Survey of
Machine Learning Feature Selection Methods: Boruta, SVD, PCA, LASSO, RFE, ...

Earl F Glynn

Kansas City R Users Group

2018-09-08

https://github.com/EarlGlynn/kc-r-users-feature-selection

Continuation from last year …

Using R's Caret Package for Machine Learning

Earl F Glynn

Kansas City R Users Group

9 September 2017

https://github.com/EarlGlynn/kc-r-users-caret-2017

From last year ....

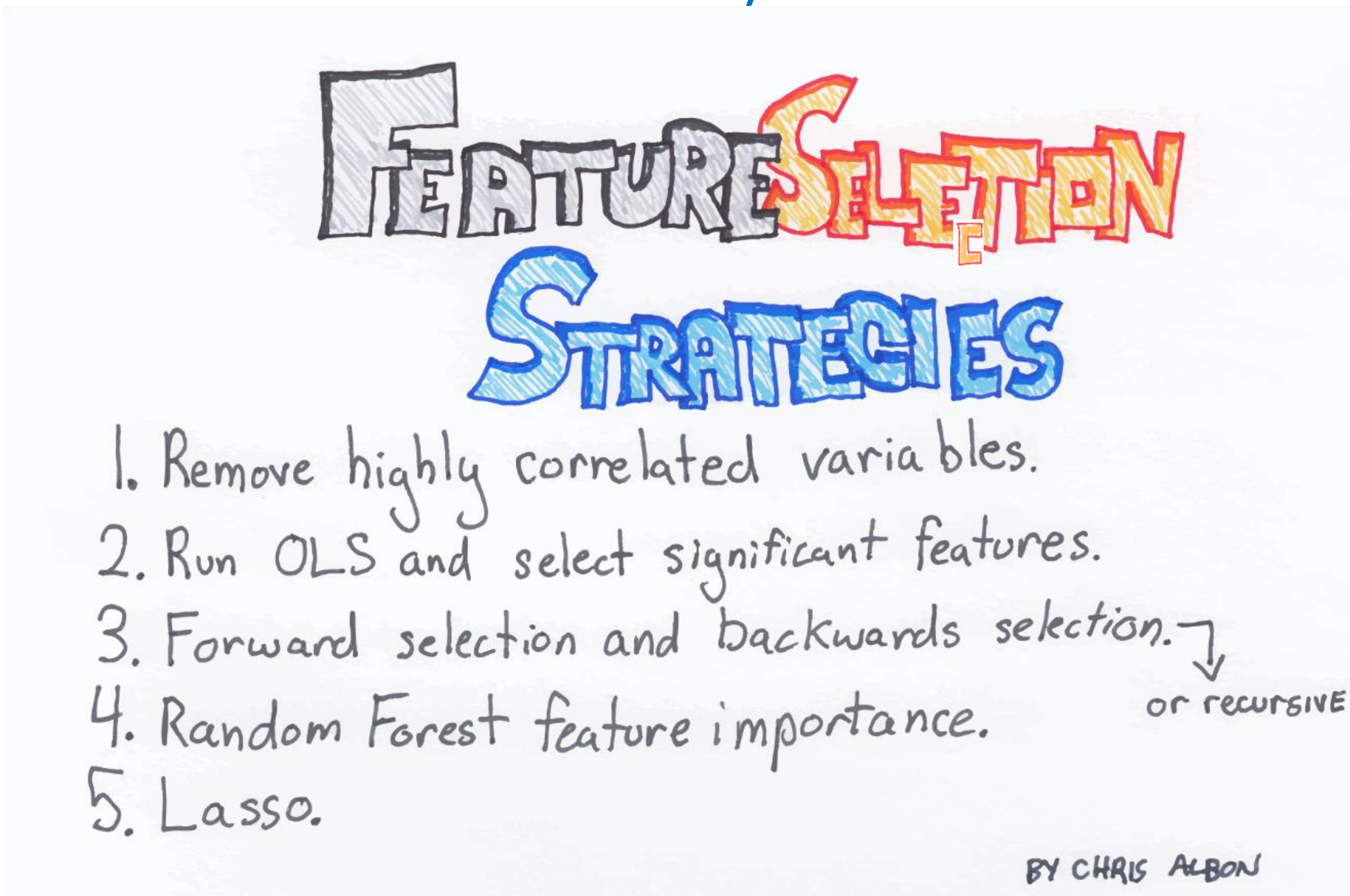
Caret Machine Learning Examples

Group	Algorithms	Caret Model	FILE
Linear Methods	Linear Discriminant Analysis	`lda`	`lda`
	Linear Discriminant Analysis with YeoJohnson preprocessing	`lda w/YeoJohnson`	`lda-YeoJohnson`
	LASSO, Ridge, and Elastic Net	`glmnet`	`glmnet`
	LASSO, Ridge, and Elastic Net with Synthetic Minority Over-Sampling Technique (SMOTE)	`glmnet w/SMOTE`	`glmnet-SMOTE`
Non-Linear Methods	Neural Network	`nnet`	`nnet`
	Support Vector Machine with Radial Basis Function Kernel	`svmRadial`	`svmRadial`
	Naïve Bayes	`nb`	`nb`
	Naïve Bayes with Independent Component Analysis (ICA)	`nb w/ICA`	`nb-ica-transform`
	k Nearest Neighbors	`knn`	`knn`
Trees and Rules	J48	`J48`	`J48`
	Classification and Regression Trees	`rpart`	`rpart`
Ensembles of Trees	C5.0	`C5.0`	`C50`
	Random Forest	`rf`	`rf`
	Random Forest with SMOTE	`rf w/SMOTE`	`rf-SMOTE`

https://github.com/EarlGlynn/kc-r-users-caret-2017

Files: **Forensic-Glass-caret-FILE.Rmd** and **Forensic-Glass-caret-FILE.html**

Survey of …



6. Boruta "All Relevant" Variables
7. SVD (Singular Value Decomposition)
8. PCA (Principal Component Analysis)

Small dataset used in examples:

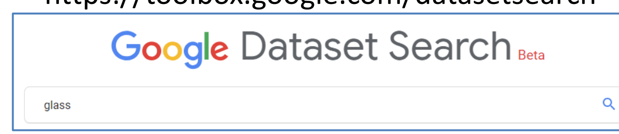
Forensic Glass Dataset

https://sha.org/bottle/links.htm

fgl dataset in MASS package

- RI = refractive index
- Percentages by weight of oxides: Na, Mg, Al, Si, K, Ca, Ba, Fe
- Predict *type* from measured values above
  - window float glass (WinF)
  - window non-float glass (WinNF)
  - vehicle window glass (Veh)
  - containers (Con)
  - tableware (Tabl)
  - vehicle headlamps (Head)

https://toolbox.google.com/datasetsearch



Also available through UCI Repository.

Discussed in book Data Mining and Business Analytics with R.

Forensic Glass Dataset

```
library(MASS)          # fgl data
```

Forensic Glass Data

```
rawData <- fgl
dim(rawData)
```

```
[1] 214  10
```

```
rawData                                                      %>%
  kable("html", caption="Forensic Glass Data")     %>%
  kable_styling(bootstrap_options=c("striped", "bordered", "condensed"),
                position="left", font_size=12,
                full_width=FALSE)                            %>%
  scroll_box(height="200px")
```

Forensic Glass Data

RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	type
3.01	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.00	WinF
-0.39	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.00	WinF
-1.82	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.00	WinF
-0.34	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.00	WinF

```
table(rawData$type)
```

```
 WinF WinNF   Veh   Con  Tabl  Head
   70    76    17    13     9    29
```

Class imbalance

Files: **Forensic-Glass-Boruta.Rmd** and **Forensic-Glass-Boruta.html**

Feature Selection

1. Remove Highly Correlated Variables

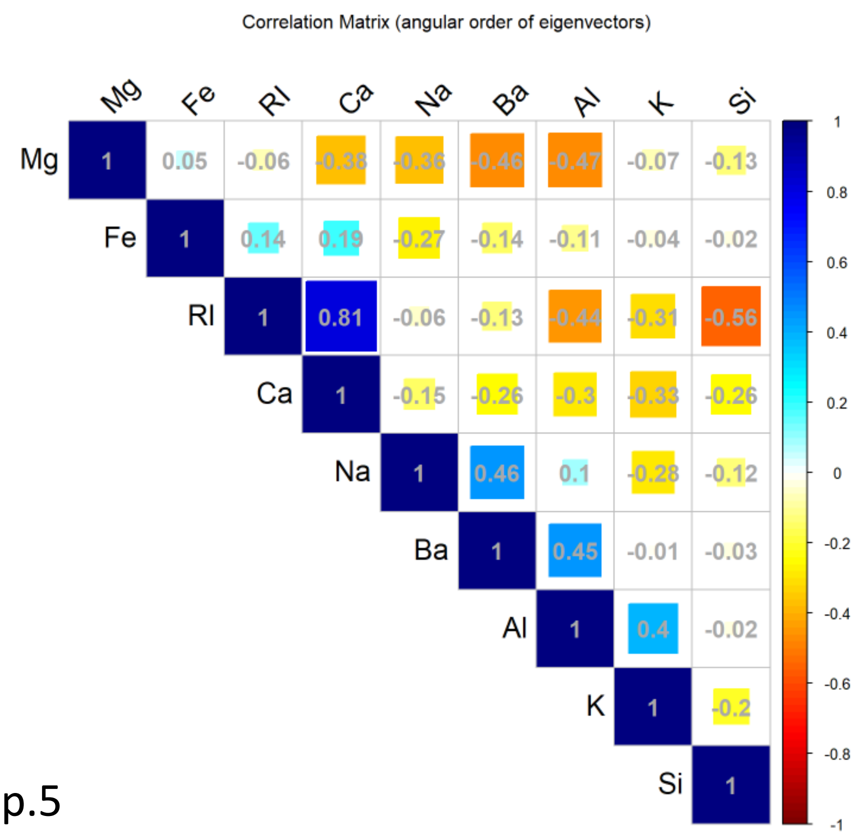
trainSet Correlation Matrix

Use only training set data or we'll have a data leak.

```r
colorScale <- colorRampPalette(c("#7F0000","red","#FF7F00","yellow","white",
                                 "cyan", "#007FFF", "blue","#00007F"))(100)

corMatrix <- cor(trainSet %>% dplyr::select(-type))   # Create correlation matrix

corrplot(corMatrix, type="upper", method="square", order="AOE",
         tl.col="black", tl.srt=45, tl.cex=1.5,
         addCoef.col="darkgrey", number.cex=1.25,
         col=colorScale)
mtext("Correlation Matrix (angular order of eigenvectors)", line=3)
```



order="AOE" (angular order of eigenvalues)
http://www.datavis.ca/papers/corrgram.pdf, p.5

Files: **Forensic-Glass-Correlation.Rmd** and **Forensic-Glass-Correlation.html**

Feature Selection

Remove Highly Correlated Variables

I often use a cutoff of 0.9 or above, but here 0.8 is used to trigger a removal.

Removing highly correlated pairs with caret

Some machine learning algorithms are impeded by highly correlated predictors. Caret's findCorrelation procedure can be used to remove one of the highly-correlated pairs.

```
HIGH_CORRELATION_CUTOFF <- 0.80

corHigh <- findCorrelation(corMatrix, HIGH_CORRELATION_CUTOFF)
if (length(corHigh) > 0)
{
  cat("Removing highly-correlated variable(s): ", names(trainSet)[corHigh])
  trainSet <- trainSet[, -corHigh]
  testSet  <- testSet[, -corHigh]
}
```

```
Removing highly-correlated variable(s):  Ca
```

Note: Remove constant variables, too.

Files: **Forensic-Glass-Correlation.Rmd** and **Forensic-Glass-Correlation.html**

Feature Selection

2. Run OLS and select significant features

OLS = Ordinary Least Square, but approach also applies to
NLS = Non-linear Least Squares, e.g.,
Levenberg-Marquardt non-linear curve fitting

```
library(minpack.lm)  # nls.lm
```

```
summary(nlsFit)
```

```
Parameters:
       Estimate Std. Error t value Pr(>|t|)
A     29.518818  23.888092   1.236    0.263
alpha -4.615019   3.485175  -1.324    0.234
B     86.974831   2.210565  39.345 1.80e-08 ***
beta  -0.092361   0.005677 -16.269 3.43e-06 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 2.002 on 6 degrees of freedom
Number of iterations to termination: 9
Reason for termination: Relative error in the sum of squares is at most `ftol'.
```

Feature Selection

3. Caret's Recursive Feature Extraction (RFE)

Random Forests

```
rfeController <- rfeControl(functions   = rfFuncs,   # random forests
                            seeds       = seeds,
                            method      = "repeatedcv",
                            number      = nCV,
                            repeats     = nRepeats,
                            verbose     = FALSE)
```

Setup parallel processing

```
rCluster <- makePSOCKcluster(6)   # use 6 cores
registerDoParallel(rCluster)

rfProfile <- rfe(x, y, sizes=subsets, rfeControl=rfeController)

stopCluster(rCluster)
```

```
rfProfile
```

```

Recursive feature selection

Outer resampling method: Cross-Validated (3 fold, repeated 50 times)

Resampling performance over subset size:

 Variables Accuracy  Kappa AccuracySD KappaSD Selected
         1   0.3860 0.1740    0.05975 0.07562
         2   0.6170 0.4725    0.07747 0.10338
         3   0.6670 0.5383    0.05204 0.07326
         4   0.7048 0.5954    0.06086 0.08318
         5   0.7300 0.6306    0.06245 0.08546
         6   0.7507 0.6583    0.05779 0.07939
         7   0.7647 0.6779    0.05450 0.07429        *
         8   0.7544 0.6622    0.05308 0.07298
         9   0.7581 0.6669    0.05218 0.07218

The top 5 variables (out of 7):
   Mg, Al, Ba, RI, Na
```

Output shows the best subset size was 7 predictors.

```
predictors(rfProfile)
```

```
[1] "Mg" "Al" "Ba" "RI" "Na" "Ca" "K"
```

-Si -Fe

Naive Bayes

```
rfeController <- rfeControl(functions   = nbFuncs,   # naive bayes
                            seeds       = seeds,
                            method      = "repeatedcv",
                            number      = nCV,
                            repeats     = nRepeats,
                            verbose     = FALSE)
```

Setup parallel processing

```
rCluster <- makePSOCKcluster(6)   # use 6 cores
registerDoParallel(rCluster)

nbProfile <- rfe(x, y, sizes=subsets, rfeControl=rfeController)

stopCluster(rCluster)
```

```
nbProfile
```

```

Recursive feature selection

Outer resampling method: Cross-Validated (3 fold, repeated 50 times)

Resampling performance over subset size:

 Variables Accuracy  Kappa AccuracySD KappaSD Selected
         1   0.5020 0.2889    0.08511 0.12103
         2   0.6093 0.4544    0.05870 0.08073
         3   0.6274 0.4888    0.06283 0.08532
         4   0.6311 0.4994    0.06330 0.08583
         5   0.6358 0.5087    0.06242 0.08254        *
         6   0.6252 0.4944    0.06187 0.08170
         7   0.6234 0.4931    0.06358 0.08370
         8   0.6046 0.4663    0.06458 0.08685
         9   0.5846 0.4358    0.06598 0.08868

The top 5 variables (out of 5):
   Al, Mg, Na, Ca, K
```

Output shows the best subset size was 5 predictors.

```
predictors(nbProfile)
```

```
[1] "Al" "Mg" "Na" "Ca" "K"
```

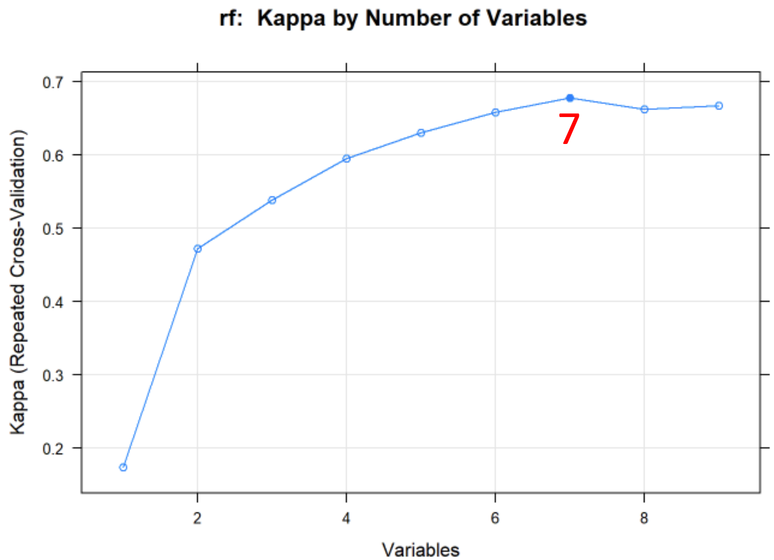
-RI -Si -Ba -Fe

Files: **Forensic-Glass-caret-RFE.Rmd** and **Forensic-Glass-caret-RFE.html**

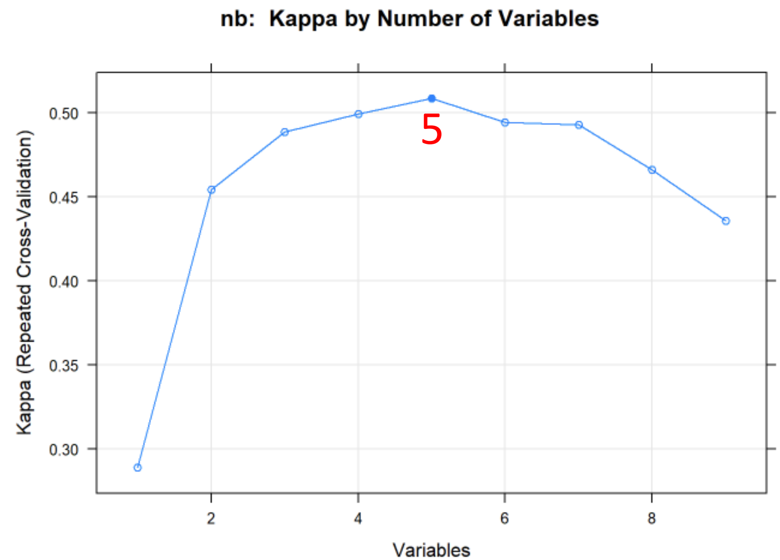
Feature Selection

Caret’s Recursive Feature Extraction (RFE)

```
plot(rfProfile, type=c("g", "o"), metric="Kappa",
     main="rf:  Kappa by Number of Variables")
```




```
plot(nbProfile, type=c("g", "o"), metric="Kappa",
     main="nb:  Kappa by Number of Variables")
```

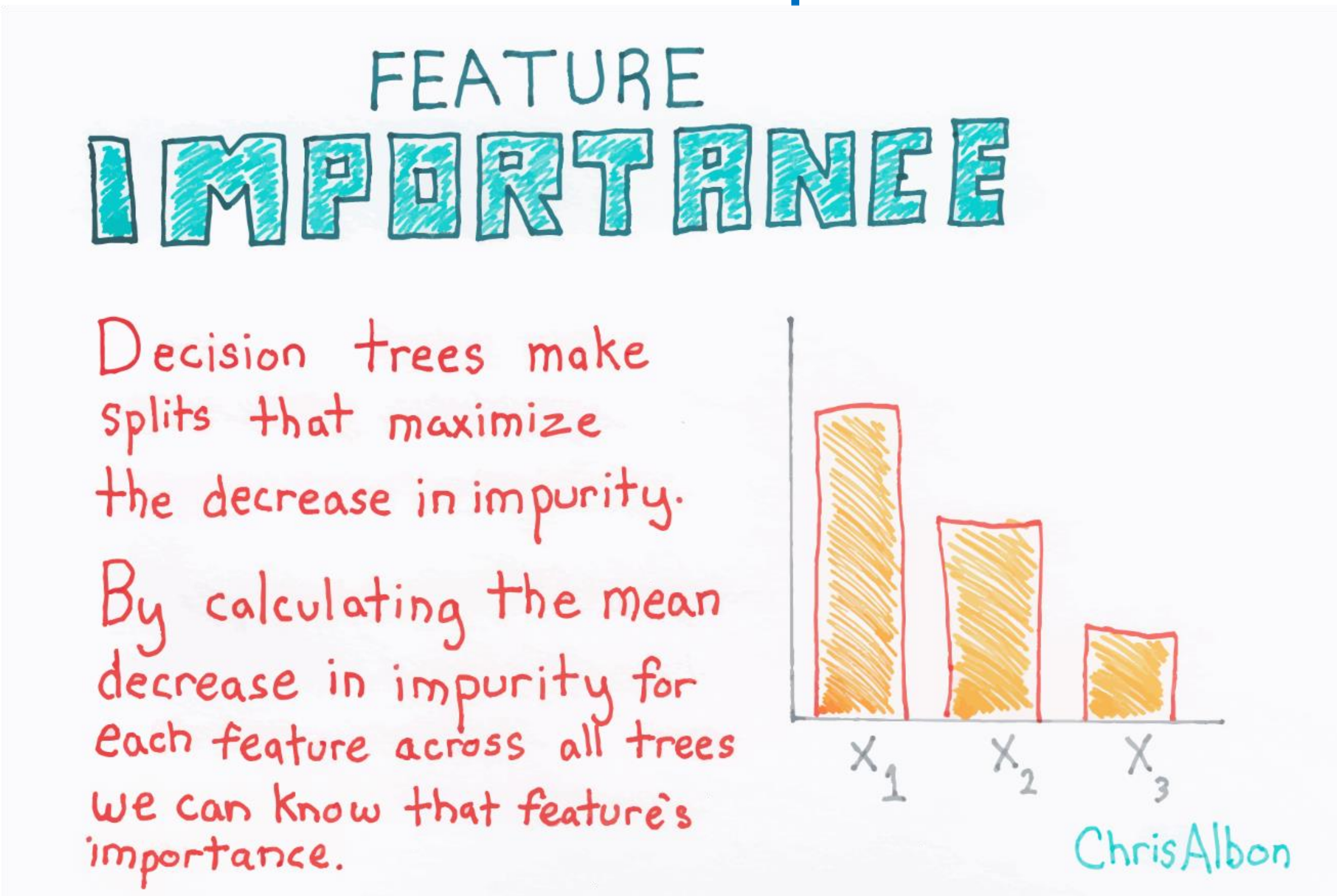


Kappa is generally a better metric than *Accuracy* with imbalanced classification.

Files: **Forensic-Glass-RFE.Rmd** and **Forensic-Glass-RFE.html**

Feature Selection

4. Feature Importance



@ChrisAlbon

machinelearningflashcards.com

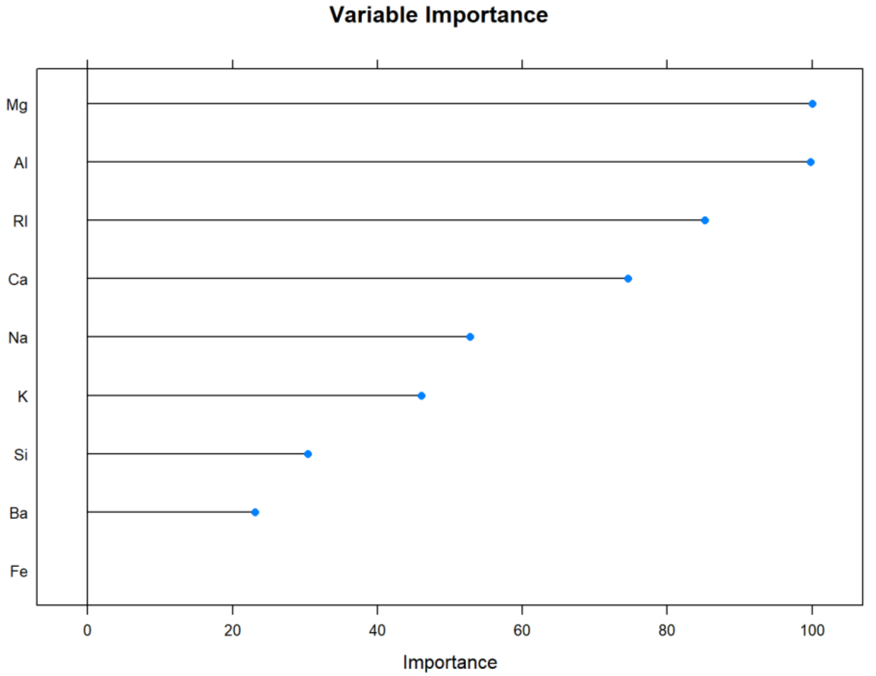
Feature Selection

4. Feature Importance

Use feature importance from one or more machine learning models to select features.
Study consistency and differences among models. Single Predictors here.

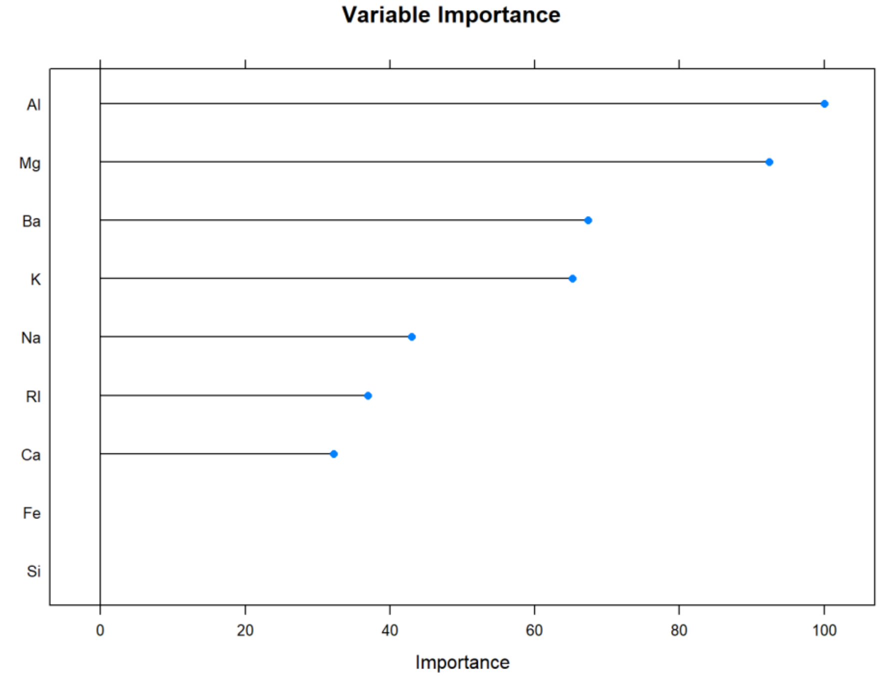
Tree Method

Random Forest Classification



Ensemble of Trees Method

Classification and Regression Trees



2017 Files: **Forensic-Glass-caret-rf.html** and **Forensic-Glass-caret-rpart.html**

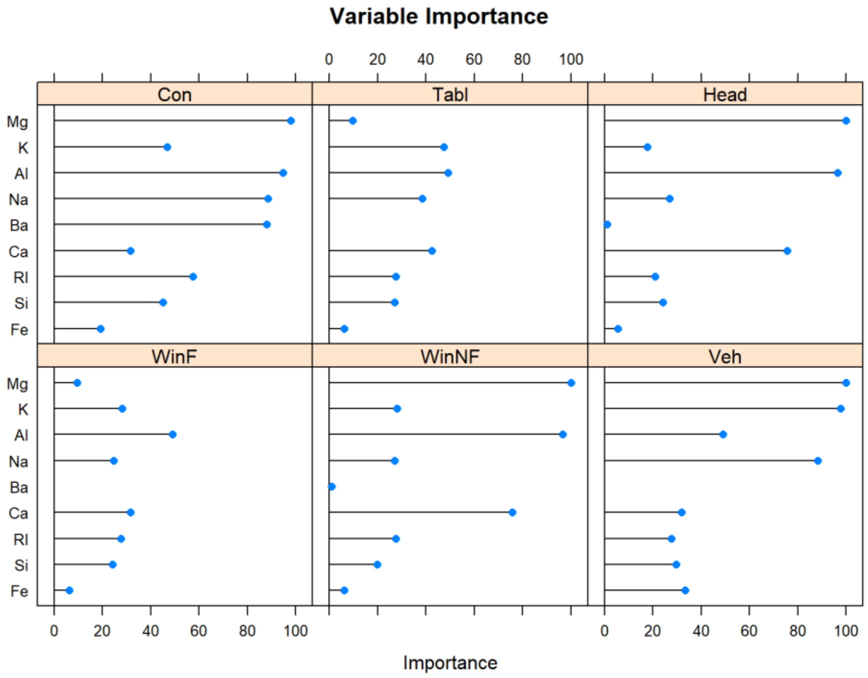
Feature Selection

Feature Importance

Use feature importance from one or more machine learning models to select features. Study consistency and differences among models. Separate Predictors, Very Similar.

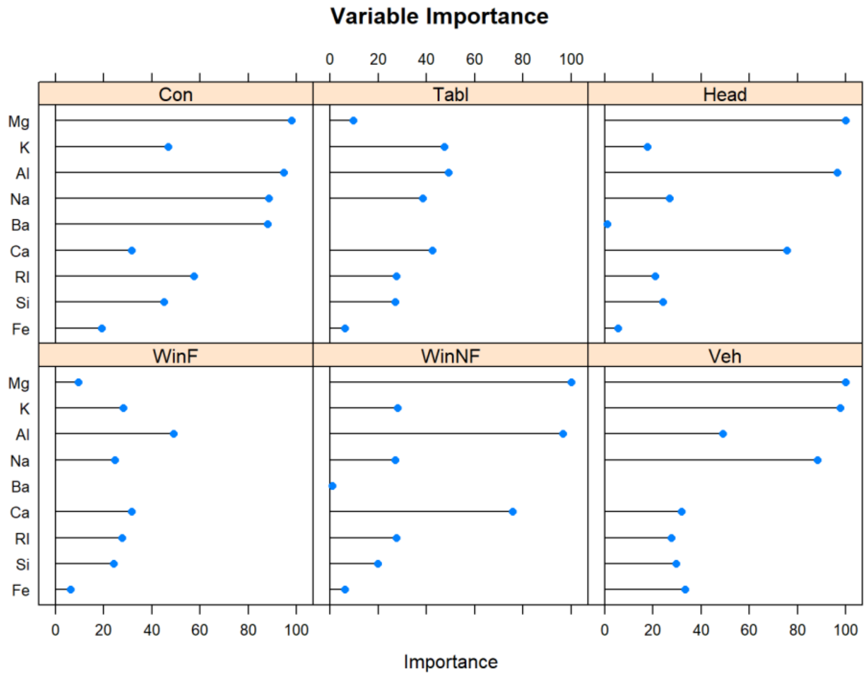
Linear Method

Linear Discriminant Analysis



Non-Linear Method

Naïve Bayes Classifier



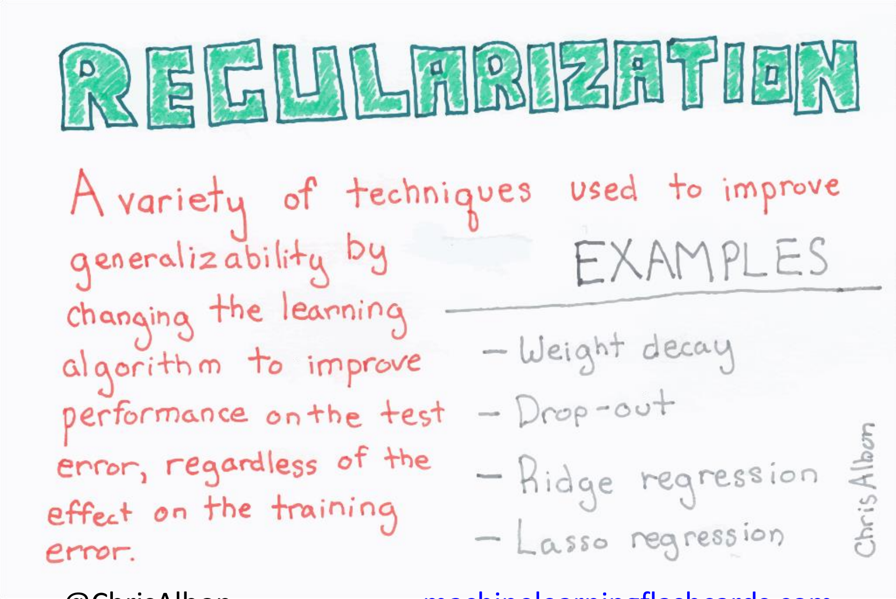
2017 Files: **Forensic-Glass-caret-lda.html** and **Forensic-Glass-caret-nb.html**

Feature Selection

5. glmnet

Ridge, Lasso and Elastic-Net Regularized Generalized Linear Models
(α=0 for Ridge, α=1 for Lasso)

- Math is a bit complicated:
  https://quantmacro.wordpress.com/2016/04/26/fitting-elastic-net-model-in-r/
  https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html
- Idea is to force some coefficients to zero to exclude from model.



@ChrisAlbon machinelearningflashcards.com

Feature Selection
glmnet

glmnet (lasso and elastic-net regularization)

```
tuneGrid <- expand.grid(alpha  = seq(0.25, 0.75, by=0.05),  # alpha 1 for Lasso, 0 for Ridge
                        lambda = c(0.05, 0.005, 0.0005))    # strength of penalty on coefficients

set.seed(29)
CVfolds   <-  5  # 5-fold cross validation (not enough data for 10 fold here)
CVrepeats <- 10  # repeat 10 times

# Used createMultiFolds to study
indexFolds <- createMultiFolds(trainSet$type, CVfolds, CVrepeats)  # for repeated CV

trainControlParms <- trainControl(method = "repeatedcv",  # repeated cross validation
                                  number  = CVfolds,
                                  repeats = CVrepeats,
                                  index   = indexFolds,
                                  classProbs = TRUE,       # Estimate class probabilities
                                  summaryFunction = defaultSummary)

fit <- train(type ~ ., data=trainSet,
             preProcess = c("center", "scale"),
             method = "glmnet",
             metric = "Kappa",
             tuneGrid = tuneGrid,
             trControl = trainControlParms)
```

Forensic-Glass-caret-glmnet.Rmd and **Forensic-Glass-caret-glmnet.html**

Feature Selection
glmnet

Each class has a separate set of coefficients.

Fit coefficients for best λ:

```
coef(fit$finalModel, s = fit$bestTune$lambda)
```

```
$WinF
10 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)  1.207375322
RI           0.250065503
Na          -0.224390621
Mg           1.630701058
Al          -1.402125023
Si           0.004345775
K            .
Ca           .
Ba           0.277709942
Fe           .

$WinNF
10 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept)  2.0909299
RI           .
Na          -0.7154609
Mg           0.1743042
Al           .
Si          -0.9109666
K            0.1107151
Ca          -0.1779174
Ba           .
Fe           0.2913854

$Veh
10 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept)  0.16121177
RI          -1.02662928
Na           .
Mg           0.56240805
Al          -1.36706675
Si          -1.29460588
K           -0.48120821
Ca           0.05261939
Ba           .
Fe           0.14586035
```

```
$Con
10 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) -1.5639444
RI           .
Na          -0.6234961
Mg          -1.3894046
Al           1.4634459
Si           0.1103138
K            1.0771763
Ca           0.5982293
Ba          -0.5683051
Fe           0.4363922

$Tabl
10 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) -1.1941202
RI           .
Na           1.4674086
Mg          -0.1975055
Al           .
Si           0.4315638
K           -1.7011573
Ca           0.1962890
Ba          -0.6153015
Fe          -0.6018880

$Head
10 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) -0.70145244
RI           0.86756205
Na           1.23281956
Mg          -0.78050324
Al           1.29083825
Si           0.02129146
K            .
Ca          -0.82978126
Ba           1.29411893
Fe          -0.65922012
```

glmnet creates a separate model for each class.

Some coefficients are driven to zero by the method, which effectively excludes predictor from model.

glmnet can be used as a prediction model, or as a tool to select relevant variables for other models.

Forensic-Glass-caret-glmnet.Rmd and **Forensic-Glass-caret-glmnet.html**

Feature Selection

6. Boruta "All Relevant" Variables

Define train and test datasets

```
set.seed(71)

trainSetIndices <- createDataPartition(rawData$type, p=0.50, list=FALSE)

trainSet <- rawData[ trainSetIndices, ]
testSet  <- rawData[-trainSetIndices, ]
```

```
dim(trainSet)
```

```
## [1] 109  10
```

Boruta 'All Relevant' Variables

Setup parallel processing

```
rClusterr <- makePSOCKcluster(6)    # use 6 cores
registerDoParallel(rCluster)
```

Will vary by platform

```
set.seed(73)    # reproducible random numbers
BorutaModel <- Boruta(type ~ ., data=trainSet,
                      getImp = getImpFerns,
                      maxRuns = 1000, num.threads=6)
stopCluster(rCluster)
```

No data pre-processing needed!

Boruta was a demon/god in Slavic mythology, who lived in trees.
https://www.pinterest.com/pin/497577458807361039/

Feature Selection

Boruta “All Relevant” Variables

```
print(BorutaModel)
```

```
Boruta performed 48 iterations in 1.187219 secs.
 8 attributes confirmed important: Al, Ba, Ca, K, Mg and 3 more;
 1 attributes confirmed unimportant: Fe;
```

Basic Idea of Boruta Algorithm

> *Perform shuffling of predictors' values and join them with the original predictors and then build random forest on the merged dataset. Then make comparison of original variables with the randomised variables to measure variable importance. Only variables having higher importance than that of the randomised variables are considered important.*

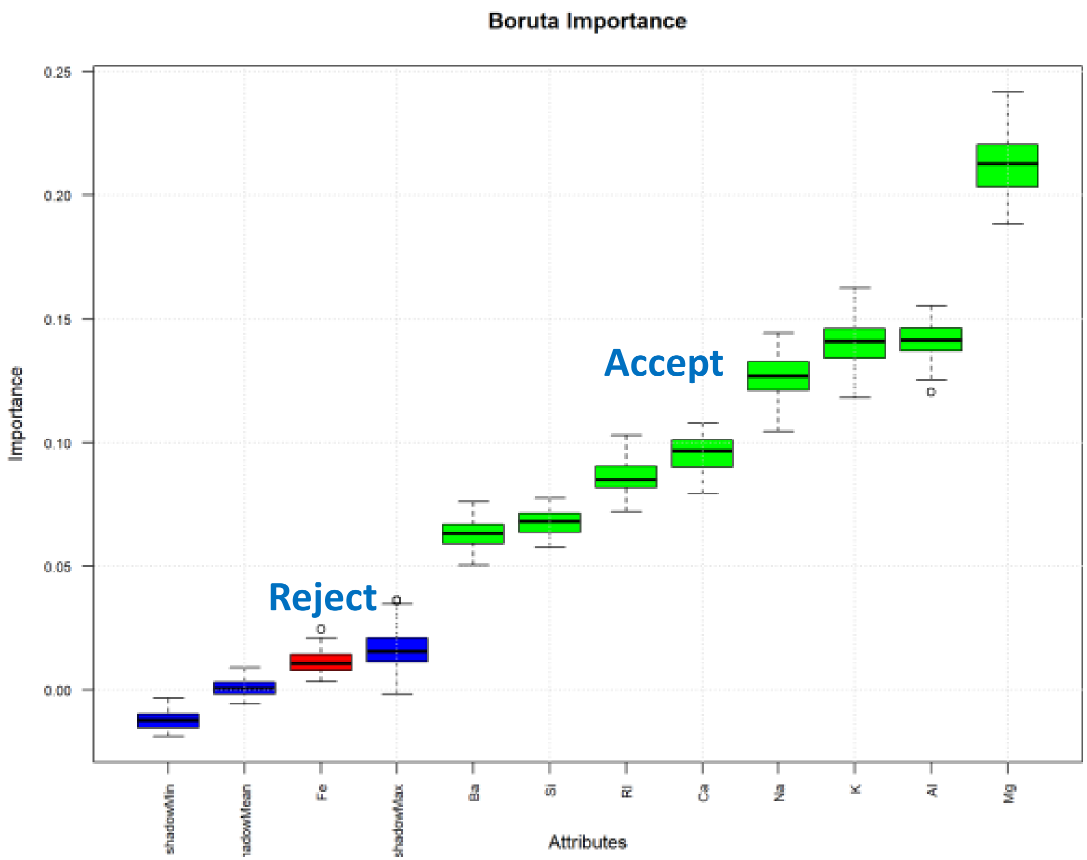
https://www.listendata.com/2017/05/feature-selection-boruta-package.html
https://www.datacamp.com/community/tutorials/feature-selection-R-boruta

Files: **Forensic-Glass-Boruta.Rmd** and **Forensic-Glass-Boruta.html**

Feature Selection

Boruta “All Relevant” Variables

```
plot(BorutaModel, las=2, cex.axis=0.75, main="Boruta Importance")
grid()
```

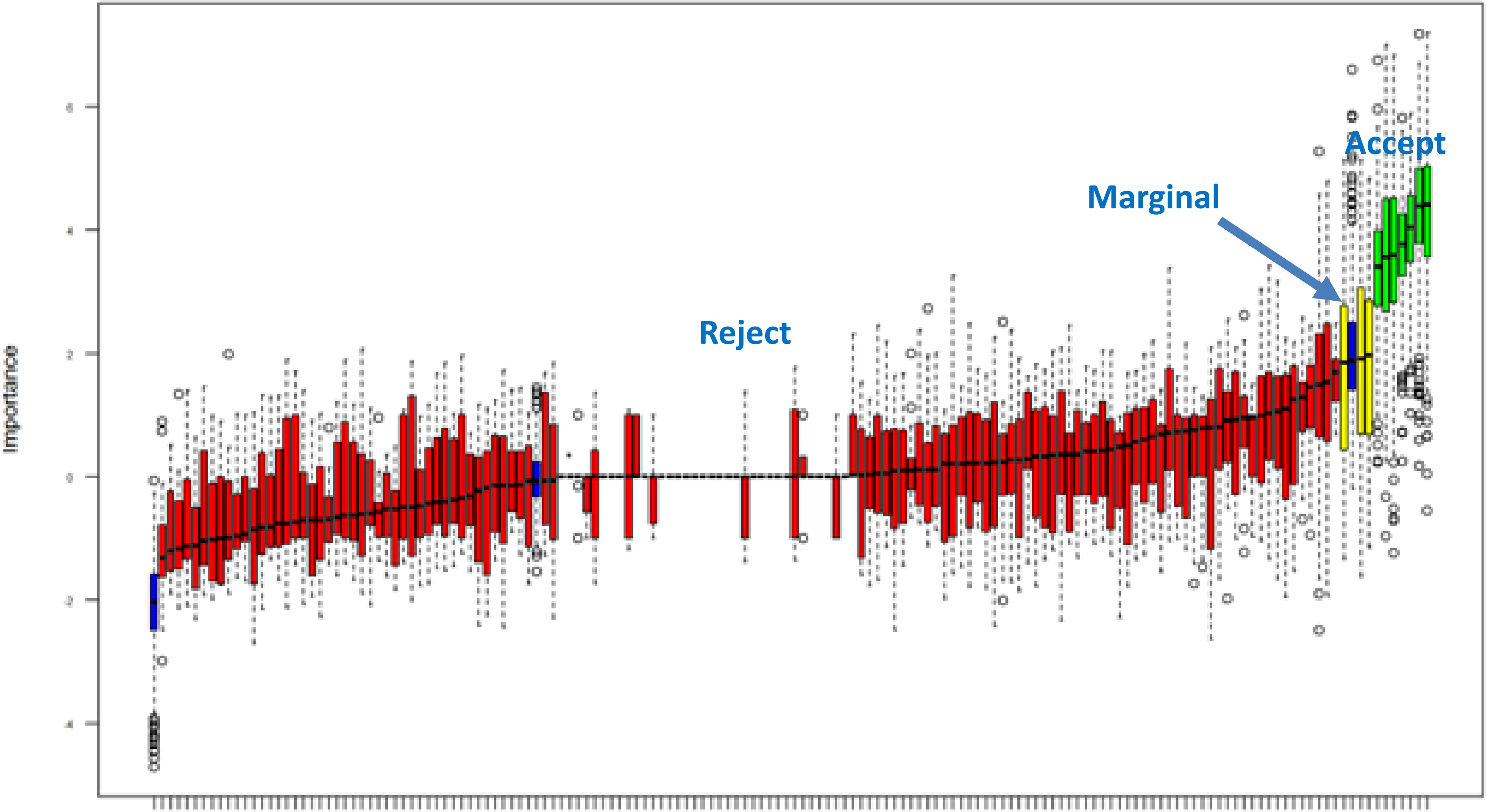


Files: **Forensic-Glass-Boruta.Rmd** and **Forensic-Glass-Boruta.html**

Feature Selection

Boruta “All Relevant” Variables

Sometimes Boruta approach can be very helpful with selections



Feature Selection

Boruta “All Relevant” Variables

```
stats <- attStats(BorutaModel)
statsOrdered <- stats[order(stats$medianImp, decreasing=TRUE), ]
```

```
BorutaFeatures <- statsOrdered %>% rownames_to_column(var="Feature")

BorutaFeatures                                          %>%
  kable("html", caption="Boruta Features")          %>%
  kable_styling(bootstrap_options=c("striped", "bordered", "condensed"),
                position="left", font_size=12,
                full_width=FALSE)
```

Boruta Features

Feature	meanImp	medianImp	minImp	maxImp	normHits	decision
Mg	0.2127641	0.2125817	0.1883694	0.2417506	1.0000000	Confirmed
Al	0.1406346	0.1412464	0.1204132	0.1553545	1.0000000	Confirmed
K	0.1400205	0.1408165	0.1181545	0.1625252	1.0000000	Confirmed
Na	0.1260519	0.1267271	0.1045717	0.1443913	1.0000000	Confirmed
Ca	0.0953332	0.0965700	0.0795888	0.1080054	1.0000000	Confirmed
RI	0.0860814	0.0849388	0.0722158	0.1030754	1.0000000	Confirmed
Si	0.0676871	0.0681237	0.0576441	0.0776154	1.0000000	Confirmed
Ba	0.0630710	0.0633130	0.0504127	0.0762487	1.0000000	Confirmed
Fe	0.0113589	0.0108034	0.0033247	0.0246047	0.2708333	Rejected

Feature Selection

Boruta "All Relevant" Variables

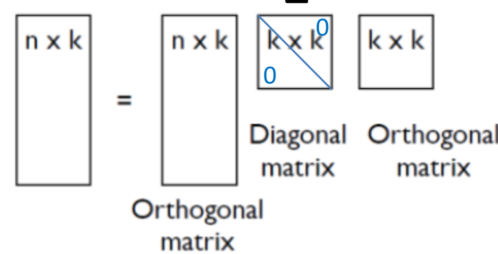
```
plotImpHistory(BorutaModel, main="Importance History")
grid()
```

Feature Selection

7. Singular Value Decomposition

- Dataset is a matrix of 214 rows of 9 predictors.
- Original matrix can be written as a product of three matrices with svd.
- Diagonal of middle matrix contains the singular values (eigenvalues if centered data).
- Right-most matrix contains the right singular vectors, which can give information about feature importance.

Train = **U** x **∑** x **V^T^**



Train Matrix = U x **∑** x **V^T**

In R:

```
svd1 <- svd(scale(trainData))
svd1$u %*% diag(svd1$d) %*% t(svd1$v)
```

U = left singular vectors = 214-by-9 unitary matrix. **svd1$u** in R.

∑ = singular values matrix = 9-by-9 diagonal matrix. **diag(svd1$d)** in R.

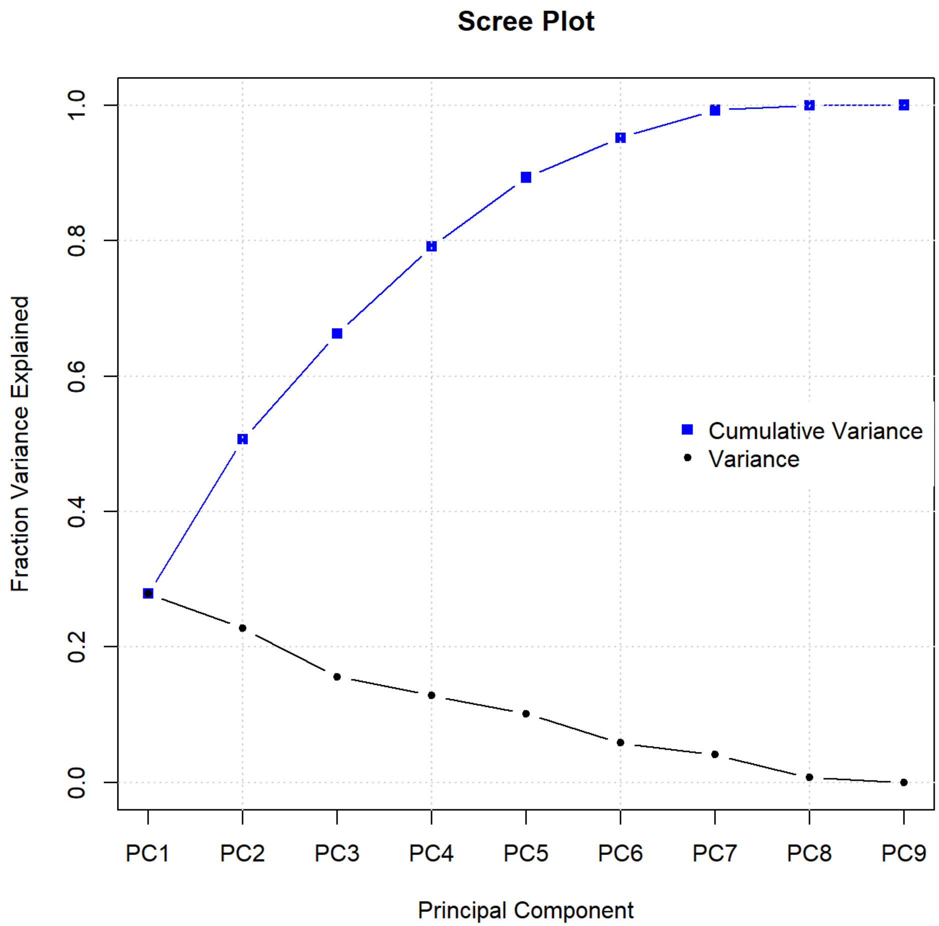
The diagonal terms are the singular values, usually listed in decreasing order.

V^T = transpose of V, where **V = right singular vectors** = 9x9 unitary matrix. **svd1$v** in R.

Matrix diagram: http://bigdata-madesimple.com/decoding-dimensionality-reduction-pca-and-svd/

Feature Selection

Singular Value Decomposition



Singular values are usually in decreasing order. 1st one largest.

Square of nth singular value proportional to variance associated with nth singular vector.

Often few terms explain large % of total variance.

Cumulative variance explained:

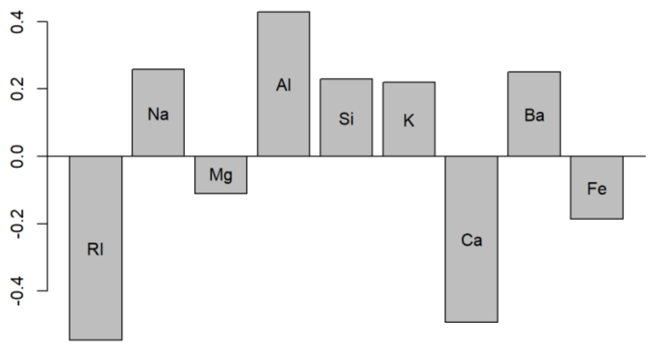
```
[1] 0.2790 0.5068 0.6629 0.7915
    0.8931 0.9517 0.9927 0.9998
    1.0000
```

Files: **Forensic-Glass-SVD.Rmd** and **Forensic-Glass-SVD.html**

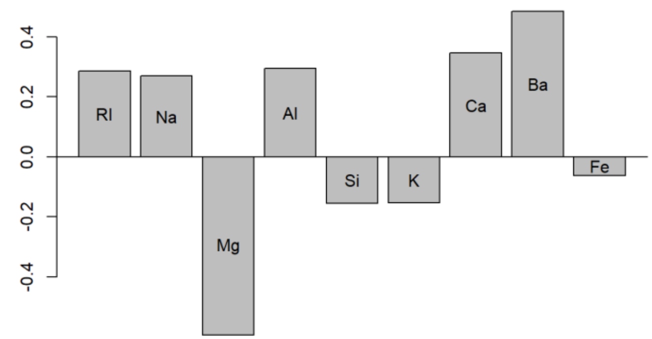
Feature Selection

Singular Value Decomposition

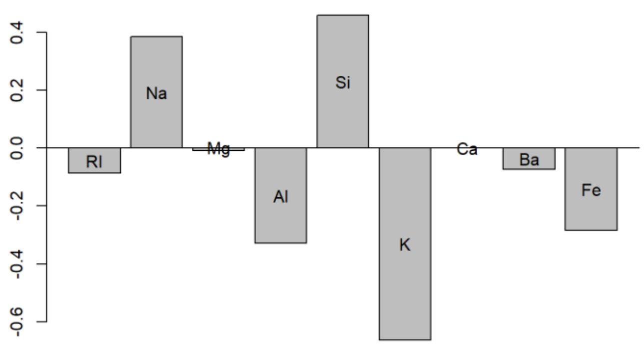
28% of variance **Right singular vector 1**



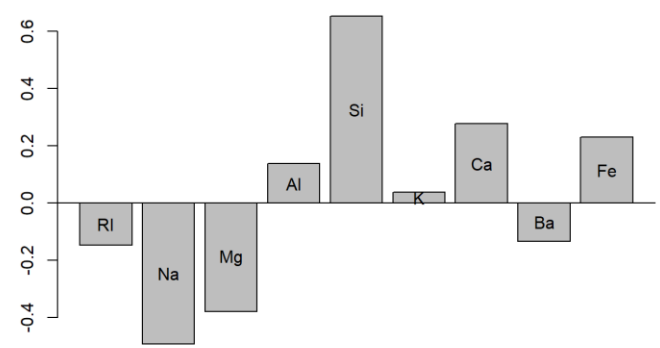
23% of variance **Right singular vector 2**



16% of variance **Right singular vector 3**



13% of variance **Right singular vector 4**

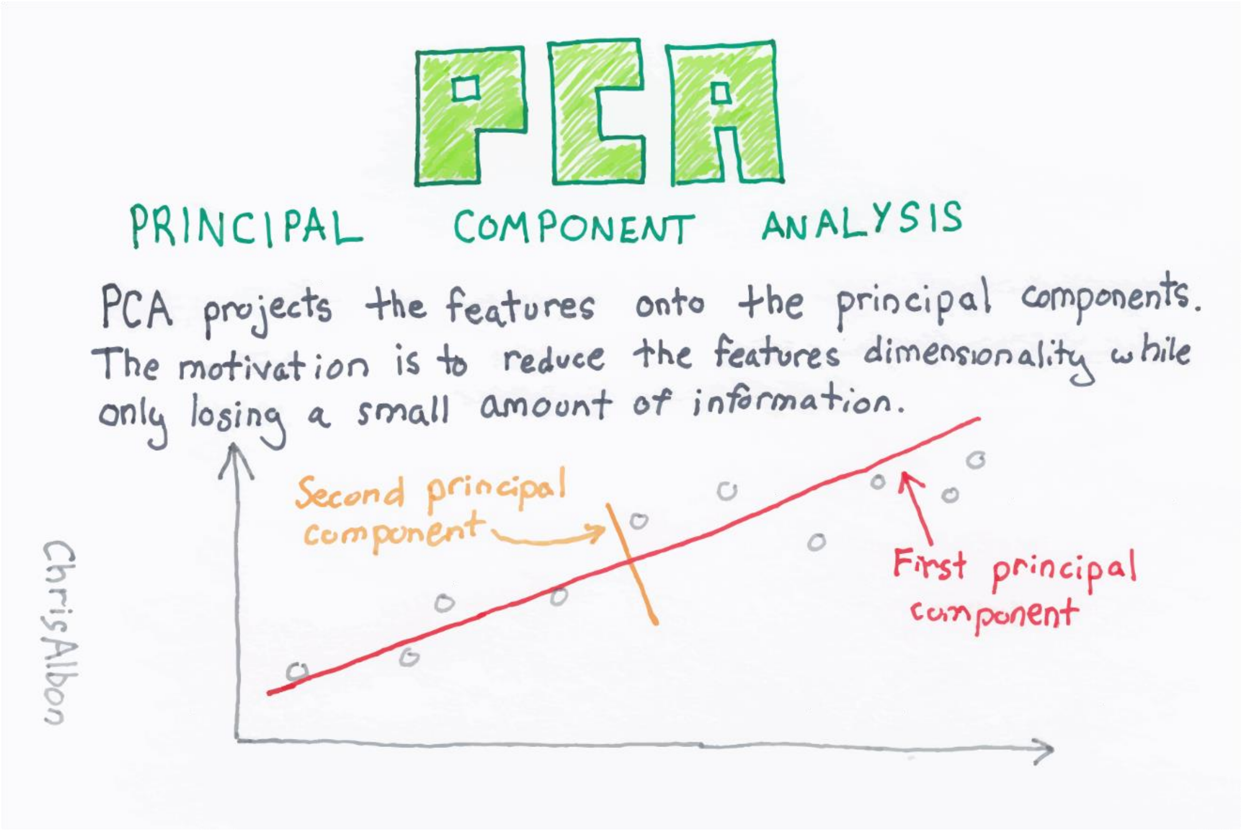


Pick variables with large contributions (+ or -): 1: RI, Ca, Al; 2: Mg, Ba; 3: K, Si

Feature Selection

8. Principal Component Analysis

PCA often computed from SVD

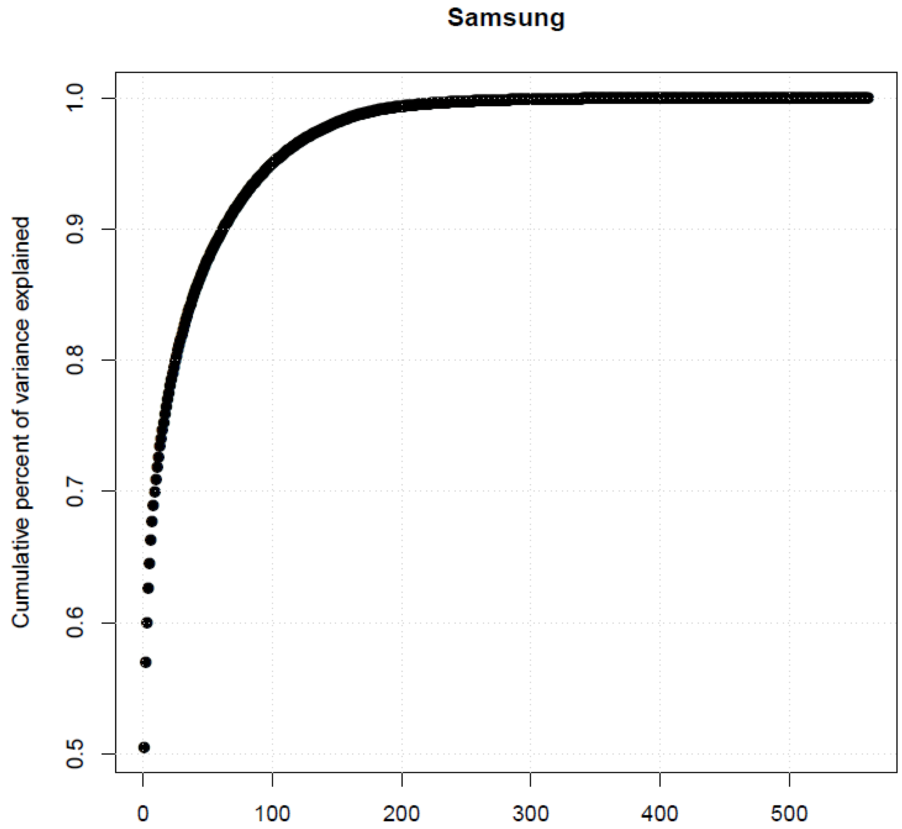


@ChrisAlbon

machinelearningflashcards.com

Feature Selection

SVD / PCA



Right singular vector can help with interpretation of a principal component.
Here, 9 of 500+ PCs explained over 70% of total variance.

Could use 9 PCs in machine learning that explain over 70% of variance instead of 500+ predictors.

Feature Selection

Principal Component Analysis

Setup

NOTE: place `tidyverse` after `MASS` below to avoid dplyr::select function clashes with MASS::select.

```
library(MASS)          # fgl data
library(tidyverse)     # place after MASS to avoid select conflict
library(caret)         # preProcess, predict
library(rgl)           # par3d, plot3d, movie3d, rglwidget
library(RColorBrewer)  # brewer.pal
```

Principal Component Analysis

Let's compute the values for the first 4 principal components using caret's pca pre-processing.

"pca" requires "center" and "scale".

These four PCs account for nearly 80% of variance.

```
nPCAcomponents <- 4
transformSetup <- preProcess(rawData, method=c("center", "scale", "pca"), pcaComp=nPCAcomponents)
pcaScores <- predict(transformSetup, rawData)
```

```
pcaScores %>% head(10)
```

```
          PC1        PC2       PC3         PC4
1 -1.148446843 -0.5282491 0.3712253 -1.72485681
2  0.572794160 -0.7580105 0.5554059 -0.75845396
3  0.937960515 -0.9276609 0.5536094 -0.20577184
4  0.141750924 -0.9594279 0.1168507 -0.41475157
5  0.350271021 -1.0886966 0.4839440 -0.06894065
```

Files: **Forensic-Glass-PCA.Rmd** and **Forensic-Glass-PCA.html**

Feature Selection

Principal Component Analysis

- *caret's preProcess* gives the same PCAscores as computed with SVD.
- Each PC is a weighted linear combination of all variables.
- PCs are orthogonal.
- **PCs can be used as variables in other machine learning algorithms**.
- Machine learning algorithm using PCs as predictors are limited by the amount of variance explained by the original variables in the given number of PCs.

Interactive Exploratory Analysis

Principal Component Analysis

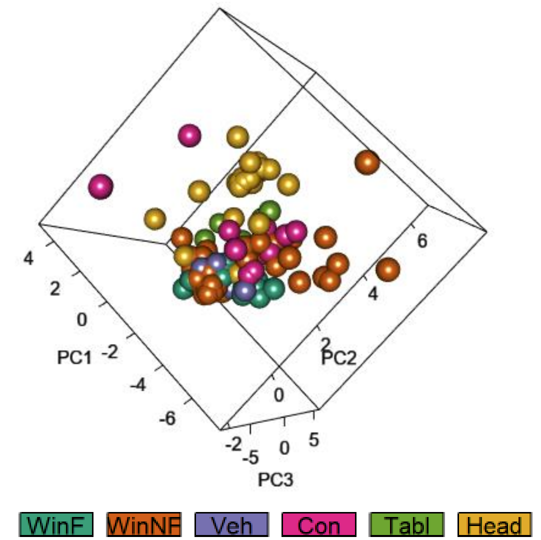
Use **plot3d** in **rgl** package to create interactive 3D scatterplot of any three PCs

The first 3 PCs account for about 66% of variance in data.

Other sets of 3 PCs could be dislayed alternatively in 3D space, such as PC2, PC3, PC4.

```
typeColors <- brewer.pal(length(levels(fgl$type)), "Dark2")
```

```
par3d("windowRect"=c(50,50,800,800))
plot3d(x=pcaScores$PC1, y=pcaScores$PC2, z=pcaScores$PC3,
       col=typeColors[typeColorIndex],
  xlab="PC1", ylab="PC2", zlab="PC3", type="s", size= 3)
rglwidget(elementId="FGL1")
```



Chrome browser works best to display above figure.

Drag mouse over figure to rotate. Use mouse wheel to zoom in and out.

Files: **Forensic-Glass-PCA.Rmd** and **Forensic-Glass-PCA.html**

Exploratory Analysis

Principal Component Analysis

magick from **ImageMagick** needed to create animated GIF of 3D PCA scatterplot

Animated GIF

Create the animated GIF using **magick** from ImageMagick – this takes some time. Display below using HTML.

150 PNG images will be computed for 15 sec duration * 10 frames/second.

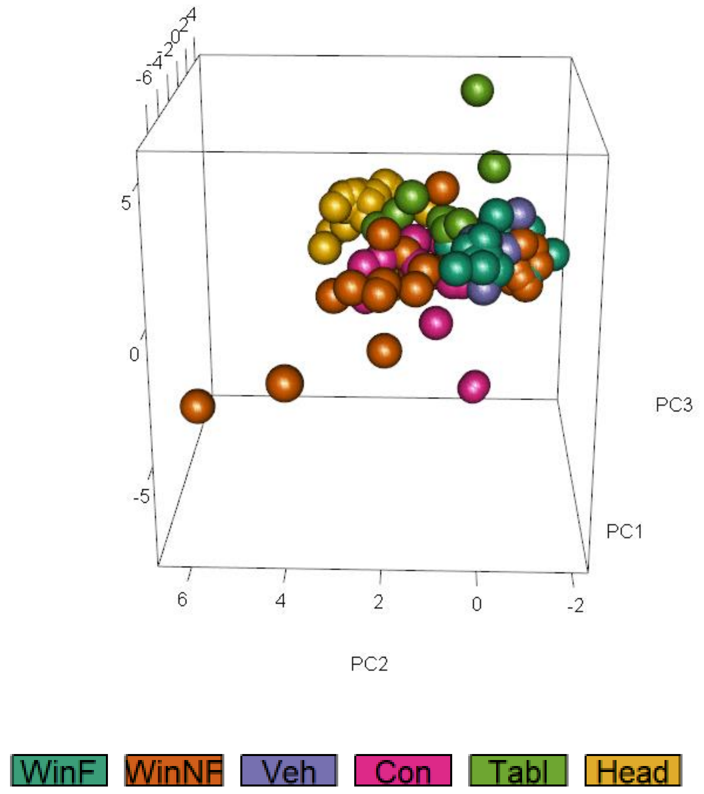
```
movie3d(spin3d(), duration = 15, dir = getwd(),
        movie="ForensicGlass-PCA",
        verbose=FALSE, convert="magick -delay 1x%d %s*.png %s.%s")
```

Here's the HTML needed in the R Markdown document to embed the GIF into the HTML file created with knitr.

```
<div id="PCA">
  <img src="ForensicGlass-PCA.gif" alt="">
</div>
```

Files: **Forensic-Glass-PCA.Rmd** and **Forensic-Glass-PCA.html**

Exploratory Analysis
Principal Component Analysis



Files: **Forensic-Glass-PCA.Rmd** and **Forensic-Glass-PCA.html**

Take Home

- Variety of ways to select features for machine learning models.
- Explore several methods.
- Boruta often easy to apply without much data preparation. Many methods require pre-processing, like centering and scaling.
- Experiment only with training data to avoid "data leaks" and overfitting.