Makine Öğrenmesi (Genombilim Eğitimi/2023)

M. Çisel Kemahlı Aytekin

2023-07-13

```
## Makine Öğrenmesi Uygulaması

1. Probleme hazırlık
a) Kütüphaneleri yükleme
b) Veriseti yükleme
c) Verisetini doğrulama kümelerine bölme

2. Data özeti
a) Tanımlayıcı istatistiklere bakma
b) Data görüntüleme

3. Datayı hazırlama
a) Data temizliği
b) Özellik seçimi
c) Data dönüştürme

4. Algoritmaları değerlendirme
a) Test denemeleri değerlendirme metrikleri
b) Algoritma kontrolü
c) Algoritmaları karşılaştırma

5. Doğruluğu (Accuracy) geliştirme
a) Algoritmayı düzenleme
b) Toparlama

6. Modeli sonuçlandırma
a) Doğrulama veri setiyle ilgili tahminler
b) Eğitim veri setinin tamamında bağımsız model oluşturun
c) Sonraki kullanımlar için modeli kaydetme

Biz bu çalışmadan genomik çalışmalarda genellikle tercih edilen sınıflandırma
algoritmalarını test edeceğiz. Kullanacağımız veri seti de bu amaç için
uygundur.

R programlama dili kullanılarak makine öğrenmesi uygulamak mümkündür. Bunu
uygulamak için caret, mlbench, randomforest gibi bir çok kütüphane
bulunmaktadır.

## Gerekli kütüphaneler
```

```r
library(caret)
library(mlbench)
library(randomForest)
```

1. Probleme hazırlık

Pima Indians Diabetes

Bu veri kümesi aslen Ulusal Diyabet, Sindirim ve Böbrek Hastalıkları Enstitüsü'ndendir. Veri setinin amacı, veri setinde yer alan belirli teşhis ölçümlerine dayalı olarak bir hastanın diyabet hastası olup olmadığını teşhis amaçlı olarak tahmin etmektir. Özellikle, buradaki tüm hastalar en az 21 yaşında Pima Kızılderili mirasına sahip kadınlardır.

Pima Kızılderelileri

- Pregnancies: Hamile kalma sayısı
- Glucose: Oral glukoz tolerans testinde 2 saatlik plazma glukoz konsantrasyonu
- BloodPressure: Diyastolik kan basıncı (mm Hg)
- SkinThickness: Triceps cilt kıvrım kalınlığı (mm)
- Insulin: 2 saatlik serum insülini (mu U/ml)
- BMI: Vücut kitle indeksi (kg cinsinden ağırlık/(m cinsinden boy)^2)
- DiabetesPedigreeFunction: Diyabet soyağacı işlevi
- Age: Yaş (yıl)
- Outcome: Sınıf değişkeni (0 veya 1)

```r
# Veri setini yükleme
data(PimaIndiansDiabetes)
```

2. Data özeti

Öncelikle datamızın içeriğini inceleyelim.

```r
head(PimaIndiansDiabetes)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72      35       0 33.6    0.627  50      pos
## 2        1      85       66      29       0 26.6    0.351  31      neg
## 3        8     183       64       0       0 23.3    0.672  32      pos
## 4        1      89       66      23      94 28.1    0.167  21      neg
## 5        0     137       40      35     168 43.1    2.288  33      pos
## 6        5     116       74       0       0 25.6    0.201  30      neg
```

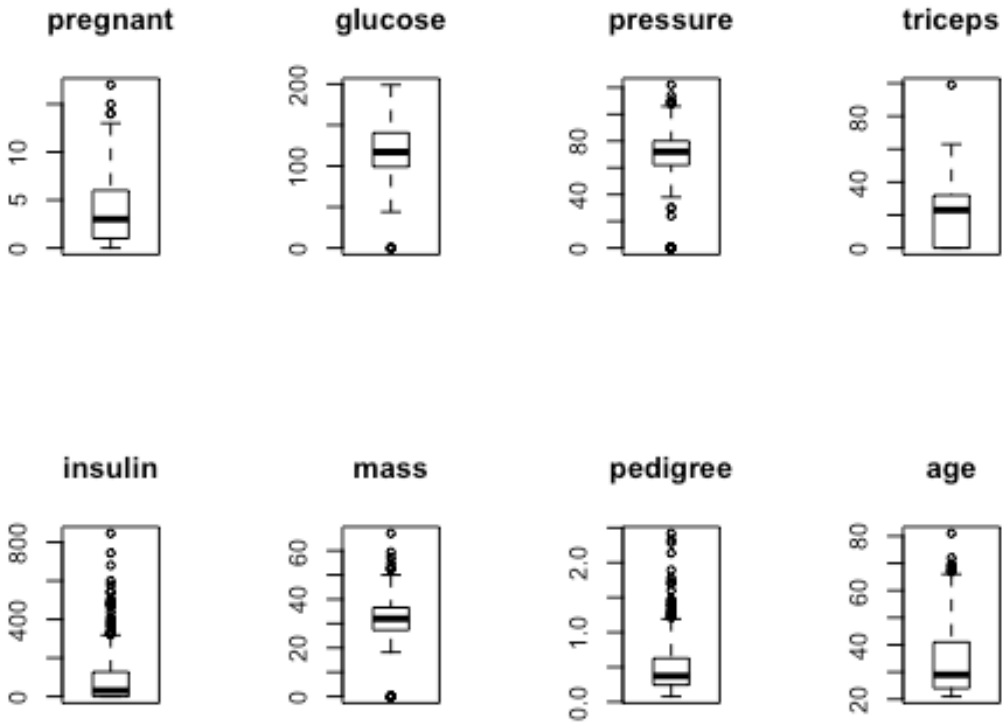
```
summary(PimaIndiansDiabetes)
```

```
##     pregnant         glucose         pressure         triceps     
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00  
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00  
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00  
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54  
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00  
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00  
##     insulin           mass          pedigree           age        diabetes  
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00   neg:500  
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00   pos:268  
##  Median : 30.5   Median :32.00   Median :0.3725   Median :29.00            
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24            
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00            
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

Bu çalışmada amacımız bu veri setindeki değişkenlikleri kullanarak kişinin diyabet hastalığı olup olmadığını tespit etmek ve bunu tahmin eden bir makina öğrenmesi algoritması tanımlamaktır.

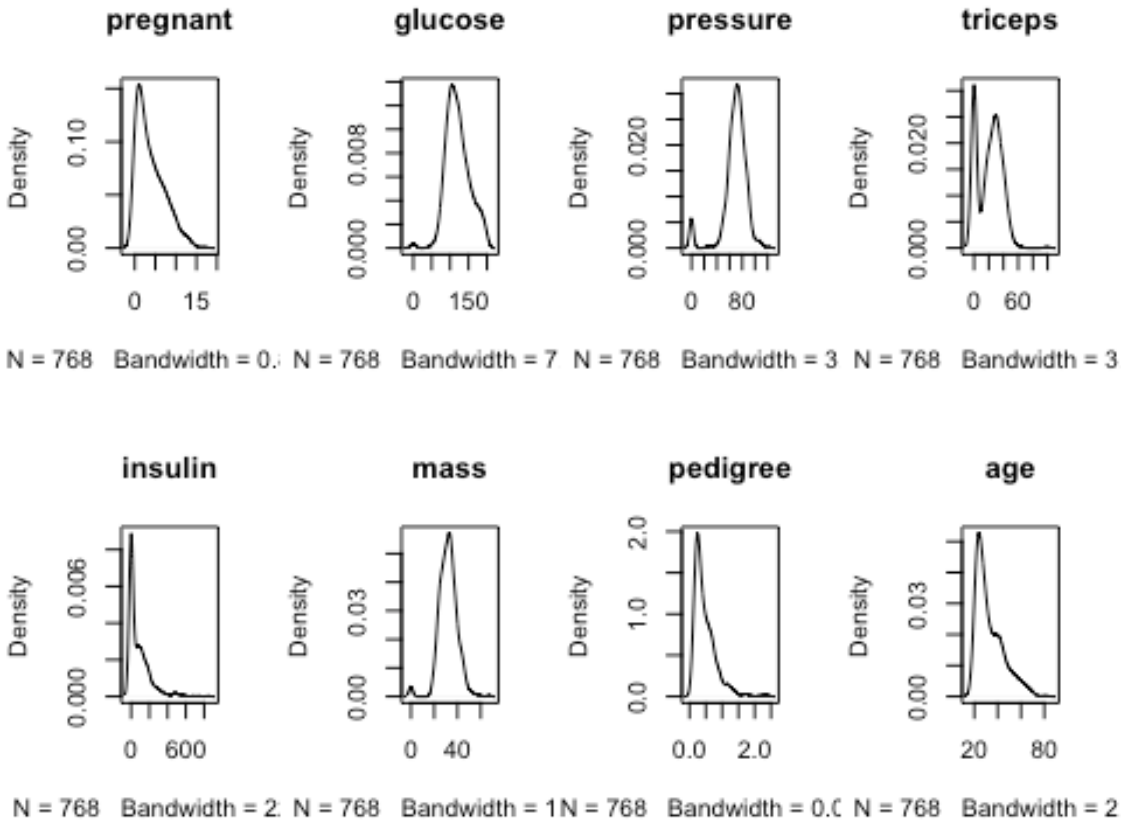
Her bir özellik için boxplot oluşturalım.

```
par(mfrow=c(2,4))
for(i in 1:8) {
  boxplot(PimaIndiansDiabetes[,i], main=names(PimaIndiansDiabetes)[i])
}
```



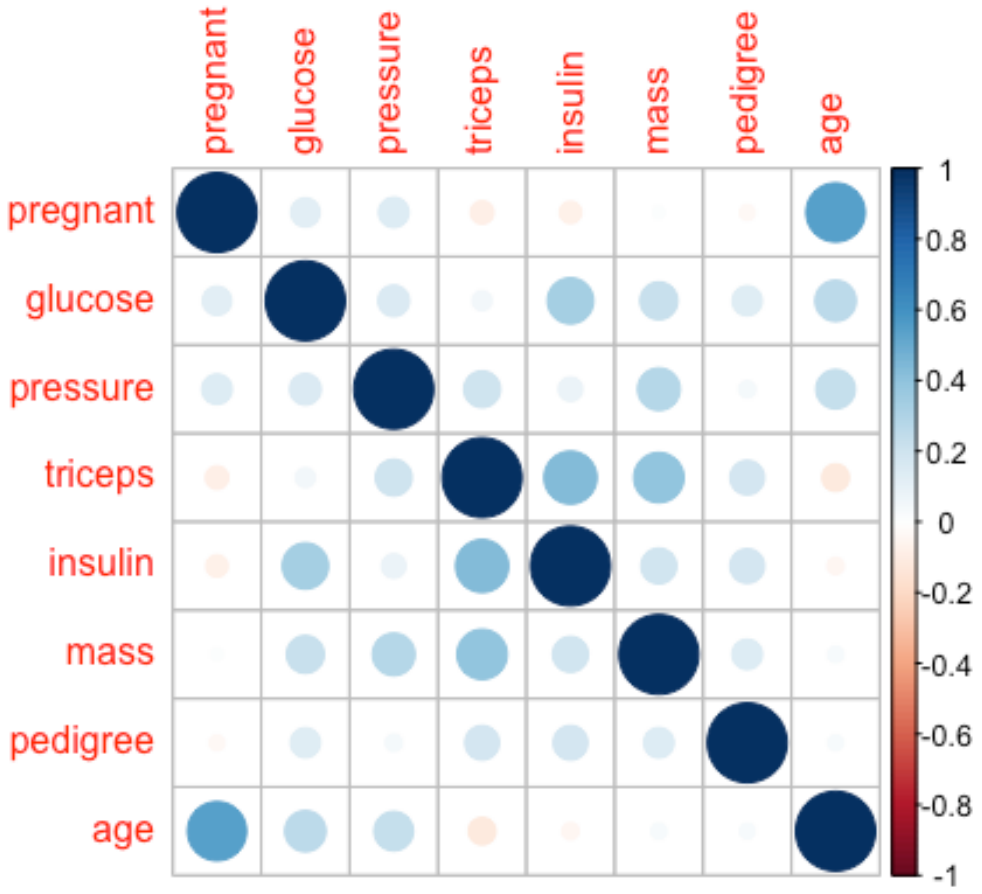
Her bir özelliğin bulunma sıklığına bakabiliriz.

```
par(mfrow=c(2,4))
for(i in 1:8) {
  plot(density(PimaIndiansDiabetes[,i]), main=names(PimaIndiansDiabetes)[i])
}
```



corrplot kütüphanesini kullanarak özellikler arasındaki korelasyona bakabiliriz.

```
#par(mfrow=c(1,1))
library(corrplot)
# Korelasyonun hesaplanması
correlations <- cor(PimaIndiansDiabetes[,1:8])
# Korelasyon grafiği oluşturma
corrplot(correlations, method="circle")
```



3. Datayı hazırlama

- Bu veri setinde 'NA' ve 0 değerleri önceden temizlenmiş ve hazırdır.
- 'Diabetes' özelliği hariç hepsi sayısal girdiler olduğu için korelasyon, ve grafiksel görüntüleme için kullanılabilmektedir.
- Bu çalışmada amaç bu özellikleri bireyin diyabet olması ile ilişkisini bulmak ve yeni bir veri girdiğimizde kişinin diyabet olup olmadığını test etmektir.
- Verimizi test ve train olmak üzere ikiye böleceğiz. Test setini en iyi algoritmayı bulmak için kullanacağız. Train setini de bu algoritmaları tahmin edilebilirliğini test etmek için kullanacağız.

4. Algoritmaları değerlendirme

Makine öğrenmesi için bir çok farklı algoritma mevcuttur. Temel olarak sınıflandırma ve regresyon analizleri için bir çok farklı algoritma kullanılabilir.

Aralarından seçim yapabileceğiniz birçok olası değerlendirme ölçütü vardır.

Sınıflandırma:

Doğruluk (Accuracy): x doğru bölü y toplam örnek. Anlaşılması kolay ve yaygın olarak kullanılır.

Kappa: sınıfların temel dağılımını hesaba katan doğruluk olarak kolayca anlaşılır.

Regresyon:

RMSE: Ortalama hatanın karekökü (root mean squared error). Yine, anlaşılması kolay ve yaygın olarak kullanılıyor.

Rsquared: uyumun iyiliği veya belirleme katsayısı (the goodness of fit or coefficient of determination).

```
# Eğitme planı hazırlamak
control <- trainControl(method="repeatedcv", number=10, repeats=3)
```

Aslında burada ne kadar çok tekrar yapabilirsek elde ettiğimiz değerlendirme ölçütlerimizi gerçeğe yakın ve doğruluğu yüksek bir şekilde elde edebiliriz.

Şimdi farklı modelleri deneyebiliriz.

```
# SVM
set.seed(7)
fit.svm <- train(diabetes~., data=PimaIndiansDiabetes, method="svmRadial",
metric="Accuracy", trControl=control)
print(fit.svm)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 768 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 691, 691, 691, 691, 691, 691, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.7712919  0.4621585
##   0.50  0.7625769  0.4485309
##   1.00  0.7560549  0.4339951
##
## Tuning parameter 'sigma' was held constant at a value of 0.124824
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.124824 and C = 0.25.
```

```
# LDA
set.seed(7)
fit.lda <- train(diabetes~., data=PimaIndiansDiabetes, method="lda",
metric="Accuracy", preProc=c("center", "scale"), trControl=control)
print(fit.lda)
```

```
## Linear Discriminant Analysis
##
## 768 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 691, 691, 691, 691, 691, 691, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7791069  0.4862025
```

```
# Random Forest
set.seed(7)
fit.rf <- train(diabetes~., data=PimaIndiansDiabetes, method="rf",
metric="Accuracy", trControl=control)
print(fit.rf)
```

```
## Random Forest
##
## 768 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 691, 691, 691, 691, 691, 691, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7638528  0.4630809
##   5     0.7634256  0.4664261
##   8     0.7599738  0.4596437
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

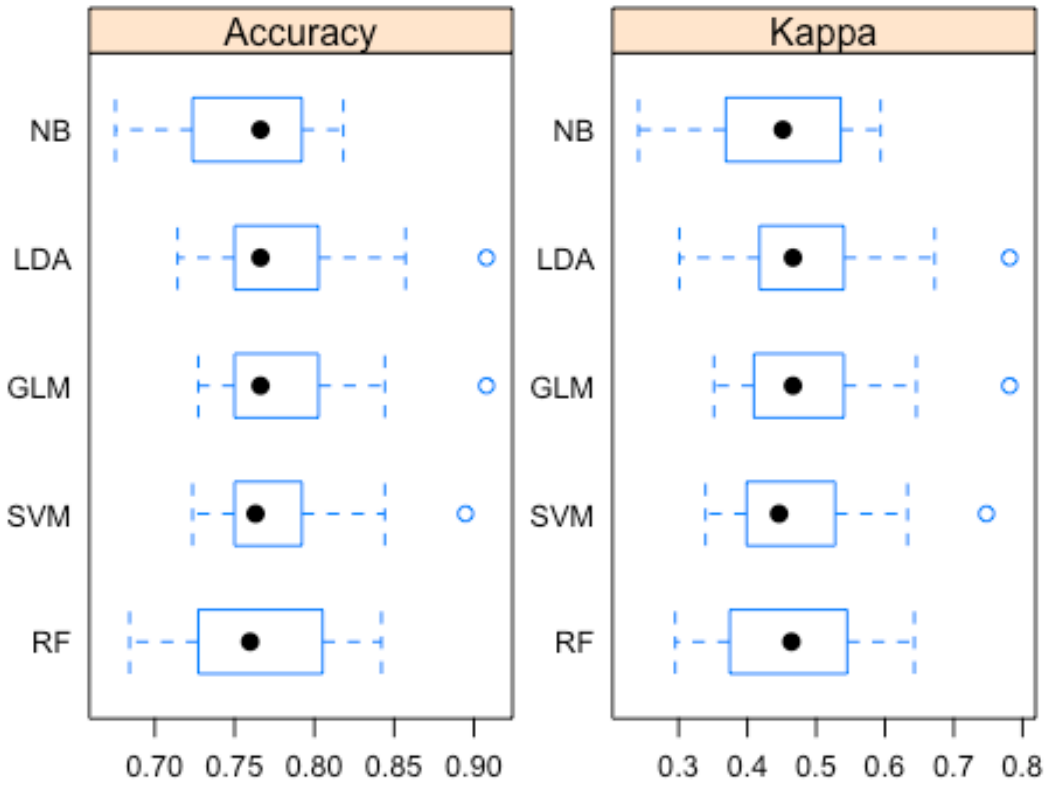
```
# Naive Bayes
set.seed(7)
fit.nb <- train(diabetes~., data=PimaIndiansDiabetes, method="nb",
metric="Accuracy", trControl=control)
print(fit.nb)
```

```
## Naive Bayes
##
## 768 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
```

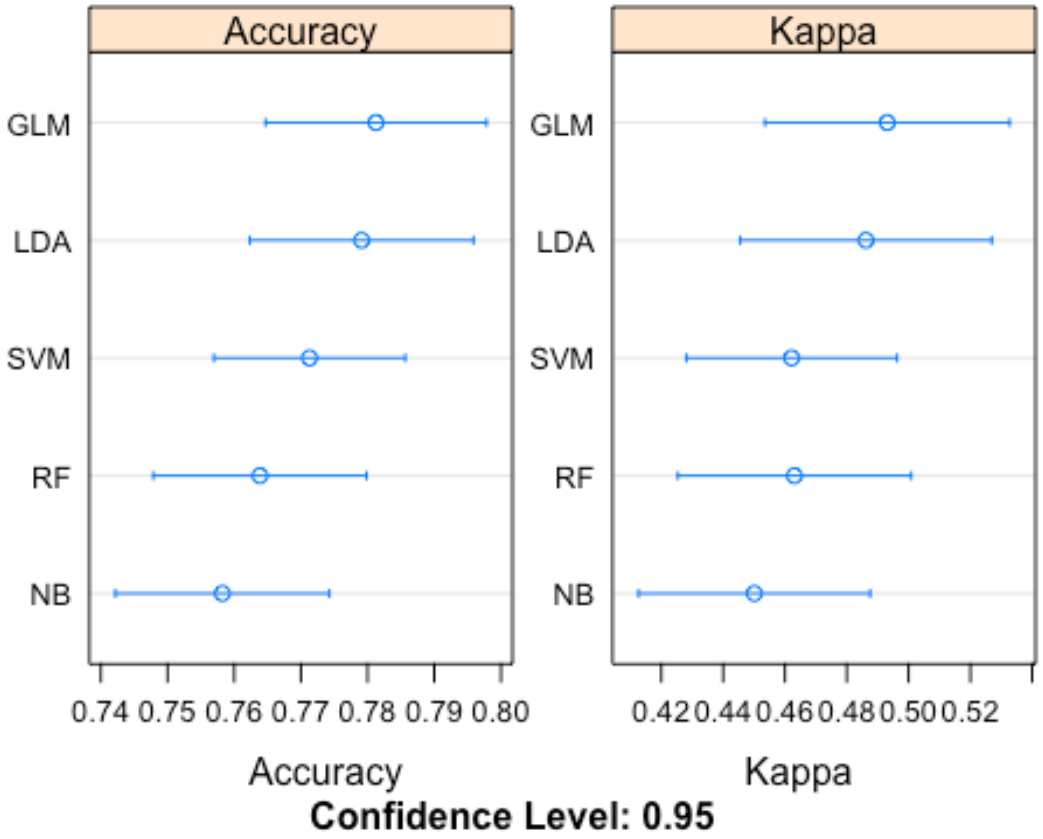
```
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 691, 691, 691, 691, 691, 691, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.7548189  0.4481068
##    TRUE      0.7582308  0.4501022
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = TRUE and
adjust
##  = 1.

# Logistic Regression
set.seed(7)
fit.glm <- train(diabetes~., data=PimaIndiansDiabetes, method="glm",
metric="Accuracy", preProc=c("center", "scale"), trControl=control)
print(fit.glm)

## Generalized Linear Model
##
## 768 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 691, 691, 691, 691, 691, 691, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7812657  0.4931161

# Tekrar örnekleminin toparlanması
results <- resamples(list(LDA=fit.lda, SVM=fit.svm, NB=fit.nb, RF=fit.rf,
GLM=fit.glm))
# Farklı modların özeti
summary(results)

##
## Call:
## summary.resamples(object = results)
##
## Models: LDA, SVM, NB, RF, GLM
## Number of resamples: 30
##
## Accuracy
```

```
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA 0.7142857 0.7508117 0.7662338 0.7791069 0.8000256 0.9078947    0
## SVM 0.7236842 0.7508117 0.7631579 0.7712919 0.7915243 0.8947368    0
## NB  0.6753247 0.7236842 0.7662338 0.7582308 0.7922078 0.8181818    0
## RF  0.6842105 0.7305195 0.7597403 0.7638528 0.8019481 0.8421053    0
## GLM 0.7272727 0.7508117 0.7662338 0.7812657 0.8000256 0.9078947    0
##
## Kappa
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LDA 0.3011551 0.4192537 0.4662541 0.4862025 0.5308596 0.7812500    0
## SVM 0.3391908 0.3997116 0.4460612 0.4621585 0.5234605 0.7475083    0
## NB  0.2418275 0.3716119 0.4514770 0.4501022 0.5349232 0.5938206    0
## RF  0.2951613 0.3778304 0.4640696 0.4630809 0.5447483 0.6426332    0
## GLM 0.3513839 0.4168485 0.4662541 0.4931161 0.5391907 0.7812500    0
```

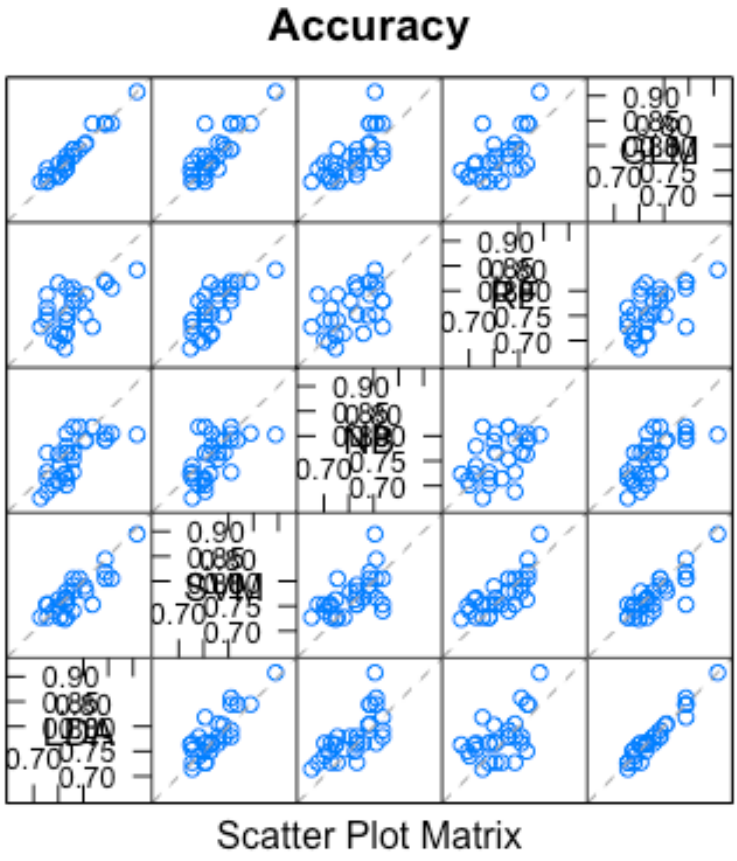
```r
# Farklı modellerin karşılaştırması için box ve whisker grafikleri oluşturma
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(results, scales=scales)
```



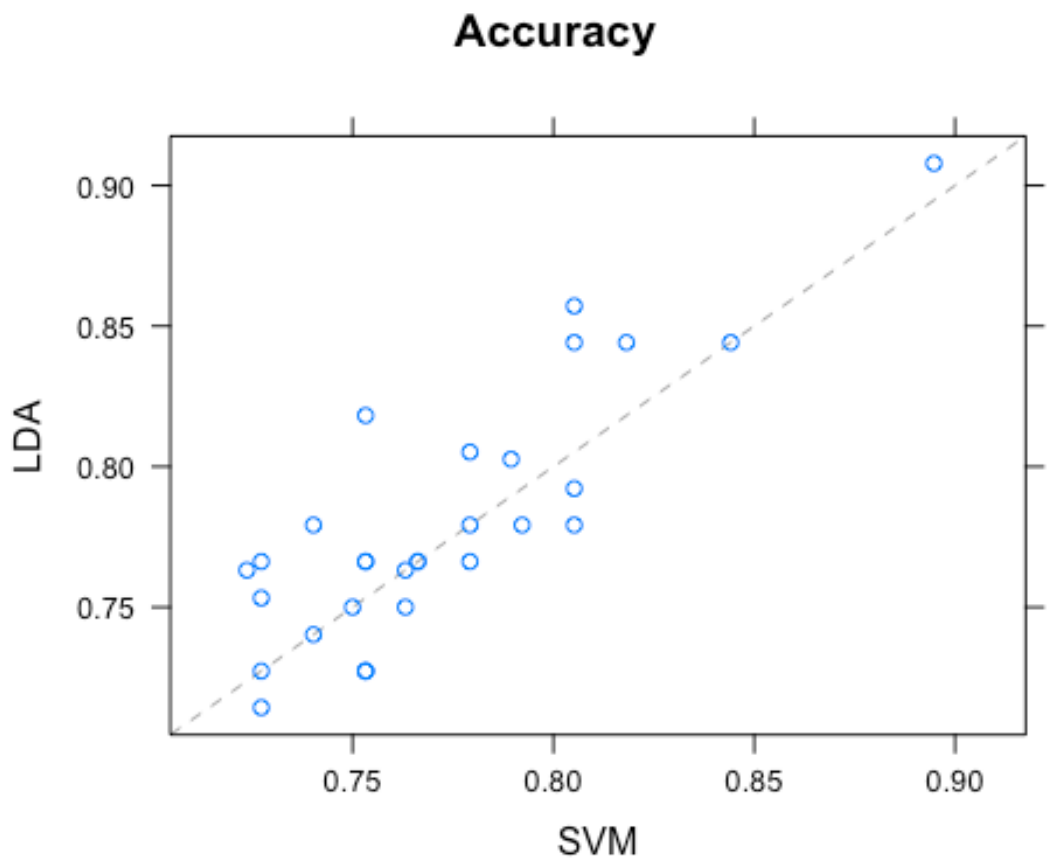

```r
# Doğruluğun nokta grafik halinde gösterimi
scales <- list(x=list(relation="free"), y=list(relation="free"))
dotplot(results, scales=scales)
```



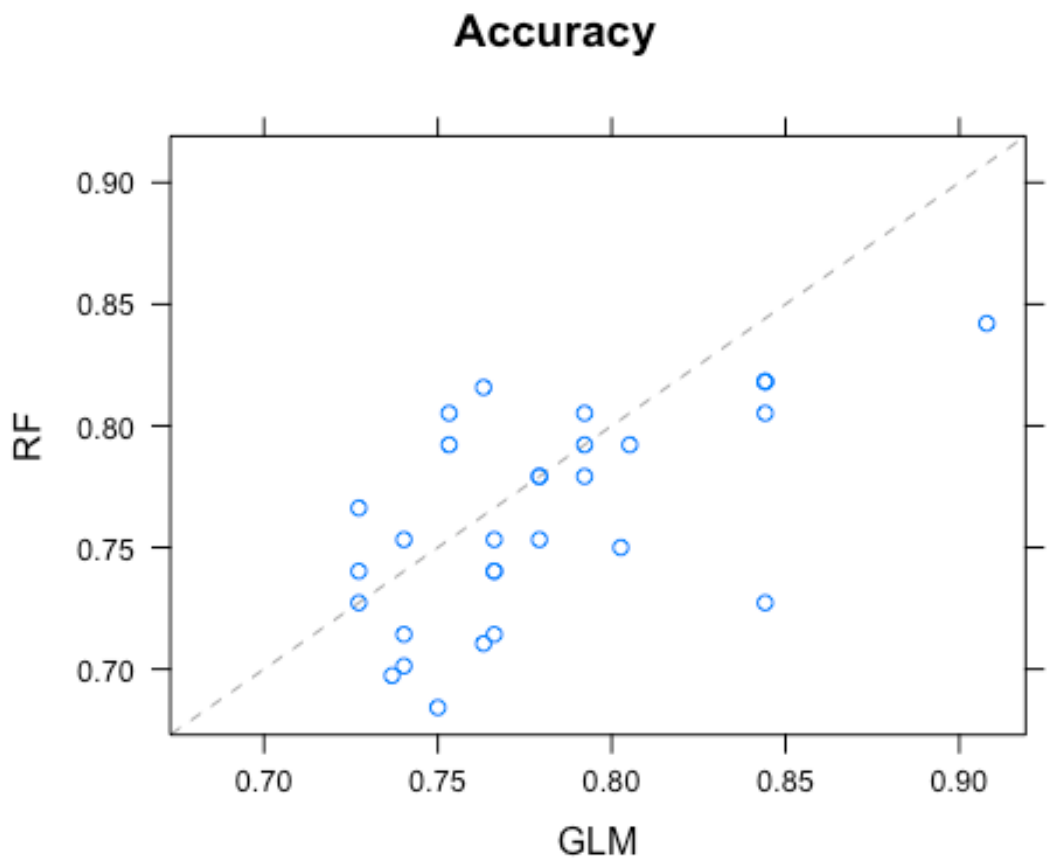

```r
# İkili tahminleri dağılım grafikleri halinde gösterimi
splom(results)
```




```r
# Modellerin karşılıklı olarak gösterimi
xyplot(results, models=c("LDA", "SVM"))
```




```
xyplot(results, models=c("RF", "GLM"))
```



Bu aşamada belli algoritmaların veri setimiz için öğrenme işlemini yapmasını sağladık. fFarklı algoritmalarda oluşan tahminlerin kontrolünü yapabiliriz.

```
# Model tahminleri arasındaki farklar
diffs <- diff(results)
# İkili karşılaştırmaların p değerlerinin özeti
summary(diffs)

##
## Call:
## summary.diff.resamples(object = diffs)
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## Accuracy
##     LDA      SVM       NB        RF        GLM
## LDA           0.007815  0.020876  0.015254 -0.002159
## SVM 0.915689            0.013061  0.007439 -0.009974
## NB  0.021892 0.704353            -0.005622 -0.023035
## RF  0.449062 1.000000  1.000000            -0.017413
## GLM 1.000000 0.307501  0.008726  0.159916
```

```
##
## Kappa
##     LDA     SVM        NB         RF         GLM
## LDA          0.0240440  0.0361002  0.0231215 -0.0069137
## SVM 0.35627             0.0120562 -0.0009225 -0.0309577
## NB  0.25315 1.00000               -0.0129787 -0.0430139
## RF  1.00000 1.00000    1.00000                -0.0300352
## GLM 1.00000 0.07976    0.09242    0.72782
```

```
# LDA algoritmasının kontrolü
predictions_lda <- predict(fit.lda, PimaIndiansDiabetes[,1:8])
confusionMatrix(predictions_lda, PimaIndiansDiabetes$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 446 112
##        pos  54 156
##
##                Accuracy : 0.7839
##                  95% CI : (0.753, 0.8125)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 7.051e-16
##
##                   Kappa : 0.4992
##
##  Mcnemar's Test P-Value : 9.686e-06
##
##             Sensitivity : 0.8920
##             Specificity : 0.5821
##          Pos Pred Value : 0.7993
##          Neg Pred Value : 0.7429
##              Prevalence : 0.6510
##          Detection Rate : 0.5807
##    Detection Prevalence : 0.7266
##       Balanced Accuracy : 0.7370
##
##        'Positive' Class : neg
##
```

```
# SVM algoritmasının kontrolü
predictions_svm <- predict(fit.svm, PimaIndiansDiabetes[,1:8])
confusionMatrix(predictions_svm, PimaIndiansDiabetes$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 463 111
##        pos  37 157
```

```
##
##                Accuracy : 0.8073
##                  95% CI : (0.7776, 0.8346)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5469
##
##  Mcnemar's Test P-Value : 1.966e-09
##
##             Sensitivity : 0.9260
##             Specificity : 0.5858
##          Pos Pred Value : 0.8066
##          Neg Pred Value : 0.8093
##              Prevalence : 0.6510
##          Detection Rate : 0.6029
##    Detection Prevalence : 0.7474
##       Balanced Accuracy : 0.7559
##
##        'Positive' Class : neg
##
```

```
# NB algoritmasının kontrolü
predictions_nb <- predict(fit.nb, PimaIndiansDiabetes[,1:8])
confusionMatrix(predictions_nb, PimaIndiansDiabetes$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 434 102
##        pos  66 166
##
##                Accuracy : 0.7812
##                  95% CI : (0.7503, 0.81)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 2.654e-15
##
##                   Kappa : 0.5031
##
##  Mcnemar's Test P-Value : 0.006928
##
##             Sensitivity : 0.8680
##             Specificity : 0.6194
##          Pos Pred Value : 0.8097
##          Neg Pred Value : 0.7155
##              Prevalence : 0.6510
##          Detection Rate : 0.5651
##    Detection Prevalence : 0.6979
##       Balanced Accuracy : 0.7437
```

```
##
##        'Positive' Class : neg
##
```

```
# RF algoritmasının kontrolü
predictions_rf <- predict(fit.rf, PimaIndiansDiabetes[,1:8])
confusionMatrix(predictions_rf, PimaIndiansDiabetes$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 500   0
##        pos   0 268
##
##                Accuracy : 1
##                  95% CI : (0.9952, 1)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.651
##          Detection Rate : 0.651
##    Detection Prevalence : 0.651
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : neg
##
```

```
# GLM algoritmasının kontrolü
predictions_glm <- predict(fit.glm, PimaIndiansDiabetes[,1:8])
confusionMatrix(predictions_glm, PimaIndiansDiabetes$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 445 112
##        pos  55 156
##
##                Accuracy : 0.7826
##                  95% CI : (0.7517, 0.8112)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 1.373e-15
```

```
##
##                  Kappa : 0.4966
##
##  Mcnemar's Test P-Value : 1.468e-05
##
##             Sensitivity : 0.8900
##             Specificity : 0.5821
##          Pos Pred Value : 0.7989
##          Neg Pred Value : 0.7393
##              Prevalence : 0.6510
##          Detection Rate : 0.5794
##    Detection Prevalence : 0.7253
##       Balanced Accuracy : 0.7360
##
##        'Positive' Class : neg
##
```

5. Doğruluğu (Accuracy) geliştirme

Elde ettiğimiz sonuçlara bakılırsa en iyi doğruluğu veren Rastgele Orman (RF) algoritması ile eğitimimizi geliştirmeye devam edebiliriz.

Bir algoritmayı ayarlarken, parametrelerin oluşturduğunuz model üzerindeki etkisini bilmek için algoritmanızı iyi anlamanız önemlidir.

Rastgele orman modelimiz üzerinde aşağıdaki etkileri olan mtry ve ntree parametreleri olmak üzere iki parametreyi ayarlamaya bağlı kalacağız. Başka pek çok parametre var, ancak bu iki parametre belki de nihai doğruluğunuz üzerinde en büyük etkiye sahip olanlardır.

R'deki randomForest() işlevi için doğrudan yardım sayfasından:

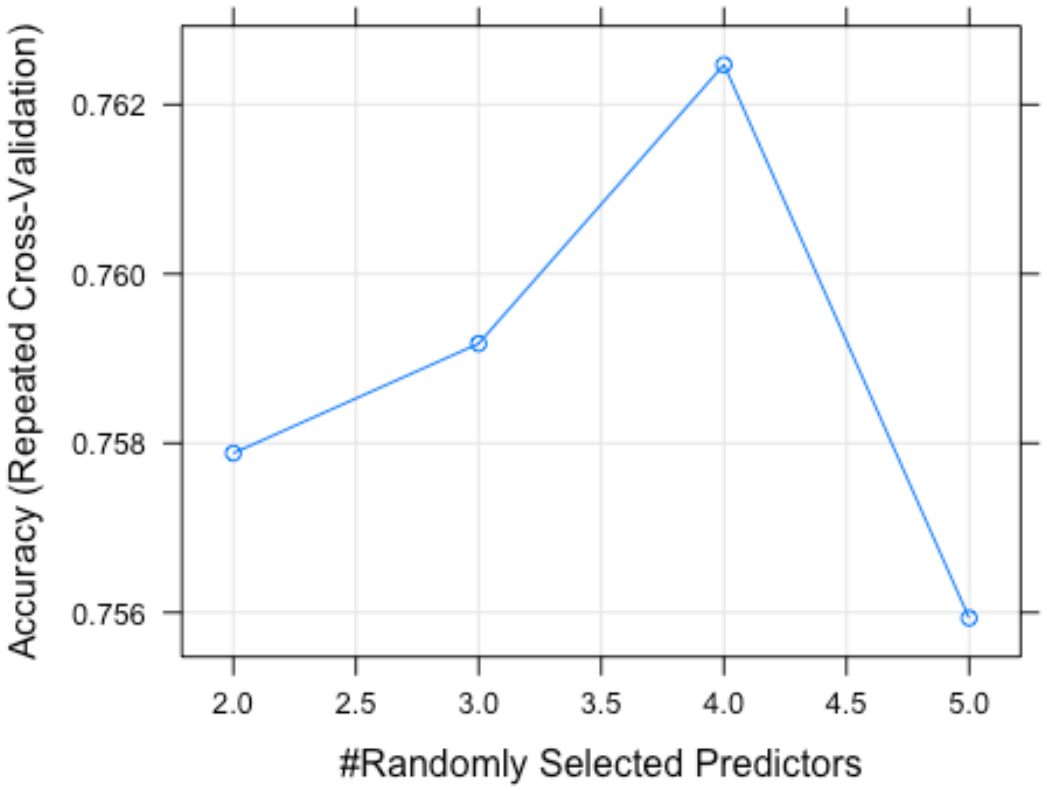
mtry: Her bölmede aday olarak rastgele örneklenen değişken sayısı. ntree: Büyüyecek ağaç sayısı.

```
# Veri seti test ve train olmak üzere rastgele doğrulama kümelerine
böleceğiz.
set.seed(7)
split <- createDataPartition(y=PimaIndiansDiabetes$diabetes, p=0.66,
list=FALSE)
trainPima <- PimaIndiansDiabetes[split,]
testPima <- PimaIndiansDiabetes[-split,]

# mtry değerini 2 ile 5 arasında deneyeceğiz.
set.seed(7)
metric <- "Accuracy"
# tunegrid <- expand.grid(.mtry=c(sqrt(ncol(trainPima))))
tunegrid <- expand.grid(.mtry=c(2:5))
rf_gridsearch <- train(diabetes~., data=trainPima, method="rf",
metric=metric, tuneGrid=tunegrid, trControl=control)
```

```r
bestmtry = rf_gridsearch$bestTune

print(rf_gridsearch)
```

```
## Random Forest
##
## 507 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 456, 456, 456, 457, 456, 456, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7578824  0.4447928
##   3     0.7591765  0.4522767
##   4     0.7624706  0.4607041
##   5     0.7559346  0.4489375
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```r
plot(rf_gridsearch)
```



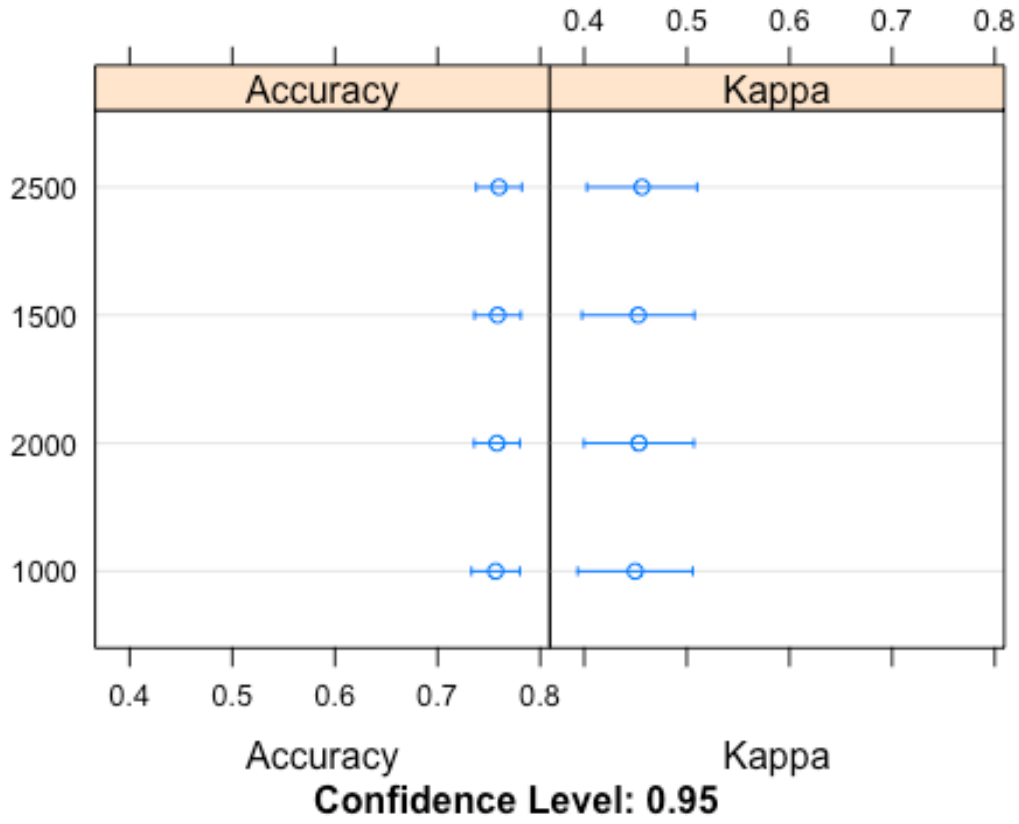

```
print(bestmtry$mtry)
```

```
## [1] 4
```

```
# mtry değerini belirledik. ntree değerini de 1000, 1500, 2000, 2500
değerleri için deneyeceğiz.
tunegrid <- expand.grid(.mtry=bestmtry$mtry)
modellist <- list()
for (ntree in c(1000, 1500, 2000, 2500)) {
  set.seed(7)
  fit <- train(diabetes~., data=trainPima, method="rf", metric=metric,
tuneGrid=tunegrid, trControl=control, ntree=ntree)
  key <- toString(ntree)
  modellist[[key]] <- fit
}

# Sonuçları karşılaştırma
results <- resamples(modellist)
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
```

```
## Models: 1000, 1500, 2000, 2500
## Number of resamples: 30
##
## Accuracy
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 1000 0.64 0.7094118 0.7647059 0.7566013 0.8039216 0.8627451    0
## 1500 0.64 0.7058824 0.7843137 0.7585490 0.8039216 0.8627451    0
## 2000 0.64 0.7058824 0.7745098 0.7579085 0.8039216 0.8627451    0
## 2500 0.64 0.7058824 0.7745098 0.7598562 0.8039216 0.8627451    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## 1000 0.1766382 0.3260041 0.5033616 0.4496830 0.5582809 0.6956522    0
## 1500 0.2138728 0.3006918 0.5154627 0.4526112 0.5669852 0.6956522    0
## 2000 0.1766382 0.3067398 0.5033616 0.4533272 0.5669852 0.6956522    0
## 2500 0.2138728 0.3019728 0.5033616 0.4563155 0.5679267 0.6956522    0

dotplot(results)
```



6. Modeli sonuçlandırma

Rastgele orman algoritmasını kullanarak en iyi parametreleri belirledik.

```r
# En iyi parametreler:
chosenNTree <- 2500
chosenmtry <- bestmtry$mtry

rf_pima <- randomForest(trainPima[,1:7], y=trainPima$diabetes,
data=trainPima, proximity=TRUE, ntree=chosenNTree, nodesize=5,
importance=TRUE, mtry=chosenmtry)

print(rf_pima)
```

```
##
## Call:
##  randomForest(x = trainPima[, 1:7], y = trainPima$diabetes, ntree =
chosenNTree,      mtry = chosenmtry, nodesize = 5, importance = TRUE,
proximity = TRUE,      data = trainPima)
##                Type of random forest: classification
##                      Number of trees: 2500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 24.85%
## Confusion matrix:
##     neg pos class.error
## neg 283  47   0.1424242
## pos  79  98   0.4463277
```

Ortalama azalma doğruluğu (Mean decrease accuracy), her parametrenin olmadan modelin performansının ölçüsüdür. Daha yüksek bir değer, grubu (diyabetik ve sağlıklı) tahmin etmede o parametrenin önemini gösterir. Bu parametrenin çıkarılması, modelin tahmindeki doğruluğunu kaybetmesine neden olur.

```r
importance(rf_pima, type=1)
```

```
##          MeanDecreaseAccuracy
## pregnant            31.543995
## glucose            103.169554
## pressure             7.327240
## triceps              5.849174
## insulin              7.972205
## mass                44.044643
## pedigree             7.247046
```

Görüldüğü üzere glikoz değeri diyabet hastası olma durumunun sınıflandırılmasında en yüksek öneme sahip parametre olarak bulunmuştur, ki bu da bekleyebildiğimiz bir sonuç olacaktır.

```r
predictions <- predict(rf_pima, newdata = testPima[,1:8])
confusionMatrix(predictions, testPima$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction neg pos
##        neg 140  30
##        pos  30  61
##
##                Accuracy : 0.7701
##                  95% CI : (0.7142, 0.8197)
##     No Information Rate : 0.6513
##     P-Value [Acc > NIR] : 2.177e-05
##
##                   Kappa : 0.4939
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8235
##             Specificity : 0.6703
##          Pos Pred Value : 0.8235
##          Neg Pred Value : 0.6703
##              Prevalence : 0.6513
##          Detection Rate : 0.5364
##    Detection Prevalence : 0.6513
##       Balanced Accuracy : 0.7469
##
##        'Positive' Class : neg
##
```

```
table(predictions, testPima$diabetes)
```

```
##
## predictions neg pos
##         neg 140  30
##         pos  30  61
```

Bu analizler örneklem, deneme sayılarının artırılması ile daha güvenilir sonuçlar verecektir. Bunun için yüksek işlemci gücüne sahip TRUBA gibi serverlar kullanılarak makine öğrenmesi analizleri gerçekleştirilebilir.

Sonuç olarak modelin kullanmadığı bir test veri seti verdiğimizde en iyi parametreleri seçerek oluşturduğumuz modelimiz bize bireyleri diyabet olma durumlarını söyleyecektir.

Kendi çalışmalarınızda kullanacağınız verilerinize, sorularınıza göre algoritmalar, kullandığınız parametreler değişecektir. En iyisini bulmak ve en iyi makine öğrenme yöntemini belirlemek için denemeler yapmak en önemli aşamadır.

Umarım bu çalışma size genomik vb. çalışmalarda makine öğrenmesi analizlerinin uygulanabilir olduğunu göstermiştir. Korkmayın, bir sürü hata alsanız da denemeye devam edin.

Bana dilediğiniz zaman mckemahli@gmail.com mail adresinden ulaşabilirsiniz.

Kodlarınızla kalın :)