Support Vector Machines (Destek Vetör Makineleri)

DSM 5007 Denetimli İstatistiksel Öğrenme 2. Ara Ödev

Fatih Emre Öztürk, Gökhan Acisulu, Veysel Karani Baris

DEÜ Fen Bilimleri Enstitüsü Veri Bilimi Yüksel Lisans Programı

2023-01-03

Destek Vektör Makinaları *(Support Vector Machines)*

Destek vektör makineleri hem sınıflandırma hem de regresyon analizlerinde kullanılabilmekle beraber en çok sınıflandırma işlemlerinde tercih edilmektedir.

Denetimli öğrenme modeline dayanmaktadır.

Çekirdek (Kernel) fonksiyonlar yardımıyla hem doğrusal hem de doğrusal olmayan sınıflandırmalar gerçekleştirebilmektedir.

Kullanım Alanları

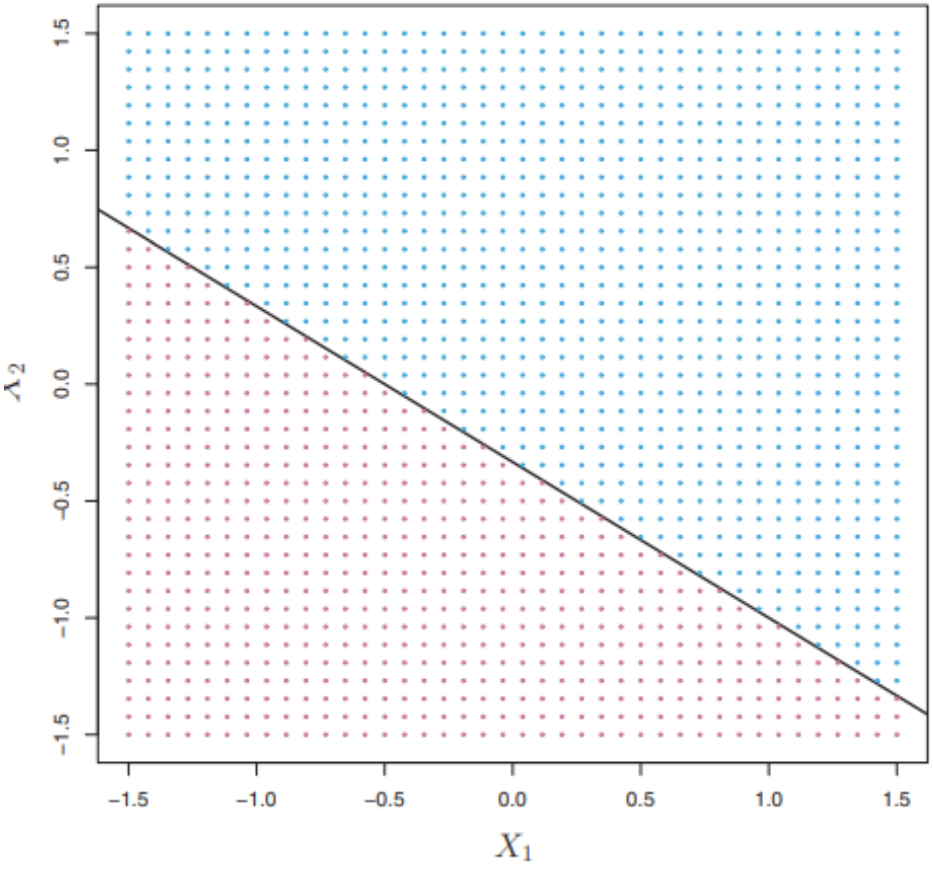
- Yüz Algılama *(Görüntünün bölümlerini yüz ve yüz olmayan bölümler şeklinde sınıflandırma)*
- Metin Sınıflandırma *(Haber makalelerinin iş ve filmler şeklinde sınıflandırılması)*
- Biyoinformatik *(Hastaların genlerine göre sınıflandırılması)*
- Jeoloji ve Çevre Bilimleri *(Uygu görüntüleri üzerinden haritalama uygulamaları)*

Support Vector Machines - *Kavramlar*

- Maximal margin classifier
- Support vector classifier
- Support vectors
- Support vector machine

Maximal Margin Classifier - *Hyperplane-1*

En kısa tanımla **Hyperplane** *p boyutlu* bir uzayın, *p-1 bir boyutlu* düz bir alt uzayıdır. Örneğin bizim uzayımız iki boyutlu ise hyperplane tek boyutlu düz bir (çizgi) yapısında olacaktır. Üç boyutlu bir uzayda ise Hyperplane iki boyutlu bir yapıda olaaktır.



Maximal Margin Classifier - *Hyperplane-2*

Hyperplane'nin denklemi;

- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_p X_p = 0$

Bir gözlem $X = (X_1, X_2, \ldots..X_p)$ değerleri yukarıdaki denklemde yerine konduğunda eğer eşitlik sağlanıyorsa (sonuç 0'a eşit oluyorsa) o gözlem hyperplane üzerinde yer alır.

Eğer X'in değerine göre aşağıdaki denklem **0'dan büyükse** bir düzlemde;

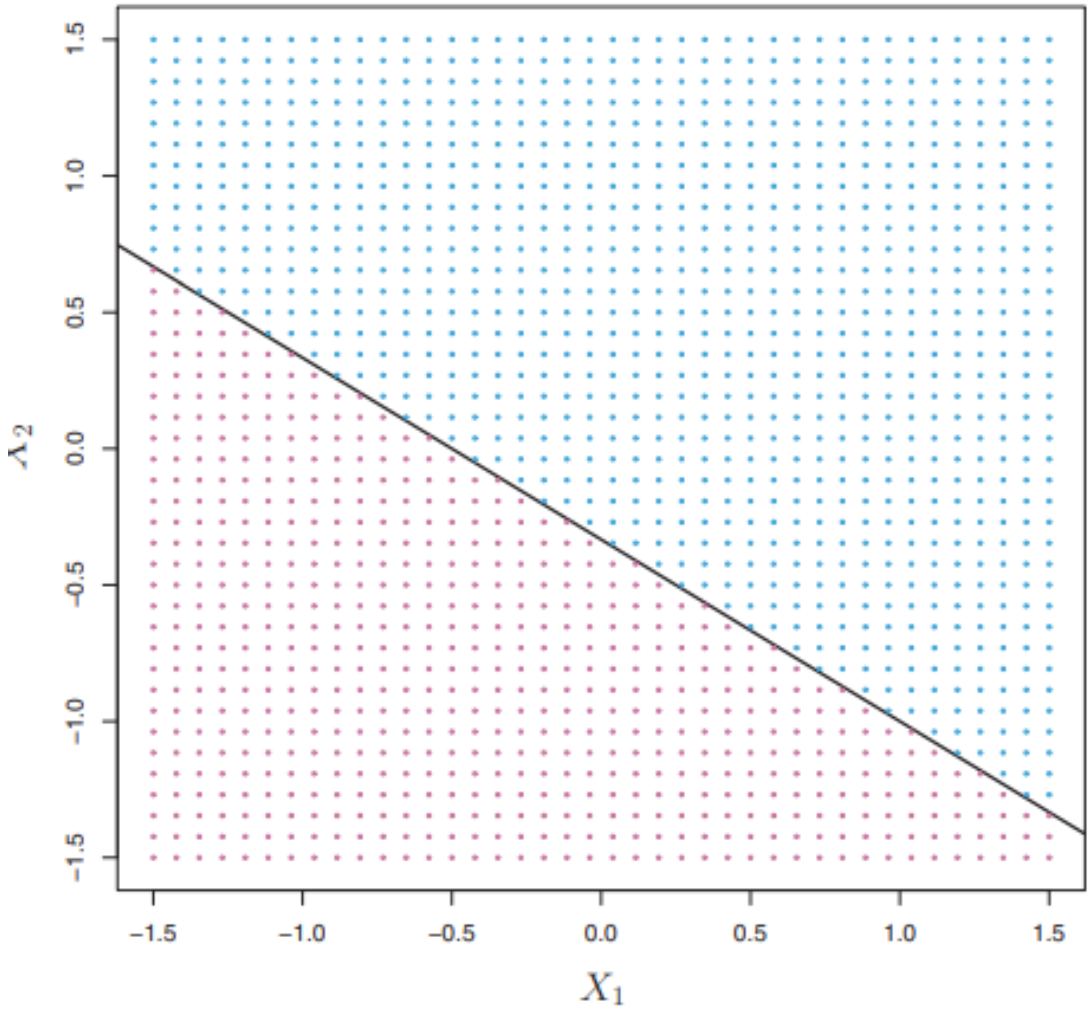
- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_p X_p > 0$

Eğer **0'dan küçükse** diğer düzlemde yer alır.

- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_p X_p < 0$

Maximal Margin Classifier - *Hyperplane-3*

Aşağıdaki şekilde yer alan hyperplane ait denklem $1 + 2X1 + 3X2 = 0$'dır. Şekilde **mavi bölge** $1 + 2X1 + 3X2 > 0$, **mor bölge** ise $1 + 2X1 + 3X2 < 0$ denklemini sağlayan noktaları göstermektedir.



Maximal Margin Classifier - *Hyperplane-4*

Mavi noktaların yi $= 1$ sınıfında, Mor noktaların ise yi $=$ -1 sınıfında yer aldığını varsayalım.

Buna göre y değerlerine göre denklemler aşağıdaki şekilde kurulabilir;

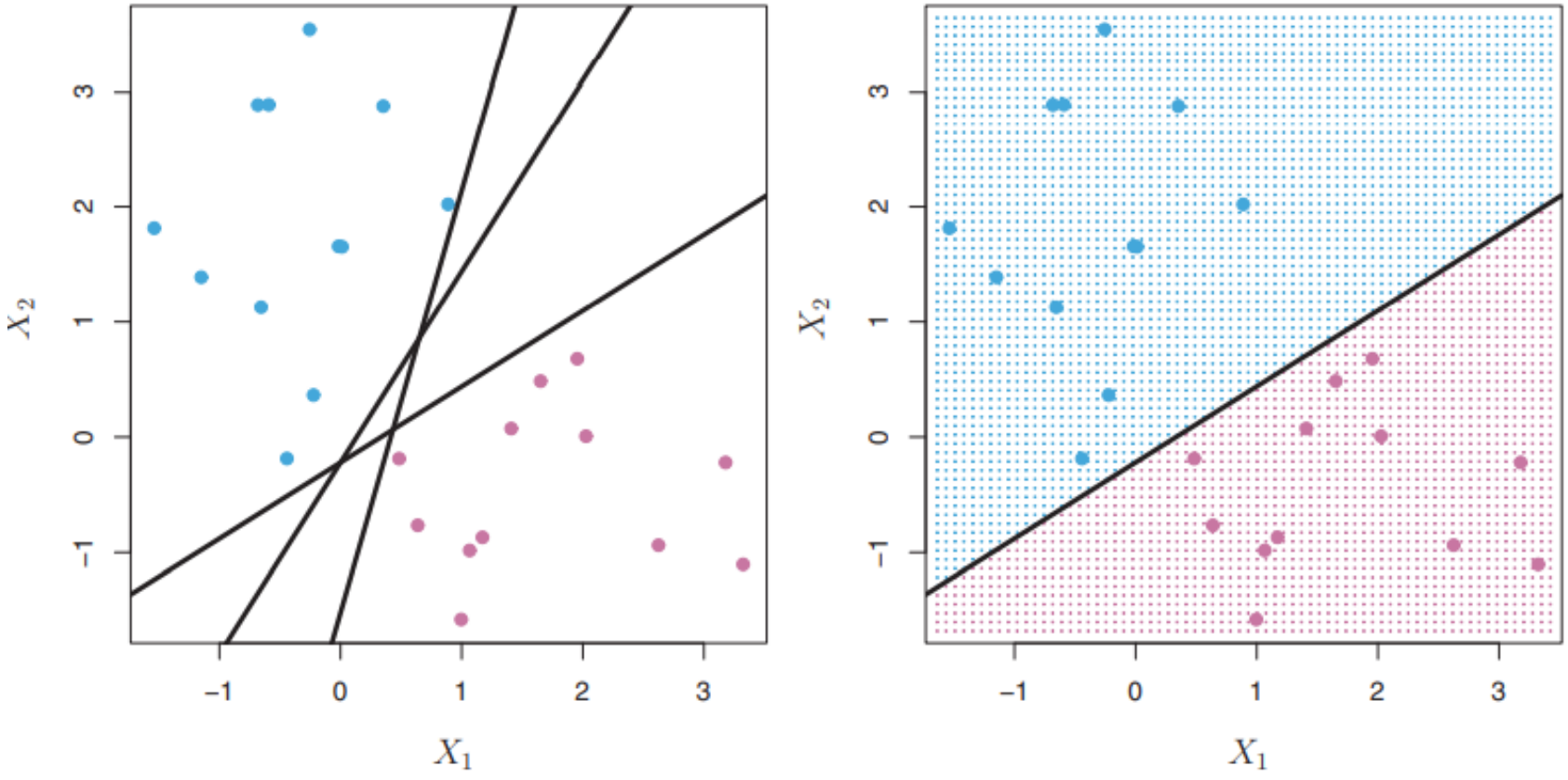
- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_p X_p > 0$ / Eğer yi $= 1$ ise
- $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_p X_p < 0$ / Eğer yi $=$ -1 ise

Yukarıdaki iki denklemi aşağıdaki şekilde ayırmak mümkün;

- $y_i(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_p X_p) > 0$

Maximal Margin Classifier

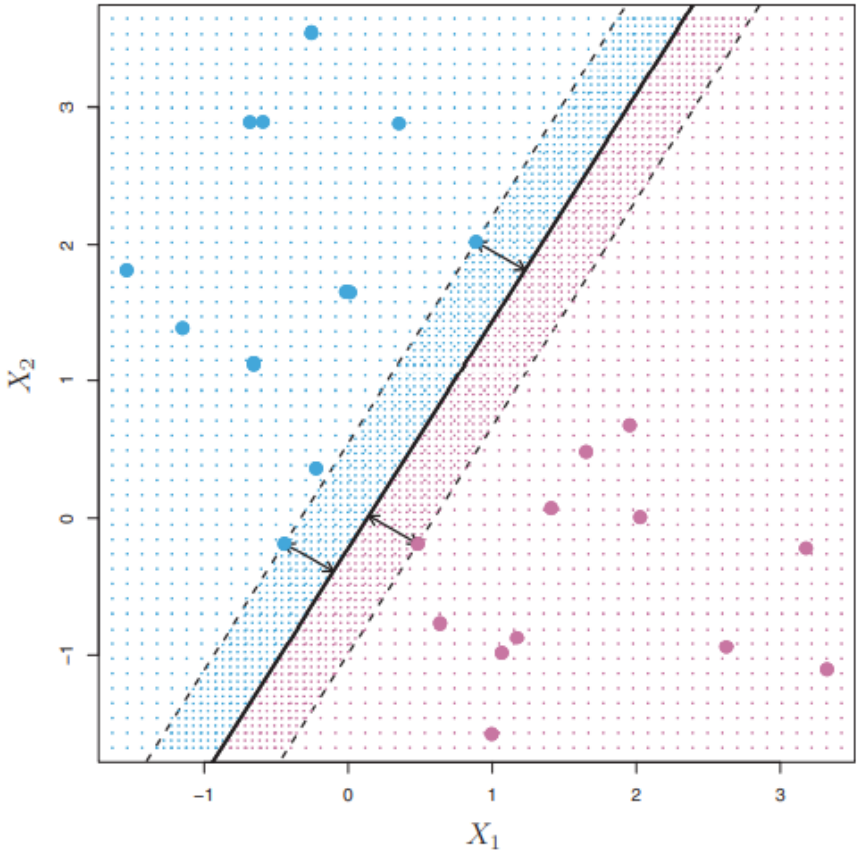
Verilerimiz eğer doğrusal bir hyperplane ile sınıflandırılmaya uygunsa birbirinden farklı sonsuz sayıda hyperplane belirleme ihtimalimiz olabilir.



Maximal Margin Classifier

Optimal Hyperplane Karar Verme;

- Farklı sınıflardaki gözlemlerden birbirine en yakın olan noktalardan, en uzak noktada olacak şekilde hyperplane belirlenir.

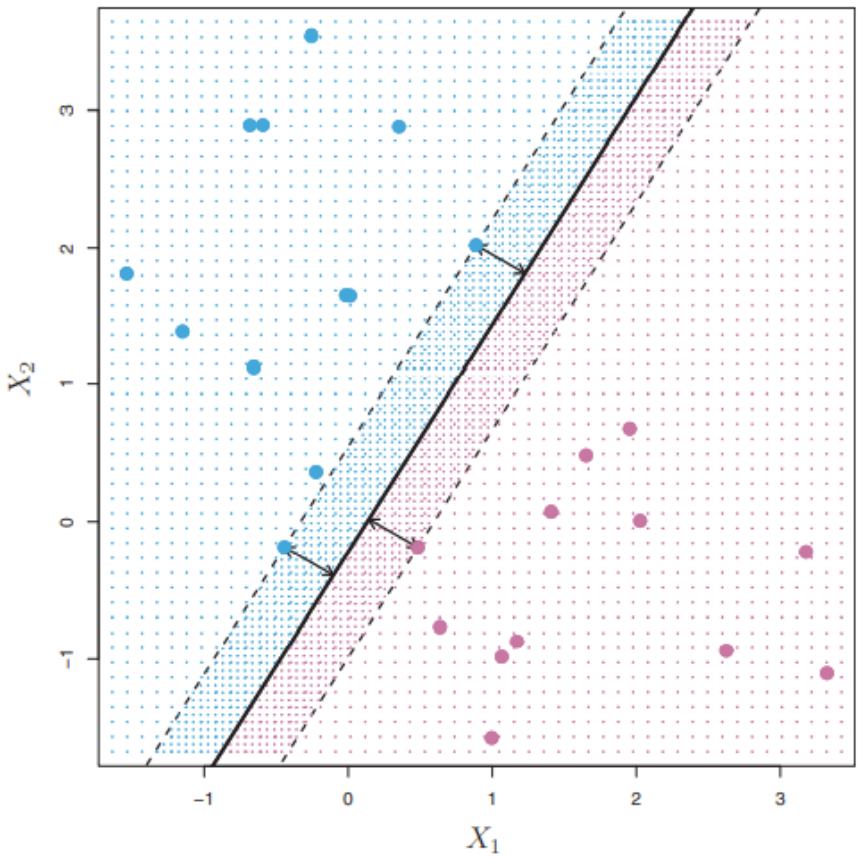


Maximal Margin Classifier

Eğer $\beta_0, \beta_1 \ldots \beta_p$ maximal margin hyperplane'nin katsayılarıysa test veri setinde yer alan x gözlemini
$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots .. + \beta_p X_p$ fonksiyonunun işaretine göre sınıflandırabiliriz. Bu durumda belirlediğimiz maximal margin hyperplane aynı zamanda **maximal margin classifier** olarak isimlendirilidir.

Support Vectors (Destek Vektörleri)

Aşağıdaki incelendiğinde 3 adet gözlemin maximal margin hyperplane düzleminin üzerinde uç kenar sınırlarında yer aldığı görülmektedir. Bu gözlemlere **support vector** denilmektedir



Support Vectors

Bu gözlemlere support vector denmesinin nedeni maximal marginal hyperplane'nin bu gözlemlere duyarlı olmasından *(bu gözlemler doğrultusunda oluşturulmuş olması)* kaynaklıdır. Eğer bu gözlemlerin konumları değişirse, maximal marginal hyperplanenin de konumu onlara bağlı olarak değişecektir.

Maximal Margin Classifier'ın Belirlenmesi

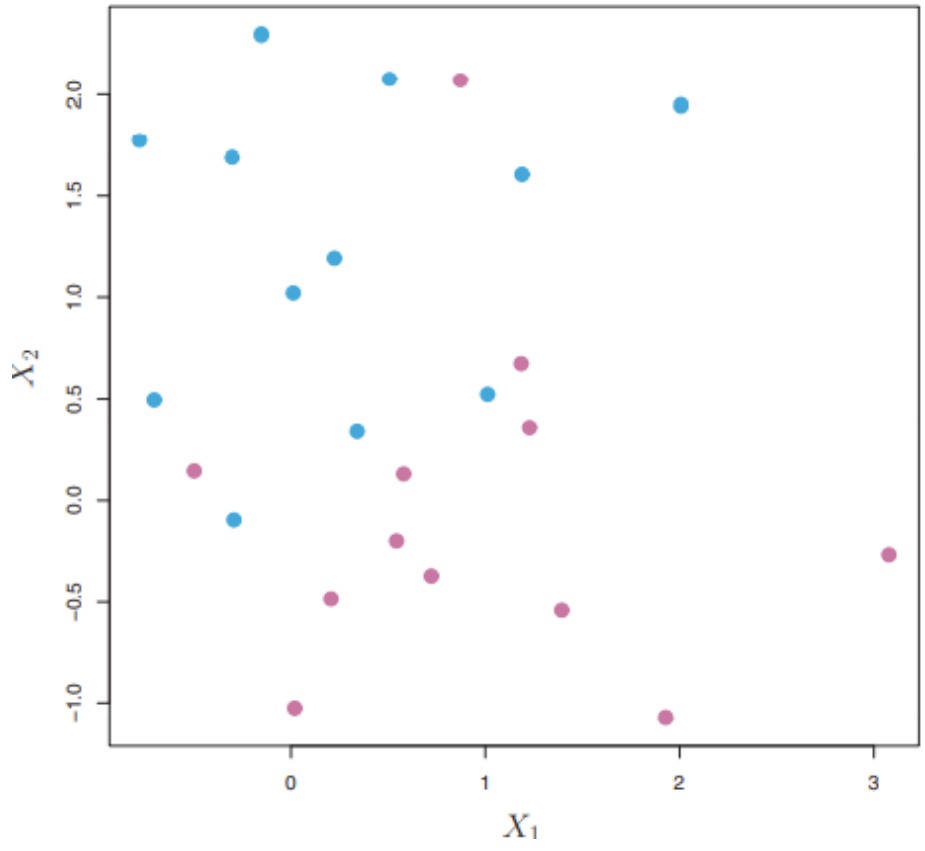
Maximal margin classifier'ın belirlenmesi bir optimizasyon problemidir.

- $y_i(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_p X_p) \geq M \ / \ \mathrm{i} = 1$

Yukarıdaki denklemde M margin'i ifade etmektedir. Biz mümkün olan en büyük margini elde etmek isteriz. Bu nedenle optimizasyon sürecinde de en büyük M'yi elde edebileceğimiz $\beta_0 + \beta_1 \ldots \beta_p$ parametrelerine karar verilir.

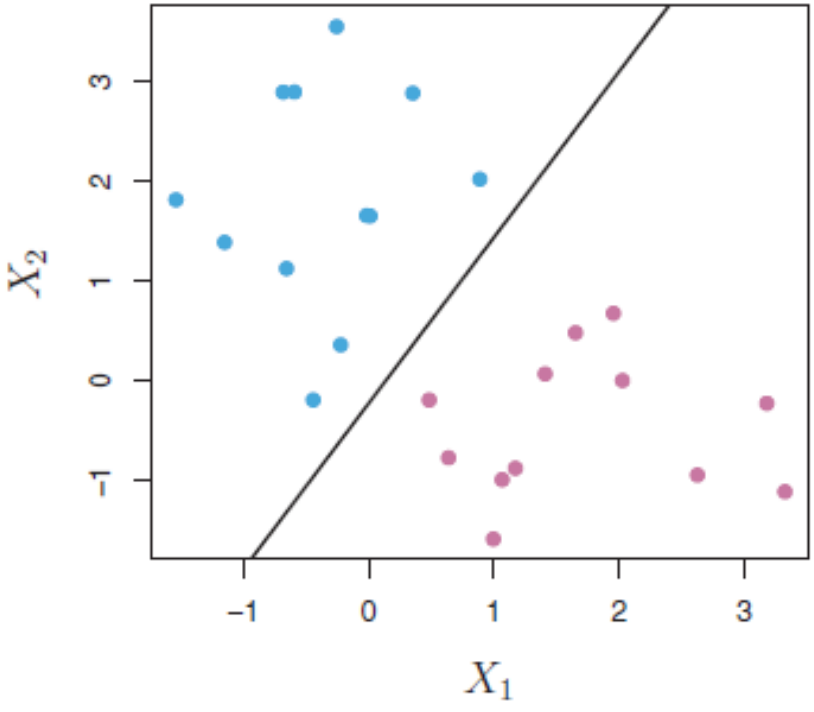
Support Vector Classifier

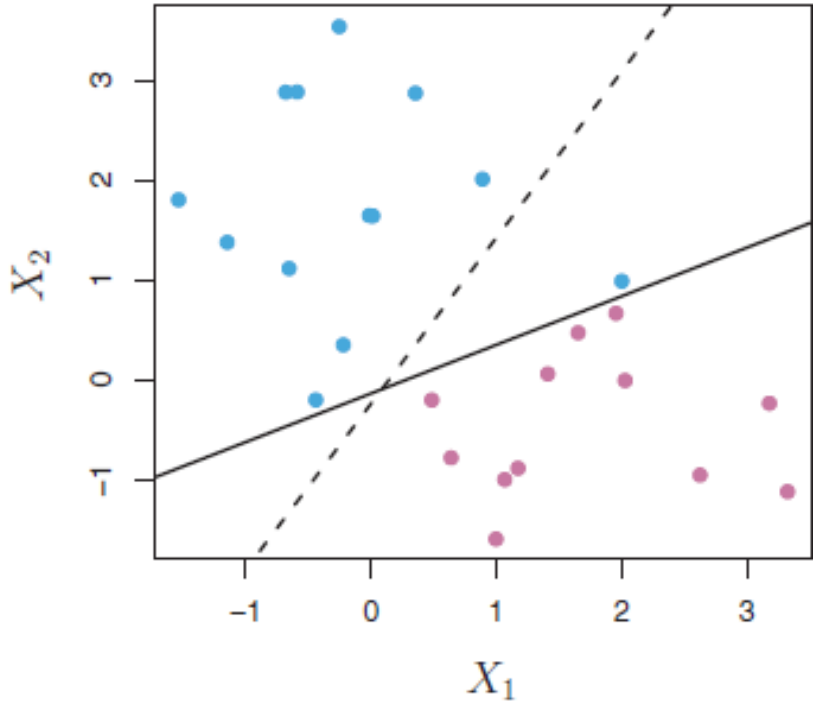
Doğrusal bir sınıflandırıcı ile sınıflandırılabilen veri setlerinde maximal margin classifier en iyi seçenektır. Ancak çoğu veri seti bu şekilde doğrusal bir sınıflandırıcı ile tam doğru bir şekilde sınıflandırılamayabilir.



Support Vector Classifier

Veriler mükemmel şekilde sınıflandırılmış olsa bile maximal margin classifier support vector'lere aşırı duyarlı olduğu için margin içerisine yeni bir gözlem eklenmesi durumunda hyperplane'de buna göre tekrar değişim gösterecektir.





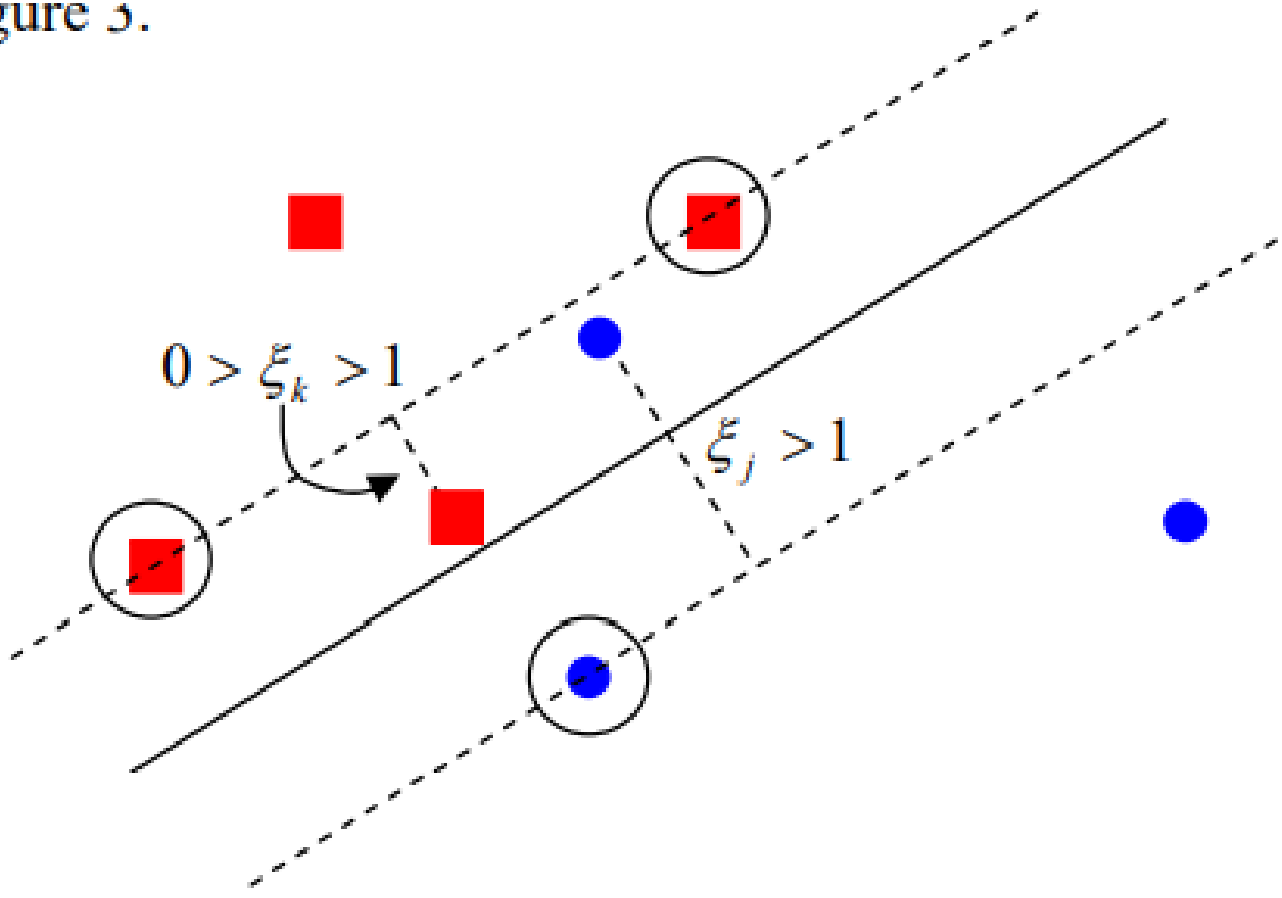
Support Vector Classifier

Yaşanabilecek bu gibi sorunlar nedeniyle gerçek yaşam verilerinde gözlemleri tam doğru bir şekilde sınıflandırmak yerine;

- Bireysel gözlemlere karşı daha dayanılık (robust),
- Gözlemleri mümkün olan en doğru şekilde sınıflandırmayı hedeflemeliyiz.

Soft margin classifier olarak da isimlendirilebilen **support vector classsifer** ta olarak bu işi yapmaktadır.

Support Vector Classifier



Support Vector Classifier

Support vector classifier'da da ne kadar yanlış gözleme izin verileceği optimazasyon ile belirlenmektedir.

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

Yukarıdaki optimizasyon problemi çalıştırıldığında;

- eğer slack variable $\epsilon = 0$ değerini alırsa gözlemin hyperplanenin doğru tarafında sınıflandırıldığı,
- eğer $\epsilon > 0$ değerini alırsa gözlemin marginin yanlış tarafında sınıflandırılığı,
- eğer $\epsilon > 1$ değerini alırsa gözlemin hyperplanenin yanlış tarafında sınıflandırıldığını gösterir.

Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i$$

$$\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C,$$

C parametresi ϵ slack variable'ların toplamını sınırlandıran bir tunning parametresidir. Bu nedenle C tolere edilebilecek olan yanlış sınıflandırmaya izin vermektedir.

Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

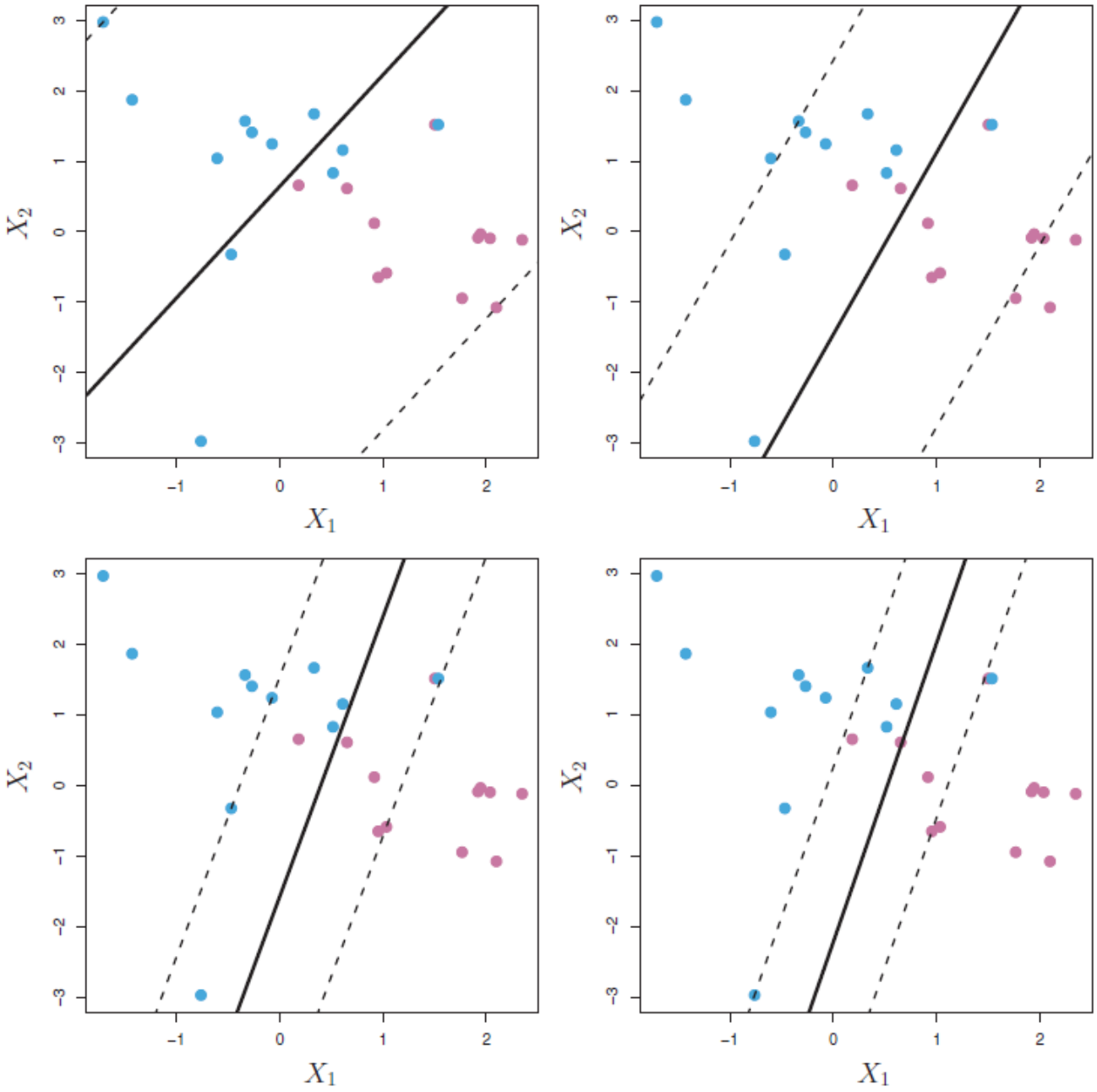
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i$$

$$\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C,$$

- Eğer c = 0 olarak belirlenirse, buna bağlı olarak tüm ϵ slack variable'larda 0 değerini alacak ve artık basit bir maximal margin classifier problemine geçilmiş olunacaktur.
- C değeri ne kadar büyük olursa gözlemlerin yanlış sınıflandırımasına o kadar fazla izin vermiş olacağız. Dolayısıya marginimiz daha geniş olacak.
- C parametresi ne kadar küçük olursa yanlış sınıflandırmalara o kadar az izin vermiş olacağız, dolayısıyla marginimiz da o kadar dar olacaktr.

Support Vector Classifier - *'C' Parametresi*



Lineer Sınıflandırılamayan Veriler *(Support Vector Machine)*

Support Vector Machine, support vector'un genişletilmiş halidir.

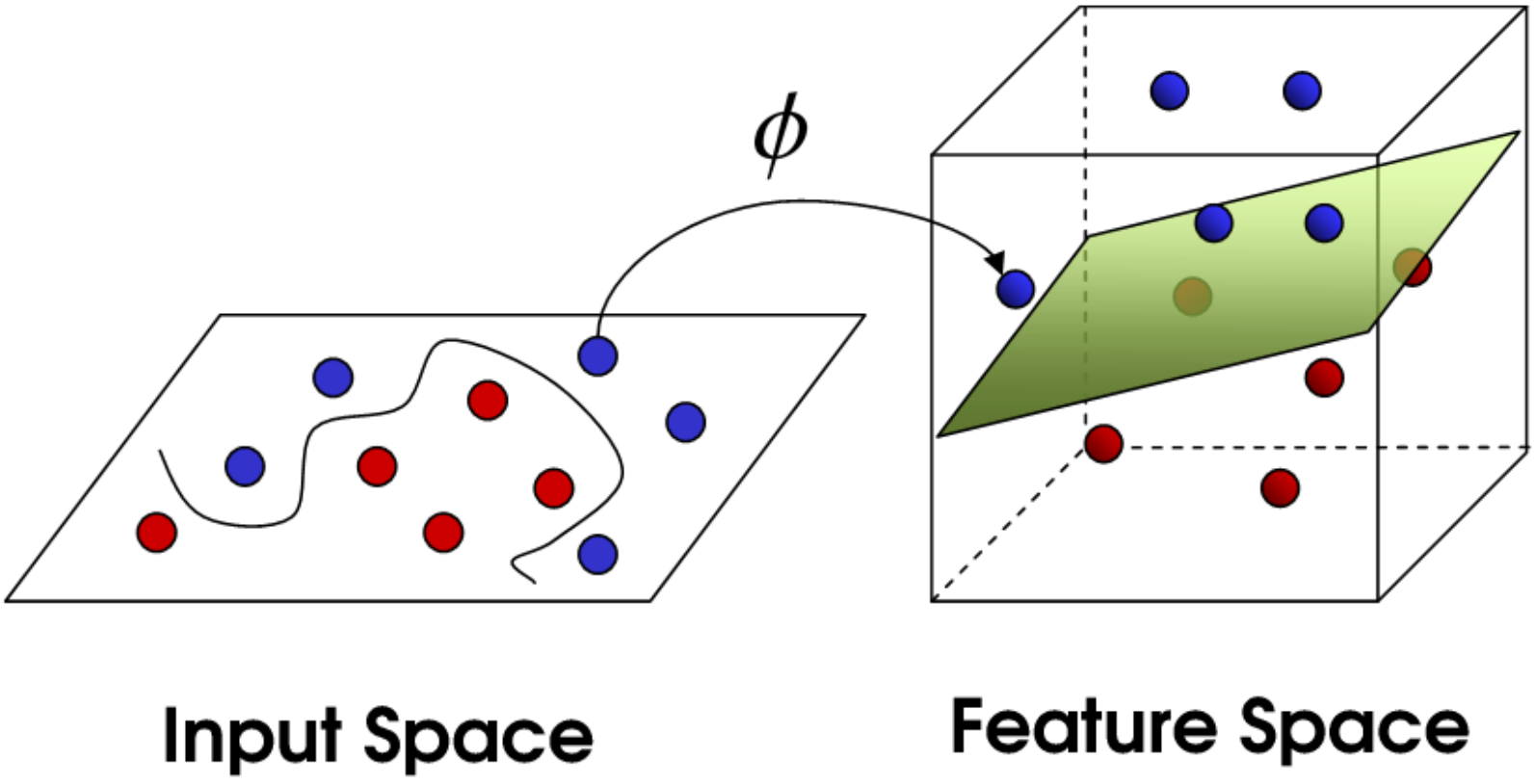
SVM değişken uzayının genişlemesini sağlayıp Kernel'lar kullanarak orijinal uzayda lineer olmayan sınıflandırmalara olanak sağlar.

Support Vector Machine

Support vectoru doğrusal olmayan karar yüzeyine/düzlemine genişletmek için ilk olarak; veriler doğrusal olmayan bir dönüşüm olan $\phi(x)$ aracılığıyla başka bir vektör uzayına eşlenir.

Elde edilen bu yeni vektör uzayına özellik uzayı denir ve genellikle boyutluluğu orijinal girdi uzayından çok daha yüksektir (2 yerine 3 boyutlu vb.). Bu da verilerin daha doğru ve kolay bir biçimde sınıflandırılmasını sağlar.

Support Vector Machine



Support Vector Machine

Dönüşüm sonrasında yeni vektör uzayında SVM algoritması kullanılarak maxima margin hyperplane veya soft margin hyperplane belirlenmeye çalışılır.

$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to } y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1-\epsilon_i),$$

$$\sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1.$$

Support Vector Machine

Aslında buradaki optimizasyon problemi çözüldüğünde gözlemlerin gerçek değerlerinin değil iç çarpımlarının (inner product) modele dahil olduğu belirlenir. Yukarıdaki optimizasyon problemi çözüldüğünde elde edilen lineer svm fonksiyonu aşağıdaki gibidir.

- $f(x) = sign(\sum_{ij=1}^{M} a_i y_i (x_i x_j) + b)$

Ancak dönüşüm yapıldığı için fonksiyon aşağıdaki gibi ifade edilecektir;

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i,j=1}^{M} \alpha_i y_i (\mathbf{\Phi}^T(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)) + b\right).$$

Support Vector Machine

- Yüksek boyutlu özellik uzayında çalışmak karmaşık fonksiyonların ifade edilmesini sağlar, ancak özellikle geniş vektörler hesaplama sorunlarına yol açarken, yüksek boylutluluk da overfitting sorununa neden olabilir.
- Yüksek boyutluluğun neden olduğu bu sorunlara yönelik **Kernel fonksiyonu** kullanılabilir.
- Kernel, orijinal veri noktalarının özellik uzayı eşlemelerinin **nokta çarpımını döndüren** bir fonksiyondur.
- Bir kernel fonksiyonu uygulandığında, özellik uzayında öğrenme ϕ'nin açık bir şekilde değerlendirilmesini gerektirmez. Böylelikle biz yukarıdaki formülü uygulayarak dönüşüm yapmadan maximal ya da soft margin hyperplane rahatlıkla elde edebilir.

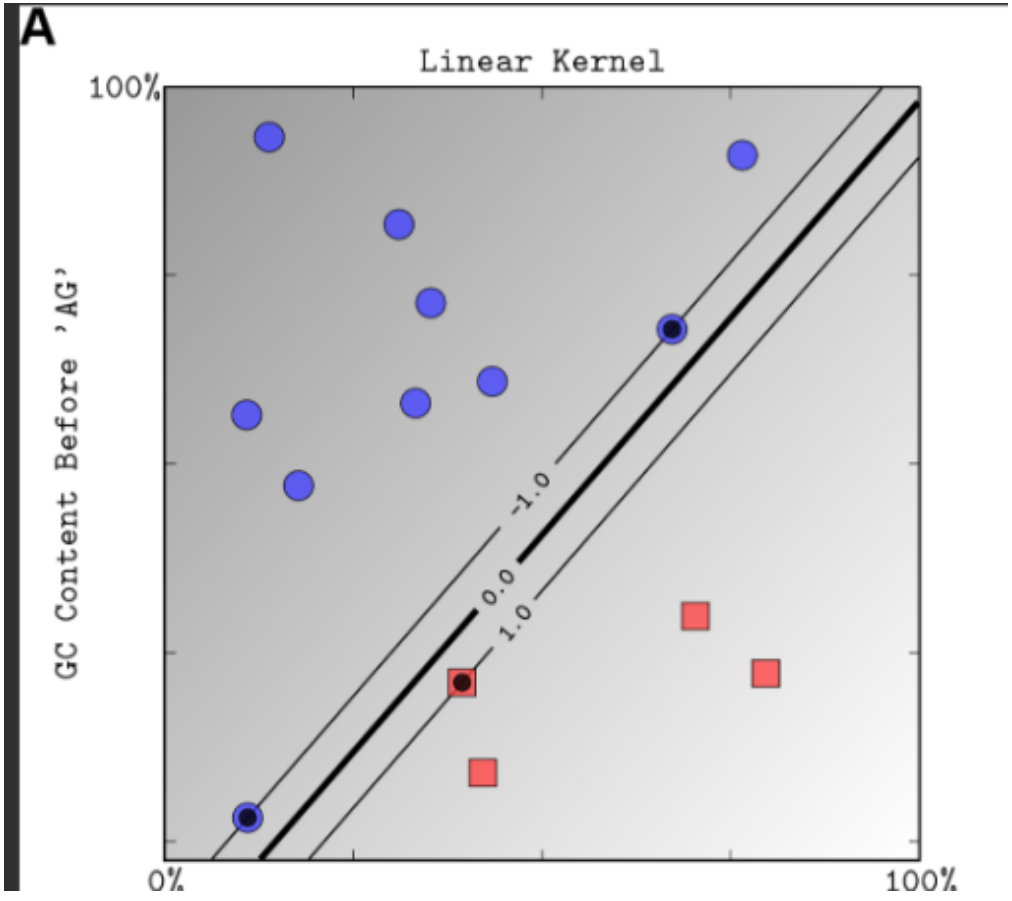
Support Vector Machine

Kernel kullandığında fonksiyon aşağıdaki şekilde ifade edilir.

- $f(x) = sign(\sum_{ij=1}^{M} a_i y_i K(x_i, x_j) + b)$

Kernel Trick Türleri

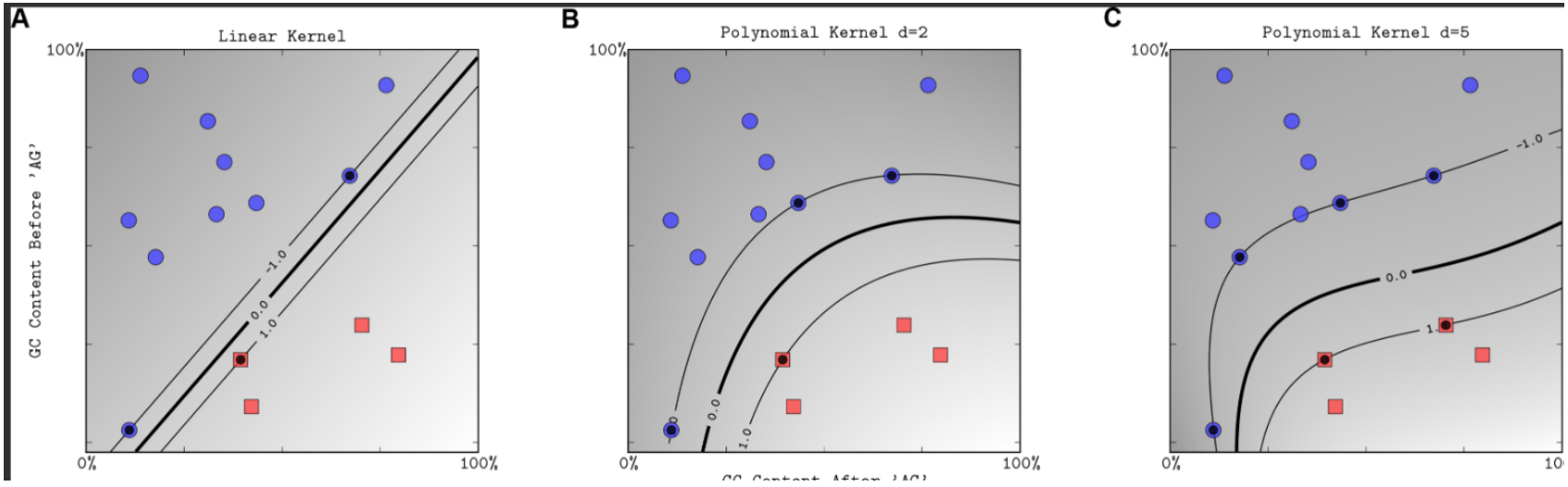
- **1. Lineer Kernel** = Doğrusal bir ayırım sağlar. Bu nedenle aslında lineer kernel bize support vector classifier'ı vermektedir.
  - $K(X_i, X_i') = \sum_{j=1}^{p} X_{ij}.X_{i'j}$



Kernel Trick Türleri

- **2. Polynomial Kernel** = Lineer kernel yanı sıra daha esnek bir hyperplane oluşturur. Orjinal özellik uzayı yerine daha yüksek boyutlu bir örneklem uzayında bir destek vektör sınıflandırıcısının elde edilmesi anlamına gelir.
  - $K(X_i, X_i') = (1 + \sum_{j=1}^{p} X_{ij}.X_{i'j})^d$

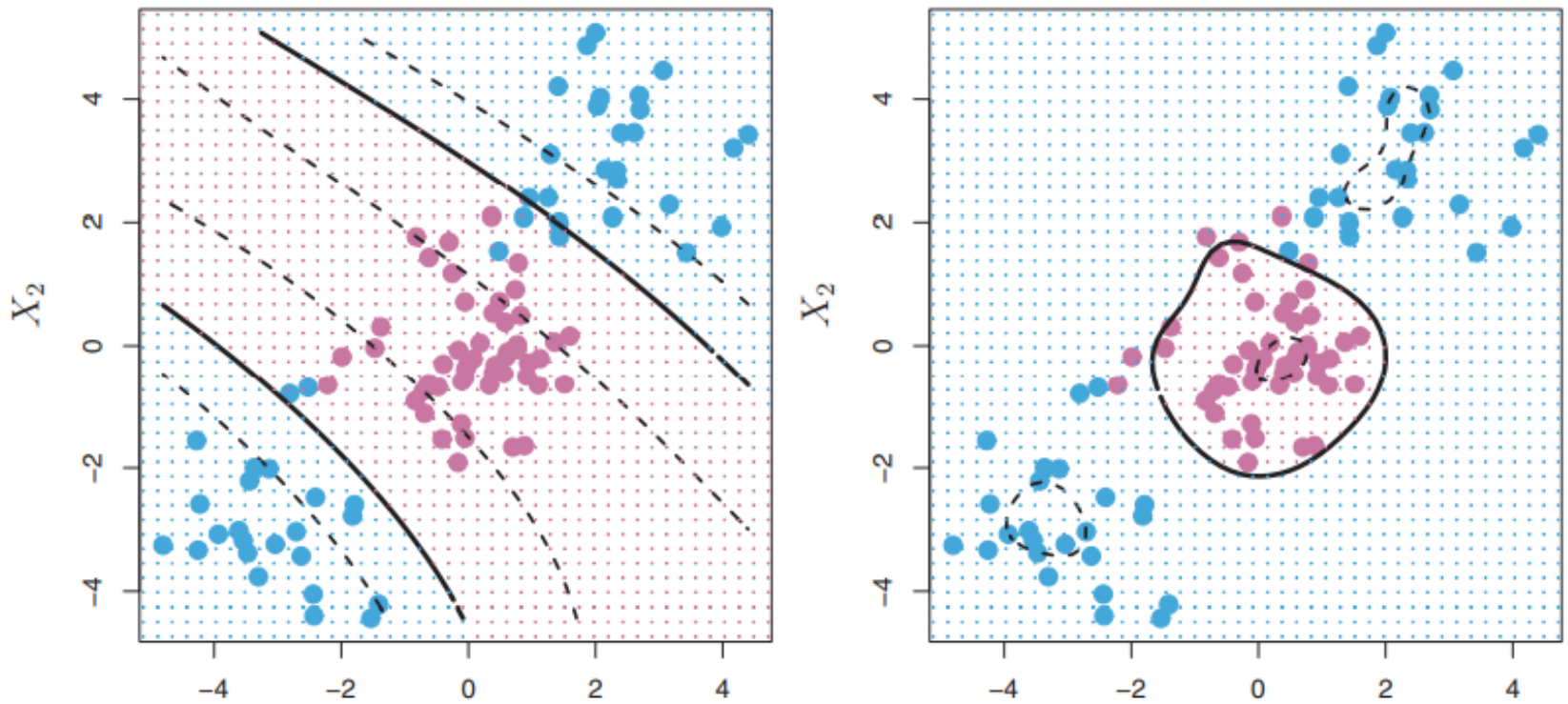
Yukarıdaki formülde d her zaman pozitif bir değer alır ve d'nin artırılması hyperplane'nin esnekliğini ayarlar.



Kernel Trick Türleri

- **3. Radial Kernel** = Lineer kernel yanı sıra daha esnek bir hyperplane oluşturur. Orjinal özellik uzayı yerine daha yüksek boyutlu bir örneklem uzayında bir destek vektör sınıflandırıcısının elde edilmesi anlamına gelir.
  - $K(X_i, X_i') = exp(-\gamma \sum_{j=1}^{p}(X_{ij}.X_{i'j})^2)$

Yukarıdaki formülde γ her zaman pozitif bir değer alır ve γ'nın artırılması hyperplane'nin esnekliğini ayarlar.



Support Vector Machine *Uygulama*

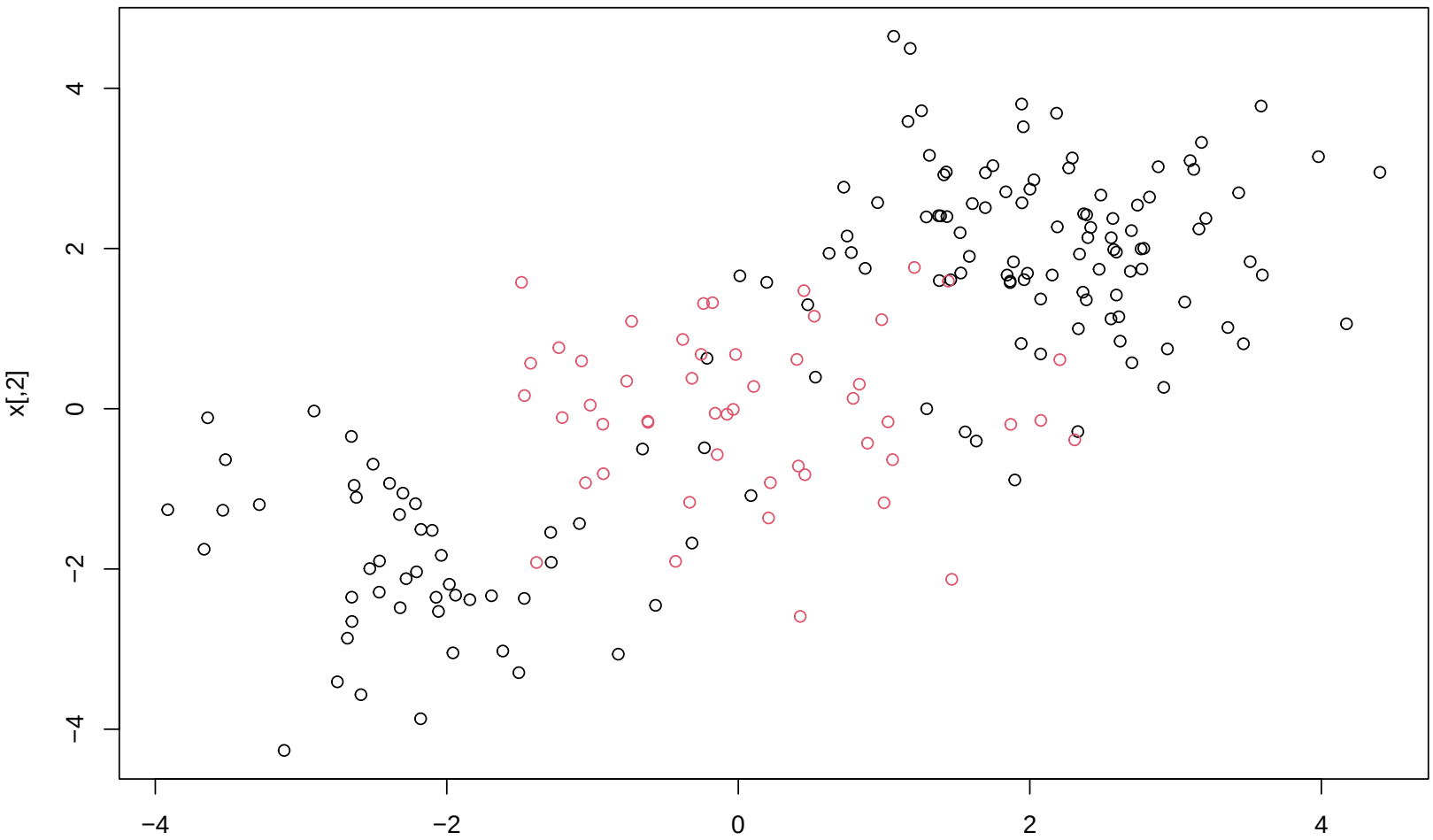
Nonlinear veri setinin oluşturulması

```
set.seed(1)
x=matrix(rnorm (200*2) , ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150 ,]=x[101:150,]-2
y=c(rep(1,150) ,rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
```

Support Vector Machine *Uygulama*

Nonlinear veri setinin oluşturulması

```
plot(x, col=y)
```



Support Vector Machine *Uygulama*

Veri setinin test ve train olarak ikiye ayrılması

```
set.seed(123)
smp_size <- floor(0.75 * nrow(dat))
train_ind <- sample(nrow(dat), size = smp_size, replace = FALSE)
train_dnm_ders <- dat[train_ind, ]
test_dnm_ders <- dat[-train_ind, ]

table(train_dnm_ders$y)
```

```
##
##   1   2
## 108  42
```

Support Vector Machine *Uygulama*

SVM fonksiyonu **e1071** kütüphenesinde yer almaktadır.

```
library("e1071")
```

```
svm_radial <- e1071::svm(train_dnm_ders$y ~.,
                         data = train_dnm_ders,
                         kernel = "radial",
                         gamma = 1,
                         cost = 1)
```

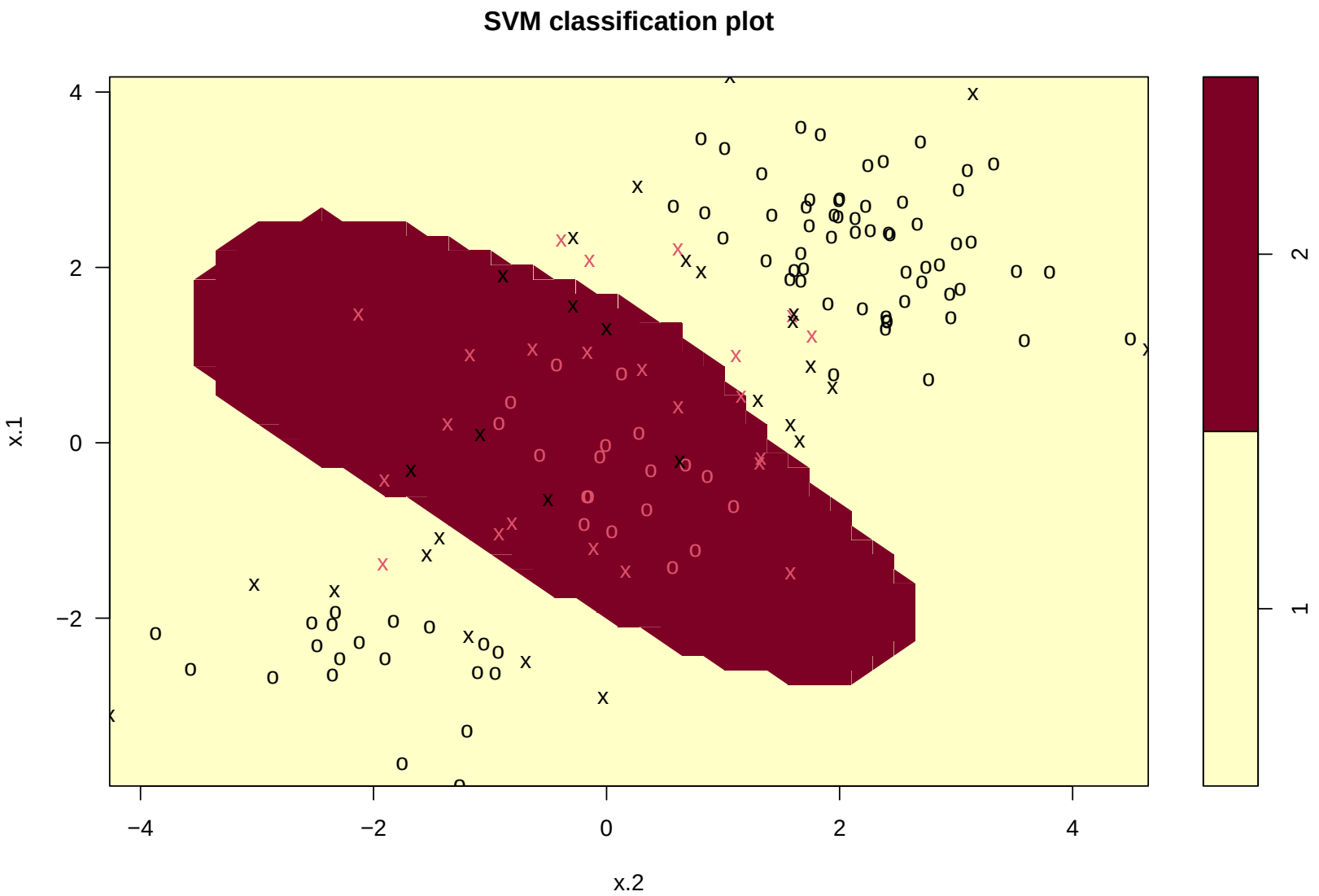
Support Vector Machine *Uygulama*

```
summary(svm_radial)
```

```
##
## Call:
## svm(formula = train_dnm_ders$y ~ ., data = train_dnm_ders, kernel = "radial"
##     gamma = 1, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  52
##
##  ( 23 29 )
##
##
## Number of Classes:  2
##
## Levels:
##  1 2
```

Support Vector Machine *Uygulama*

```
plot(svm_radial , train_dnm_ders)
```



Support Vector Machine *Uygulama*

C ve γ parametrellerinin Cross-Validation İle Belirlenmesi

```
set.seed(123)
tune.out=e1071::tune(e1071::svm ,
                     y~.,
                     data=train_dnm_ders,
                     kernel ="radial",
                     ranges=list(cost=c(0.1,1,10,100,1000),
                                 gamma=c(0.5,1,2,3,4) ))
```

Support Vector Machine *Uygulama*

```
summary (tune.out)
```

```
##
## Parameter tuning of 'e1071::svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     1
##
## - best performance: 0.1133333
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.1666667 0.12272623
## 2  1e+00   0.5 0.1200000 0.04216370
## 3  1e+01   0.5 0.1333333 0.05443311
## 4  1e+02   0.5 0.1533333 0.06324555
## 5  1e+03   0.5 0.1666667 0.05665577
## 6  1e-01   1.0 0.1200000 0.06885304
## 7  1e+00   1.0 0.1133333 0.03220306
## 8  1e+01   1.0 0.1666667 0.06478835
## 9  1e+02   1.0 0.1600000 0.06440612
## 10 1e+03   1.0 0.1533333 0.07062333
## 11 1e-01   2.0 0.1666667 0.10061539
```

Support Vector Machine *Uygulama*

```
bestparamet <-  tune.out$best.parameters
bestparamet
```

```
##   cost gamma
## 7    1     1
```

```
bestmodel <- tune.out$best.model
bestmodel
```

```
##
## Call:
## best.tune(METHOD = e1071::svm, train.x = y ~ ., data = train_dnm_ders,
##     ranges = list(cost = c(0.1, 1, 10, 100, 1000), gamma = c(0.5,
##         1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  52
```

Support Vector Machine *Uygulama*

Train Prediction

```
table(train_dnm_ders$y)
```

```
##
##   1   2
## 108  42
```

```
pred_radial <- stats::predict(bestmodel, train_dnm_ders)
table(predict = pred_radial, train_dnm_ders$y)
```

```
##
## predict   1   2
##       1 100   7
##       2   8  35
```

Support Vector Machine *Uygulama*

Train Prediction

```
caret::confusionMatrix(train_dnm_ders$y, pred_radial)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 100   8
##          2   7  35
##
##                Accuracy : 0.9
##                  95% CI : (0.8404, 0.9429)
##     No Information Rate : 0.7133
##     P-Value [Acc > NIR] : 2.517e-08
##
##                   Kappa : 0.7538
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9346
##             Specificity : 0.8140
##          Pos Pred Value : 0.9259
##          Neg Pred Value : 0.8333
##              Prevalence : 0.7133
##          Detection Rate : 0.6667
```

Support Vector Machine *Uygulama*

Test Prediction

```
table(test_dnm_ders$y)
```

```
##
##  1  2
## 42  8
```

```
pred_radial_test <- stats::predict(bestmodel, test_dnm_ders
table(predict = pred_radial_test, test_dnm_ders$y)
```

```
##
## predict  1  2
##       1 39  2
##       2  3  6
```

Support Vector Machine *Uygulama*

Test Prediction

```
caret::confusionMatrix(test_dnm_ders$y, pred_radial_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 39  3
##          2  2  6
##
##                Accuracy : 0.9
##                  95% CI : (0.7819, 0.9667)
##     No Information Rate : 0.82
##     P-Value [Acc > NIR] : 0.09286
##
##                   Kappa : 0.6459
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.9512
##             Specificity : 0.6667
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.7500
##              Prevalence : 0.8200
##          Detection Rate : 0.7800
```

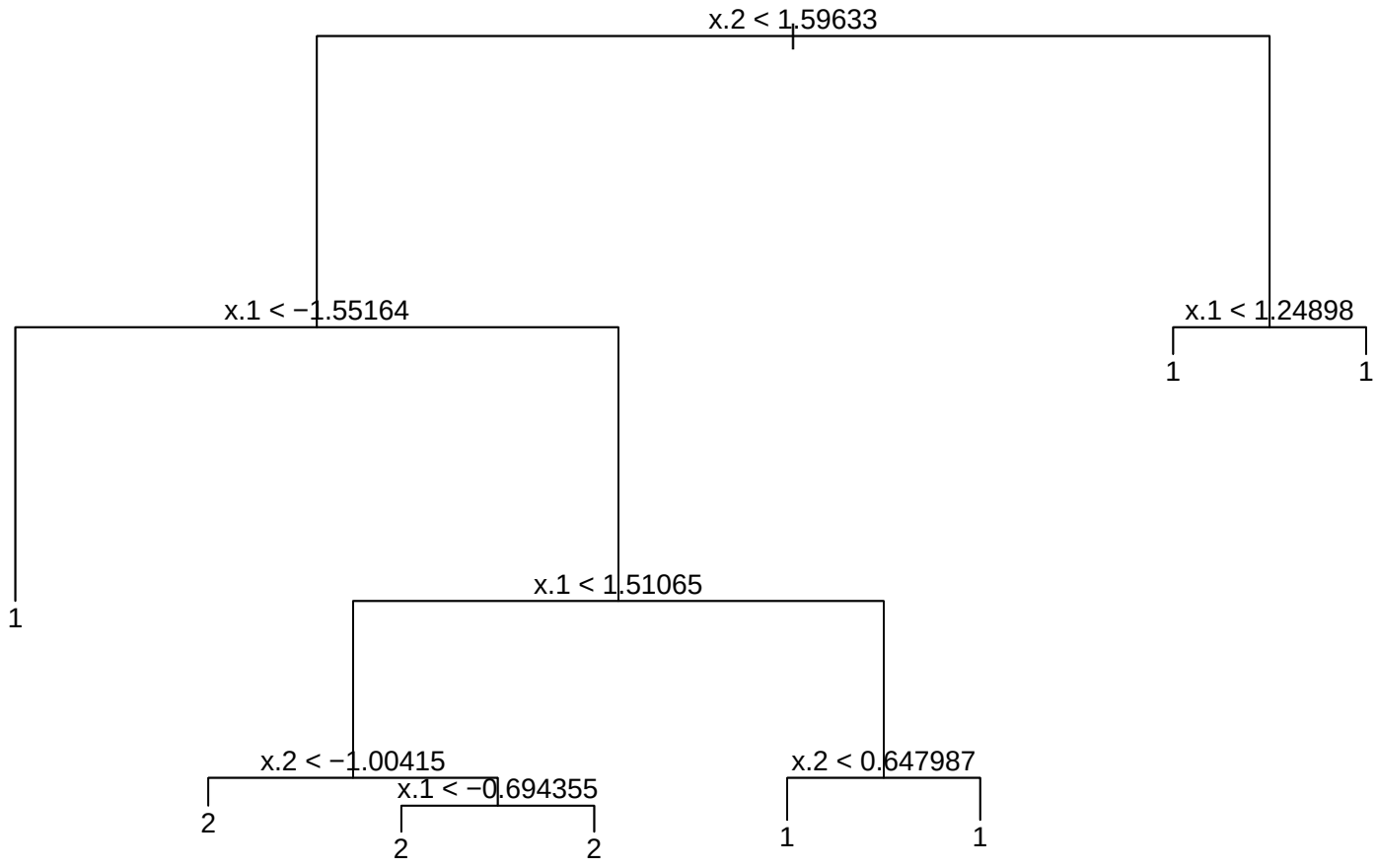
Sınıflandırma Ağaçları *Uygulama*

```
tree_ders <- tree::tree(train_dnm_ders$y~., train_dnm_ders)
summary(tree_ders)
```

```
##
## Classification tree:
## tree::tree(formula = train_dnm_ders$y ~ ., data = train_dnm_d
## Number of terminal nodes:  8
## Residual mean deviance:  0.3881 = 55.1 / 142
## Misclassification error rate: 0.08667 = 13 / 150
```

Sınıflandırma Ağaçları *Uygulama*

```
plot(tree_ders)
text(tree_ders, pretty = 0)
```



Sınıflandırma Ağaçları *Uygulama*

```
tree_ders_pred <- predict(tree_ders, train_dnm_ders,
                          type = "class")
table(tree_ders_pred, train_dnm_ders$y)
```

```
##
## tree_ders_pred  1  2
##              1 99  4
##              2  9 38
```

Sınıflandırma Ağaçları *Uygulama*

```
caret::confusionMatrix(train_dnm_ders$y, tree_ders_pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 99  9
##          2  4 38
##
##                Accuracy : 0.9133
##                  95% CI : (0.8564, 0.953)
##     No Information Rate : 0.6867
##     P-Value [Acc > NIR] : 2.773e-11
##
##                   Kappa : 0.7926
##
##  Mcnemar's Test P-Value : 0.2673
##
##             Sensitivity : 0.9612
##             Specificity : 0.8085
##          Pos Pred Value : 0.9167
##          Neg Pred Value : 0.9048
##              Prevalence : 0.6867
##          Detection Rate : 0.6600
##    Detection Prevalence : 0.7200
##       Balanced Accuracy : 0.8848
```

Sınıflandırma Ağaçları *Uygulama*

Cross-Validation

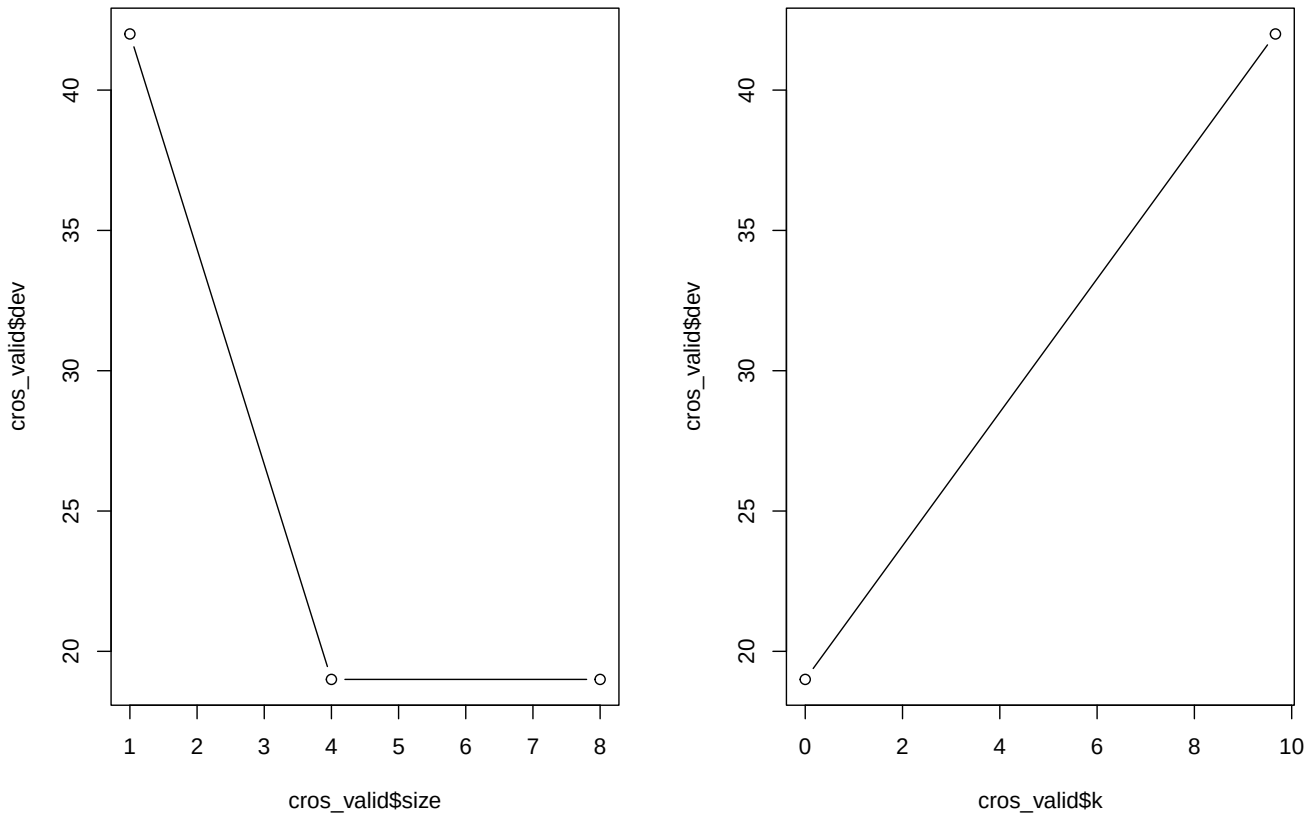
```
set.seed(123)
cros_valid <- tree::cv.tree(tree_ders, FUN = prune.misclass)
cros_valid
```

```
## $size
## [1] 8 4 1
##
## $dev
## [1] 19 19 42
##
## $k
## [1]      -Inf 0.000000 9.666667
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Sınıflandırma Ağaçları *Uygulama*

Cross-Validation

```
par(mfrow=c(1,2))
plot(cros_valid$size, cros_valid$dev, type = "b")
plot(cros_valid$k, cros_valid$dev, type = "b")
```



Sınıflandırma Ağaçları *Uygulama*

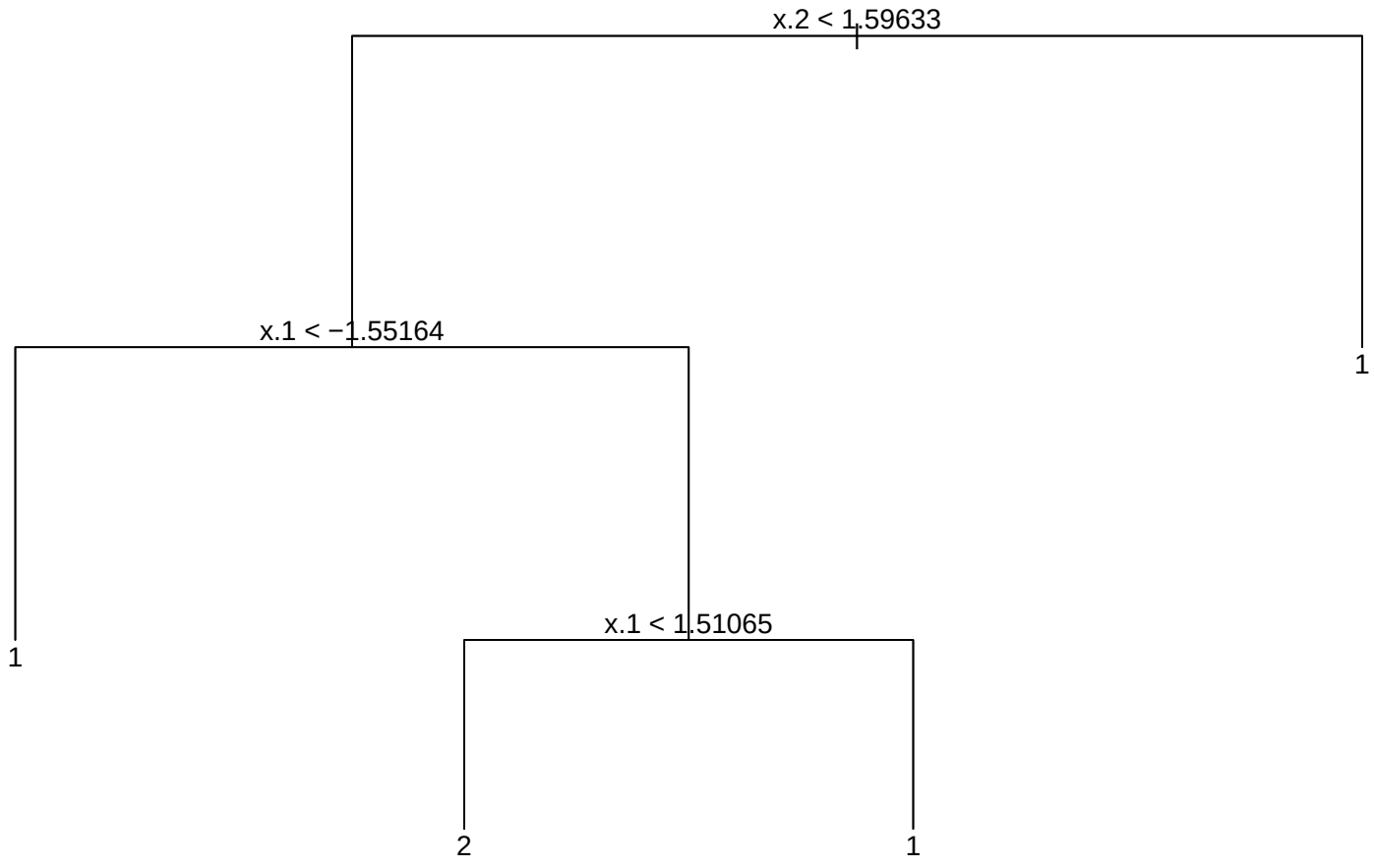
Cross-Validation

```
prune_ders <- tree::prune.misclass(tree_ders, best = 4)
summary(prune_ders)
```

```
##
## Classification tree:
## snip.tree(tree = tree_ders, nodes = c(3L, 10L, 11L))
## Number of terminal nodes:  4
## Residual mean deviance:  0.4972 = 72.58 / 146
## Misclassification error rate: 0.08667 = 13 / 150
```

Sınıflandırma Ağaçları *Uygulama*

```
plot(prune_ders)
text(prune_ders, pretty = 0)
```



Sınıflandırma Ağaçları *Uygulama*

```
tree_ders_pred2 <- predict(prune_ders, train_dnm_ders, type
table(tree_ders_pred2, train_dnm_ders$y)
```

```
##
## tree_ders_pred2  1  2
##               1 99  4
##               2  9 38
```

Sınıflandırma Ağaçları *Uygulama*

```
caret::confusionMatrix(train_dnm_ders$y, tree_ders_pred2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 99  9
##          2  4 38
##
##                Accuracy : 0.9133
##                  95% CI : (0.8564, 0.953)
##     No Information Rate : 0.6867
##     P-Value [Acc > NIR] : 2.773e-11
##
##                   Kappa : 0.7926
##
##  Mcnemar's Test P-Value : 0.2673
##
##             Sensitivity : 0.9612
##             Specificity : 0.8085
##          Pos Pred Value : 0.9167
##          Neg Pred Value : 0.9048
##              Prevalence : 0.6867
##          Detection Rate : 0.6600
##    Detection Prevalence : 0.7200
##       Balanced Accuracy : 0.8848
```

Sınıflandırma Ağaçları *Uygulama*

Test Prediction

```
tree_ders_pred3 <- predict(prune_ders, test_dnm_ders, type
table(tree_ders_pred3, test_dnm_ders$y)
```

```
##
## tree_ders_pred3  1  2
##               1 35  1
##               2  7  7
```

Sınıflandırma Ağaçları *Uygulama*

```
caret::confusionMatrix(test_dnm_ders$y, tree_ders_pred3)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 35  7
##          2  1  7
##
##                Accuracy : 0.84
##                  95% CI : (0.7089, 0.9283)
##     No Information Rate : 0.72
##     P-Value [Acc > NIR] : 0.03646
##
##                   Kappa : 0.5434
##
##  Mcnemar's Test P-Value : 0.07710
##
##             Sensitivity : 0.9722
##             Specificity : 0.5000
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.8750
##              Prevalence : 0.7200
##          Detection Rate : 0.7000
##    Detection Prevalence : 0.8400
##       Balanced Accuracy : 0.7361
```

TRAIN SONUCLARININ KARSILASTIRILMASI

SVM	***Truth***	***Truth***
	Class 1	**Class 2**
Prediction		
Class 1	100	7
Class 2	8	35

CT	***Truth***	***Truth***
	Class 1	**Class 2**
Prediction		
Class 1	99	4
Class 2	9	38

TEST SONUCLARININ KARSILASTIRILMASI

SVM	***Truth***	***Truth***
	Class 1	**Class 2**
Prediction		
Class 1	39	2
Class 2	3	6

CT	***Truth***	***Truth***
	Class 1	**Class 2**
Prediction		
Class 1	35	1
Class 2	7	7

ALGORİTMALARIN KARŞILAŞTIRILMASI

	Train Set			Test Set		
Model	**Accr.**	**Sens.**	**Spec.**	**Accr.**	**Sens.**	**Spec.**
SVM	0.900	0.934	0.814	0.900	0.951	0.666
CT	0.913	0.961	0.808	0.840	0.972	0.500

KAYNAKÇA

Cherkassky, V., & Ma, Y. (2004). Practical selection of SVM parameters and noise estimation for SVM regression. Neural networks, 17(1), 113-126.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning with Applications in R. New York, NY: Springer.

Mammone, A., Turchi, M., & Cristianini, N. (2009). Support vector machines. Wiley Interdisciplinary Reviews: Computational Statistics, 1(3), 283-289.

Widodo, A., & Yang, B. S. (2007). Support vector machine in machine condition monitoring and fault diagnosis. Mechanical systems and signal processing, 21(6), 2560-2574.

Zhang, R., & Wang, W. (2011). Facilitating the applications of support vector machine by using a new kernel. Expert systems with applications, 38(11), 14225-14230.

TEŞEKKÜRLER...