Estimate Model Accuracy

Veerasak Kritsanapraphan

10/4/2017

How To Estimate Model Accuracy in R Using The Caret Package

When you are building a predictive model, you need a way to evaluate the capability of the model on unseen data.

This is typically done by estimating accuracy using data that was not used to train the model such as a test set, or using cross validation. The caret package in R provides a number of methods to estimate the accuracy of a machines learning algorithm.

Estimating Model Accuracy

They are as follows and each will be described in turn:

- Data Split
- Bootstrap
- k-fold Cross Validation
- Repeated k-fold Cross Validation
- Leave One Out Cross Validation

Generally, I would recommend Repeated k-fold Cross Validation, but each method has its features and benefits, especially when the amount of data or space and time complexity are considered. Consider which approach best suits your problem

```
# load the libraries
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'default/Asia/
## Bangkok'
```

```
library(klaR)
```

```
## Loading required package: MASS
```

```
# load the iris dataset
data(iris)
```

Data Split

Data splitting involves partitioning the data into an explicit training dataset used to prepare the model and an unseen test dataset used to evaluate the models performance on unseen data.

It is useful when you have a very large dataset so that the test dataset can provide a meaningful estimation of performance, or for when you are using slow methods and need a quick approximation of performance.

The example below splits the iris dataset so that 70% is used for training a Naive Bayes model and 30% is used to evaluate the models performance.

```r
# define an 70%/30% train/test split of the dataset
split=0.70
trainIndex <- createDataPartition(iris$Species, p=split, list=FALSE)
data_train <- iris[ trainIndex,]
data_test <- iris[-trainIndex,]
# train a naive bayes model
model <- NaiveBayes(Species~., data=data_train)
# make predictions
x_test <- data_test[,1:4]
y_test <- data_test[,5]
predictions <- predict(model, x_test)
# summarize results
confusionMatrix(predictions$class, y_test)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         14         0
##   virginica       0          1        15
##
## Overall Statistics
##
##                Accuracy : 0.9778
##                  95% CI : (0.8823, 0.9994)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9667
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9333           1.0000
## Specificity                 1.0000            1.0000           0.9667
## Pos Pred Value              1.0000            1.0000           0.9375
## Neg Pred Value              1.0000            0.9677           1.0000
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3111           0.3333
## Detection Prevalence        0.3333            0.3111           0.3556
## Balanced Accuracy           1.0000            0.9667           0.9833
```

Bootstrap

Bootstrap resampling involves taking random samples from the dataset (with re-selection) against which to evaluate the model. In aggregate, the results provide an indication of the variance of the models performance. Typically, large number of resampling iterations are performed (thousands or tends of thousands).

The following example uses a bootstrap with 10 resamples to prepare a Naive Bayes model.

```r
# define training control
train_control <- trainControl(method="boot", number=100)
```

```r
# train the model
model <- train(Species~., data=iris, trControl=train_control, method="nb")
# summarize results
print(model)
```

```
## Naive Bayes
##
## 150 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Bootstrapped (100 reps)
## Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.9513358  0.9263969
##    TRUE      0.9545145  0.9312076
##
## 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.
```

k-fold Cross Validation

The k-fold cross validation method involves splitting the dataset into k-subsets. For each subset is held out while the model is trained on all other subsets. This process is completed until accuracy is determine for each instance in the dataset, and an overall accuracy estimate is provided.

It is a robust method for estimating accuracy, and the size of k and tune the amount of bias in the estimate, with popular values set to 3, 5, 7 and 10.

The following example uses 10-fold cross validation to estimate Naive Bayes on the iris dataset.

```r
# define training control
train_control <- trainControl(method="cv", number=10)
# fix the parameters of the algorithm
#grid <- expand.grid(.fL=c(0), .usekernel=c(FALSE))
# train the model
#model <- train(Species~., data=iris, trControl=train_control, method="nb", tuneGrid=grid)
model <- train(Species~., data=iris, trControl=train_control, method="nb")
# summarize results
print(model)
```

```
## Naive Bayes
##
## 150 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.9600000  0.94
##    TRUE      0.9533333  0.93
##
## 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 = FALSE
##  and adjust = 1.
```

Repeated k-fold Cross Validation

The process of splitting the data into k-folds can be repeated a number of times, this is called Repeated k-fold Cross Validation. The final model accuracy is taken as the mean from the number of repeats.

The following example uses 10-fold cross validation with 3 repeats to estimate Naive Bayes on the iris dataset.

```
# define training control
train_control <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model <- train(Species~., data=iris, trControl=train_control, method="nb")
# summarize results
print(model)
```

```
## Naive Bayes
##
## 150 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.9533333  0.9300000
##    TRUE      0.9555556  0.9333333
##
## 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.
```

Leave One Out Cross Validation

In Leave One Out Cross Validation (LOOCV), a data instance is left out and a model constructed on all other data instances in the training set. This is repeated for all data instances.

The following example demonstrates LOOCV to estimate Naive Bayes on the iris dataset.

```
# define training control
train_control <- trainControl(method="LOOCV")
# train the model
model <- train(Species~., data=iris, trControl=train_control, method="nb")
# summarize results
print(model)
```

```
## Naive Bayes
##
## 150 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 149, 149, 149, 149, 149, 149, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.9533333  0.93
##    TRUE      0.9600000  0.94
##
## 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.
```

Summary

You have discovered 5 different methods that you can use to estimate the accuracy of your model on unseen data.

Those methods were: Data Split, Bootstrap, k-fold Cross Validation, Repeated k-fold Cross Validation, and Leave One Out Cross Validation.