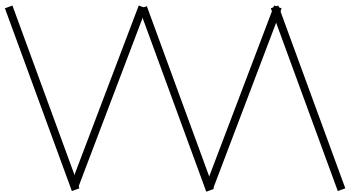
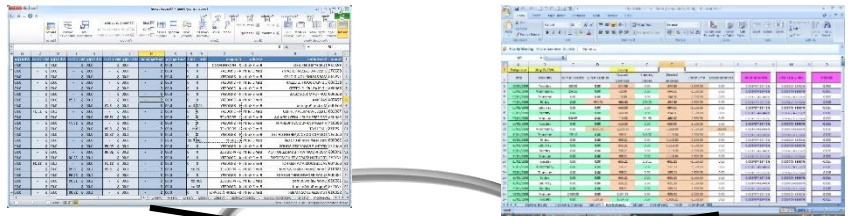


VERİ MADENCİLİĞİ

Fırat İsmailoğlu, PhD

Sınıflandırmaya Giriş – KNN – Perseptron



veri toplama

veri
ön işleme

veri madenciliği
algoritma uygulaması

bilgi

Dönem sonuna kadar burdayiz.

Sınıflandırma

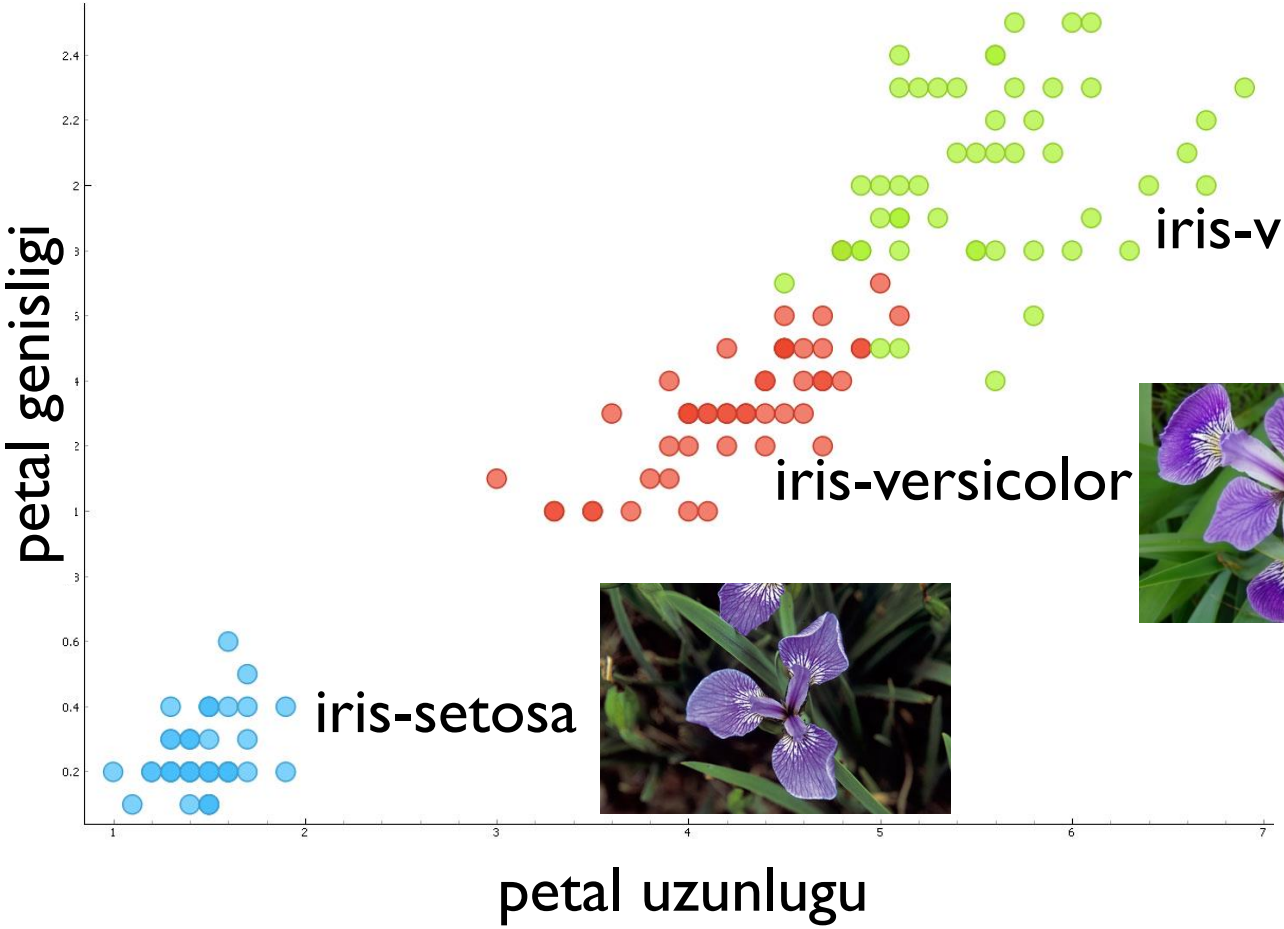
Sınıflandırma veri madenciligindeki en önemli görevlerden biridir.

Sınıflandırmada amacımız bize verilen etiketli/siniflandırılımış veri setini (eğitim setini) kullanarak bir veri madenciligi algoritmasi vasitasiyla bir siniflandirici (model) elde etmektir. Daha sonra bu siniflandiriciyi sinifi bilinmeyen (etiketlenmemis) örnekler (test seti) uzerine uygular ve bu orneklerin sinifini tahmin ederiz.

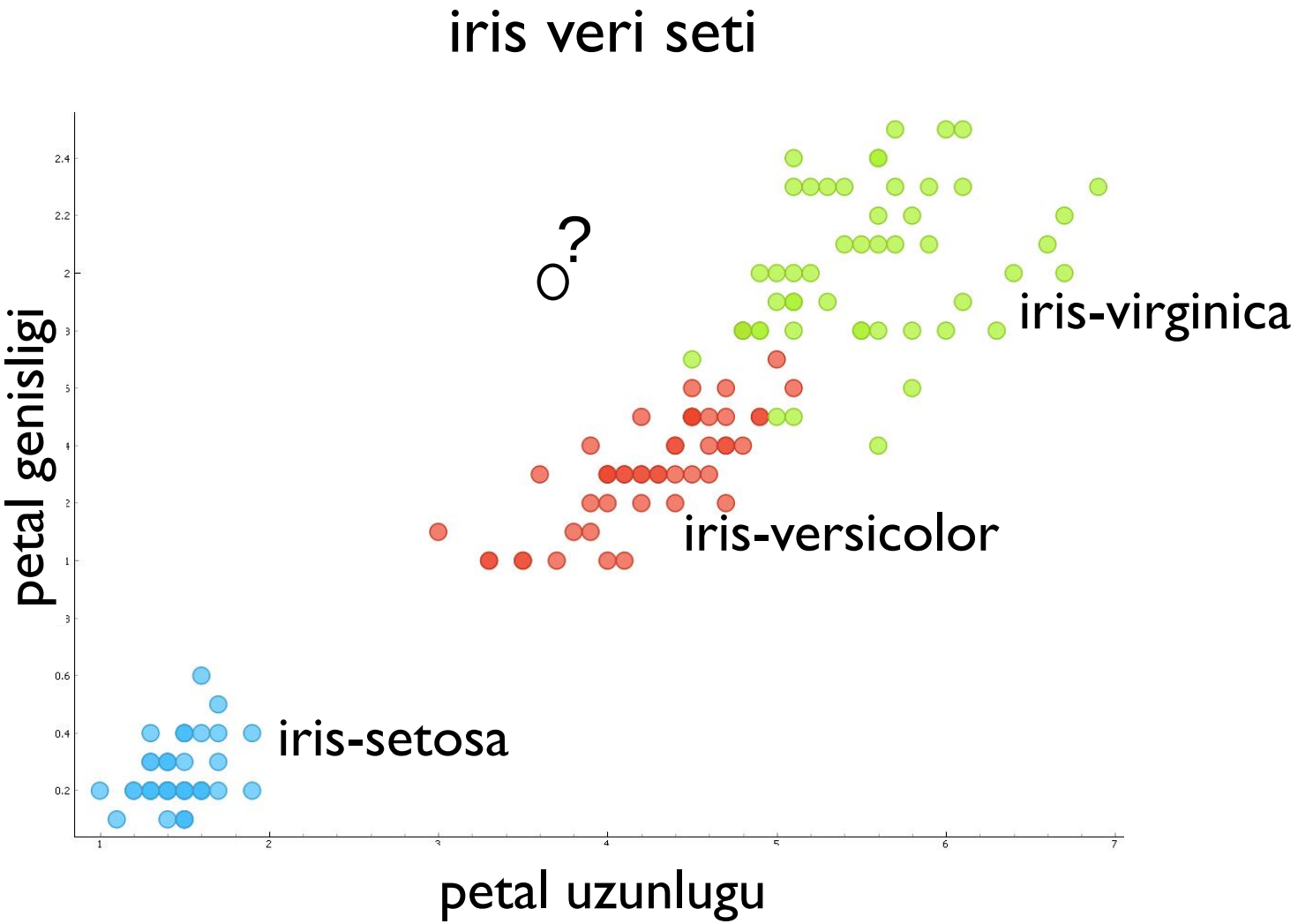
Siniflandirma Örnekleri:

- Tümor hucrelerini iyi yada kötü huylu olarak siniflandirmak
- Kredi karti harcamalarini yasal yada hileli olarak siniflandirmak
- Haberleri finans, hava, spor... olarak siniflandirmak
- Mailleri spam yada degil olarak siniflandirmak
- Atilan tweet'leri icerdigi duyguya gore (üzgün, sinirli, mutlu, heyecanlı) siniflandirmak (duygu analizi yapmak)

iris veri seti



- Burada iris çiçeginin üç farkli sınıfı (türü) goruluyor.
- Her bir sınıfın kendine has özellikleri oldugunu varsayacagiz.
- Ayni sınıfa ait örnekler birbirine benzer iken farkli sinifa ait ornekler birbirinden farkidir.



Amaç:
Sınıflandirmada amacimiz sinifi bilinmeyen bir ornegin (objenin) sinifi bilinen objeleri kullanarak sinifini tahmin etmektır.

Yukaridaki ornekte renkli noktalarin (eğitim örnekleri) sınıfları biliniyor, renksiz örnegin (test ornegi) ise sinifi bilinmiyor. Amacimiz bu örnegi buradaki üç siniftan biriyle eşleştirmek, yani sinifini tahmin etmek.

Veri Seti

özellikler (features)

sınıf (etikitet/kategori)

örnekler (objeler)

sepal length	sepal width	petal length	petal width	iris
4.9	3.1	1.5	0.1	Iris-setosa
4.4	3.0	1.3	0.2	Iris-setosa
5.1	3.4	1.5	0.2	Iris-setosa
5.0	3.5	1.3	0.3	Iris-setosa
4.5	2.3	1.3	0.3	Iris-setosa
4.4	3.2	1.3	0.2	Iris-setosa
5.0	3.5	1.6	0.6	Iris-setosa
5.1	3.8	1.9	0.4	Iris-setosa
4.8	3.0	1.4	0.3	Iris-setosa
5.1	3.8	1.6	0.2	Iris-setosa
4.6	3.2	1.4	0.2	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor
4.9	2.4	3.3	1.0	Iris-versicolor
6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor
5.0	2.0	3.5	1.0	Iris-versicolor
5.9	3.0	4.2	1.5	Iris-versicolor
6.0	2.2	4.0	1.0	Iris-versicolor
6.1	2.9	4.7	1.4	Iris-versicolor
5.6	2.9	3.6	1.3	Iris-versicolor

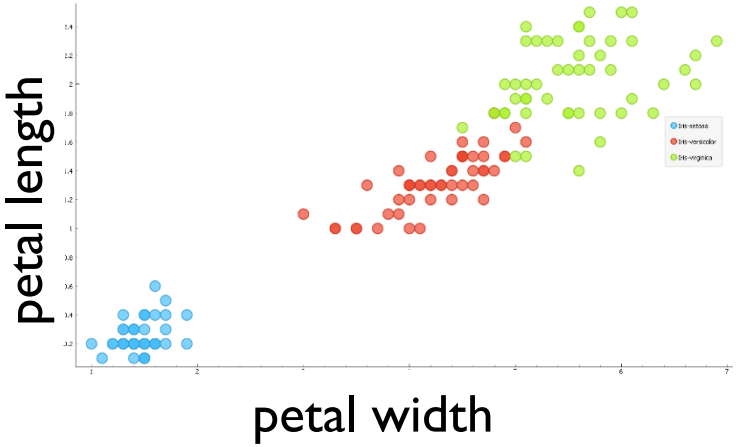
Sınıflandırma görevi için bir sınıflandırıcı/model (classifier) inşa ederiz.

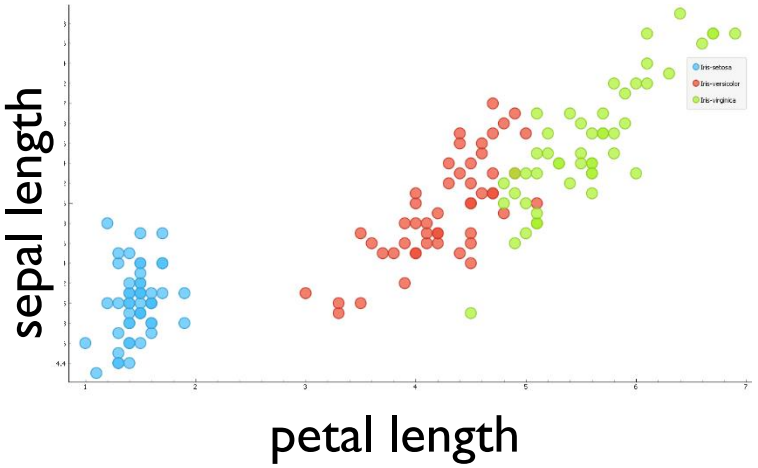
Sınıflandırıcı giriş uzayindan çikis uzayina bir fonksiyondur:

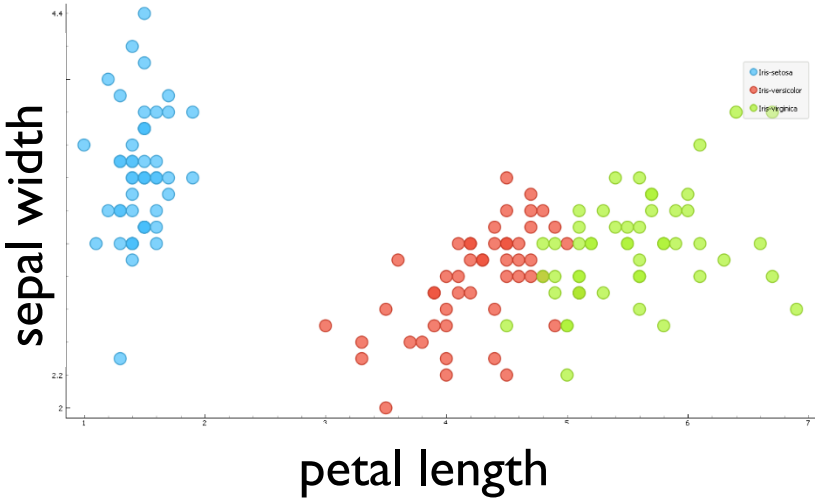
$$f: X \rightarrow Y$$

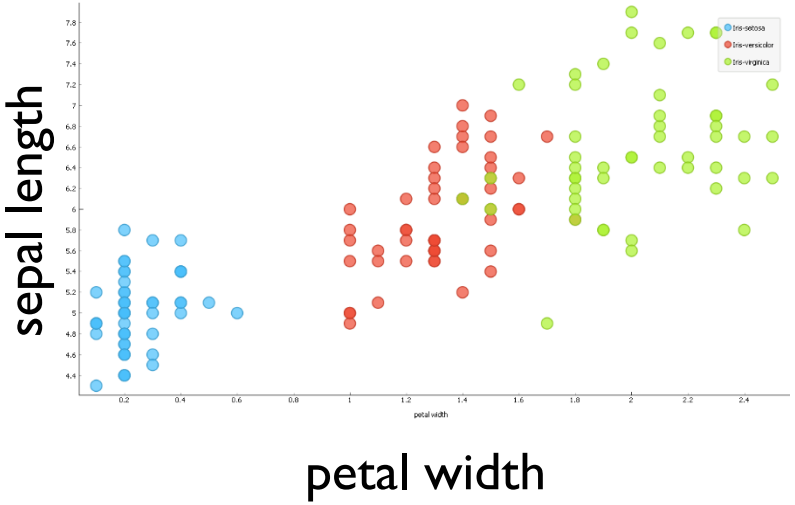
Burada X'in elemanlari bir vektördür.
ör. $(4.9, 3.1, 1.5, 0.1) \in X$

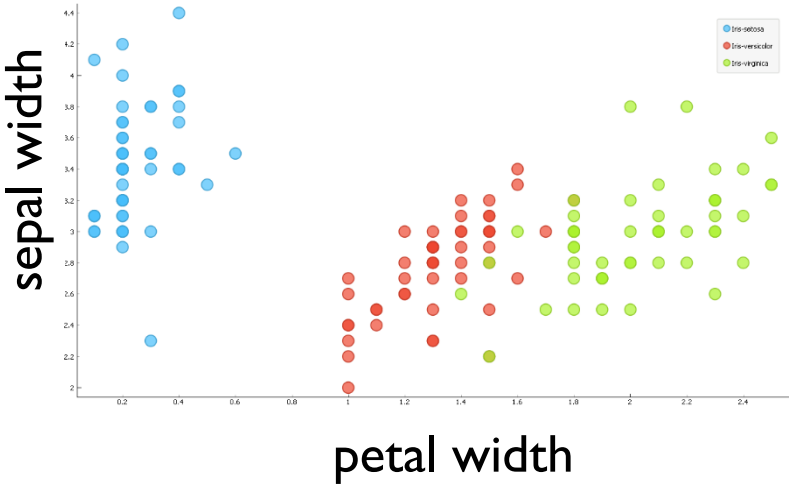
X giriş uzayi (input space) Y çıkış uzayi (output space)

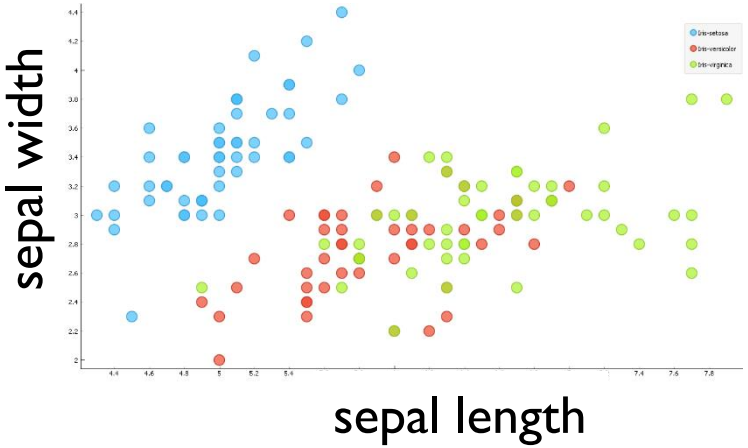












Çeşitli özellik çiftlerine göre iris türlerinin dagilimi

Sınıflandırıcı (Classifier) İnşaası

- Bir siniflandirici inşa ederken genel olarak verilen veri seti %70 ve %30 ve oraninda *rastgele* ikiye ayrilir.

- Verinin %70'lik kısmı eğitim seti (training set) olur. Eğitim seti siniflandirma algoritmasina verilir. Yada başka bir deyişle siniflandirma algoritmasi eğitim setindeki örneklerle *eğitilir*.

  Eğitilen siniflandirma algoritmasi ürün olarak bir siniflandirici verir.

- Elde edilen siniflandirici verinin kalan %30'lik kısmına (test seti) uygulanir. Siniflandiricinin buradaki perfomansı test edilir, ve daha sonra rapor edilir.

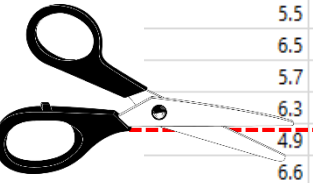
Sınıflandırıcı (Classifier) İnşaası

sınıflandırma algoritması

sepal length	sepal width	petal length	petal width	iris
4.9	3.1	1.5	0.1	Iris-setosa
4.4	3.0	1.3	0.2	Iris-setosa
5.1	3.4	1.5	0.2	Iris-setosa
5.0	3.5	1.3	0.3	Iris-setosa
4.5	2.3	1.3	0.3	Iris-setosa
4.4	3.2	1.3	0.2	Iris-setosa
5.0	3.5	1.6	0.6	Iris-setosa
5.1	3.8	1.9	0.4	Iris-setosa
4.8	3.0	1.4	0.3	Iris-setosa
5.1	3.8	1.6	0.2	Iris-setosa
4.6	3.2	1.4	0.2	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor
4.9	2.4	3.3	1.0	Iris-versicolor
6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor
5.0	2.0	3.5	1.0	Iris-versicolor
5.9	3.0	4.2	1.5	Iris-versicolor
6.0	2.2	4.0	1.0	Iris-versicolor
6.1	2.9	4.7	1.4	Iris-versicolor
5.6	2.9	3.6	1.3	Iris-versicolor

%70

%30



eğitim seti

test seti

sepal length	sepal width	petal length	petal width	iris
4.9	3.1	1.5	0.1	Iris-setosa
4.4	3.0	1.3	0.2	Iris-setosa
5.1	3.4	1.5	0.2	Iris-setosa
5.0	3.5	1.3	0.3	Iris-setosa
4.5	2.3	1.3	0.3	Iris-setosa
4.4	3.2	1.3	0.2	Iris-setosa
5.0	3.5	1.6	0.6	Iris-setosa
5.1	3.8	1.9	0.4	Iris-setosa
4.8	3.0	1.4	0.3	Iris-setosa
5.1	3.8	1.6	0.2	Iris-setosa
4.6	3.2	1.4	0.2	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor

sınıflandırıcı
f
(classifier)

Notlar...

- Veri seti %70- %30 ayrilirken örneklerden rastgele secilim yapilir. Yani örneklerin rastgele %70 'i secilir, bunlar eğitim setini olusturur. Kalan %30 'luk kisim test seti olur.

- Bazen, veri setinde bazi siniflardan cok az ornek olur. (ornegin 90 tane *A* sinifindan 5 tane *B* sinifindan). Böyle bir durumda rastegele secilim yaptigimizda az sinifin orneklerinin (*B*'nin) *tamamı* egitim setine gidebilir;siniflandiriciyi test ederken yalnizca baskin siniftaki (A) performansini test etmiş oluruz. O yuzden böyle durumlarda hem egitim setinde hem de test setinde butun siniflardan ornekler oldugundan emin olunmaya calisilir.

- Siniflandiricinin perfomansi test seti uzerine olculur. Test setindeki örnekleri siniflandirici daha önce hic görmediginden bu test uzerinden elde edilen perfomans degeri siniflandiricinin perfomansini tarafsiz/önyargisiz (unbiased) olarak verir.

  Bu performans degeri siniflandiricinin genelleme yapabilme kapasitesi (generalization power) ile igili ipucu verir. Böylece ileride örnekleri ne kadar iyi siniflandiracagi hakkinda bir öngörümüz olur.

k-En Yakın Komşu Algoritmasi (*k*-Nearest Neighbour) (Komşunun Yaptigini Yap)

k-en yakın komşu algortimasinda her bir test örneginin eğitim setindeki bütün örneklere uzakliklari hesaplanir. Egitim setindeki orneklerden en yakin *k* tanesi bulunur. Bulunan *k* tane komşu icerisinde hangi sınıf en yaygınsa o sınıf test örneginin sınıfı olarak tahmin edilir.

ör.

Cinsiyet	Yaş	Medeni Hal	Eğitim	Aylik Gelir(TL)	Kredi Geçmişi	Kredi Ödendi mi?
1	61	1	4	6600	1	1
1	26	0	2	1900	0	0
0	33	1	3	4800	1	0
0	39	0	4	8200	1	1
0	30	0	3	5600	1	→ Test örneği

Eğitim Seti

Sınıf

cinsiyet 1:erkek 0: kadın; medeni hal: 1: evli 0 bekar; eğitim: 1:ilkokul, 2: lise, 3: universite, 4: lisansustu; kredi geçmişi: 1:temiz, 0 temiz degil; kredi odendi mi: 1 aldigi krediyi odedi, 0 odemedi

Öklid uzakligina göre test örneginin eğitim örneklerine uzakliklari ilk sutunda verilmistir.

k'yi 3 alalim. Yani en yakın üç tane komşusuna bakalim.

Uzaklık	Kredi ödendi mi?
1000	1
3600	0
800	0
2600	1

En yakın komşular. Bu komşularin ikisi aldigi krediyi ödemis, biri odememistir. Şu halde test ornegini 1 olarak siniflandiririz. Yani krediyi odeyeceğini tahmin ediyoruz.

Öklid Uzakligi: x ve x', n boyutunda iki vektor olsun. x_i, x'in i.elemani olsun ($i = 1, \dots, n$).

x ve x' arasindaki oklid uzakligi:

$$d_{oklid}(x, x') = \sqrt{(x_1 - x_1')^2 + \cdots + (x_n - x_n')^2} = \left(\sum_{i=1}^{n} (x_i - x_i')^2\right)^{\frac{1}{2}}$$

ör. Test örneginin ikinci egitim ornegine uzakliginin hesaplanmasi:

$$\sqrt{(0-1)^2+(30-26)^2+(0-0)^2+(3-2)^2+(5600-1900)^2+(1-0)^2}$$

Bu uzaklik hesabi $(5600-1900)^2$ tarafindan domine edilir!! Bu yuzden aylik gelir, siniflandirma tahminimize en buyuk katkiyi yaparken, diger özelliklerin (örnegin kredi geçmişinin) tahminimize katkisi çok az olur.

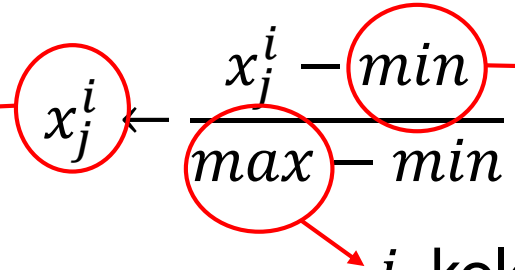
Bu problemi aşmak için kolonlari (özellikleri) normalize ederiz. Böylece her kolon aynı aralıkta yer alır. Bu ise her özelliğin uzaklık hesabına ayni derecede katkıda bulunmasini sağlar!

Min-Max Normalizasyonu

Min-max normalizasyonu ile her bir kolondaki değerler $[0-1]$ araliginda yer alir.

Degerleri $[0-1]$ araligina getirmek için her bir kolonda, o kolonun minimum değeri kolondaki degerleden cikarilir daha sonra bu değerler $max-min$ farkına bolunur.

i. örnegin (satirin) j. özelligi (kolonu)



Normalizyon sonrasi:

Cinsiyet	Yaş	Medeni Hal	Eğitim	Aylik Gelir(TL)	Kredi Geçmişi	Kredi Ödendi mi?
1	1	1	1	0.74	1	1
1	0	0	0	0	0	0
0	0.2	1	0.5	0.46	1	0
0	0.37	0	1	1	1	1

Eğitim Seti

0	0.11	0	0.5	0.58	1	→ Test örneği

Normalizyon sonrasi oluşan uzaklıklar:

Uzaklık	Kredi ödendi mi?
1.75	1
1.61	0
1.01	0
0.7	1

En yakın komşular. Bu komşularin ikisi aldigi krediyi ödememiş, biri ödemiştir. Test ornegini 0 olarak siniflandiririz. Yani krediyi odeyemeyecegini tahmin ediyoruz!

Ağırlıklandılımış Uzaklık Fonksiyonu (Weighted Distance Funtion)

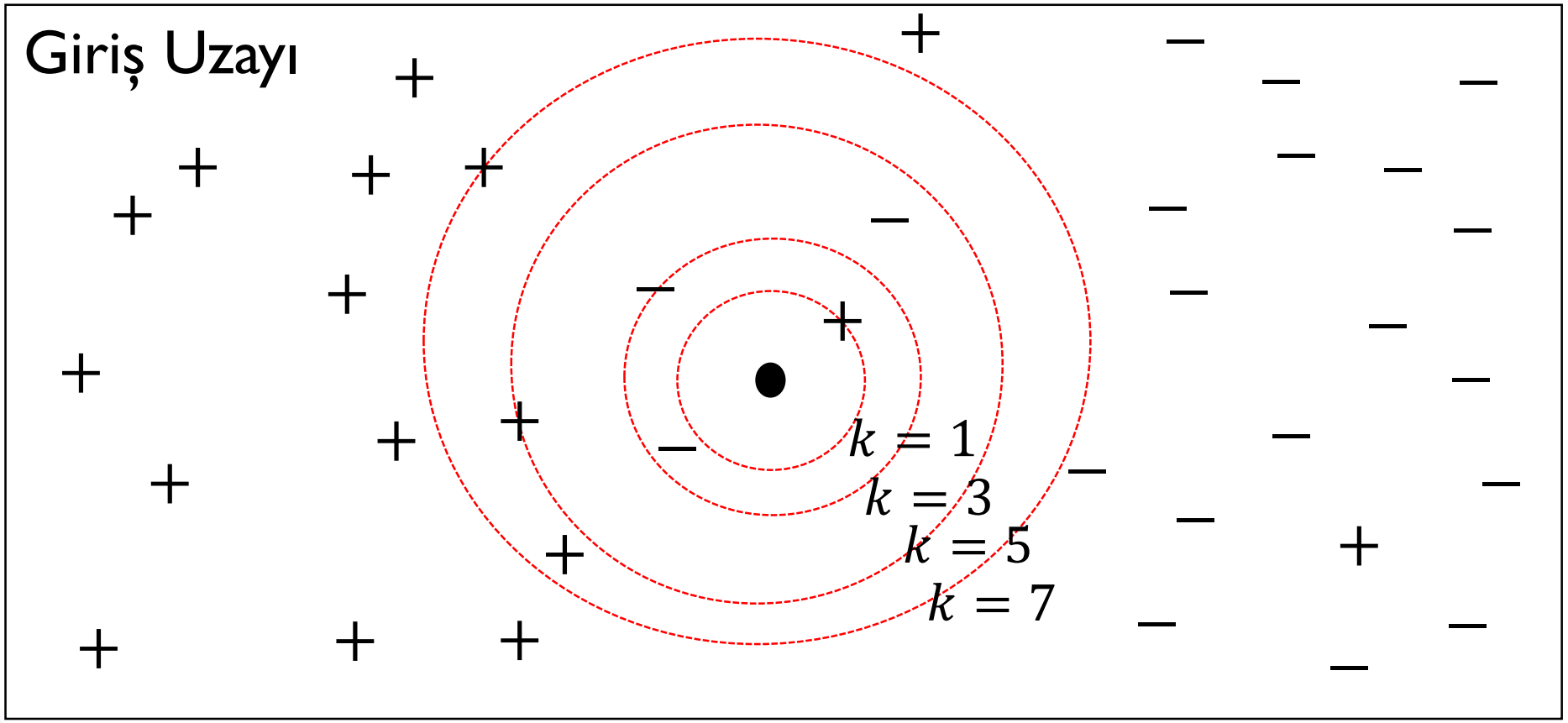
Eğer bazi ozelliklerin uzaklık hesabında daha önemli bazilarinin ise daha az önemli olduğunu düşünüyorsak, bu düşüncemizi uzaklık hesabına yansıtabiliriz. Bunun için önemli gordugumuz ozellige (kolona) yüksek bir agirlik atariz, önemsize ise daha dusuk bir agirlik atariz.

$$d_{w\_oklid}(x, x') = \sqrt{w_1(x_1 - x_1')^2 + \cdots + w_n(x_n - x_n')^2} = \left(\sum_{i=1}^{n} w_i(x_i - x_i')^2\right)^{\frac{1}{2}}$$

Peki k Ne Olmalıdır?

k-en yakın komşu algortimasinda k, test orneginin uzaklık hesabından sonara dikkate alinacak komsu sayisini belirtir.

k genellikle $1, 3, 5$ ... gibi bir tek sayi olarak alinir; fakat k bir *parametredir* ve kaç alinacagina dikkatlice karar verilmelidir.



Peki k Ne Olmalıdır?

k	# Poz. Eğt. Örneği	# Neg. Eğt. Örneği	Karar
1	1	0	Pozitif
3	1	2	Negatif
5	2	3	Negatif
7	4	3	Pozitif

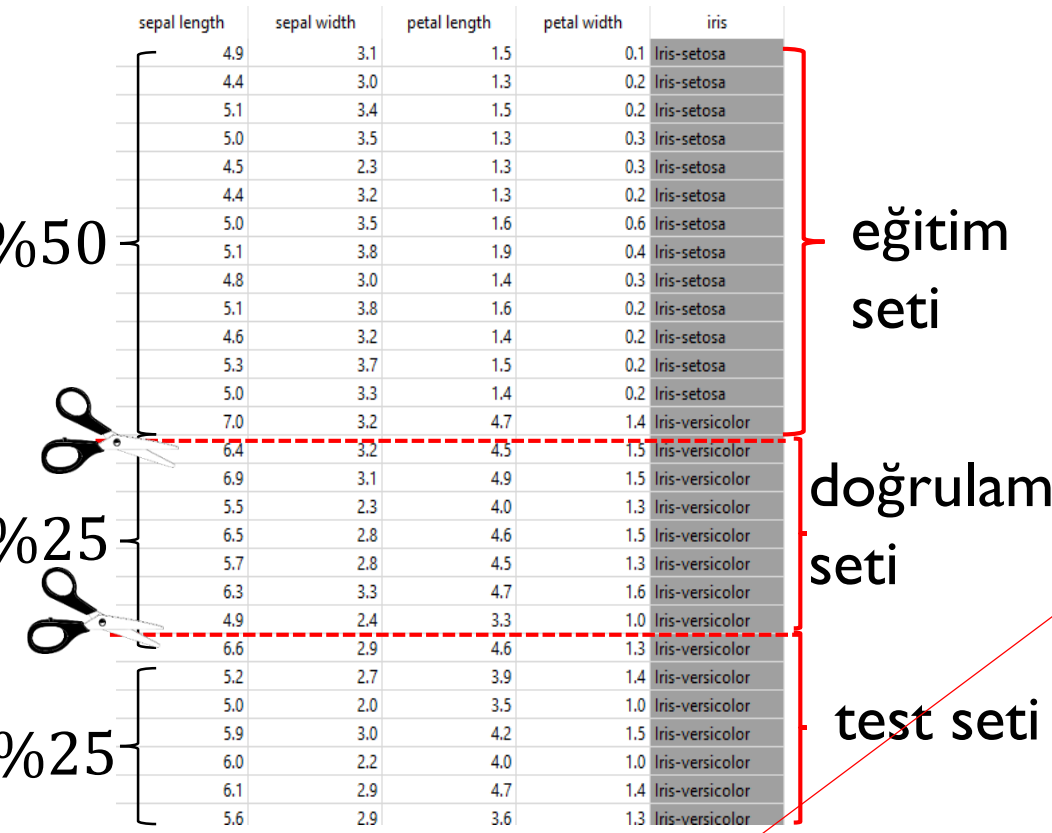
Test örneğini pozitif olarak mı negatif olarak mı sınıflandıracağımız k seçimine bağlıdır!

Doğrulama (Validation) Seti ile Parametre Belirleme

Eger kullandigimiz siniflandirma algoritmasi paremetrik ise (yani ayarlanmasi gereken parametre yada parametreler içeriyorsa) bir doğrulama kümesi kullaniriz.

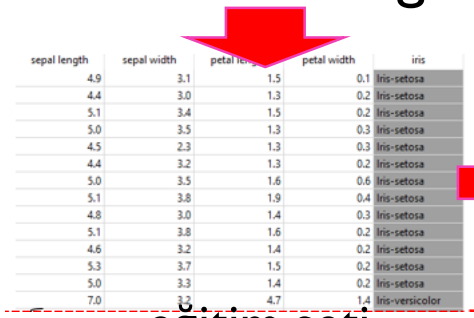
Burada temel mantık algoritmanin farklı parametreler kullandiğimiz zamanki perfomansini bağimsiz bir set (doğrulama kümesi) üzerinde ölçmektir. En yüksek perfomansi getiren parametre, algoritmanin parametresi olarak seçilir.

Genel kural veri setini %50 eğitim seti, %25 doğrulama seti ve %25 test seti olacak şekilde üçe bölmektir.

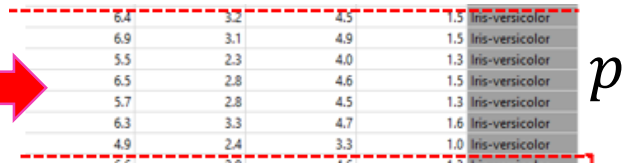


sınıflandırma algoritması her defasında farklı bir parametre kullaniyor!

sınıflandırma algoritması

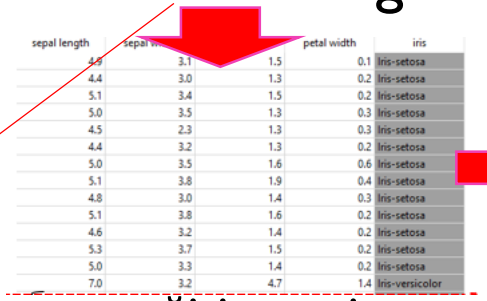


f_1

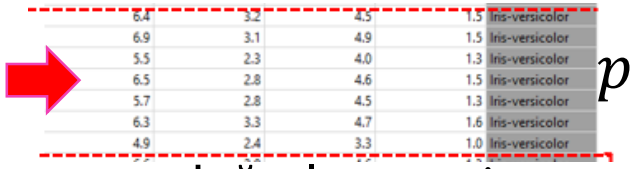


$perform_1$

sınıflandırma algoritması



f_2



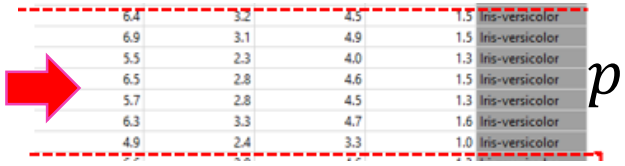
$perform_2$

...

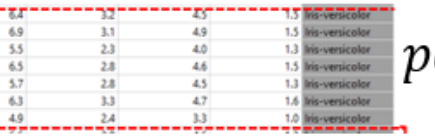
...

sınıflandırma algoritması

f_n

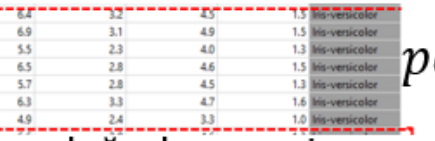


$perform_n$



doğrulama seti

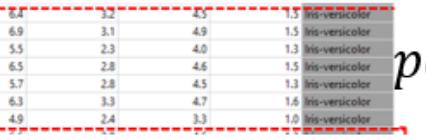
$perform_1$



doğrulama seti

$perform_2$

...

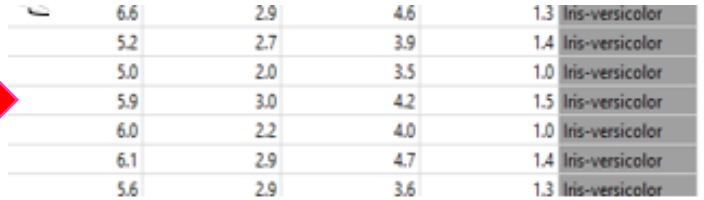


doğrulama seti

$perform_n$

Doğrulama setindeki performansi en yüksek olan siniflandirici seçilir. Bu siniflandirici final siniflandiricimiz olur. Diyelim ki bu f_2 olsun. f_2'yi test set için çalistirip, f_2'nin test setindeki perfomansini rapor ederiz.

f_2



6.6	2.9	4.6	1.3	Iris-versicolor
5.2	2.7	3.9	1.4	Iris-versicolor
5.0	2.0	3.5	1.0	Iris-versicolor
5.9	3.0	4.2	1.5	Iris-versicolor
6.0	2.2	4.0	1.0	Iris-versicolor
6.1	2.9	4.7	1.4	Iris-versicolor
5.6	2.9	3.6	1.3	Iris-versicolor

test seti

Doğrulama (Validation) Seti ile k-En Yakın Komşu'da k Belirleme

1) Veri seti $\%50$ eğitim seti, $\%25$ doğrulama seti ve $\%25$ test seti olacak şekilde üçe bölünür.

2) Her bir farkli k değeri ($k = 1, 3, 5, 7 \ldots$) için:
   - Doğrulama setindeki her eleman için:
     - Eğitim setindeki k tane komşusu bulunur.
     - Bulunan komsulardaki en yaygın sınıf bulunur.
     - Bulunan sınıf doğrulama setinin elemanının sınıfı olarak tahmin edilir.
   - Dogrulama setindeki elemanlarinin yüzde kacinin sinifinin doğru olarak tahmin edildiği hesaplanir.

3) En yüksek tahmin yüzdesi veren k seçilir.

k-En Yakın Komşu Algoritmasi Üzerine Notlar..

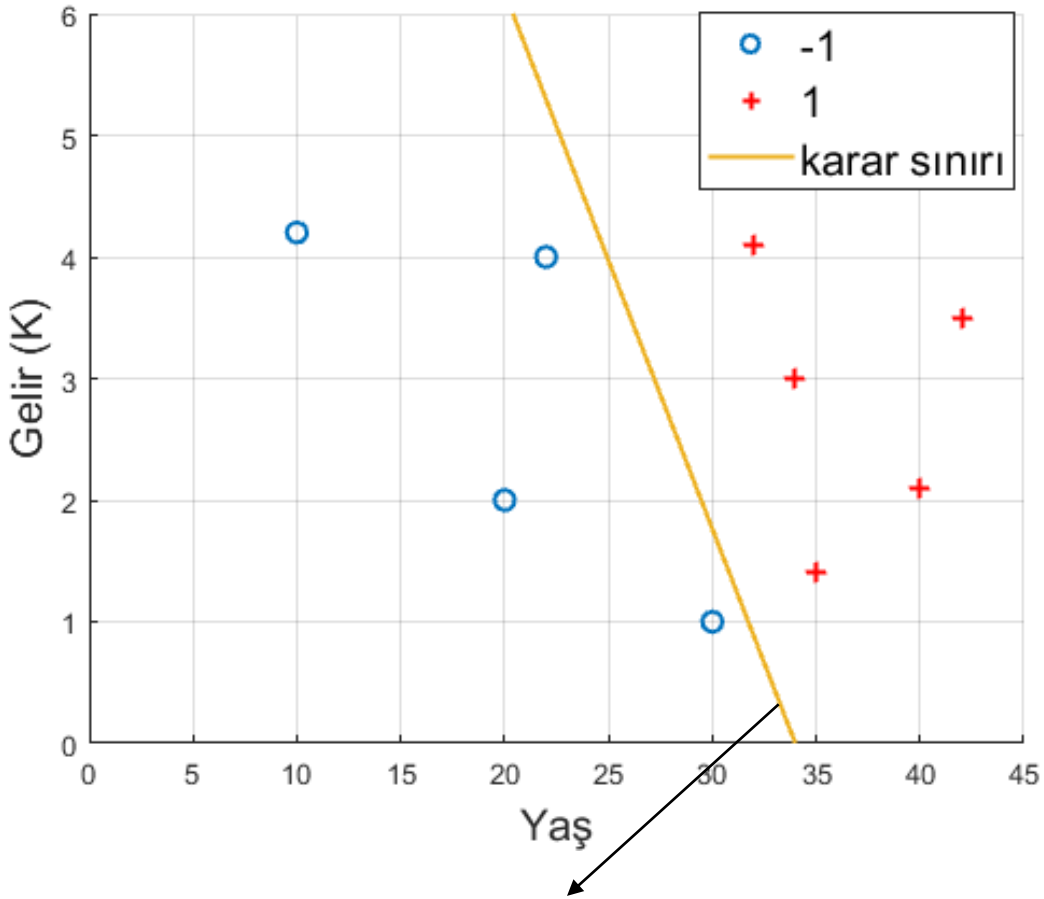
1) k-en yakın komşu algoritmasi bir *tembel öğrenme* örneğidir. Bir test örneğiyle karsilaşilincaya kadar hiçbir şey yapılmaz, bir siniflandirici, bir model inşa edilmez. Onun yerine eğitim setinin tamami hafızada tutulur.

2) Her bir test ornegi için eğitim setinin tamamı taranarak en yakin komşular bulunmaya calisilir. Bu da tahminimize yavaşlık getirir.

3) k-en yakın komşu, örnekler arasindaki uzakligi temel alır. Bu uzaklıklar boyutun genişliğinden (ozellik saysinindan) negatif olarak etkilenir: boyut arttikca ornekler biribirinden uzaklaşır. Dolayisiyla k-en yakın komşu boyutun lanetinden (curse of dimensionality) etkilenir. Bu yüzden k-en yakın komşu algortimasina baslamadan önce ozellik sayisini azaltmak önemli bir adimdir.

Perceptron Algoritması

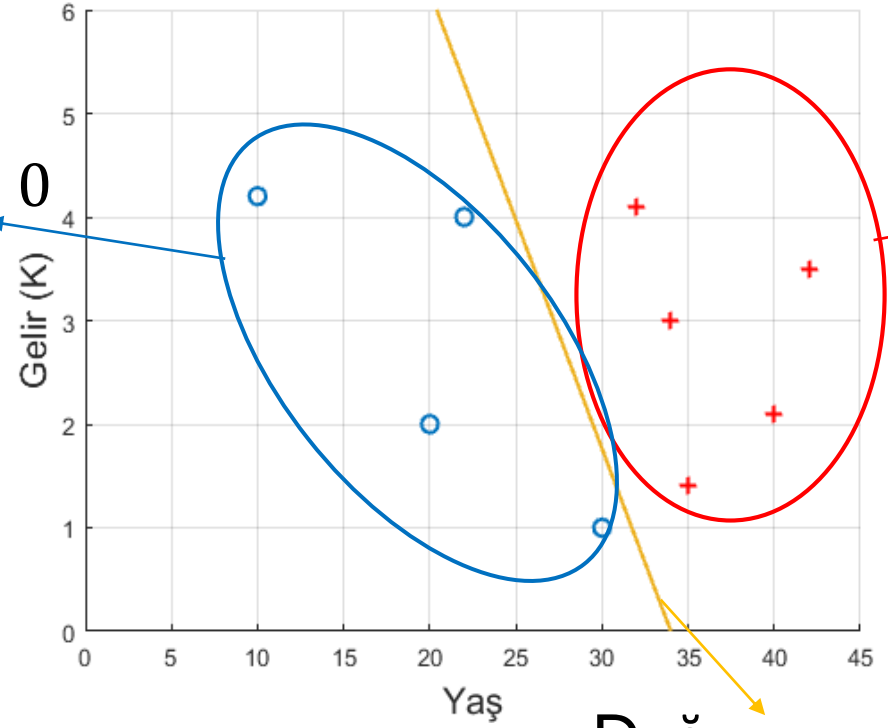
Yaş (x_1)	Gelir(K) (x_2)	Kredi Ödeme Durumu
30	1	-1
20	2	-1
22	4	-1
10	4.2	-1
35	1.4	1
34	3	1
32	4.1	1
40	2.1	1
42	3.5	1



Pozitif ve negatif örnekleri dogrusal olarak ayıran $w_0 + w_1x_1 + w_2x_2 = 0$ doğrusunu arıyoruz. Bu doğrunun diğer adı 'karar sınırı' (decision boundary) dir.

Perceptron Algoritması

Negatif örneklerde
$w_0 + w_1 x_1 + w_2 x_2 < 0$



Pozitif örneklerde
$w_0 + w_1 x_1 + w_2 x_2 > 0$

Doğru uzerinde
$w_0 + w_1 x_1 + w_2 x_2 = 0$

Amacımız eğitim setini kullanarak w_0, w_1 ve w_2 katsayilarini bulmak. Bu katsayilar bir kere bulundugunda test örneklerini

$$f = sign(w_0 + \sum_{i=1}^{2} w_i x_i)$$

fonksiyonu ile -1 veya $+1$ olarak sınıflandırabiliriz.

Not 1: $sign$, işaret (signature) fonksiyonudur: $sign(x) = \begin{cases} +1, x > 0 \\ -1, x < 0 \end{cases}$

Not 2: *w_0, w_1 ve w_2 katsayilarini bulmak demek veri setini özetlemek demektir! Bu katsayıları bulduktan sonra, egitim setini hafizada tutmak zorunda degiliz. Hatirlarsak k-en yakin komsu algoritmasinda egitim setini her zaman hafizada tutuyorduk.*

Perceptron (Algılayıcı) Algoritmasi

Perceptron algoritmasi eğer pozitif ve negatif örnekler birbirinden bir doğru ile ayrılabiliyorsa bu doğruyu bulmayı garanti eder.

İteratif bir algoritmadir. Her bir iterasyonda eğitim setinin bütün örnekleri kontrol edilir. Eğer bir örnek yanlış sınıflandırılmışsa o örnek sınıfı ile çarpılıp o anki $w = (w_0, w_1, \ldots, w_n)$ vektörüne eklenir.

Yanlış sınıflandırılan bir örnek kalmadığında algoritma sonlanır.

örneğin (x^j, y^j) yanlış sınıflandırılmiş bir egitim örnegi olsun: $sign(w_0 + \sum_{i=1}^{n} w_i x_i^j) \neq y^j$

Bu durumda $w = (w_0, w_1, \ldots, w_n)$ vektörünün güncellenmesi şu şekilde olur.

$$w_0 := w_0 + y^j$$
$$w_1 := w_1 + y^j \cdot x_1^j$$
$$\ldots$$
$$w_n := w_n + y^j \cdot x_n^j$$

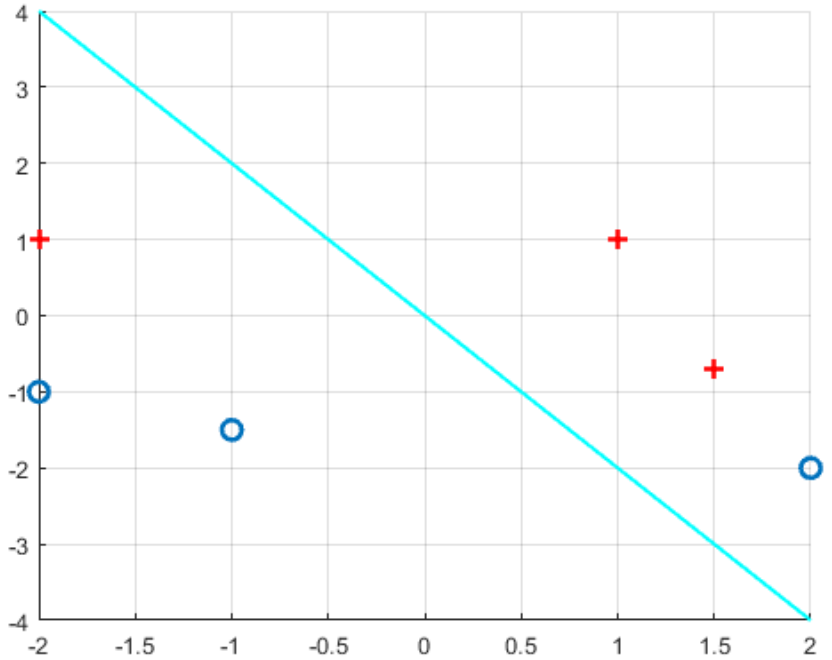
Son olarak,

Perceptron algoritmasinin iterasyonun daha cabuk sonlanmasi icin adım büyüklüğü (step size) yada öğrenme oranı (learning rate) denilen bir η sabiti kullanılır. Örnekler η ile carpilarak w'ya eklenir:

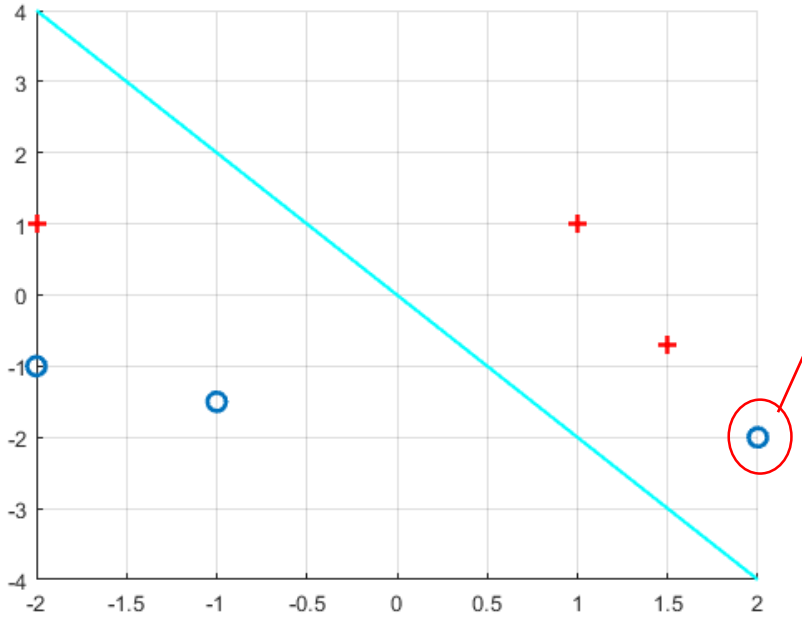
$$w_0 := w_0 + \eta \cdot y^j$$
$$w_1 := w_1 + \eta \cdot y^j \cdot x_1^j$$
$$\ldots$$
$$w_n := w_n + \eta \cdot y^j \cdot x_n^j$$

- η (eta) $[0,1]$ arasi bir deger alir. η veriden öğrenilmez. Algoritmanin başında ne alınılacagina karar verilir. Örneğin $\eta = 0.1$ alalim, yada $\eta = 0.2$ alalim gibi.
- $w = (w_0, w_1, \ldots, w_n)$ vektörünün başlangıç degeri olarak genelde $w = (0,0, \ldots, 0\ )$ vektörü alınır.

ör. Pozitif (+) ve negatif (o) örnekler giriş uzayinda aşagidaki gibi dagilmis olsun. Ve w vektoru başlangiç olarak $w = (0, 1, 0.5)$ olarak verilsin. $\eta = 0.2$ alalim



iter #1:



yanlış sınıflandırılmış örnek:

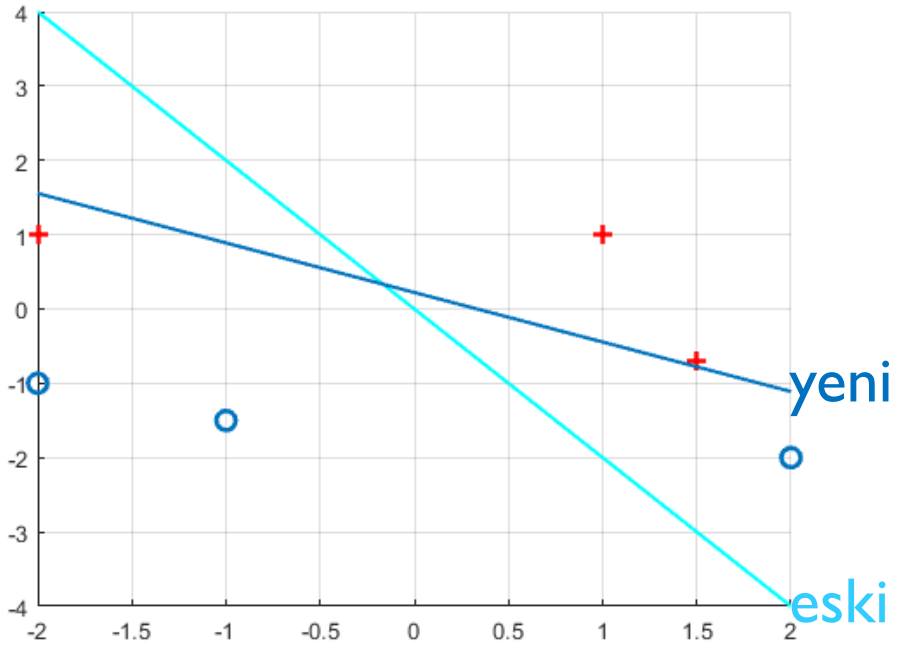
$sign(w_0 + \sum_{i=1}^{2} w_i x_i) =$

$sign(0 + 2 \cdot 1 + (-2) \cdot 0.5) = sign(1) = 1 \neq -1.$

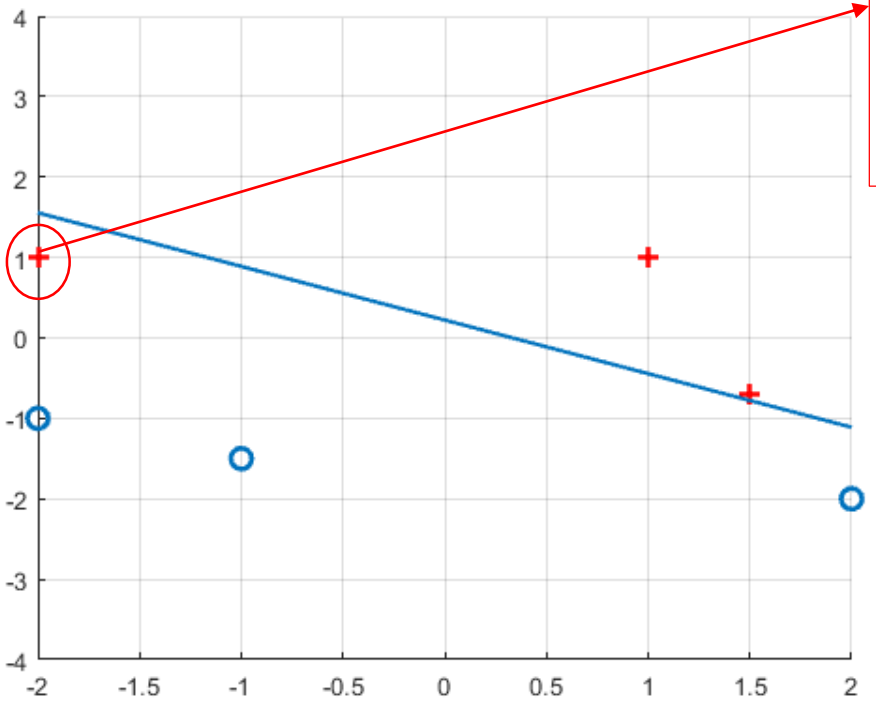
$w_0 := 0 - 0.2 \cdot 1 = -0.2$

$w_1 := 1 - 0.2 \cdot 2 = 0.6$

$w_2 := 0.5 - 0.2 \cdot (-2) = 0.9$



iter #2:

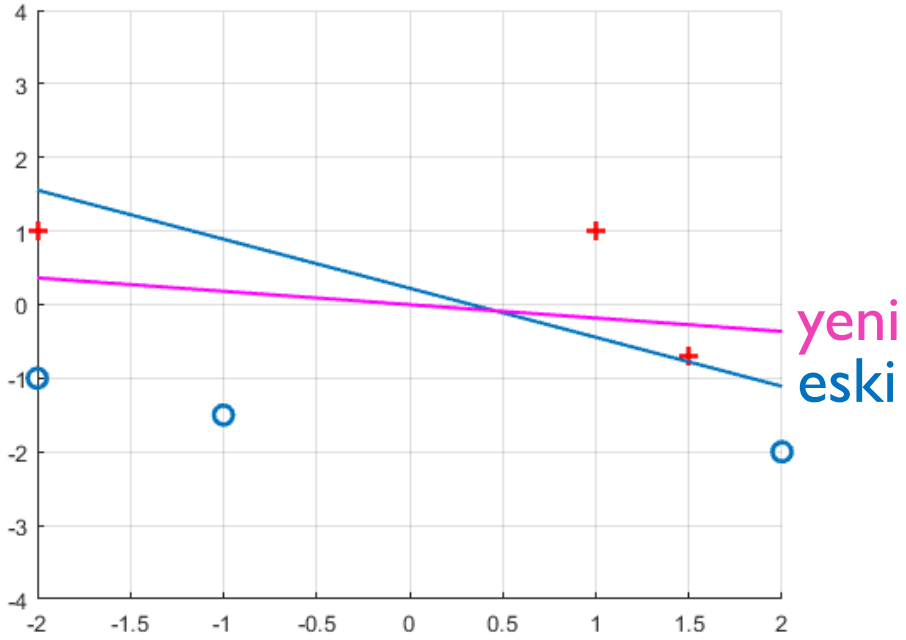


yanlış sınıflandırılmış örnek:

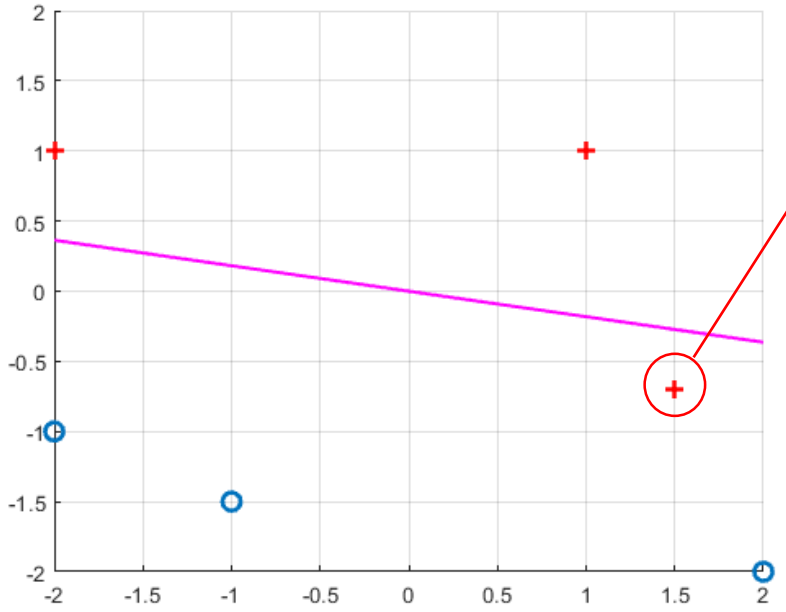
$sign(w_0 + \sum_{i=1}^{2} w_i x_i) =$

$sign(-0.2 \ -2 \cdot 0.6 + 1 \cdot 0.9) = sign(-0.5) = -1 \neq 1$

$$w_0 := -0.2 + 0.2 \cdot 1 = 0$$
$$w_1 := 0.6 + 0.2 \cdot (-2) = 0.2$$
$$w_2 := 0.9 + 0.2 \cdot 1 = 1.1$$



iter #3:



yanlış sınıflandırılmış örnek:

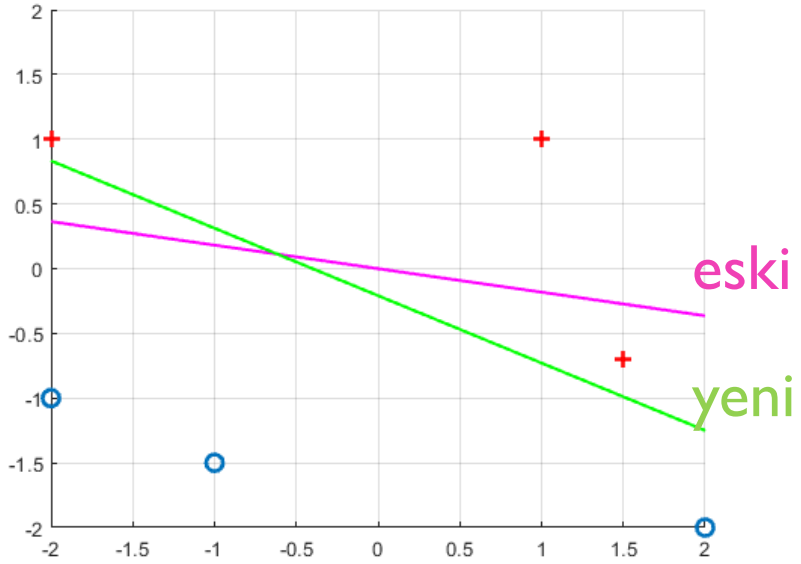
$sign(w_0 + \sum_{i=1}^{2} w_i x_i) =$

$sign(0 + 1.5 \cdot 0.2 - 0.7 \cdot 1.1) = sign(-0.4) = -1 \neq 1$

$w_0 := 0 + 0.2 \cdot 1 = 0.2$

$w_1 := 0.2 + 0.2 \cdot 1.5 = 0.5$

$w_2 := 1.1 + 0.2 \cdot (-0.7) = 0.96$



Bulunan bu son doğru pozitif ve negatif örnekleri hatasız bir sekilde ikiye ayırır!

Perceptron Algoritmasi Pseudo Kod:

Giris: $D = \{(x^1, y^1), \ldots, (x^m, y^m)\}, \left(x^i \in \mathbb{R}^n, y^i \in \{-1,1\}, i = 1, \ldots, m\right)$ veri seti

η adim buyuklugu, $w = (w_0, w_1, \ldots, w_n)$ baslangic vektoru

Cikis: $w = (w_0, w_1, \ldots, w_n)$ vektoru

```
1.flag=true
2.while flag
3.      flag=false
4.       for (x,y) ∈ D  // her bir örnek için
5.           if sign(w0 + Σ_{i=1}^{n} wi xi) ≠ y   //x eger yanlis siniflandirilmissa
6.               w0 := w0 + η · y    //w guncelle
7.               for i=1:n      //  w guncelle
8.                   wi := wi + η · y · xi
9.               end for
10.               flag=true //  dongu devam etsin
11.           end if
12.       end for
13.end while
```