CLASSIFICATION AND REGRESSION
With Caret Package
Cheat Sheet - 2016

Ziad Al Bkhetan

Basics

Caret package solves complex regression and classification problems, by utilizing a number of R packages, but it loads them when needed. Covered aspects:

1. Data Partitioning.
2. Features Importance.
3. Pre-Processing
4. Features Selection.
5. Models with Built-In Feature Selection
6. Classification And Regression Training.
7. Model Performance Evaluation.
8. Data Visualization.

URL: https://github.com/topepo/caret/

Data Partitioning

Create series of training and testing partitions:
`createDataPartition(): y, times, p, list, groups`
To create bootstrap samples:
`createResample(): y, times, list`
To split the data into k group:
`createFolds(): y, k, lis, returnTrain`
`createMultiFolds() : y, k, times`
Create corss-validatation sample information for time series data:
`createTimeSlices() : y, initialWindow, horizon, fixedWindow, skip`

Features Importance

Calculate scaled features importance
`varImp(model, scale = FALSE)`
scale : normalize the results from 0 to 100
Get the area under ROC curve for each predictor
`filterVarImp(x, y) x: the features, y: the target`

Pre-Processing

Create Dummy Variables

Generate a set of dummy variables from one or more factors.
`dummyVars(formula, data)`

Finding Correlated Predictor

To flag some correlated predictors for removal
`Cor_mat<-cor(filteredDescr)`
`findCorrelation(Cor_mat, cutoff)`

Linear Dependencies

Using QR decomposition of a matrix
`findLinearCombos(data)`

PreProcess Function

To apply some operations on predictors.
`preProcess(data, method)`
`Method` Options: BoxCox, YeoJohnson, expoTrans, center, scale, pca, ica, range, knnImpute, bagImpute, medianImpute and spatialSign

Features Selection

Recursive Feature Elimination

A backwards selection method
`rfeControl(): functions, method, repeats, verbose`
`rfe() : x, y, sizes, rfeControl`

Univariate Filters

Pre-screen the predictors using simple univariate statistical methods.
`sbf(predictors, outcome, sbfControl)`
`sbf(formula, data, sbfControl)`

Genetic Algorithms

Applying genetic algorithm to find the optimal features set.
`gafs() : x, y, iters , gafsControl, method`

Simulated Annealing

Applying small random changes then check if it improve the solution to accept them.
`safs() : x, y, iters, safsControl`

Classification And Regression Training

Train function tunes parameters for classification or regression models, evaluate the effect of tuned parameters on the performance, find the best model across the parameters, and estimate the model performance.

```
train() : x, y, form, data, method,
preProcess, weights, metric,
maximize, trControl, tuneGrid,
tuneLength, subset, na.action,
contrasts, tuneGrid, tuneLength.
```

Classification and Regression Models

216 models can be used with train function:

Classification: *88* models
Regression: *50* models
Dual Use: *78* models

The full list exists at this link:
http://topepo.github.io/caret/modelList.html

Tuning Training Parameters

To Generate the parameters that control models creation.

`Method`: takes one of these values:
boot, boot632, cv, repeatedcv, LOOCV, LGOCV, none , oob, adaptive_cv, adaptive_boot, adaptive_LGOCV

Model Performance Evaluation

Evaluate Test Sets Binary Classification

```
testPred = predict(model, date)
postResample(testPred, data$Class)
sensitivity(testPred, data$Class)
specificity(testPred, data$Class)
posPredValue(testPred, data$Class)
negPredValue(testPred, data$Class)
```

To summerize the results of classification model, and calculate most of the known performance measurements :
`confusionMatrix(testPred, data$Class)`

Evaluate Test Sets Multiple Classification

Get the negative of the multinomial log-likelihood

```
test_results = predict(model, testing,
type = "prob")
test_results$obs = testing$Class
mnLogLoss(test_results, lev =
levels(test_results$obs))
```

To calculate overall accuracy and Kappa, negative multinomial log loss , sensitivity, specificity, the area under the ROC curve:

```
test_results$pred = predict(model,
testing)
multiClassSummary(test_results, lev =
levels(test_results$obs))
```

Evaluating Class Probabilities

To Evaluate probabilities thresholds that can capture a certain percentage of hits
`lift(formula, data)`
for probability calibration
`calibration(formula, data)`

Models with Built-In Feature Selection

ada, bagEarth, bagFDA, bstLs, bstSm, C5.0, C5.0Cost, C5.0Rules, C5.0Tree, cforest, ctree, ctree2, cubist, earth, enet, evtree, extraTrees, fda, gamboost, gbm, gcvEarth, glmnet, glmStepAIC, J48, JRip, lars, lars2, lasso, LMT, LogitBoost, M5, M5Rules, nodeHarvest, oblique.tree, OneR, ORFlog, ORFpls, ORFridge, ORFsvm, pam, parRF, PART, penalized, PenalizedLDA, qrf, relaxo, rf, rFerns, rpart, rpart2, rpartCost, RRF, RRFglobal, smda, sparseLDA

FULL EXAMPLE

#Partitioning And RFE Features Selection:

```
indxTrain=createDataPartition(y =
pimaTrain$test, p = 0.75)
pimaTrain=pima[indxTrain$Resample1,]
pimaTest=pima[-indxTrain$Resample1,]
```

```
x=pimaTrain[, 1:8]; subsets=c(1:7)
ctrl=rfeControl(functions = lmFuncs,
method="repeatedcv", repeats=5, verbose=FALSE)
mProfile=rfe(x, pimaTrain$test, sizes=subsets,
rfeControl=ctrl)
predictors(mProfile)
```

#Fitting Stochastic Gradient Boosting model with cross-validation using all features :

```
fitControl=trainControl(method="cv",
number=10)
model=train(test ~ ., data=pimaTrain,
method= "gbm", trControl= fitControl)
```

#Evaluate the model using 3 functions :

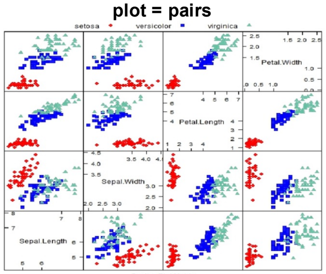
```
testPred=predict(model, newdata=pimaTest)
postResample(testPred, pimaTest$test)
posPredValue(testPred, pimaTest$test)
confusionMatrix(testPred, pimaTest$test)
```

Data Visualization

featurePlot() : x, y, plot, auto.key, scales, adjust, pch, layout, type, span

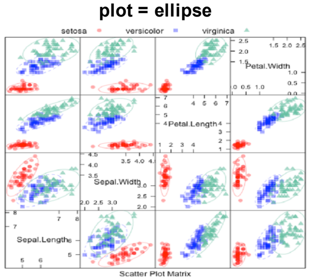
Scatterplot Matrix

plot = pairs



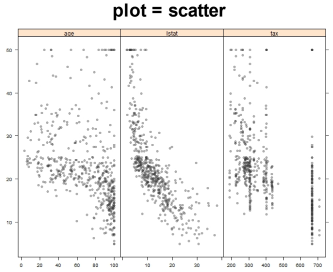
Scatterplot Matrix with Ellipses

plot = ellipse

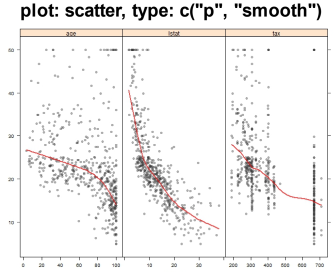


Scatter Plots

plot = scatter

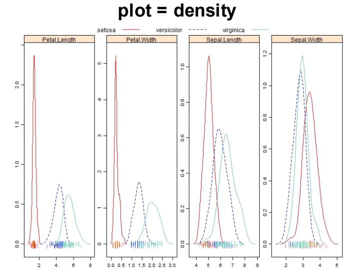


plot: scatter, type: c("p", "smooth")



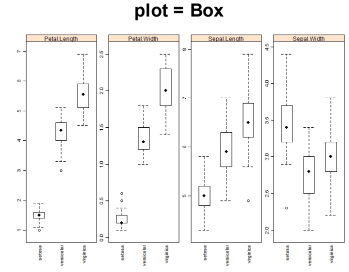
Overlayed Density Plots

plot = density



Box Plots

plot = Box



Final Project For The Course: Advances In Data Mining, Prof.Przemysław Biecek, Warsaw University Of Technology