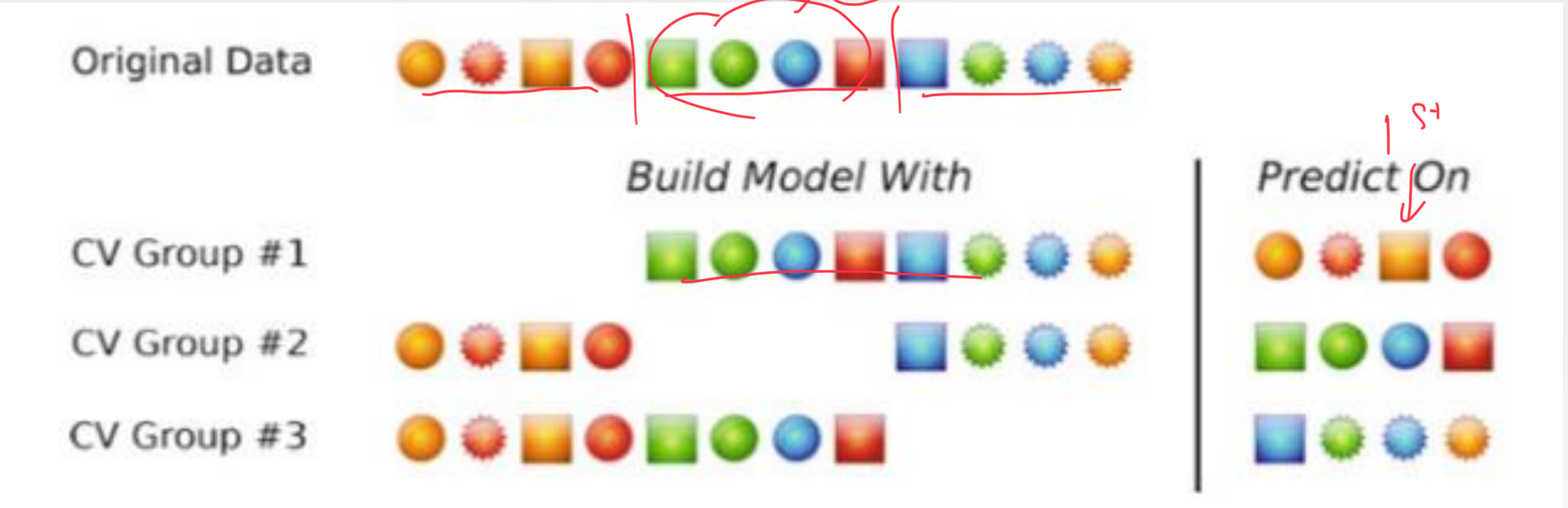
Resampling techniques

- Idea: a subset of samples are used to fit a model and the remaining samples are used to estimate the efficacy of the model. This process is repeated multiple times and the results are aggregated and summarized.
  - k-fold cross-validation
  - Generalized cross-validation
  - Repeated training/test splits
  - The bootstrap

k-fold cross-validation

- The choice of k is usually 5 or 10, but there is no formal rule.
- As *k* gets larger, the difference in size between the training set and the resampling subsets gets smaller. As this difference decreases, the bias of the technique becomes smaller (i.e., the bias is smaller for *k* = 10 than *k* = 5).
- From a practical viewpoint, larger values of k are more computationally burdensome.
- When *k* is the number of samples, we have leave-one-out cross-validation (LOOCV). k=n → n is small

A schematic of threefold (*k*=3) cross-validation



Generalized cross-validation

- For linear regression model, the generalized cross-validation (GCV) statistic is an approximation of the leave one-out error rate given by

$$\mathrm{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - df/n}\right)^2,$$

$y = \beta_0 + \beta_1 x + \varepsilon$

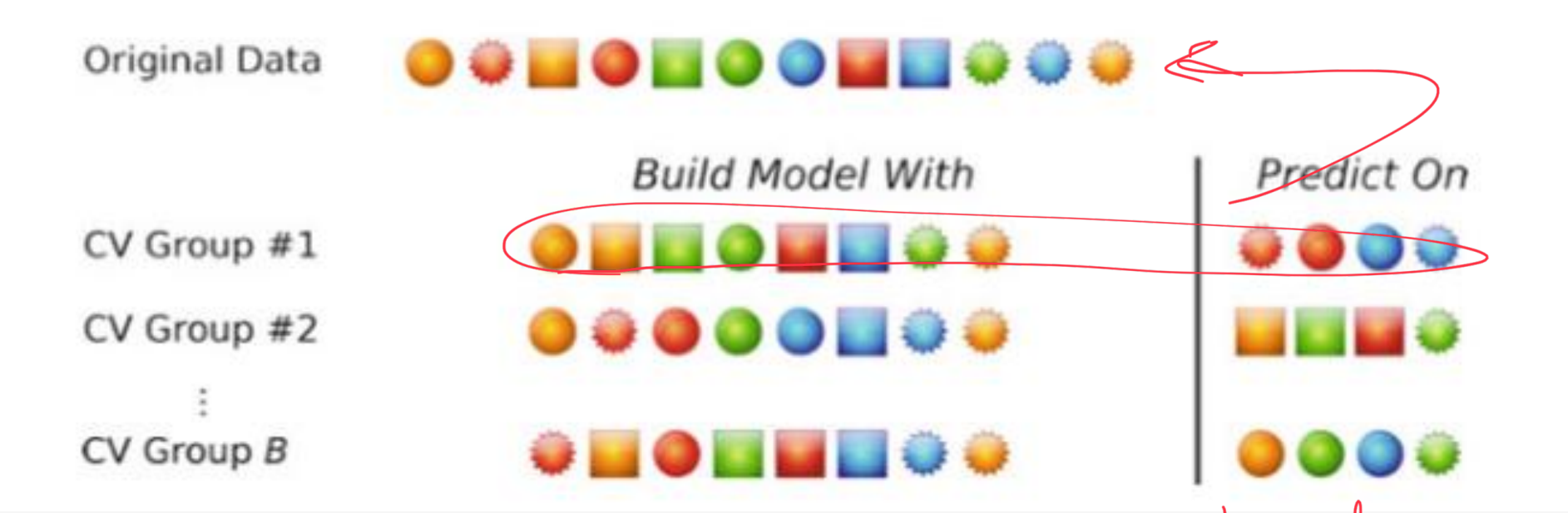
- where *n* is the sample size and *df* is the degrees of freedom of the model.

$df = 2.$

Repeated training/test splits

- Leave-group-out cross validation or Monte Carlo cross-validation
- It creates multiple splits of the data into modeling and prediction sets
- A good rule of thumb for the proportion of data into the training data is about 75–80%. Higher proportions are a good idea if the number of repetitions is large.
- A good rule of thumb for the number of repetitions is about 25. To get stable estimates of performance, it is suggested to choose a larger number of repetitions (say 50–200).

A schematic of B repeated training and test set partitions



B = 100

No repetition is allowed

The bootstrap (need data repetition)

- A bootstrap sample is a random sample of the data taken *with replacement*.
- The bootstrap sample is the same size as the original data set.
- Some samples will *be represented multiple times* in the bootstrap sample while others will not be selected at all. The samples not selected are usually referred to as the "out-of-bag" samples.
- In general, bootstrap error rates tend to have less uncertainty than *k*-fold cross-validation.

A schematic of bootstrap resampling

A simple classification example

```
library(AppliedPredictiveModeling)
data(twoClassData)
str(predictors)
str(classes)

## Split the data into training (80%) and test sets (20%)
set.seed(100)
x = cbind(predictors, classes)
inTrain <- createDataPartition(classes, p = .8)[[1]]
ClassTrain <- x[inTrain, ]
ClassTest  <- x[-inTrain, ]
```

training data

test data

'sample' as well since 'class 1' and 'class 2' are roughly balanced.

A simple classification example

```
#Hyperparameter Estimation For The Gaussian Radial Basis Kernel
library(kernlab)
set.seed(231)
#frac: fraction of data to use for estimation.
#By default a quarter of the data is used to estimate
#the range of the sigma hyperparameter.
sigDist <- sigest(classes~ ., data = ClassTrain, frac = 1)
sigDist
svmTuneGrid <- data.frame(sigma = as.vector(sigDist)[1], C = 2^(-2:7))
svmTuneGrid
```

sigest for SVM with Gaussian radial basis kernel

```
> sigDist
      90%       50%       10%
0.1385608 0.4950346 4.8479681
> svmTuneGrid <- data.frame(sigma = as.vector(sigDist)[1], C = 2^(-2:7))
> svmTuneGrid
       sigma      C
1  0.1385608   0.25
2  0.1385608   0.50
3  0.1385608   1.00
4  0.1385608   2.00
5  0.1385608   4.00
6  0.1385608   8.00
7  0.1385608  16.00
8  0.1385608  32.00
9  0.1385608  64.00
10 0.1385608 128.00
```

Two tuning parameters : sigma C

https://www.rdocumentation.org/packages/kernlab/versions/0.9-29/topics/sigest

SVM with svmTuneGrid

```
set.seed(1056)
svmFit <- train(classes~ .,
                data = ClassTrain,
                method = "svmRadial",
                preProc = c("center", "scale"),
                tuneGrid = svmTuneGrid,
                trControl = trainControl(method = "repeatedcv",
                                         repeats = 5,
                                         classProbs = TRUE))
## classProbs = TRUE was added since the text was written

## Print the results
svmFit
```

two tuning parameters: σ and C

```
> svmFit
Support Vector Machines with Radial Basis Function Kernel

167 samples
  2 predictor
  2 classes: 'Class1', 'Class2'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.7657353  0.5247871
    0.50  0.7657353  0.5254151
    1.00  0.7680147  0.5293976
    2.00  0.7644853  0.5224310
    4.00  0.7704412  0.5347356
    8.00  0.7703676  0.5345100
   16.00  0.7765441  0.5469482
   32.00  0.7849265  0.5630442
   64.00  0.7908088  0.5745254
  128.00  0.7955147  0.5844457

Tuning parameter 'sigma' was held constant at a value of 0.09728643
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.09728643 and C = 128.
```

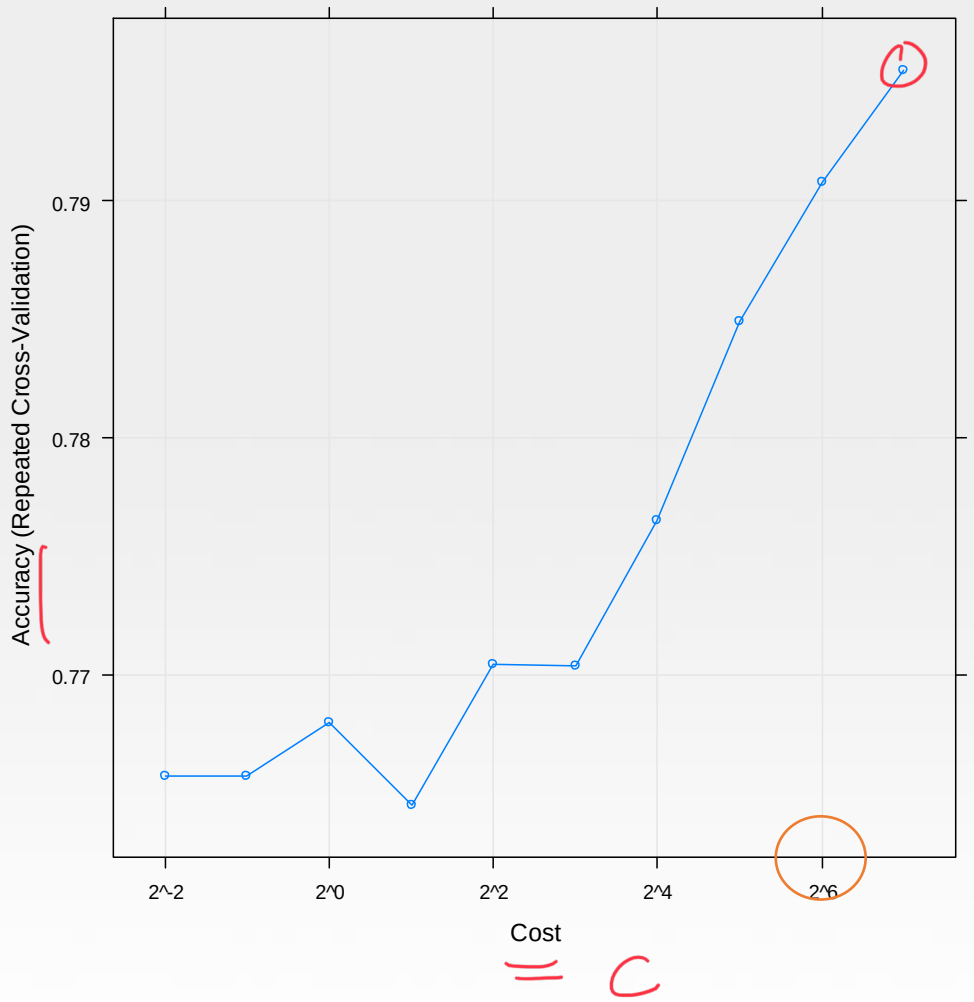
largest accuracy.

SVM with svmTuneGrid

```
## A line plot of the average performance.
## The 'scales' argument is actually an argument
## to xyplot that converts the x-axis to log-2 units.

plot(svmFit, scales = list(x = list(log = 2)))

## Test set predictions

predictedClasses <- predict(svmFit, ClassTest)
str(predictedClasses)

## Use the "type" option to get class probabilities
predictedProbs <- predict(svmFit, newdata = ClassTest, type = "prob")
head(predictedProbs)
```

model

new data


Accuracy (Repeated Cross-Validation)
0.79
0.78
0.77
2^-2
2^0
2^2
2^4
2^6
Cost

C = 128

= C

```
> predictedProbs <- predict(svmFit, newdata = ClassTest, type = "prob")
> head(predictedProbs)
     Class1    Class2
1 0.7509123 0.2490877
2 0.6848781 0.3151219
3 0.1484117 0.8515883
4 0.7860613 0.2139387
5 0.5083886 0.4916114
6 0.3983306 0.6016694
```

threshold = 0.5

```
> predictedProbs <- predict(svmFit, newdata = ClassTest)
> head(predictedProbs)
[1] Class1 Class1 Class2 Class1 Class1 Class2
Levels: Class1 Class2
```

Fit the same model using different resampling methods.

```
#10-fold cross validation

set.seed(1056)
svmFit10CV <- train(classes ~ .,
                    data = ClassTrain,
                    method = "svmRadial",
                    preProc = c("center", "scale"),
                    tuneGrid = svmTuneGrid,
                    trControl = trainControl(method = "cv", number = 10))
svmFit10CV
```

10-fold

```
> svmFit10CV
Support Vector Machines with Radial Basis Function Kernel

167 samples
  2 predictor
  2 classes: 'Class1', 'Class2'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.7724265  0.5386097
    0.50  0.7724265  0.5386097
    1.00  0.7724265  0.5386097
    2.00  0.7599265  0.5136097
    4.00  0.7599265  0.5136097
    8.00  0.7665441  0.5257069
   16.00  0.7665441  0.5265500
   32.00  0.7841912  0.5608819
   64.00  0.7959559  0.5846522
  128.00  0.7959559  0.5853314

Tuning parameter 'sigma' was held constant at a value of 0.09728643
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.09728643 and C = 64.
```

Fit the same model using different resampling methods.

```
#LOOCV

set.seed(1056)
svmFitLOO <- train(classes ~ .,
                   data = ClassTrain,
                   method = "svmRadial",
                   preProc = c("center", "scale"),
                   tuneGrid = svmTuneGrid,
                   trControl = trainControl(method = "LOOCV"))
svmFitLOO
```

k = n

```
> svmFitLOO
Support Vector Machines with Radial Basis Function Kernel

167 samples
  2 predictor
  2 classes: 'Class1', 'Class2'

Pre-processing: centered (2), scaled (2)
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 166, 166, 166, 166, 166, 166, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.7664671  0.5275299
    0.50  0.7544910  0.5033007
    1.00  0.7544910  0.5033007
    2.00  0.7544910  0.5040921
    4.00  0.7544910  0.5040921
    8.00  0.7544910  0.5040921
   16.00  0.7544910  0.5040921
   32.00  0.7664671  0.5282828
   64.00  0.7664671  0.5282828
  128.00  0.7604790  0.5165726

Tuning parameter 'sigma' was held constant at a value of 0.1385608
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1385608 and C = 0.25.
>
```

Fit the same model using different resampling methods.

```
#Repeated Training/leave-group-out cross validation

set.seed(1056)
svmFitLGO <- train(classes ~ .,
                   data = ClassTrain,
                   method = "svmRadial",
                   preProc = c("center", "scale"),
                   tuneGrid = svmTuneGrid,
                   trControl = trainControl(method = "LGOCV",
                                            number = 50,
                                            p = .8))
svmFitLGO
```

```
> svmFitLOO
Support Vector Machines with Radial Basis Function Kernel

167 samples
  2 predictor
  2 classes: 'Class1', 'Class2'

Pre-processing: centered (2), scaled (2)
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 166, 166, 166, 166, 166, 166, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.7664671  0.5252569
    0.50  0.7664671  0.5252569
    1.00  0.7664671  0.5252569
    2.00  0.7604790  0.5134741
    4.00  0.7604790  0.5134741
    8.00  0.7724551  0.5385398
   16.00  0.7844311  0.5614240
   32.00  0.7784431  0.5503238
   64.00  0.8083832  0.6101547
  128.00  0.8083832  0.6107793

Tuning parameter 'sigma' was held constant at a value of 0.09728643
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.09728643 and C = 64.
```

Fit the same model using different resampling methods.

```
#The bootstrap

set.seed(1056)
svmFitBoot <- train(classes ~ .,
                    data = ClassTrain,
                    method = "svmRadial",
                    preProc = c("center", "scale"),
                    tuneGrid = svmTuneGrid,
                    trControl = trainControl(method = "boot", number = 50))
svmFitBoot
```

↳ repetitions.

```
> svmFitBoot
Support Vector Machines with Radial Basis Function Kernel

167 samples
  2 predictor
  2 classes: 'Class1', 'Class2'

Pre-processing: centered (2), scaled (2)
Resampling: Bootstrapped (50 reps)
Summary of sample sizes: 167, 167, 167, 167, 167, 167, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.7442448  0.4808603
    0.50  0.7415823  0.4747385
    1.00  0.7385618  0.4684565
    2.00  0.7394617  0.4699234
    4.00  0.7405735  0.4718349
    8.00  0.7430877  0.4772625
   16.00  0.7461736  0.4830559
   32.00  0.7480467  0.4864420
   64.00  0.7486119  0.4872764
  128.00  0.7457998  0.4820563

Tuning parameter 'sigma' was held constant at a value of 0.1385608
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1385608 and C = 64.
```

Fit the same model using different resampling methods.

```
#The bootstrap632

set.seed(1056)
svmFitBoot632 <- train(classes ~ .,
                       data = ClassTrain,
                         method = "svmRadial",
                         preProc = c("center", "scale"),
                         tuneGrid = svmTuneGrid,
                         trControl = trainControl(method = "boot632",
                                                  number = 50))
svmFitBoot632
```

```
> svmFitBoot
Support Vector Machines with Radial Basis Function Kernel

167 samples
  2 predictor
  2 classes: 'Class1', 'Class2'

Pre-processing: centered (2), scaled (2)
Resampling: Bootstrapped (50 reps)
Summary of sample sizes: 167, 167, 167, 167, 167, 167, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.7744663  0.5420548
    0.50  0.7750058  0.5428191
    1.00  0.7745126  0.5422242
    2.00  0.7754992  0.5442220
    4.00  0.7750657  0.5439890
    8.00  0.7743676  0.5426775
   16.00  0.7734254  0.5411820
   32.00  0.7737925  0.5428141
   64.00  0.7735546  0.5425626
  128.00  0.7749954  0.5446829

Tuning parameter 'sigma' was held constant at a value of 0.09728643
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.09728643 and C = 2.
```

Between model comparisons

- As an illustration, we contrast *the SVM model* with a *logistic regression model*.
- We can compare the models with respect to the values of Accuracy and Kappa.
- One sample-t test or ANOVA test with Bonferroni correction can be conducted for the differences of accuracy and/or Kappa
- We then make a decision based on the resulting p-values from a statistical test.

```
set.seed(1056)
svmFit <- train(classes~ .,
                data = ClassTrain,
                method = "svmRadial",
                preProc = c("center", "scale"),
                tuneGrid = svmTuneGrid,
                trControl = trainControl(method = "repeatedcv",
                                         repeats = 5,
                                         classProbs = TRUE))
svmFit

set.seed(1056)
glmProfile <- train(classes ~ .,
                    data = ClassTrain,
                    method = "glm",
                    trControl = trainControl(method = "repeatedcv",
                                             repeats = 5))
glmProfile
```

svmFit

```
> svmFit
Support Vector Machines with Radial Basis Function Kernel

167 samples
  2 predictor
  2 classes: 'Class1', 'Class2'

Pre-processing: centered (2), scaled (2)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.7657353  0.5247871
    0.50  0.7657353  0.5254151
    1.00  0.7680147  0.5293976
    2.00  0.7644853  0.5224310
    4.00  0.7704412  0.5347356
    8.00  0.7703676  0.5345100
   16.00  0.7765441  0.5469482
   32.00  0.7849265  0.5630442
   64.00  0.7908088  0.5745254
  128.00  0.7955147  0.5844457

Tuning parameter 'sigma' was held constant at a value of 0.09728643
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.09728643 and C = 128.
```

glmProfile

```
> glmProfile
Generalized Linear Model

167 samples
  2 predictor
  2 classes: 'Class1', 'Class2'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 150, 150, 150, 150, 150, 150, ...
Resampling results:

  Accuracy   Kappa
  0.7580882  0.5131848
```

Between model comparisons

```
resamp <- resamples(list(SVM = svmFit, Logistic = glmProfile))
summary(resamp)

modelDifferences <- diff(resamp)
summary(modelDifferences)

## The actual paired t-test:
modelDifferences$statistics$Accuracy
```

name of model

The results of resamples

```
> summary(resamp)

Call:
summary.resamples(object = resamp)

Models: SVM, Logistic
Number of resamples: 50

Accuracy
             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
SVM      0.5294118 0.7647059 0.7647059 0.7955147 0.8750000 0.9411765    0
Logistic 0.5294118 0.7058824 0.7647059 0.7580882 0.8235294 0.9411765    0

Kappa
               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
SVM      0.04225352 0.5159960 0.5277778 0.5844457 0.7500000 0.8827586    0
Logistic 0.02857143 0.3992709 0.5211268 0.5131848 0.6433566 0.8811189    0
```

five number summary

Summary of model differences

```
> summary(modelDifferences)

Call:
summary.diff.resamples(object = modelDifferences)

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

Accuracy
         SVM       Logistic
SVM                0.03743
Logistic 0.0003704

Kappa
         SVM       Logistic
SVM                0.07126
Logistic 0.000761
```

0.03743 < α = 0.05 indicating that the two models are different.

0.07126 > α = 0.05 indicating that the two model are not statistically significantly different.

One Sample t-test for model comparison

```
> ## The actual paired t-test:
> modelDifferences$statistics$Accuracy
$SVM.diff.Logistic

        One Sample t-test

data:  x
t = 3.8251, df = 49, p-value = 0.0003704
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.01776372 0.05708923
sample estimates:
 mean of x
0.03742647
```

- What conclusion can be drawn from the p-value of 0.0003704?

$\alpha = 0.01$ The two model perform differently!