Regression and Classification using the caret package

The caret package (short for _C_lassification _A_nd _RE_gression _T_raining) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:

- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation

as well as other functionality.

Predictive models are also known as regression models when the output variable takes continuous values, whereas we'll refer to classification when the output variable takes class labels.

Model Training and Tuning

Machine learning algorithms typically work with two data sets namely a training set and a test set. This 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.

The function for partitioning the data using the caret package is `createDataPartition()`. If the outcome is a factor, the random sampling occurs within each class and should preserve the overall class distribution of the data. For example, to create a single 80% / 20% split of the iris data:

```
library(caret)
```

```
Loading required package: lattice
```

```
Loading required package: ggplot2
```

```
set.seed(3456)
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

irisTrain <- iris[ trainIndex,]
irisTest  <- iris[-trainIndex,]

dim(irisTrain)
```

```
[1] 120   5
```

```
table(irisTrain$Species)
```

```

    setosa versicolor  virginica
        40         40         40
```

```
dim(irisTest)
```

```
[1] 30  5
```

```
table(irisTest$Species)
```

```

    setosa versicolor  virginica
        10         10         10
```

The second step of the machine learning will be to specify the parameters of the model. This can be done using then `trainControl` function. The function `trainControl` generates parameters that further control how models are created. The main argument of this function is the method for resampling. The options are: * boot : bootstrap * boot632: 0.632 rule bootstrap * cv: K-fold cross-validation (K will be specified with the argument `numbers`) * LOOCV: leave-one-out cross validation, also known as jacknife. * LGOCV: leave-group-out cross validation, variant of LOOCV for hierarchical data. * repeatedcv: repeated cross valisation (The number of repeats is specified inthe argument `repeats`) * oob: out-of-bag estimation.

If for example we want to run algorithms using a 10-fold cross validation, we'll run the following command:

```
control <- trainControl(method="cv", number=10)
```

Let's apply several predictive models and compare their performance. The "accuracy" metric is going to be used to evaluate the models.

```
metric <- "Accuracy"
```

Model Building

The next step is to Build the models. Let's evaluate 5 different algorithms:

- Linear Discriminant Analysis (LDA)
- Classification and Regression Trees (CART)
- k-Nearest Neighbors (kNN)
- Support Vector Machines (SVM)
- Random Forest (RF)

This is a good mixture of simple linear (LDA), nonlinear (CART, kNN) and complex nonlinear methods (SVM, RF). We reset the random number seed before reach run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable.

```
library(e1071)
#Linear Discriminant Analysis
set.seed(7)
fit.lda <- train(Species~., data=irisTrain, method="lda", metric=metric, trControl=control)
```

```
Loading required package: MASS
```

```
fit.lda
```

```
Linear Discriminant Analysis

120 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
Resampling results:

  Accuracy  Kappa
  0.975     0.9625
```

```
#CART
set.seed(7)
fit.cart <- train(Species~., data=irisTrain, method="rpart", metric=metric, trControl=control)
```

```
Loading required package: rpart
```

```
fit.cart
```

```
CART

120 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
Resampling results across tuning parameters:

  cp     Accuracy   Kappa
  0.000  0.9166667  0.8750
  0.425  0.7583333  0.6375
  0.500  0.3333333  0.0000

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.
```

```
#kNN
set.seed(7)
fit.knn <- train(Species~., data=irisTrain, method="knn", metric=metric, trControl=control)
fit.knn
```

```
k-Nearest Neighbors

120 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.9583333  0.9375
  7  0.9583333  0.9375
  9  0.9333333  0.9000

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 7.
```

```
#SVM
set.seed(7)
fit.svm <- train(Species~., data=irisTrain, method="svmLinear2", metric=metric, trControl=control)
fit.svm
```

```
Support Vector Machines with Linear Kernel

120 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
Resampling results across tuning parameters:

  cost  Accuracy   Kappa
  0.25  0.9500000  0.9250
  0.50  0.9583333  0.9375
  1.00  0.9666667  0.9500

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was cost = 1.
```

```
#Random Forest
library(randomForest)
```

```
randomForest 4.6-12

Type rfNews() to see new features/changes/bug fixes.


Attaching package: 'randomForest'

The following object is masked from 'package:ggplot2':

    margin
```

```
set.seed(7)
fit.rf <- train(Species~., data=irisTrain, method="rf", metric=metric, trControl=control, prox=TRUE)
fit.rf
```

```
Random Forest

120 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 108, 108, 108, 108, 108, 108, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9333333  0.9000
  3     0.9416667  0.9125
  4     0.9416667  0.9125

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 3.
```

Select the best model

We can compare the accuracy of each model by first creating a list of the created models and using the summary function.

```r
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn, svm=fit.svm, rf=fit.rf))
summary(results)
```

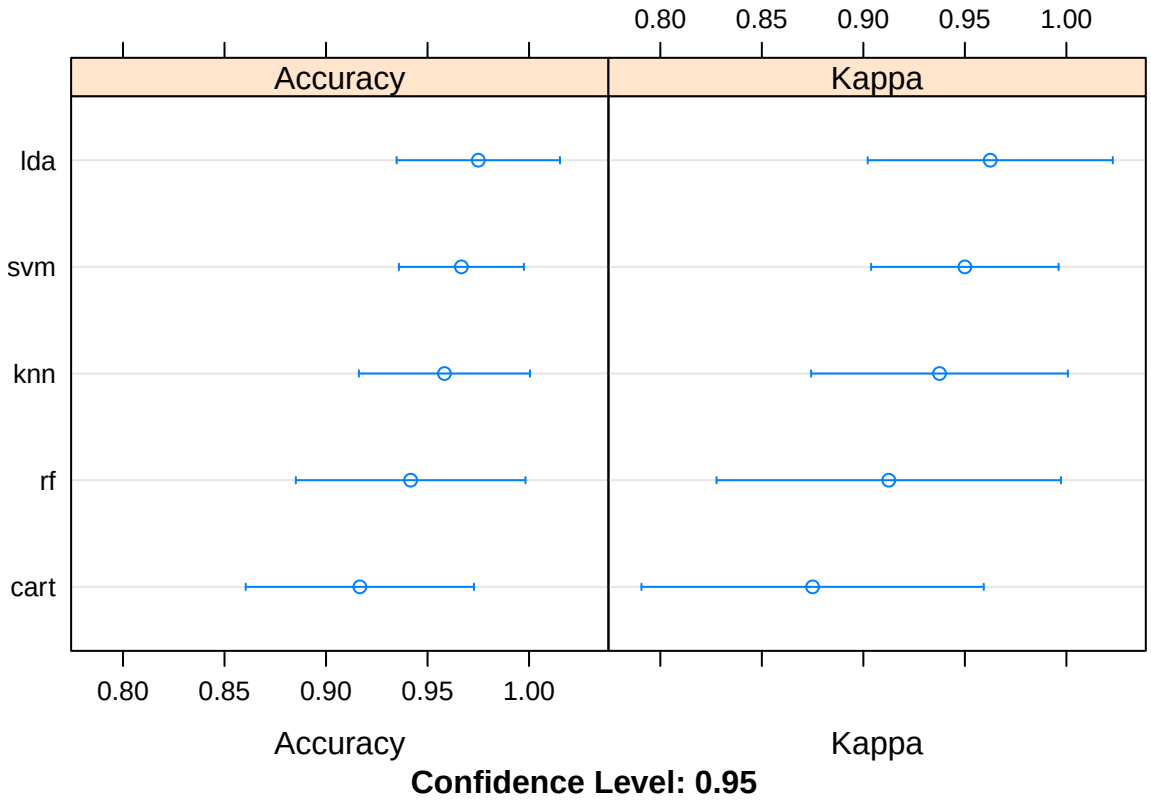
```
Call:
summary.resamples(object = results)

Models: lda, cart, knn, svm, rf
Number of resamples: 10

Accuracy
       Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
lda  0.8333  1.0000 1.0000 0.9750       1    1    0
cart 0.8333  0.8333 0.9167 0.9167       1    1    0
knn  0.8333  0.9167 1.0000 0.9583       1    1    0
svm  0.9167  0.9167 1.0000 0.9667       1    1    0
rf   0.8333  0.8542 1.0000 0.9417       1    1    0

Kappa
      Min. 1st Qu. Median   Mean 3rd Qu. Max. NA's
lda  0.750  1.0000  1.000 0.9625       1    1    0
cart 0.750  0.7500  0.875 0.8750       1    1    0
knn  0.750  0.8750  1.000 0.9375       1    1    0
svm  0.875  0.8750  1.000 0.9500       1    1    0
rf   0.750  0.7812  1.000 0.9125       1    1    0
```

```r
dotplot(results)
```



We observe that the best model is the `LDA` model.

Make predictions

The LDA was the most accurate model. Now we want to get an idea of the accuracy of the model on our validation set.

We'll predict the values for the validation set (`irisTest`) This will give us a final check on the accuracy of LDA model. We can run the LDA model directly on the validation set and summarize the results in a confusion matrix.

```
predictions <- predict(fit.lda, irisTest)
confusionMatrix(predictions, irisTest$Species)
```

```
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         0
  virginica       0          0        10

Overall Statistics

               Accuracy : 1
                 95% CI : (0.8843, 1)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 4.857e-15

                  Kappa : 1
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            1.0000           1.0000
Specificity                 1.0000            1.0000           1.0000
Pos Pred Value              1.0000            1.0000           1.0000
Neg Pred Value              1.0000            1.0000           1.0000
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.3333           0.3333
Detection Prevalence        0.3333            0.3333           0.3333
Balanced Accuracy           1.0000            1.0000           1.0000
```

We can see that the accuracy is 100%. It was a small validation dataset (20%), but this result is within our expected margin of 97% +/-4% suggesting we may have an accurate and a reliably accurate model.