caret: : HATIRLATICI NOTLAR

Model Belirleme

Model içerisindeki değişkenleri belirlemek için olası söz dizimi:

```
train(kararDegiskeni ~ x1 + x2, data = veriSeti, ...)
train(x = egitim_veriSeti, y = kararDegiskeni, ...)
train(recipe_object, data = veriSeti, ...)
```

- "rfe", "sbf", "gafs" ve "safs" fonksiyonları x/y arayüzüne sahiptir.
- **train** fonksiyonu her zaman **kukla değişken** (*dummy variable)* üretir.
- x/y arayüzü kukla değişken üretmez; ancak altında yatan model fonksiyonu üretebilir.

Hatırlatma:

- Veri setinde değişken isimleri bulunmalıdır.
- Sınıflandırmada çıktı değişkeni olarak `factor` tipi kullanılmalıdır (0/1 veya tamsayı olmamalıdır).
- Sınıf kategorileri (`levels`) R için geçerli isimler kullanılmalıdır ("0"/"1" olmamalı).
- Her seferinde aynı örneklerle çalışmak için **train** fonksiyonunu çağırmadan önce rastgele bir başlangıç değeri (`seed`) belirleyin.
- Eksik veriyi tamamlamak için **train** fonksiyonunda **na.action = na.pass** parametresini kullanın. Ayrıca, bu seçenek kayıp değer içeren yeni veriyi tahmin etmede de kullanılabilir.

Model fonksiyonundaki seçenekleri geçmek için üç nokta kullanılabilir:

```
train(y ~ ., data = veriSeti, method = "rf",
# `randomForest` icin secenekler:
importance = TRUE)
```

Paralel İşleme

foreach paketi modellerin paralel olarak çalıştırılması için kullanılır. **train** fonksiyonundaki kodlar değişmez; ancak öncesinde "**do**" paketinin çağırılması gerekir.

```
#MacOS veya Linux için
library(doMC)
registerDoMC(cores=4)
```

```
#Windows için
library(doParallel)
Cl<-makeCluster(2)
registerDoParallel(cl)
```

parallel::`detectCores` fonksiyonu da yardımcı olabilir.

Ön İşleme

Niteliklere; dönüştürme, filtreleme ve başka işlemler **preProc** seçeneği ile uygulanabilir.

```
train(, preProc = c("metot1", "metot2"), ...)
```

Metotlar şunlardır:

- Nitelikleri normalize etmek için "`center`", "`scale`" ve "`range`"
- Nitelikleri dönüştürmek için "`BoxCox`", "`YeoJohnson`" veya "`expoTrans`"
- Tamamlamak için "`knnImpute` ", "`bagImpute`" veya "`medianImpute`"
- Filtrelemek için "`corr`", "`nzv`", "`zv`" ve "`conditionalX`"
- Grupları dönüştürmek için "`pca`", "`ica`", or "`spatialSign`"

train fonksiyonu işlemlerin sırasını belirler; metotların sırası fark etmemektedir.

recipes paketinde ön işleme için daha geniş bir liste bulunmaktadır.

Ekleme Seçenekleri

train fonksiyonunda birçok seçenek **trainControl** fonksiyonu ile belirtilebilir:

```
train(y ~ ., data = veriSeti, method = "cubist",
      trControl = trainControl(<secenekler>))
```

Yeniden Örnekleme Seçenekleri

trainControl yeniden örnekleme yöntemini seçmek için kullanılır:

```
trainControl(method = <metot>, <secenekler>)
```

Metotlar ve seçenekler şunlardır:

- k-kat çapraz geçerleme için "`cv`" (`number` kat sayısını belirtir).
- Tekrarlı çapraz geçerleme için "`repeatedcv`" (`repeats` tekrar sayısını belirtir).
- Bootstrap için "`boot`" (`number` iterasyon sayısını belirtir).
- Bir grubu dışarıda bırak için "`LGOCV`" (seçenekler: `number` ve `p`).
- Birini dışarıda bırak çapraz geçerleme için "`LOO`".
- "`oob`" out-of-bag için (sadece bazı modeller için).
- Zaman serisi verisi için "`timeslice`" (seçenekler: `initialWindow`, `horizon`, `fixedWindow` ve `skip`)

Performans Ölçütleri

Modelin nasıl özetleneceğini seçmek için yine **trainControl** fonksiyonu kullanılır.

```
trainControl(summaryFunction = <R fonksiyonu>,
classProbs = <mantıksal>)
```

Özel R fonksiyonları da kullanılabilir; ancak **caret** paketi birkaç fonksiyon içermektedir:

defaultSummary (doğruluk, RMSE vb. için), **twoClassSummary** (ROC eğrileri için) ve **prSummary** (bilgi getirimi için). Son iki fonksiyon için `classProbs` seçeneği **TRUE** olmalıdır.

Grid Arama (Grid Search)

train fonksiyonunun **iyileştirme** (*tuning*) parametresini(lerini) belirlemesine izin vermek için, `tuneLength` seçeneği, **iyileştirme parametresi başına** kaç değer verilebileceğini denetlemektedir.

Alternatif olarak, **tuneGrid** kullanılarak spesifik değerler verilebilir:

```
grid <- expand.grid(alpha = c(0.1, 0.5, 0.9),
lambda = c(0.001, 0.01))
train(x = x, y = y, method = "glmnet", preProc =
c("center", "scale"), tuneGrid = grid)
```

Rastgele Arama

İyileştirme için **train** fonksiyonu çok çeşitli rastgele iyileştirme parametre kombinasyonları üretebilir.

`tuneLength` değerlendirilecek kombinasyonların toplam sayısını kontrol eder.

Rastgele aramayı kullanmak için:

```
trainControl(search = "random")
```

Alt Örnekleme (Subsampling)

Büyük bir sınıf dengesizliği söz konusu olduğunda **train** , model oluşturmadan önce sınıfları dengelemek için alt örnekleme yapar.

```
trainControl(sampling = "down")
```

Diğer değerler: "`up`", "`smote`" ya da "`rose`". Son ikisi başka paketlerin indirilmesini gerektirebilir.

Çeviri: İlkim Ecem Emre • ecem.emre@marmara.edu.tr • Çeviri Tarihi: 01/18