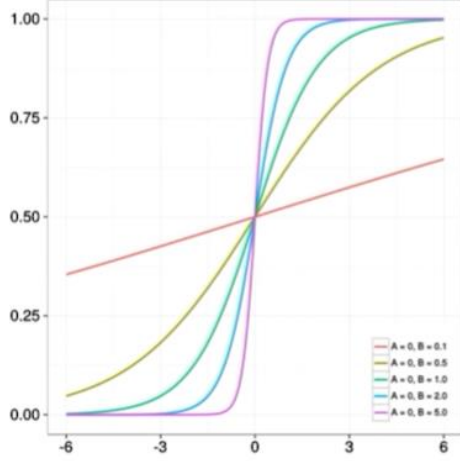


CLASSIFICATION

Logistic Function

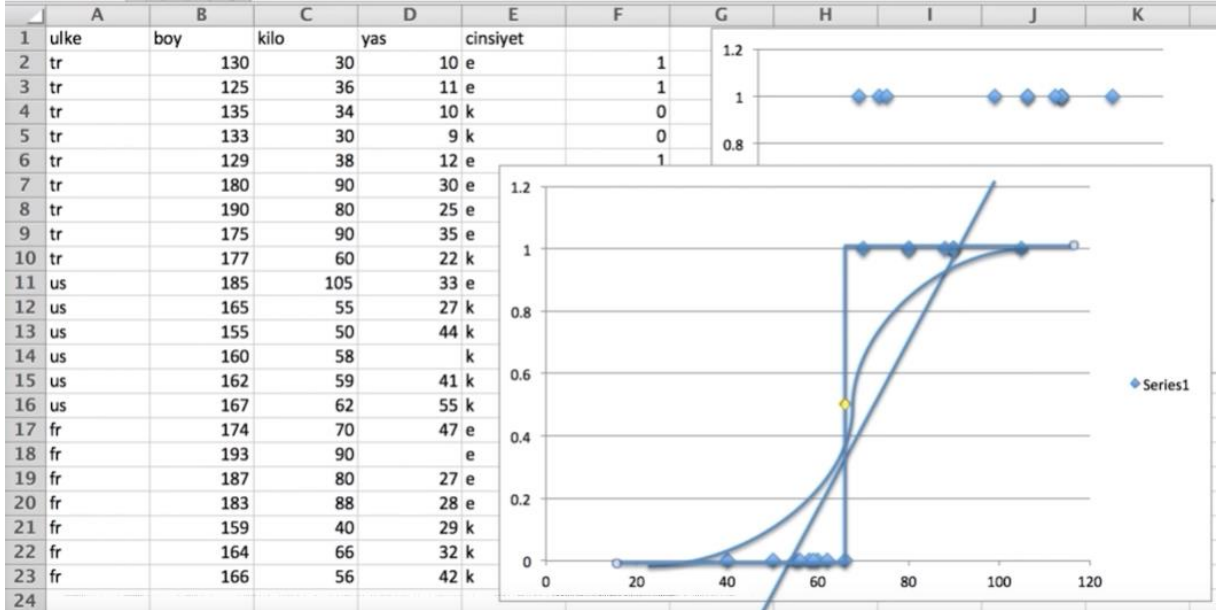


$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

$$t = \beta_0 + \beta_1 x \quad t = A + Bx$$

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_m x_m = \beta_0 + \sum_{i=1}^m \beta_i x_i$$



veri setini eğitim ve test kümelerine bölerken `train_test_split` fonksiyonunu kullanırsanız ve `random_state = 0` olarak ayarlarsanız, her zaman aynı şekilde bölecektir. Bu, çalışmanızın tekrarlanabilirliğini sağlar ve sonuçların diğerleriyle karşılaştırılmasını kolaylaştırır.

KARMAŞIKLIK MATRİSİ(confusion matrix)

from sklearn.metrics import confusion_matrix

Karmaşıklık Matrisi

	C ₁	C ₂
C ₁	True positive	False negative
C ₂	False positive	True negative

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.52

- Accuracy M, acc(M): model M için yüzde kaç doğru sınıflandırma olduğudur
 - Error rate (misclassification rate) = 1 – acc(M)
 - Alternatif ölçümler (e.g., for cancer diagnosis)
 - sensitivity = t-pos/(t-pos+f-neg) /* true positive recognition rate */
 - specificity = t-neg/(t-neg+f-pos) /* true negative recognition rate */
 - precision = t-pos/(t-pos + f-pos)
 - accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)

(sınıflandırma başarısını ölçmek)

```
log_reg = LogisticRegression(random_state=0)
log_reg.fit(x_train_sc, y_train)

y_pred = log_reg.predict(x_test_sc)

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
wdir='C:/Users/t...
basari oranı= %
C:\Users\terzi\...
vector y was pas...
using ravel().
y = column_or_...

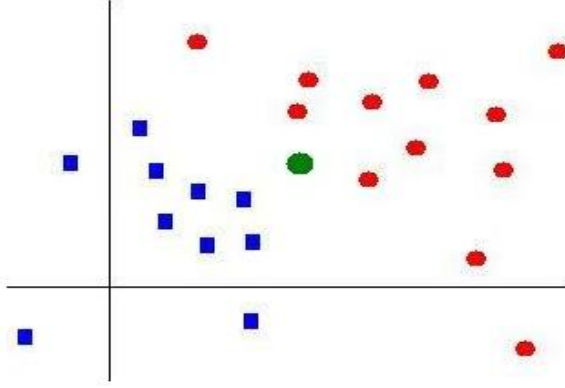
In [30]: runfile
wdir='C:/Users/t...
[[ 9  5]
 [ 0 14]]
C:\Users\terzi\...
vector y was pas...
using ravel().
y = column_or_...

In [31]:
```

```
[[ 9  5]
 [ 0 14]]
```

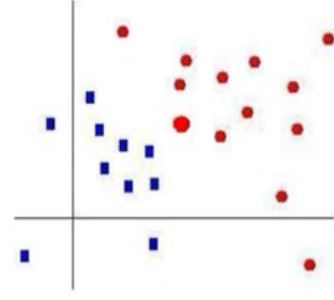
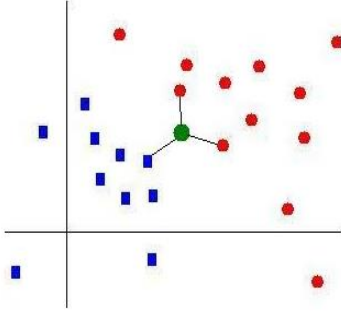
- Model, 9 örneği doğru bir şekilde negatif olarak sınıflandırmıştır (True Negatives - TN).
- Model, 5 örneği yanlış pozitif olarak sınıflandırmıştır (False Positives - FP).
- Model, 0 örneği yanlış negatif olarak sınıflandırmıştır (False Negatives - FN).
- Model, 14 örneği doğru bir şekilde pozitif olarak sınıflandırmıştır (True Positives - TP).

KNN (K nearest neighborhood, en yakın k komşu)



Yeşil örneğin sınıflandırılması için K-NN algoritması kullanılırsa,

En yakın 3 (k=3 ise) komşudan 2 komşu kırmızı olduğu için yeşil örnek kırmızı sınıfına dahil edilecektir



K == 4[cift] için en yakın komşuların 2 kırmızı 2 mavi olduğu durumda hangilerinin mesafesi daha yakınsa o sınıfa dahil edilir

Yakın örneklerin bulunması için *oklid bağlantısı* kullanıyor

$$[(x_1, y_1) \text{ ve } (x_2, y_2) \rightarrow d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}]$$

K-NN'de kullanılan bazı öğrenme (learning) yöntemleri şunlardır:

1. **Lazy Learning (Tembel Öğrenme):** K-NN, tembel öğrenme olarak bilinen bir yaklaşımı kullanır. Bu modelin eğitim aşamasında hiçbir şey yapmaması ve sadece tahmin yapmak için gelen veriye dayanması anlamına gelir. Veri seti alındığında, K-NN tüm veriyi belleğe yükler ve tahmin yapmak için gelen yeni örneklerle karşılaştırır. Bu nedenle, K-NN gerçek anlamda öğrenme yapmaz, sadece veri setinin doğrudan öğrenilmesi ve saklanması şeklinde çalışır.
2. **Instance-Based Learning (Örnek Temelli Öğrenme):** K-NN, örnek temelli bir öğrenme yöntemidir. Bu, modelin sınıflandırma yapmak için etiketli örneklerin bir koleksiyonunu kullanması anlamına gelir. Yeni bir örneği sınıflandırmak için, K-NN, bu örneği en yakın komşularına dayanarak sınıflandırır. Yani, öğrenme süreci, veri setinin örneklerine dayanır ve herhangi bir özneteliği çıkarmaz veya özetlemez.
3. **Distance-Based Learning (Uzaklık Temelli Öğrenme):** K-NN, sınıflandırma yaparken örnekler arasındaki uzaklığı ölçerek karar verir. Bu nedenle, K-NN'nin bir tür uzaklık temelli öğrenme olduğu söylenebilir. Örneğin, en yakın komşularını seçerken, öklid mesafesi gibi bir uzaklık metriği kullanılır. Uzaklık temelli öğrenme, K-NN'nin temelinde yatan prensiplerden biridir.

NearestNeighbors(n_neighbors=3, algorithm='any_distance_function')

Metrics intended for real-valued vector spaces:

identifier	class name	args	distance function
"euclidean"	EuclideanDistance	•	$\sqrt{\sum((x - y)^2)}$
"manhattan"	ManhattanDistance	•	$\sum(x - y)$
"chebyshev"	ChebyshevDistance	•	$\max(x - y)$
"minkowski"	MinkowskiDistance	p, w	$\sum(w * x - y ^p)^{1/p}$
"seuclidean"	SEuclideanDistance	V	$\sqrt{\sum((x - y)^2 / V)}$
"mahalanobis"	MahalanobisDistance	V or VI	$\sqrt{(x - y)' V^{-1} (x - y)}$

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5, metric='minkowski')
'''
metric = 'euclidean'
        'manhattan'
        'chebyshev'
        'minkowski'
        'seuclidean'
        'mahalanobis'
'''

knn.fit(x_train_sc, y_train)

y_pred = knn.predict(x_test_sc)

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

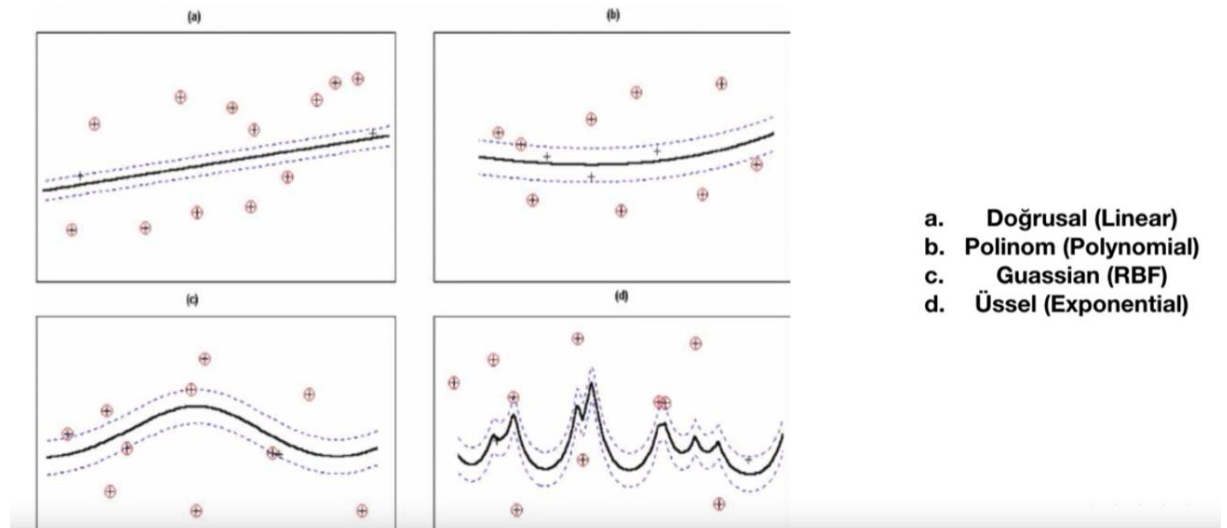
```
In [22]: runf
KNM(k_nearest
machine_learn
[[14 1]
 [ 2 11]]
C:\Users\terz
column-vector
example using
return self

In [23]: runf
KNM(k_nearest
machine_learn
[[14 1]
 [ 2 11]]

In [24]: runf
KNM(k_nearest
machine_learn
[[14 1]
 [ 2 11]]
```

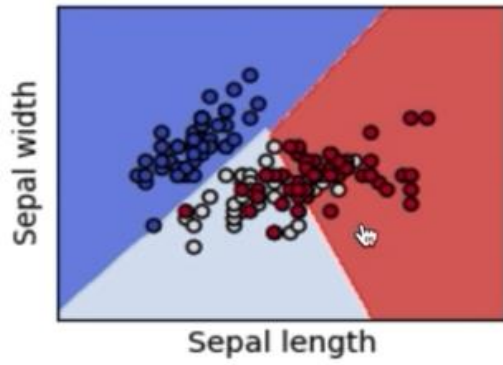
- Model, 14 örneği doğru bir şekilde negatif olarak sınıflandırmıştır (True Negatives - TN).
- Model, 1 örneği yanlış pozitif olarak sınıflandırmıştır (False Positives - FP).
- Model, 2 örneği yanlış negatif olarak sınıflandırmıştır (False Negatives - FN).
- Model, 11 örneği doğru bir şekilde pozitif olarak sınıflandırmıştır (True Positives - TP).

Destek Vektör Regresyonu (Support Vector Regression)

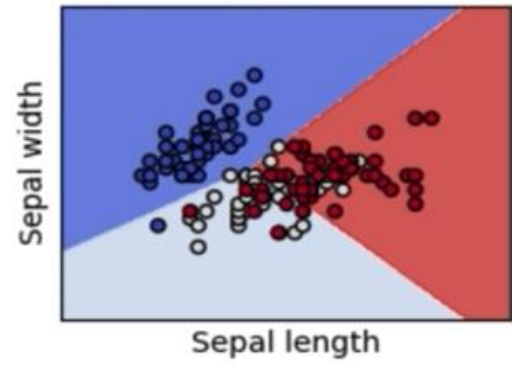


Support vector classifier

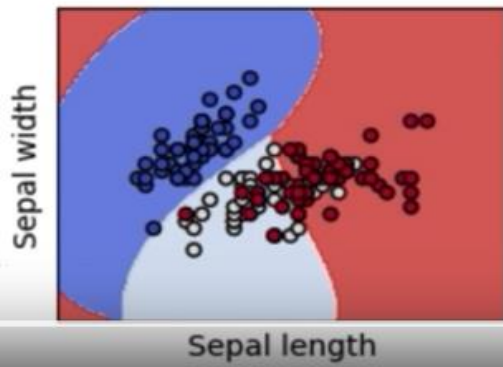
SVC with linear kernel



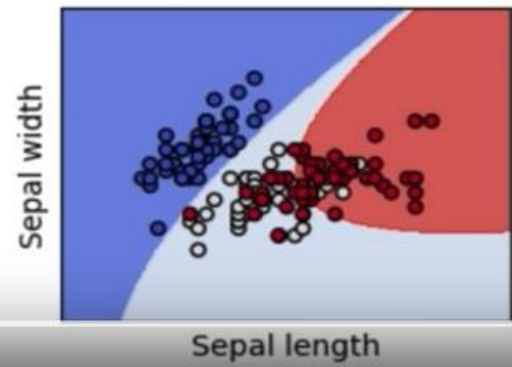
LinearSVC (linear kernel)



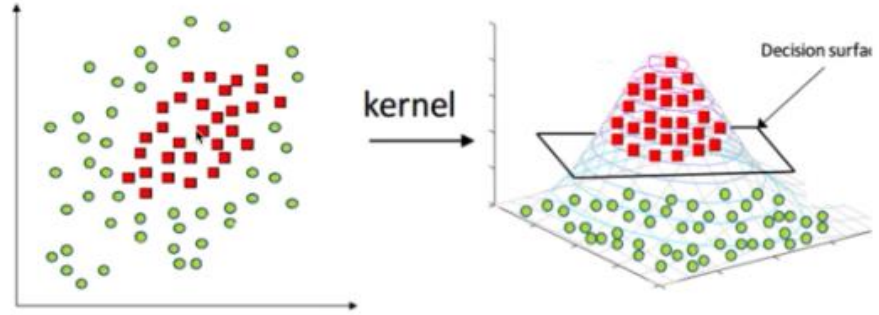
SVC with RBF kernel



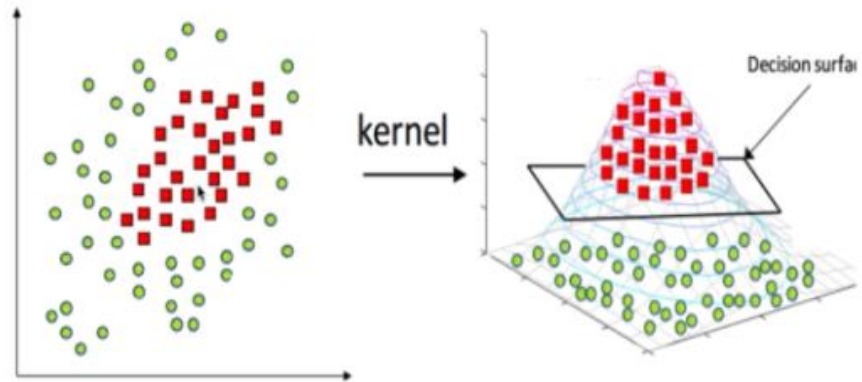
SVC with polynomial (degree 3) kernel



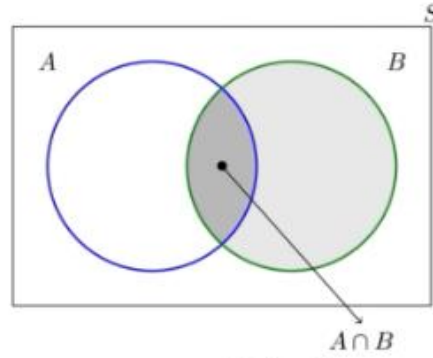
DVM Çekirdek Hilesi



SVM'NİN doğrusal ayrılma özelliğini 3.boyuta taşıyarak bir kernel trick(çekirdek hilesi) yaparak doğrusal olarak ayrıştırılamayacak verileri ayrıştırmayı hedefliyoruz



Naive Bayes (Koşullu Olasılık)



$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Naive Bayes (Naif Bayes)

$$P(C_i | \mathbf{X}) = \frac{P(\mathbf{X} | C_i) P(C_i)}{P(\mathbf{X})}$$

Sınıf:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Örnek Veri

X = (age ≤ 30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	com
≤30	high	no	fair	no
≤30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
≤30	medium	no	fair	no
≤30	low	yes	fair	yes
>40	medium	yes	fair	yes
≤30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

X'in gerçekleşmesinden sonra C'nin gerçekleşme ihtimali (veya tam tersi)

Örnek

- $P(C_i)$: $P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643$
 $P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357$

- Compute $P(X|C_i)$ for each class
 $P(\text{age} = \text{"<=30"} | \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$
 $P(\text{age} = \text{"<=30"} | \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$
 $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$
 $P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$
 $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 $P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$
 $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$
 $P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$

- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$

$$P(X|C_i) : P(X|\text{buys_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X|\text{buys_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i) \cdot P(C_i) : P(X|\text{buys_computer} = \text{"yes"}) \cdot P(\text{buys_computer} = \text{"yes"}) = 0.028$$

$$P(X|\text{buys_computer} = \text{"no"}) \cdot P(\text{buys_computer} = \text{"no"}) = 0.007$$

age	income	student	credit_rating	con
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

Naive Bayes bir sınıflandırma algoritması ancak sınıflandırma işlemi sırasında olasılıkları kullanıyor. Verilerin olasılıksal dağılımlarına bakarak bir sınıflandırma işlemi yapıyor.

Bütün dağılıma bakarak her bir bireyin(kolonun) dağılım üzerindeki etkisini ölçerek olasılıksal bir şekilde modelleniyor.

Naive bayes de KNN gibi lazy learning yönetimini kullanıyor.

Lazy learning (tembel öğrenme): Veri geldikten sonra elimizdeki veri setine göre gelen veriyi sınıflayacak. Yeni bir veri gelmeden herhangi bir işlem yapmaz. (veri geldikten sonra sormak)

Eager learning (istekli öğrenme): veri gelmeden önce elindeki veri kümesi üzerinden öğrenir (bütün ihtimalleri veri gelmeden hesaplamaları yapar) bütün olasılıkları bu tarafta bekletiyor olacak.

(veri kümesini unutulabilir) öğrendiği olasılıkları aklında tutarak gelen veriyi sınıflandırabilir

Dezavantaj: veri kümesi çok detaylı ve karmaşık olduğunda eager learning çalıştırmak maliyetlidir

Temel 3 yöntem ;

Gaussian Naive Bayes : tahmin edilecek sınıf sürekli bir değerse(ondalıklı reel) ise

Multinomial Naive Bayes : Örnek olarak: üniversite isimleri 1,2,3,4,5 gibi tam sayılar verip tahmin için kullanılabilir

Bernouilli Naive Bayes : Örnek olarak: kadın erkek , sigara içiyor içmiyor (binary nominal değerler)