Model Evaluation - Data splitting

source("~/MCD/MCD/m.R")  
  
library(mlbench)  
data(PimaIndiansDiabetes)  
  
print("Loaded")

## [1] "Loaded"

## Notes

* Given a fixed amount of data, the modeler must decide how to "spend" their data points to accomodate a fixed amount of data.
* Resampling can be considered to be an alternative approach to evaluating a model on a single test set.
* The first decision when modeling is to decide which samples will be used to evaluate performance.
* Ideally the model should be evaluated on samples that were not used to build or fine-tune the model, so they provide an unbiased sense of model effectivness.
* However when the number of samples is not large, a strong case can be made that a test set should be avoided because every sample may be needed for model building.
* Additionally the size of the test set may not have sufficient power or precision to make reasonable judgements.
* Sometime non-random methods for splitting the data are somestimes appropiate.

EG. In medical trials model may be tested using certain patients (Eg same clininical site or disease stage) then tested on different samples to understand how the model generalizes.

Also in same cases we are interested in seeing how the model generalizes into a new "Search space" of interest.

In most cases the simplest way to split data is a random sample.

However when one class has a disproportionately small frequency, there is a chance the distribution will be substantiallly different between the training and test sets.

To account for the outcome when splitting the data, stratified random sampling applies random sampling within subgroups.

When the outcome is a number, a similar strategy can be used. The numeric values are broken into similar groups (e.g. low, medium, high) and the randomization is executed within these groups.

Alternatively the data can be split on the basis of the predictor values. An example being maximum dissimlarity sampling. (Described in APM - Section 4.4)

## Computing

### Stratified Data Split

**Load the data**

library(AppliedPredictiveModeling)  
data(twoClassData)  
  
twoClassData <- as\_tibble(predictors)  
twoClassData$classes <- classes  
  
p <- m\_data(twoClassData) %>% m\_response("classes") %>% m\_stratified\_split(p = 0.8) %>% m\_pre\_process()

**Frequencies for both classes will be the same**

Training data

m\_get\_level\_statistics(m\_get\_training\_data(p))

## # A tibble: 3 × 5  
## .task .variable .level count freqs  
## <chr> <chr> <chr> <int> <dbl>  
## 1 m\_get\_training\_data(p) classes Class1 89 0.5329341  
## 2 m\_get\_training\_data(p) classes Class2 78 0.4670659  
## 3 m\_get\_training\_data(p) classes .NA 0 0.0000000

Validation data

m\_get\_level\_statistics(m\_get\_validation\_data(p))

## # A tibble: 3 × 5  
## .task .variable .level count freqs  
## <chr> <chr> <chr> <int> <dbl>  
## 1 m\_get\_validation\_data(p) classes Class1 22 0.5365854  
## 2 m\_get\_validation\_data(p) classes Class2 19 0.4634146  
## 3 m\_get\_validation\_data(p) classes .NA 0 0.0000000

### Maximum dissimilarity sampling

TBD: Would use the caret::maxdissm function.