizin rastgele orman için geliştirdiğim güven oyu yöntemini tekrar kullanacağım. Elbette ki formülde ufak değişiklikler yapacağım ancak güven oyu mantığı ve hesaplanması aynı kalacak.

Algoritmamız şu şekilde çalışacak :

1. Her bir ağacın güven oyunu hesapladıktan sonra (bu bir önceki slaytta var) bunları toplayacak ve toplam güven oyunu bulacak.
2. Ağacın güven oyu ile toplam güven oyunu oranlanarak, o ağacın tahmininin toplam tahmine etkileyeceği ağırlık bulunacak
3. Ağacın yaptığı tahmin bu ağırlık ile çapılacak
4. Bütün ağaçların tahminleri toplanacak
5. Çıkan sonuç algoritmamızın sonucu olacak.

$$\sum_{i=0}^{n} \theta_i = \theta_T \ \rightarrow Toplam\ Güven\ Oyu$$

$$\sum_{i=0}^{n} \frac{\theta_i}{\theta_T} p_i = PREDICT\ (TAHMIN)$$

EREN KARA