A major risk in classification: overfitting
Assume we have a small data set
We fit a model that separates red and blue
When more data becomes available, we see that the model is poor.
A simpler model might have worked better.
A predictor always works best on the data set on which it was trained!
Solution: divide data into training and test sets
Solution: divide data into training and test sets

- Training data
- Best model for training data
Solution: divide data into training and test sets

Evaluate model on test data
Frequently used approach: 
\textit{k-fold cross-validation}

- Divide data into \( k \) equal parts
- Use \( k-1 \) parts as training set, 1 as test set
- Repeat \( k \) times, so each part has been used once as test set
Also: Leave-one-out cross-validation

- Fit model on \( n-1 \) data points
- Evaluate on remaining data point
- Repeat \( n \) times, so each point has been left out once
And: Repeated random sub-sampling validation

- Randomly split data into training and test data sets
- Train model on training set, evaluate on test set
- Repeat multiple times, average over result
# We assume our data are stored in data table called `data`.
Random sub-sampling in R

# We assume our data are stored in data table called `data`.

# Fraction of data used for training purposes (here: 40%)
train_fraction <- 0.4
# We assume our data are stored in data table called `data`.

# Fraction of data used for training purposes (here: 40%)
train_fraction <- 0.4

# Number of observations in training set
train_size <- floor(train_fraction * nrow(data))
# We assume our data are stored in data table called `data`.

# Fraction of data used for training purposes (here: 40%)
train_fraction <- 0.4

# Number of observations in training set
train_size <- floor(train_fraction * nrow(data))

# Indices of observations to be used for training
train_indices <- sample(1:nrow(data), size = train_size)
Random sub-sampling in R

# We assume our data are stored in data table called `data`.

# Fraction of data used for training purposes (here: 40%)
train_fraction <- 0.4

# Number of observations in training set
train_size <- floor(train_fraction * nrow(data))

# Indices of observations to be used for training
train_indices <- sample(1:nrow(data), size = train_size)

# Extract training and test data
train_data <- data[train_indices, ] # get training data
test_data <- data[-train_indices, ] # get test data