K-En Yakın Komşu Algoritması

K-En Yakın Komşu (*K-Nearest Neighbor / K-NN*) algoritması Makine Öğrenmesi'nde *(Machine Learning*) kullanılan bir sınıflandırma (*classification*) algoritmasıdır.

Çoğu makine öğrenmesi algoritmasında "Eğitim (*Training*)" ve "Tahmin (*Prediction*)" olmak üzere iki farklı süreç bulunur.

Ancak, K-NN sınıflandırıcı, parametrik olmayan, örnek tabanlı ve tembel bir öğrenme algoritmasıdır.

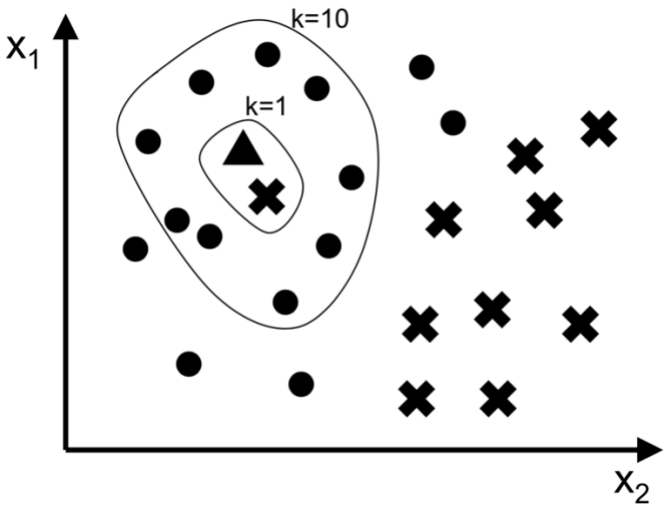
Parametrik olmayan, temel veri dağılımı hakkında bir varsayım olmadığı anlamına gelir. Diğer bir deyişle, model yapısı veri setinden belirlenir.

Örnek tabanlı öğrenme, algoritmamızın açık bir model öğrenmediği anlamına gelir. Bunun yerine, eğitim örneklerini ezberlemeyi tercih eder ve bu örnekler, tahmin aşamasında 'bilgi' olarak kullanılır.

Tembel algoritma, eğitim verilerinden bir ayrıştırıcı fonksiyon (*discriminative function*) öğrenmek yerine eğitim veri setini ezberlediği anlamına gelir. Modelin öğrenmesine veya eğitilmesine gerek yoktur ve tüm veri noktaları tahmin (*prediction*) sırasında kullanılır.

Örneğin, bir doğrusal regresyon modelinde X_1, X_2,, …, X_p gibi p tane bağımsız değişken (*independent variables*) ve Y gibi bir tane bağımlı değişken (*dependent variable*) varken, $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$ gibi bir matematiksel fonksiyon elde etmeye çalışırız. Buradaki "Eğitim (*Training*)" aşaması elimizdeki veri kümesini (*dataset*) kullanarak $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ gibi model katsayılarının tahmin elde edilmesidir. ANCAK, böyle bir durum K-En Yakın Komşu algoritması için geçerli değildir. **K-En Yakın Komşu algoritmasında sadece "Tahmin" aşaması vardır.**

K-En Yakın Komşu algoritmasında K değeri, en yakın komşu sayısını ifade eder. Komşu sayısı, karar verme faktörüdür. Genellikle K, beraberliği önlemek için tek bir sayı olarak seçilir. $K = 1$ olduğunda, algoritma en yakın komşu algoritması (*nearest neighbor algorithm*) olarak adlandırılır. Bu, en basit durumdur.



Burada, iki girdi değişkeni vardır. Bunlar X_1 ve X_2'dir. Bu nedenle iki boyutlu bir koordinat sisteminde saçılım grafiği (*scatter plot*) çizdirilmiştir. ▲ sembolü yeni veri gözlemini ifade etmektedir. ● ve ✖ iki farklı sınıfa ait veri örnekleridir. K = 1 için ▲ yeni veri gözlemi ✖ sınıfına aittir. $K = 10$ için ▲ yeni veri gözlemi ● sınıfına aittir çünkü en yakın 10 komşusunda 9 adet ● ve 1 adet ✖ vardır. Bu nedenle ▲ veri gözlemi 9/10 = 0.90 olasılıkla ● sınıfına ve 1/10 = 0.10 olasılıkla ✖ sınıfına aittir denir.

Daha resmi bir şekilde ifade etmek gerekirse, pozitif bir tam sayı K, önceden görülmemiş bir veri gözlemi x_{test} ve bir uzaklık ölçütü (*distance metric*) d verildiğinde, K-NN sınıflandırıcısı şu şekilde çalışır.

1- Öncelikle tüm veri seti taranarak, x_{test} ile her bir eğitim gözlemi arasındaki mesafe hesaplanır.
2- Daha sonra bu mesafeler sıralanır ve x_{test} ile en yakın K komşunun ait olduğu sınıfların sayısını göre, x_{test}'in hangi olasılıkla hangi sınıfa ait olduğu bulunur.

ÖRNEK:

Aşağıda $n = 6$ gözlem ve $p = 2$ girdi değişkeni ($\mathbf{X_1}$ ve $\mathbf{X_2}$) olan bir eğitim veri seti verilmiştir. Bu 6 gözlemin üç tanesi Kırmızı sınıfa, üç tanesi Mavi sınıfa aittir.

Gözlem No. (i)	$\mathbf{X_1}$	$\mathbf{X_2}$	Y
1	-1	3	Kırmızı
2	2	1	Mavi
3	-2	2	Kırmızı
4	-1	2	Mavi
5	-1	0	Mavi
6	1	1	Kırmızı

$x_{test} = [1, 2]$ veri gözlemi için çıktıyı tahmin etmekle ilgileniyoruz (Diğer bir deyişle, burada, bu test veri noktası için $X_1 = 1$ ve $X_2 = 2$'dir ve bu test örneğinin hangi sınıfa ait olabileceğini bir olasılıkla bulmak istiyoruz) . Bu amaçla, 3-en yakın komşuyu inceliyoruz, yani $K = 3$'ü seçiyoruz.

Öncelikle, her bir eğitim veri noktası x_i ile test veri noktası x_{test} arasındaki Öklid uzaklığını (*Euclidean distance*), yani $\|x_i - x_{test}\|$'i, hesaplarız ve ardından hesaplanan uzaklık değerlerini artan sırayla sıralarız

İki veri noktası (x_{1j} ve $x_{2j}, j = 1, 2, 3, \ldots, p$) arasındaki Öklid uzaklığı aşağıdaki gibi hesaplanır:

$$d(x_{1j}, x_{2j}) = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + \ldots + (x_{1p} - x_{2p})^2}$$

Yukarıdaki örnek için…

Gözlem No. (i)	$\|x_i - x_{test}\|$	Y
6	$\sqrt{(1-1)^2 + (1-2)^2} = \sqrt{1} = 1$	Kırmızı
2	$\sqrt{(2-1)^2 + (1-2)^2} = \sqrt{2} = 1.4142$	Mavi
4	$\sqrt{(-1-1)^2 + (2-2)^2} = \sqrt{4} = 2$	Mavi
1	$\sqrt{(-1-1)^2 + (3-2)^2} = \sqrt{5} = 2.236$	Kırmızı
5	$\sqrt{(-1-1)^2 + (0-2)^2} = \sqrt{8} = 2.8284$	Mavi
3	$\sqrt{(-2-1)^2 + (2-2)^2} = \sqrt{9} = 3$	Kırmızı

$K = 3$ seçtiğimiz için, x_{test}'in en yakın 3 komşusu $i = 6$ (Kırmızı), $i = 2$ (Mavi) ve $i = 4$ (Mavi) olarak belirlenir. Bu en yakın üç veri noktasının 2 tanesi Mavi ve bir tanesi Kırmızı olduğundan, x_{test} veri noktasının Mavi olması olasılığı $\frac{2}{3} = 0.66$ ve Kırmızı olması olasığı $\frac{1}{3} = 0.33$'tür. O halde, **x_{test} veri noktasının %66 olasılıkla Mavi sınıfa ait olduğunu** söyleyebiliriz.

Bu işlemi bizim için gerçekleştiren ve "bu veri noktasının %66 olasılıkla Mavi sınıfa ait olduğunu" söyleyen Python programını yazınız.

Fonksiyonu tanımını (*function definition*) gerçekleştiriniz ve fonksiyon çağrısını (*function call*) yapınız. Yukarıdaki örneği veri olarak kullanabilirsiniz. Çıktı olarak yukarıda kırmızı ile yazılmış çıktıyı görmek istiyorum.

TÜM İŞLEMLERİN OTOMATİKLEŞTİRİLMİŞ OLMASI GEREKMEKTEDİR! SERT KODLAMA (*HARDCODING*) YAPILAN ÇÖZÜMLERE PUAN VERİLMEYECEKTİR! SİZDEN İSTENİLEN BİR *predict* FONKSİYONU YAZMANIZDIR**. HERHANGİ BİR KÜTÜPHANE KULLANMAMALISINIZ**! BAŞARILAR! ☺