Doing machine learning in R

Alastair Rushworth

EdinbR, R User group meeting

Predictive analytics → **Machine learning** → **Artificial intelligence (AI)**

"mathematical rues for predicting unknown or future events"

"[giving] computers the ability to learn without being explicitly programmed"

Arthur Samuel, 1959

"the theory and development of computer systems able to perform tasks normally requiring human intelligence"

Oxford English dictionary

The CRAN task view

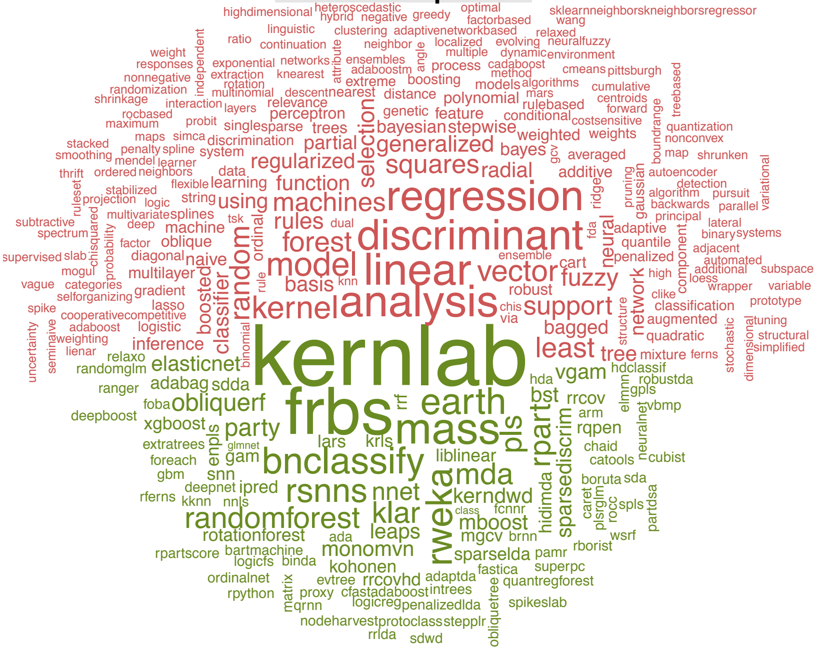
CRAN Task View: Machine Learning & Statistical Learning

Maintainer: Torsten Hothorn
Contact: Torsten.Hothorn at R-project.org
Version: 2017-05-30
URL: https://CRAN.R-project.org/view=MachineLearning

Several add-on packages implement ideas and methods developed at the borderline between computer science and statistics - this field of research is usu

- *Neural Networks and Deep Learning* : Single-hidden-layer neural network are implemented in package nnet (shipped with base R). Package RSNN recurrent neural networks. Packages implementing deep learning flavours of neural networks include darch (restricted Boltzmann machine, deep be restricted Boltzmann machine, deep belief network) and h2o (feed-forward neural network, deep autoencoders).
- *Recursive Partitioning* : Tree-structured models for regression, classification and survival analysis, following the ideas in the CART book, are impl provides an interface to this implementation, including the J4.8-variant of C4.5 and M5. The Cubist package fits rule-based models (similar to trees these.
  Two recursive partitioning algorithms with unbiased variable selection and statistical stopping criterion are implemented in package party. Functio models. Extensible tools for visualizing binary trees and node distributions of the response are available in package party as well.
  Tree-structured varying coefficient models are implemented in package vcrpart.
  For problems with binary input variables the package LogicReg implements logic regression. Graphical tools for the visualization of trees are avail Trees for modelling longitudinal data by means of random effects is offered by package REEMtree. Partitioning of mixture models is performed by Computational infrastructure for representing trees and unified methods for predition and visualization is implemented in partykit. This infrastructu censoring in addition to right-censoring.
- *Random Forests* : The reference implementation of the random forest algorithm for regression and classification is available in package randomFor variant for response variables measured at arbitrary scales based on conditional inference trees is implemented in package party. randomForestSRC numeric response on exploratory variables via a random forest approach. For binary data, LogicForest is a forest of logic regression trees (package implementations of random forests. Reinforcement Learning Trees, featuring splits in variables which will be important down the tree, are impleme
- *Regularized and Shrinkage Methods* : Regression models with some constraint on the parameter estimates can be fitted with the lasso2 and lars pac such as group MCP and group SCAD. The L1 regularization path for generalized linear models and Cox models can be obtained from functions av glmnet. The penalized package provides an alternative implementation of lasso (L1) and ridge (L2) penalized regression models (both GLM and C a specified shrinkage path and to determine the appropriate extent of shrinkage. Semiparametric additive hazards models under lasso penalties are LASSO penalty to produce sparse solutions is implemented in package penalizedLDA. The shrunken centroids classifier and utilities for gene expr in penalized models (SCAD or L1 penalties) is implemented in package penalizedSVM. Various forms of penalized discriminant analysis are impl MCP regression penalties using a coordinate descent algorithm. High-throughput ridge regression (i.e., penalization with many predictor variables) under non-Gaussian and heteroscedastic errors is estimated by hdm, inference on low-dimensional components of Lasso regression and of estimat
- *Boosting and Gradient Descent* : Various forms of gradient boosting are implemented in package gbm (tree-based functional gradient descent boos

Techniques


highdimensional
heteroscedastic
hybrid
negative
greedy
optimal
factorbased
sklearnneighborskneighborsregressor
wang
linguistic
clustering
adaptivenetworkbased
relaxed
ratio
continuation
neighbor
localized
evolving
neuralfuzzy
weight
independent
attribute
angle
multiple
dynamic
environment
responses
exponential
networks
ensembles
process
cadaboost
cmeans
pittsburgh
nonnegative
extraction
knearest
adaboostm
method
treebased
randomization
rotation
extreme
boosting
models
algorithms
cumulative
shrinkage
multinomial
descent
nearest
distance
mars
centroids
interaction
relevance
polynomial
rulebased
forward
rocbased
layers
perceptron
genetic
feature
conditional
boundrange
maximum
probit
single
sparse
trees
bayesian
stepwise
costsensitive
quantization
stacked
maps
simca
discrimination
weighted
weights
nonconvex
smoothing
penalty
spline
system
partial
selection
generalized
bayes
gcv
averaged
map
shrunken
thrift
mendel
learner
regularized
squares
radial
ordered
neighbors
data
additive
autoencoder
variational
stabilized
flexible
learning
function
pruning
gaussian
detection
ruleset
projection
logic
string
using
machines
regression
ridge
algorithm
pursuit
backwards
parallel
chisquared
multivariate
splines
principal
subtractive
deep
tsk
ordinal
rules
dual
discriminant
fda
adaptive
lateral
spectrum
machine
neural
component
binary
systems
probability
factor
oblique
random
forest
quantile
adjacent
supervised
slab
diagonal
naive
model
linear
ensemble
penalized
automated
mogul
multilayer
cart
additional
subspace
vague
categories
rule
basis
knn
vector
fuzzy
network
high
loess
wrapper
variable
selforganizing
gradient
boosted
classifier
kernel
analysis
robust
clike
prototype
spike
lasso
chis
support
classification
cooperative
competitive
via
structure
augmented
tuning
uncertainty
seminaive
adaboost
logistic
bagged
stochastic
dimensional
weighting
binomial
quadratic
structural
lienar
inference
least
tree
mixture
ferns
simplified
relaxo
kernlab
randomglm
elasticnet
hdclassif
hda
vgam
robustda
ranger
adabag
sdda
elmnn
gpls
foba
obliquerf
frbs
rrf
earth
bst
rrcov
vbmp
deepboost
neuralnet
arm
xgboost
party
mass
pls
rpart
rqpen
extratrees
enpls
glmnet
lars
krls
sparsediscrim
chaid
foreach
gam
bnclassify
liblinear
catools
cubist
gbm
snn
rweka
mda
hidimda
boruta
sda
deepnet
ipred
rsnns
nnet
kerndwd
caret
plsrglm
rocc
spls
partdsa
rferns
kknn
nnls
class
fcnnr
randomforest
klar
mboost
rotationforest
ada
leaps
mgcv
brnn
wsrf
bartmachine
monomvn
rborist
rpartscore
logicfs
binda
sparselda
pamr
kohonen
superpc
ordinalnet
evtree
rrcovhd
adaptda
fastica
matrix
quantregforest
rpython
proxy
cfastadaboost
intrees
qrnn
logicreg
penalizedlda
obliquetree
spikeslab
nodeharvest
protoclass
stepplr
rrlda
sdwd

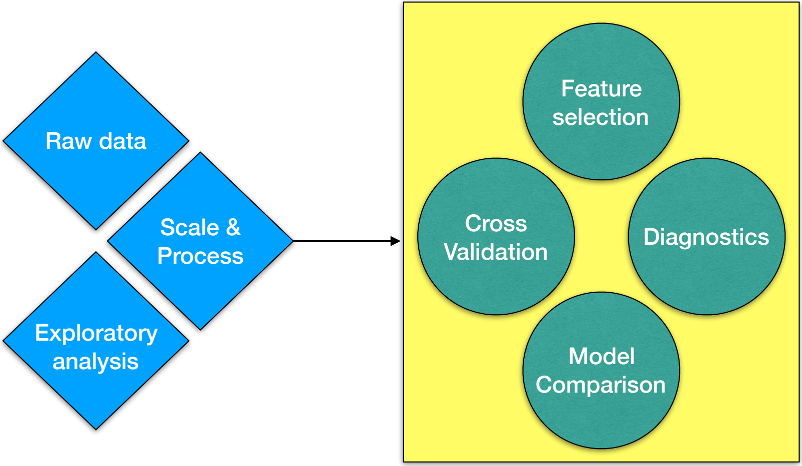
Challenges & opportunities for the user

Hurray: Statistics & machine learning are growing

- Huge array of models now available for nearly every conceiveable task
- Software is democratised with cutting-edge implementations usually OS
- Parallel computation baked in and easier than ever
  - Don't need to have PhD in CompSci to understand
  - Don't even need personal cluster, just fire up an AWS instance

But:

- Software often developed for specific communities and tasks and may have learning curve to implement
- Models are increasingly complicated to tune, with many different hyperparameters
- Overwhelming choice - sensible to choose some candidate models and compare performance


Raw data
Scale & Process
Exploratory analysis
Feature selection
Cross Validation
Diagnostics
Model Comparison

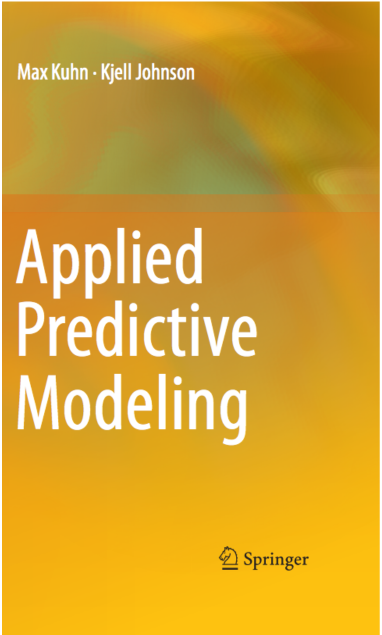
There's an R (meta-)package for that

- `mlr`: Machine Learning in R
- `h2o`: R scripting for H2O, the open source math engine for big data
- `SuperLearner`: Implements the super learner prediction method
- `caret`: **C**lassification **a**nd **Re**gression **T**raining.

`caret`

- A suite of functions for simplifying fitting of predictive models in R
- Uniform interface for tuning hyperparametsrs
- Support for automatic cross-validation, prediction and measuring accuracy

Bedtime reading



Journal of Statistical Software

November 2008, Volume 28, Issue 5. *http://www.jstatsoft.org/*

Building Predictive Models in R Using the caret Package

Max Kuhn
Pfizer Global R&D

Abstract

The **caret** package, short for classification and regression training, contains numerous tools for developing predictive models using the rich set of models available in R. The package focuses on simplifying model training and tuning across a wide variety of modeling techniques. It also includes methods for pre-processing training data, calculating variable importance, and model visualizations. An example from computational chemistry is used to illustrate the functionality on a real data set and to benchmark the benefits of parallel processing with several types of models.

Keywords: model building, tuning parameters, parallel processing, R, **NetWorkSpaces**.

1. Introduction

The use of complex classification and regression models is becoming more and more commonplace in science, finance and a myriad of other domains (Ayres 2007). The R language (R Development Core Team 2008) has a rich set of modeling functions for both classification and regression, so many in fact, that it is becoming increasingly more difficult to keep track of the syntactical nuances of each function. The **caret** package, short for classification and regression training, was built with several goals in mind:

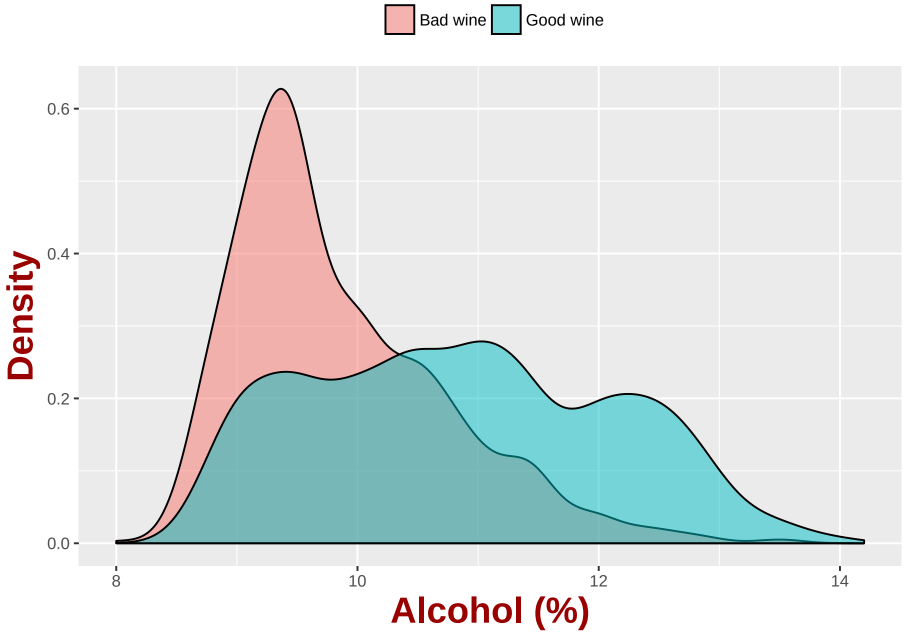
- to eliminate syntactical differences between many of the functions for building and predicting models,
- to develop a set of semi-automated, reasonable approaches for optimizing the values of the tuning parameters for many of these models and
- create a package that can easily be extended to parallel processing systems.

Wine data

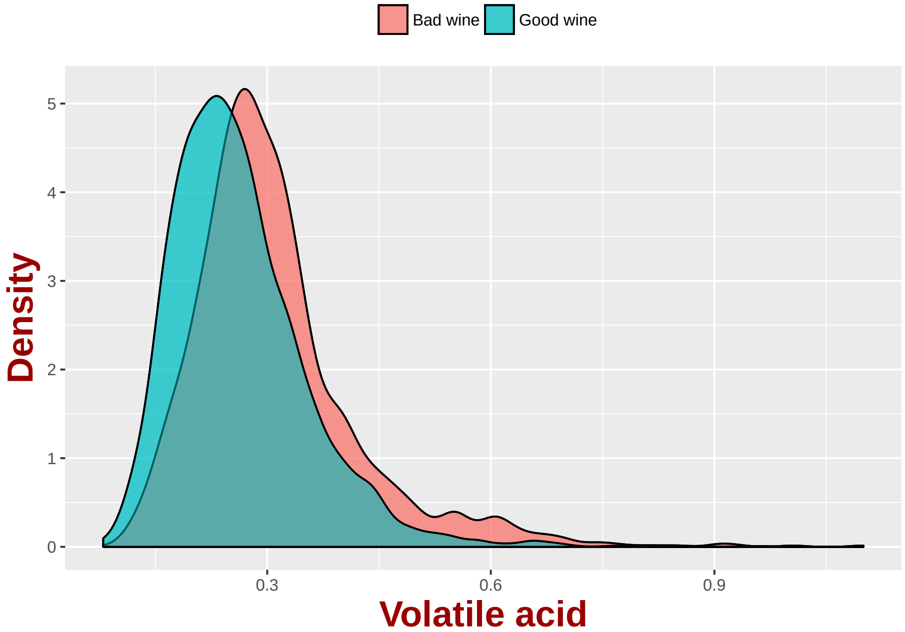
Fixed Acidity	Volatile Acidity	Citric Acid	Residual Sugar	Chlorides	Density	pH	Sulphates	Alcohol	Quality
7	0.27	0.36	20.7	0.045	1.001	3	0.45	8.8	6
6.3	0.3	0.34	1.6	0.049	0.994	3.3	0.49	9.5	6
8.1	0.28	0.4	6.9	0.05	0.9951	3.26	0.44	10.1	6
7.2	0.23	0.32	8.5	0.058	0.9956	3.19	0.4	9.9	6
7.2	0.23	0.32	8.5	0.058	0.9956	3.19	0.4	9.9	6
8.1	0.28	0.4	6.9	0.05	0.9951	3.26	0.44	10.1	6

- Source: `https://archive.ics.uci.edu/ml/datasets/wine+quality`
- Subjective quality ratings for 5000 wines scores 1 - 10
- For each wine, chemical properties are measured
- For simplicity, binarise quality score into **good** (> 5) and **bad** (≤ 5)
- **Which properties are predictive of a good score?**

People like the booze...



...and are less keen on vinegar



Logistic regression

- Old as the hills, simple to interpret
- Core function: `stats:::glm`
- Hyperparameters: 0

Regularised logistic regression

- Extends GLMs, still linear, more robust to overfitting and collinearity
- Core function: `glmnet:::glmnet`
- Hyperparameters: 1 (2 if using elastic net)

Gradient boosted regression model

- Non-linear, tree based
- Core function: `gbm:::gbm`
- Hyperparameters: 4

```r
library(caret)
library(dplyr)
```

Randomly select 80% of the rows for the training set

```r
wine.part  <- createDataPartition(y = wine$quality, p = 0.8, list = F)
```

Define the training set....

```r
X.train    <- wine[ wine.part,]  %>% dplyr::select(-quality, -good.wine)
y.train    <- wine$good.wine[wine.part]
```

...and test sets

```r
X.test     <- wine[-wine.part,]  %>% dplyr::select(-quality, -good.wine)
y.test     <- wine$good.wine[-wine.part]
```

Logistic regression

The workhorse of the `caret` package is the `train` function:

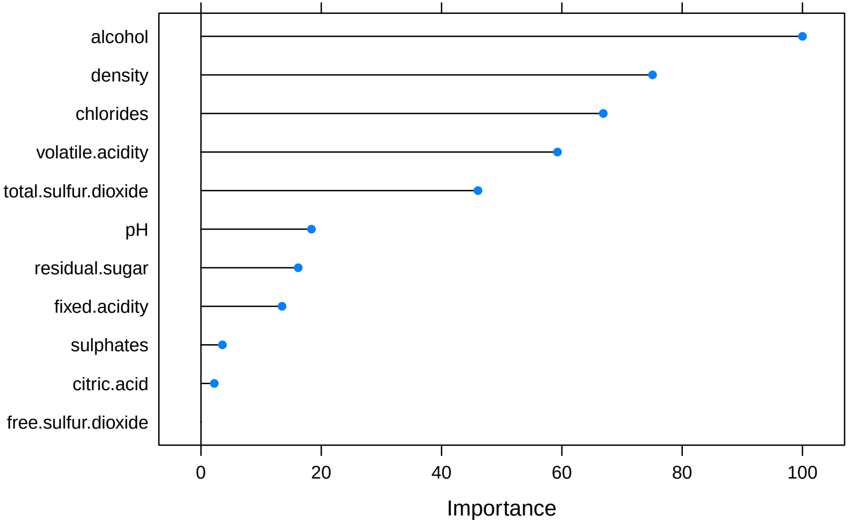
```
wine_logistic <- train(y          = as.factor(y.train),
                       x          = X.train,
                       metric     = "Accuracy",
                       method     = "glm",
                       family     = "binomial",
                       preProcess = c("scale", "center"))
```

Options:

- `y`, `x` are the training data
- `metric = 'Accuracy'`: proportion correctly classified
- `method = 'glm'`: use the `glm` function to fit this
- `family = 'binomial'`: when `y` is two-level factor, logistic regression

Logistic regression

```
plot(varImp(wine_logistic))
```



Logistic regression accuracy

```r
confusionMatrix(wine_logistic)
```

```
...
##
##             Reference
## Prediction  Bad_wine Good_wine
##   Bad_wine      16.7       8.1
##   Good_wine     16.8      58.3
##
##  Accuracy (average) : 0.7501
...
```

```r
confusionMatrix(predict(wine_logistic, newdata = X.test), y.test)
```

```
...
##             Reference
## Prediction  Bad_wine Good_wine
##   Bad_wine       178        86
##   Good_wine      150       565
##
##                Accuracy : 0.7589
...
```

Regularised logistic regression

Minimise λ over a grid of values: `tuneGrid`:

```
tuneGrid   <- expand.grid(alpha = 0, lambda = 10^-(5:2))
```

This a list of arguments controlling cross-validation

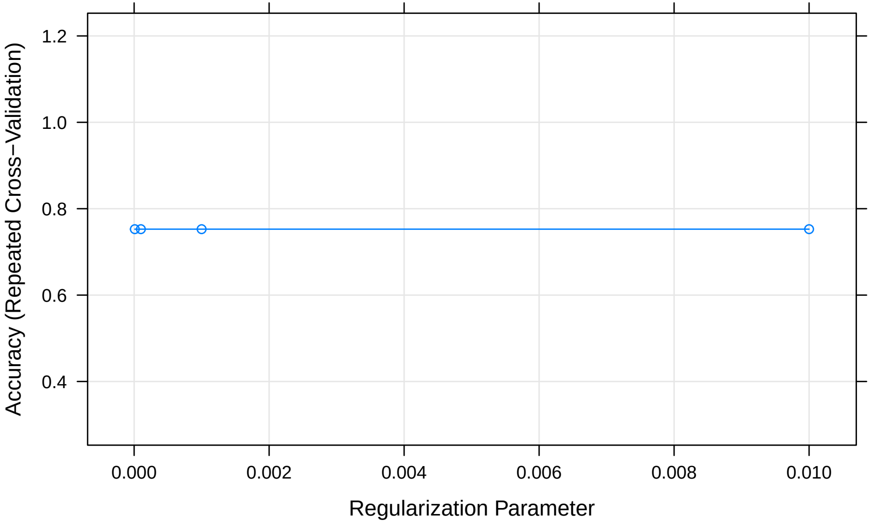
```
objControl <- trainControl(method = 'repeatedcv', number = 10, repeats = 5)
```

Now train regularised model:

```
wine_regular <- train(y          = factor(y.train),
                      x          = X.train,
                      metric     = "Accuracy",
                      method     = "glmnet",
                      family     = "binomial",
                      trControl  = objControl,
                      tuneGrid   = tuneGrid,
                      preProcess = c("scale", "center"))
```

Regularised logistic regression

```
plot(wine_regular)
```



Regularised logistic regression

```
confusionMatrix(wine_regular)

...
##
##             Reference
## Prediction  Bad_wine Good_wine
##   Bad_wine      15.5       6.8
##   Good_wine     18.0      59.7
##
##  Accuracy (average) : 0.7526
...
```

```
confusionMatrix(predict(wine_regular, newdata = X.test), y.test)

...
##             Reference
## Prediction  Bad_wine Good_wine
##   Bad_wine       152        65
##   Good_wine      176       586
##
##                Accuracy : 0.7538
...
```

Gradient boosted regression model

Four different hyperparameters in `tuneGrid`:

```
tuneGrid <- expand.grid(interaction.depth = c(1, 2, 3),
                        n.trees           = c(20, 50, 100),
                        shrinkage         = c(0.1, 0.2, 0.3),
                        n.minobsinnode    = 20)
```

Now train model, using same cross-validation scheme as before

```
library(doParallel)
registerDoParallel(3, cores = 3)

wine_gbm <- train(y         = factor(y.train),
                  x         = X.train,
                  metric    = "Accuracy",
                  method    = "gbm",
                  trControl = objControl,
                  tuneGrid  = tuneGrid,
                  verbose   = F)
```

Gradient boosted regression model

```
confusionMatrix(wine_gbm)

...
##
##             Reference
## Prediction  Bad_wine Good_wine
##   Bad_wine      19.8       8.4
##   Good_wine     13.7      58.2
##
##  Accuracy (average) : 0.7797
...
```

```
confusionMatrix(predict(wine_gbm, newdata = X.test), y.test)

...
##             Reference
## Prediction  Bad_wine Good_wine
##   Bad_wine       234        63
##   Good_wine       94       588
##
##                 Accuracy : 0.8396
...
```

caret models & extensions

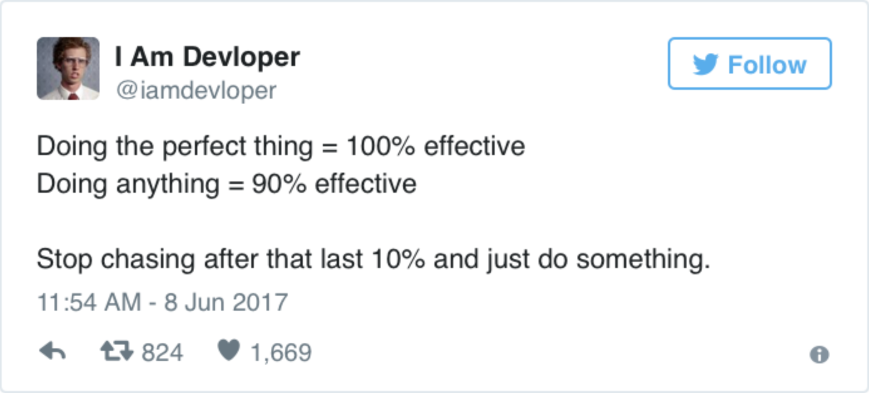
Huge variety of models, including

- Lots more in the package -
- SVM, deep learning, PLS, random forest, mixture models, additive models, Gaussian process.... **Full list:** (https://topepo.github.io/caret/available-models.html)
- Possible to add your own models (https://topepo.github.io/caret/using-your-own-model-in-train.ht

Extensions

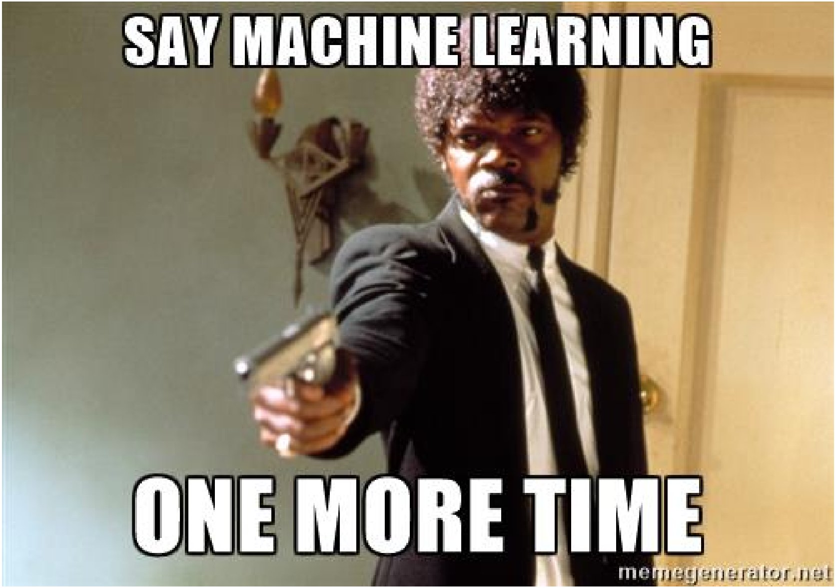
- `caretEnsemble`: Ensembling and stacking for `caret` models
- `fscaret`: Automated feature selection for `caret` models

Who should use caret?



Some external libraries with R support

External libraries	**R package**	**Description**
H_2O.ai	h2o	Deep learning, general purpose machine learning
dmlc mxnet R	mxnet	Deep learning
K	keras	Deep learning
TensorFlow	tensorflow	Deep learning


SAY MACHINE LEARNING
ONE MORE TIME
memegenerator.net