Question → input data → features (extracting variables/characteristics) → algorithm → parameters(estimate) → evaluation

**procedures**

1. define error rate (type I/type II)
2. split data into:
   * training, testing, validation (optional)
3. pick features from the training set
   * use cross-validation
4. pick prediction function (model) on the training set
   * use cross-validation
5. if no validation set
   * apply **1 time** to test set

Draw up Receiver Operating Characteristic Curve after

Cross Validation

* **procedures**
  1. split training set into sub-training/test sets
  2. build model on sub-training set
  3. evaluate on sub-test set
  4. repeat and average estimated errors
* **result**
  1. we are able to fit/test various different models with different variables included to the find the best one on the cross-validated test sets
  2. we are able to test out different types of prediction algorithms to use and pick the best performing one
  3. we are able to choose the parameters in prediction function and estimate their values
  4. ***Note****: original test set completely untouched, so when final prediction algorithm is applied, the result will be an unbiased measurement of the****out of sample accuracy****of the model*
* **approaches**
  1. random subsampling
  2. K-fold
  3. leave one out
* **CARET core functionality**
  + preprocessing/cleaning data → preProcess()
  + cross validation/data splitting → createDataPartition(), createResample(), createTimeSlices()
  + train algorithms on training data and apply to test sets → train(), predict()
  + model comparison (evaluate the accuracy of model on new data) → confusionMatrix()
* machine learning algorithms in caret package
  + linear discriminant analysis
  + regression
  + naive Bayes
  + support vector machines
  + classification and regression trees
  + random forests
  + boosting

library(ISLR); library(ggplot2); library(caret)

set.seed(96)

training <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!"))

testing <- read.csv("pml-testing.csv")

#featurePlot(x=training, y=training$classe, plot="pairs")

#featurePlot(x=training[,c(11:15)], y = training$classe,plot="pairs")

#Remove first columns of username and timestamps --> 154 columns

trainfilt <- training[,-c(1:6)]

# Impute and standardize

preObj <- preProcess(training[,-154],method="knnImpute")

trainfilt <- predict(preObj,training[,-154])$classe

#Create dummy variables

#dummies <- dummyVars(classe ~ new\_window, data = trainfilt)

#traindum <- predict(dummies, newdata = trainfilt)

#Remove only NA columns --> not needed

#trainfilt <- trainfilt[,colSums(is.na(trainfilt))<nrow(trainfilt)]

# Remove Near zero variance predictors --> 95 columns

nzv <- nearZeroVar(trainfilt)

trainfilt <- trainfilt[, -nzv]

#dim(trainnzv)

#Remove highly correlated variables --> matrix not symmetric?

highlyCor <- findCorrelation(trainfilt[,-95], cutoff = .75)

trainfilt <- trainfilt[,-na.omit(highlyCor)]

# create preprocess object

#preProcValues <- preProcess(trainfilt[,-79], method = c("center", "scale"))

preProc <- preProcess(trainfilt[,-79],method="pca")

# calculate PCs for training data

trainPC <- predict(preProc,trainfilt[,-79])

# run model on outcome and principle components

modelFit <- train(trainfilt$classe ~ .,method="glm",data=trainPC)

# calculate PCs for test data

testPC <- predict(preProc,testing[,-160])

# compare results

confusionMatrix(testing$classe,predict(modelFit,testPC))