Variable Importance Using The `caret` Package

Max Kuhn
max.kuhn@pfizer.com

June 9, 2011

1 Variable Importance

Variable importance evaluation functions can be separated into two groups: those that use the model information and those that do not. The advantage of using a model–based approach is that is more closely tied to the model performance and that it *may* be able to incorporate the correlation structure between the predictors into the importance calculation. Regardless of how the importance is calculated:

- For most classification models, each predictor will have a separate variable importance for each class (the exceptions are classification trees, bagged trees and boosted trees).
- All measures of importance are scaled to have a maximum value of 100, unless the `scale` argument of `varImp.train` is set to `FALSE`.

1.1 Model Specific Metrics

The following methods for estimating the contribution of each variable to the model are available

- **Linear Models:** the absolute value of the t–statistic for each model parameter is used.
- **Random Forest:** from the R package: "For each tree, the prediction accuracy on the out-of-bag portion of the data is recorded. Then the same is done after permuting each predictor variable. The difference between the two accuracies are then averaged over all trees, and normalized by the standard error. For regression, the MSE is computed on the out-of-bag data for each tree, and then the same computed after permuting a variable. The differences are averaged and normalized by the standard error. If the standard error is equal to 0 for a variable, the division is not done."

- **Partial Least Squares**: the variable importance measure here is based on weighted sums of the absolute regression coefficients. The weights are a function of the reduction of the sums of squares across the number of PLS components and are computed separately for each outcome. Therefore, the contribution of the coefficients are weighted proportionally to the reduction in the sums of squares.

- **Recursive Partitioning**: The reduction in the loss function (e.g. mean squared error) attributed to each variable at each split is tabulated and the sum is returned. Also, since there may be candidate variables that are important but are not used in a split, the top competing variables are also tabulated at each split. This can be turned off using the `maxcompete` argument in `rpart.control`. This method does not currently provide class–specific measures of importance when the response is a factor.

- **Bagged Trees**: The same methodology as a single tree is applied to all bootstrapped trees and the total importance is returned

- **Boosted Trees**: This method uses the same approach as a single tree, but sums the importances over each boosting iteration (see the `gbm` package vignette).

- **Multivariate Adaptive Regression Splines**: MARS models include a backwards elimination feature selection routine that looks at reductions in the generalized cross-validation (GCV) estimate of error. The `varImp` function tracks the changes in model statistics, such as the GCV, for each predictor and accumulates the reduction in the statistic when each predictor's feature is added to the model. This total reduction is used as the variable importance measure. If a predictor was never used in any MARS basis function, it has an importance value of zero. There are three statistics that can be used to estimate variable importance in MARS models. Using `varImp(object, value = "gcv")` tracks the reduction in the generalized cross-validation statistic as terms are added. However, there are some cases when terms are retained in the model that result in an increase in GCV. Negative variable importance values for MARS are set to zero. Terms with non–zero importance that were not included in the final, pruned model are also listed as zero. Alternatively, using `varImp(object, value = "rss")` monitors the change in the residual sums of squares (RSS) as terms are added, which will never be negative. Also, the option `varImp(object, value = "nsubsets")` returns the number of times that each variable is involved in a subset (in the final, pruned model). Prior to June 2008, `varImp` used an internal function to estimate importance for MARS models. Currently, it is a wrapper around the `evimp` function in the `earth` package.

- **Nearest shrunken centroids**: The difference between the class centroids and the overall centroid is used to measure the variable influence (see `pamr.predict`). The larger the difference between the class centroid and the overall center of the data, the larger the separation between the classes. The training set predictions must be supplied when an object of class `pamrtrained` is given to `varImp`.

- **Cubist**: The Cubist output contains variable usage statistics. It gives the percentage of times where each variable was used in a condition and/or a linear model. Note that this output

will probably be inconsistent with the rules shown in the output from `summary.cubist`. At each split of the tree, Cubist saves a linear model (after feature selection) that is allowed to have terms for each variable used in the current split or any split above it. Quinlan (1992) discusses a smoothing algorithm where each model prediction is a linear combination of the parent and child model along the tree. As such, the final prediction is a function of all the linear models from the initial node to the terminal node. The percentages shown in the Cubist output reflects all the models involved in prediction (as opposed to the terminal models shown in the output). The variable importance used here is a linear combination of the usage in the rule conditions and the model.

1.2 Model Independent Metrics

If there is no model–specific way to estimate importance (or the argument `useModel = FALSE` is used in `varImp`) the importance of each predictor is evaluated individually using a "filter" approach.

For classification, ROC curve analysis is conducted on each predictor. For two class problems, a series of cutoffs is applied to the predictor data to predict the class. The sensitivity and specificity are computed for each cutoff and the ROC curve is computed. The trapezoidal rule is used to compute the area under the ROC curve. This area is used as the measure of variable importance. For multi–class outcomes, the problem is decomposed into all pair–wise problems and the area under the curve is calculated for each class pair (i.e. class 1 vs. class 2, class 2 vs. class 3 etc.). For a specific class, the maximum area under the curve across the relevant pair–wise AUC's is used as the variable importance measure.

For regression, the relationship between each predictor and the outcome is evaluated. An argument, `nonpara`, is used to pick the model fitting technique. When `nonpara = FALSE`, a linear model is fit and the absolute value of the t–value for the slope of the predictor is used. Otherwise, a loess smoother is fit between the outcome and the predictor. The R^2 statistic is calculated for this model against the intercept only null model. This number is returned as a relative measure of variable importance.

1.3 Example

As an example, the multidrug resistance reversal (MDRR) agent data is used. First, we filter and standardize the predictors:

```
> options(width = 80)
> data(mdrr)
> set.seed(100)
> nzvColumns <- nearZeroVar(mdrrDescr)
> mdrrDescr <- mdrrDescr[, -nzvColumns]
```

```
> preProcVals <- apply(mdrrDescr, 2, processData)
> mdrrDescr <- applyProcessing(mdrrDescr, preProcVals)
```

Classification models were fit using random forests and partial least squares. The variable importance measures were calculated for these models and using the ROC method described above.

```
> ctrl <- trainControl(verboseIter = FALSE)
> rfFit <- randomForest(mdrrDescr, mdrrClass, ntree = 2000, importance = TRUE)
> plsFit <- train(mdrrDescr, mdrrClass, "pls", tuneGrid = data.frame(.ncomp = 2 *
+     (1:10)), trControl = ctrl)
> modelFree <- filterVarImp(mdrrDescr, mdrrClass)
> allImp1 <- data.frame(randomForest = varImp(rfFit)[, 1], PLS = varImp(plsFit$finalModel
+     1], ROC = modelFree[, 1])
> allImp1$note <- "All Descriptors"
```

A scatter plot matrices of the unscaled importances with a loess smooth curve are given in Figure . The top panel shows the results when the models use all predictors as inputs. Here, there is no 1:1 relationship between between methods. However, in the case of random forests, if a set of predictors are highly correlated, the selection of which predictor is used in a split is essentially random. This dilutes the importance of each of the correlated descriptors and may make the variable importance measures less helpful.

To evaluate the effect of between–predictor correlations, descriptors with absolute pair–wise correlations above 0.90 are removed. In this case, the three methods show better agreement (the bottom panel of).

```
> corrColumns <- findCorrelation(cor(mdrrDescr))
> mdrrDescr <- mdrrDescr[, -corrColumns]
> rfFit2 <- randomForest(mdrrDescr, mdrrClass, ntree = 2000, importance = TRUE)
> plsFit2 <- train(mdrrDescr, mdrrClass, "pls", tuneGrid = data.frame(.ncomp = 2 *
+     (1:10)), trControl = ctrl)
> modelFree <- filterVarImp(mdrrDescr, mdrrClass)
> allImp2 <- data.frame(randomForest = varImp(rfFit2)[, 1], PLS = varImp(plsFit2$finalMod
> allImp2$note <- "|Corr| < 0.90"
> allPred <- splom(~allImp1[, 1:3], type = c("p", "smooth"), main = "Variable Importance:
> filteredPred <- splom(~allImp2[, 1:3], type = c("p", "smooth"), main = "Variable Import
```

Figure 2 demonstrates the `plot` method for object of class `varImp`. In each case, needle plots are produced for the top predictors, based on their importance values. For classification models with more than 2 classes, the predictors are ordered by their average importance. With two classes, only one importance is plotted.

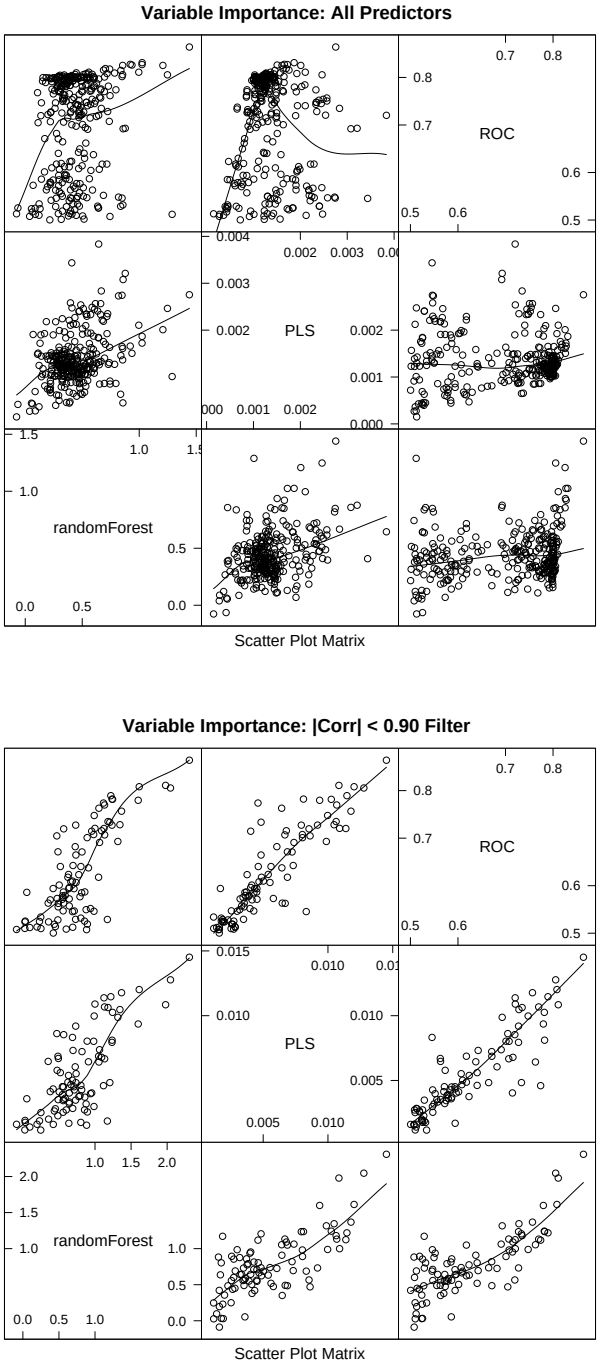

Figure 1: Relationships between different variable importance measures. The top panel shows a lack of linear correlation when all of the predictors are included in the model while the lower panel show the same techniques after a filter on between–predictor correlations.

2 References

Quinlan. Learning with continuous classes. Proceedings of the 5th Australian Joint Conference On Artificial Intelligence (1992) pp. 343–348

3 Session Information

- R version 2.11.1 (2010-05-31), x86_64-apple-darwin9.8.0
- Locale: en_US/en_US/en_US/C/en_US/en_US
- Base packages: base, datasets, graphics, grDevices, grid, methods, splines, stats, stats4, tools, utils
- Other packages: akima 0.5-4, caret 4.91, class 7.3-2, cluster 1.12.3, e1071 1.5-24, earth 2.4-5, ellipse 0.3-5, gam 1.03, gbm 1.6-3.1, Hmisc 3.8-3, ipred 0.8-8, kernlab 0.9-12, klaR 0.6-4, lattice 0.18-8, leaps 2.9, MASS 7.3-6, mlbench 2.1-0, modeltools 0.2-17, mvtnorm 0.9-95, nnet 7.3-1, plotrix 3.0-5, pls 2.1-0, plyr 1.2.1, proxy 0.4-6, randomForest 4.5-36, reshape 0.8.3, rpart 3.1-46, survival 2.35-8
- Loaded via a namespace (and not attached): coin 1.0-17, colorspace 1.0-1, party 0.9-99992, rJava 0.8-7, RWeka 0.4-4, RWekajars 3.7.2-1

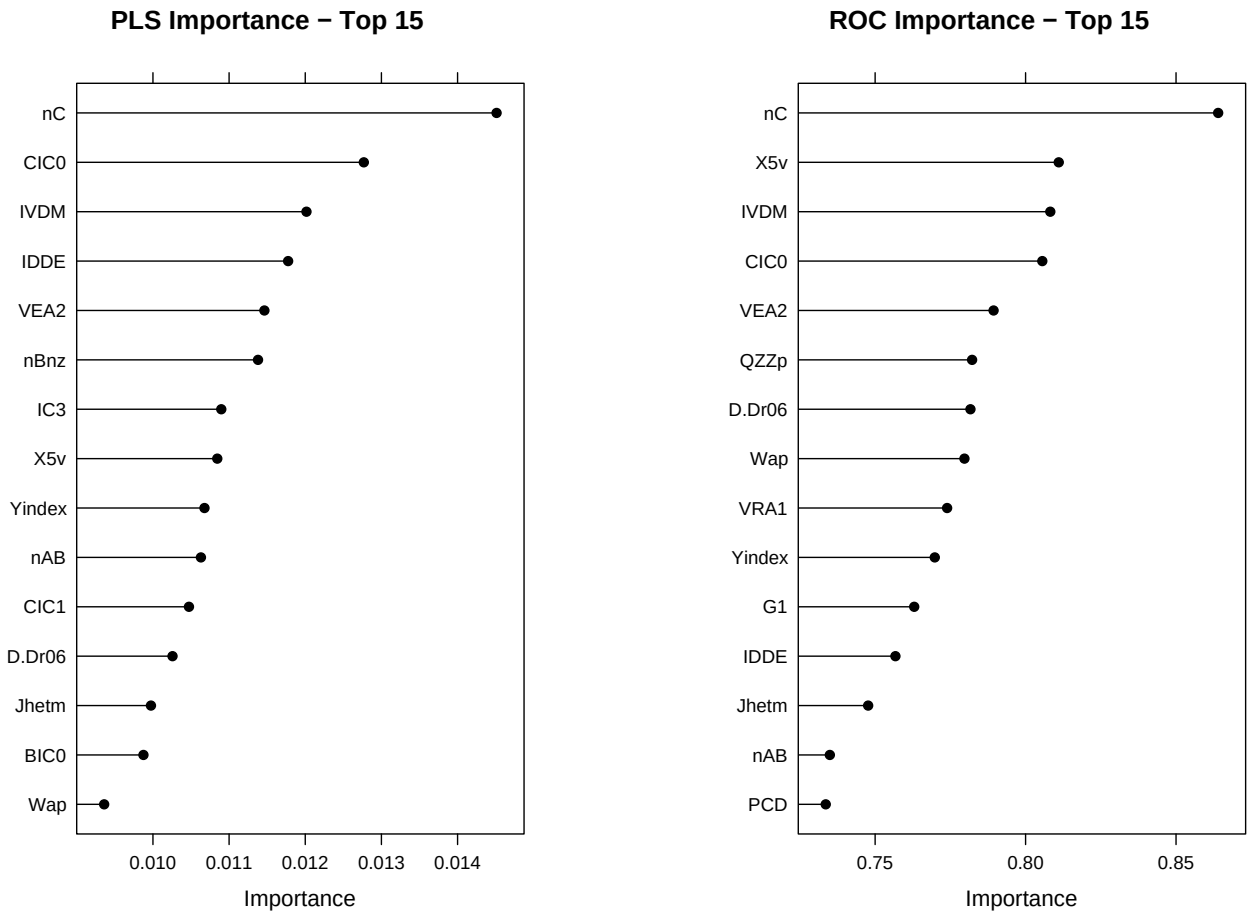

Figure 2: Examples of two variable importance plots for the MBRR data. The left–hand panel is based on the partial least squares results and was generated using `plot(varImp(plsFit2), top = 15)`. The right–hand plot based on the univariate ROC curves and was generated using `plot(varImp(plsFit2, useModel = FALSE), top = 15)`. In each case, the unsupervised correlation filter was applied to the predictors prior to modeling