Guide to Credit Scoring in R

By DS (ds5j@excite.com) *(Interdisciplinary Independent Scholar with 9+ years experience in risk management)*

Summary

To date Sept 23 2009, as Ross Gayler has pointed out, there is no guide or documentation on Credit Scoring using R (Gayler, 2008). This document is the first guide to credit scoring using the R system. This is a brief practical guide based on experience showing how to do common credit scoring development and validation using R. In addition the paper highlights cutting edge algorithms available in R and not in other commercial packages and discusses an approach to improving existing credit scorecards using the Random Forest package.

Note: This is not meant to be tutorial on basic R or the benefits of it necessarily as other documentation for e.g. http://cran.r-project.org/other-docs.html does a good job for introductory R.

Acknlowedgements: Thanks to Ross Gayler for the idea and generous and detailed feedback. Thanks also to Carolin Strobl for her help on unbiased random forest variable and party package.
Thanks also to George Overstreet and Peter Beling for helpful discussions and guidance. Also much thanks to Jorge Velez and other people on R-help who helped with coding and R solutions.

Table of Contents

Goals

The goal of this guide to show basic credit scoring computations in R using simple code.

Approach to Model Building

It is suggested that credit scoring practitioners adopt a systems approach to model development and maintenance. From this point of view one can use the SOAR methodology, developed by Don Brown at UVA (Brown, 2005). The SOAR process comprises of understanding the goal of the system being developed and specifying it in clear terms along with a clear understanding and specification of the data, observing the data, analyzing the data, and the making recommendations (2005). For references on the traditional credit scoring development process like Lewis, Siddiqi, or Anderson please see Ross Gayler's Credit Scoring references page (http://r.gayler.googlepages.com/creditscoringresources).

Architectural Suggestions

Clearly in the commercial statistical computing world SAS is the industry leading product to date. This is partly due to the vast amount of legacy code already in existence in corporations and also because of its memory management and data manipulation capabilities. R in contrast to SAS offers open source support, along with cutting edge algorithms, and facilities. To successfully use R in a large scale industrial environment it is important to run it on large scale computers where memory is plentiful as R, unlike SAS, loads all data into memory. Windows has a 2 gigbayte memory limit which can be problematic for super large data sets.

Although SAS is used in many companies as a one stop shop, most statistical departments would benefit in the long run by separating all data manipulation to the database layer (using SQL) which leaves only statistical computing to be performed. Once these 2 functions are decoupled it becomes clear R offers a lot in terms of robust statistical software.

Practical Suggestions

Building high performing models requires skill, ability to conceptualize and understand data relationships, some theory. It is helpful to be versed in the appropriate literature, brainstorm relationships that should exist in the data, and test them out. This is an ad hoc process I have used and found to be effective. For formal methods like Geschka's brainwriting and Zwicky's morphological box see Gibson's guide to Systems analysis (Gibson etal, 2004). For the advantages of R and introductory tutorials see http://cran.r-project.org/other-docs.html.

R Code Examples

In the credit scoring examples below the German Credit Data set is used (Asuncion et al, 2007). It has 300 bad loans and 700 good loans and is a better data set than other open credit data as it is performance based vs. modeling the decision to grant a loan or not. The bad loans did not pay as intended. It is common in credit scoring to classify bad accounts as those which have ever had a 60 day delinquency or worse (in mortgage loans often 90 day plus is often used).

Reading Data In

```
# read comma separated file into memory
data<-read.csv("C:/Documents and Settings/My
Documents/GermanCredit.csv")
```

Binning Example

In R dummy data variables are called factors and numeric or double are numeric types.

```
#code to convert variable to factor
data$ property <-as.factor(data$ property)

#code to convert to numeric
data$age <-as.numeric(data$age)

#code to convert to decimal
data$amount<-as.double(data$amount)
```

Often in credit scoring it is recommended that continuous variables like Loan to Value ratios, expense ratios, and other continuous variables be converted to dummy variables to improve performance (Mays, 2000).

Example of Binning or Coarse Classifying in R:

```
data$amount<-as.factor(ifelse(data$amount<=2500,'0-
2500',ifelse(data$amount<=5000,'2600-5000','5000+')))
```

Note: Having a variable in both continuous and binned (discrete form) can result in unstable or poorer performing results.

Breaking Data into Training and Test Sample

The following code creates a training data set comprised of randomly selected 60% of the data and the out of sample test sample being a random 40% sample remaining.

```
d = sort(sample(nrow(data), nrow(data)*.6))
#select training sample
train<-data[d,]
test<-data[-d,]
train<-subset(train,select=-default)
```

Traditional Credit Scoring

Traditional Credit Scoring Using Logistic Regression in R

```
m<-glm(good_bad~.,data=train,family=binomial())
# for those interested in the step function one can use m<-
step(m) for it
# I recommend against step due to well known issues with it
choosing the optimal #variables out of sample
```

Calculating ROC Curve for model

There is a strong literature based showing that the most optimal credit scoring cut off decisions can be made using ROC curves which plot the business implications of both the true positive rate of the model vs. false positive rate for each score cut off point (Beling et al, 2005)

```
#load library
library(ROCR)

#score test data set
test$score<-predict(m,type='response',test)
pred<-prediction(test$score,test$good_bad)
perf <- performance(pred,"tpr","fpr")
plot(perf)
```

For documentation on ROCR see Sing (Sing etal, 2005).

Calculating KS Statistic

To the dismay of optimal credit scoring cut off decision literature the KS statistic is heavily in use of the industry. Hand has shown that KS can be misleading and the only metric which matters should be the conditional bad rate given the loan is approved (Hand, 2005).

That said due to prevalence of KS we show how to compute it in R as it might be needed in work settings. The efficient frontier trade off approach although optimal seems to not appeal to executives as making explicit and forced trade offs seems to cause cognitive dissonance. For some reason people in the industry are entrenched on showing 1 number to communicate models whether it is KS or FICO etc.

```
#this code builds on ROCR library by taking the max delt
#between cumulative bad and good rates being plotted by
#ROCR
max(attr(perf,'y.values')[[1]]-attr(perf,'x.values')[[1]])
```

KS is the maximum difference between the cumulative true positive and cumulative false positive rate. The code above calculates this using the ROC curve.

If you do not use this cut off point the KS in essence does not mean much for actual separation of the cut off chosen for the credit granting decision.

Calculating top 3 variables affecting Credit Score Function in R

In credit scoring per regulation lenders are required to provide the top 3 reasons impacting the credit decision when a loan fails to be pass the credit score (Velez, 2008).

```
#get results of terms in regression
g<-predict(m,type='terms',test)
#function to pick top 3 reasons
#works by sorting coefficient terms in equation
# and selecting top 3 in sort for each loan scored
ftopk<- function(x,top=3){

     res=names(x)[order(x, decreasing = TRUE)][1:top]
     paste(res,collapse=";",sep="")
}

# Application of the function using the top 3 rows
topk=apply(g,1,ftopk,top=3)
#add reason list to scored tets sample
test<-cbind(test, topk)
```

Cutting Edge techniques Available in R

Using Bayesian N Using Traditional recursive Partitioning

Recursive Partitioning trees offer various benefits to credit scoring: quick, simple logic which can be converted into rules and credit policies, non parametric and can deal with interactions. The down side of trees is that they unstable, small changes in data can lead to large deviations in models, and can overfit if not built using cross validation and pruning. R's Rpart is one of best performing and robust tree algorithms and is comparable to J4.5 algorithm in java (Schauerhuber et al, 2007)

The fact that Rpart uses out of sample data to build and fit the tree makes it a very strong implementation (Therneau et al, 1997).

In an important study of logistic regression vs. tree algorithms Perlich et al show that high signal to noise data favors logistic regression while high separation favors tree algorithms and also 'apparent superiority of one method over another on small data sets' does not hold out over large samples' (Perlich et al, 2003). Bagging helps improve recursive partitioning. Using random forests is strongly recommended in lieu of trees or model based recursive partitioning but for simple needs the decision tree is still a powerful technique.

Trees can be used to clean variables, find splits in cut offs of other variables, break data in segments, and offer simple insights. Also it is possible to generate a large number of trees which perform equivalently but may look vastly different. They are perfect for generating straw credit policies for rule based systems for quick and dirty needs.

In terms of modeling rare events like fraud or low default credit portfolios using prior probabilities to configure trees can help improve performance. In particular trying 80/20, 90/10, 60/40, 50/50 type priors seems to be a quick and effective heuristic approach to getting high performing trees.

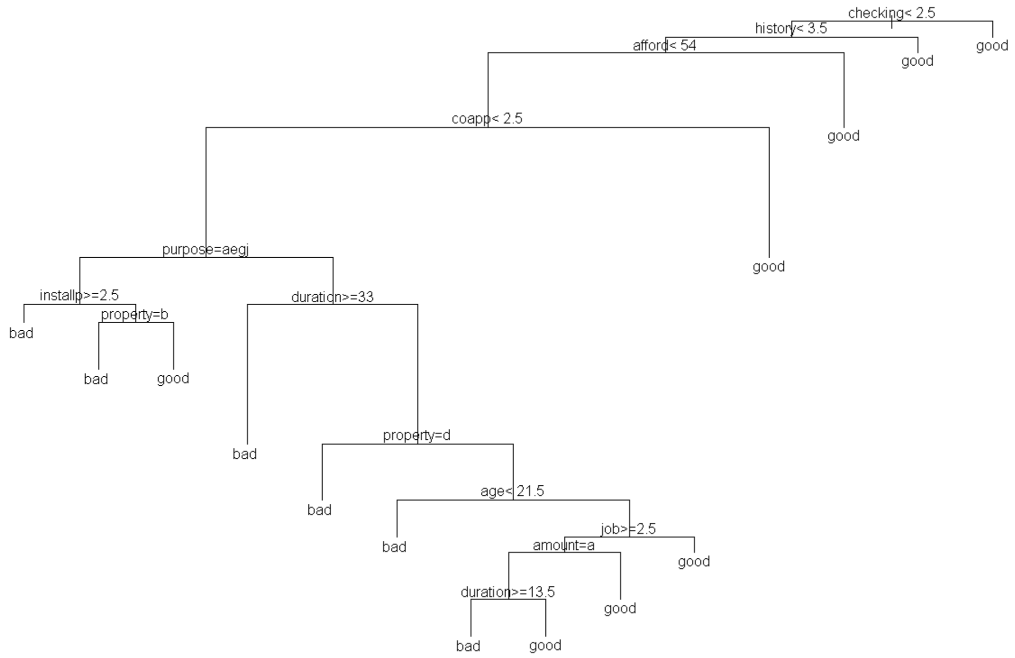
The following code builds decision trees and plots them and compares the tree with and without priors. As you can see the tree with priors performs better in this case.

The section then concludes with Graham Williams' code to convert trees into rules for rule based systems.

```
#load tree package
library(rpart)


fit1<-rpart(good_bad~.,data=train)
```

```
plot(fit1);text(fit1);
#test$t<-predict(fit1,type='class',test)
```

Plot of Tree without Priors

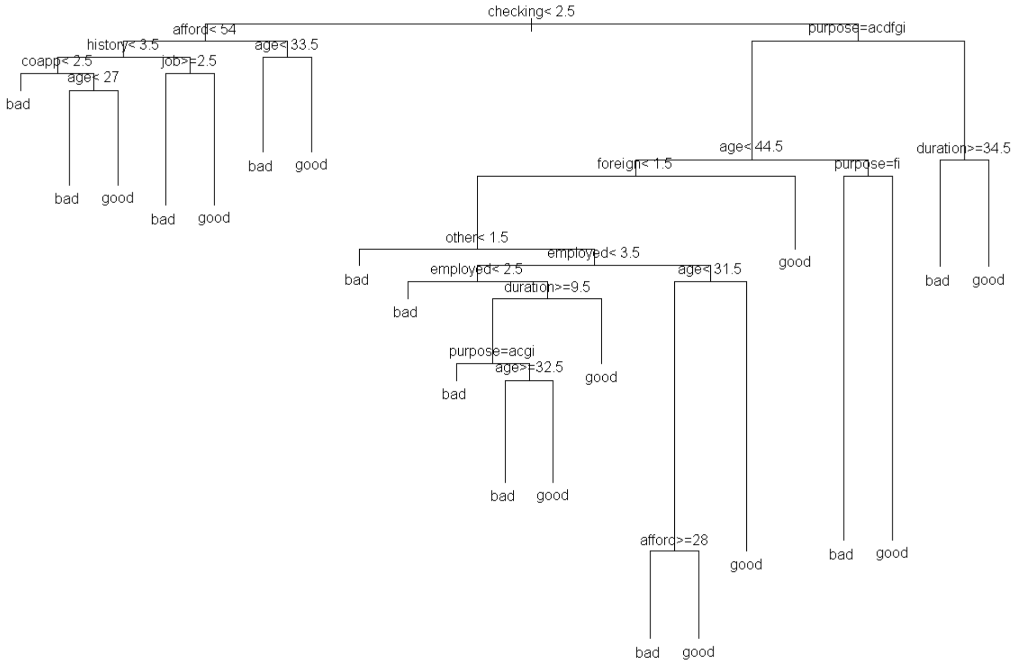


```
#score test data
test$tscore1<-predict(fit1,type='prob',test)
pred5<-prediction(test$tscore1[,2],test$good_bad)
perf5 <- performance(pred5,"tpr","fpr")

#build model using 90% 10% priors
#with smaller complexity parameter to allow more complex
trees
# for tuning complexity vs. pruning see Thernau 1997
```

```
fit2<-
rpart(good_bad~.,data=train,parms=list(prior=c(.9,.1)),cp=.
0002)
plot(fit2);text(fit2);
```

The tree shows that checking, history, and affordability appear to segment the loans well into different risk categories.

Plot of Tree with Priors and Greater Complexity



This tree built using weights for priors fits the data better and shows that loan purpose and affordability along with checking make better splits for segmenting data.

```
test$tscore2<-predict(fit2,type='prob',test)
```

```
pred6<-prediction(test$tscore2[,2],test$good_bad)
perf6<- performance(pred6,"tpr","fpr")
```

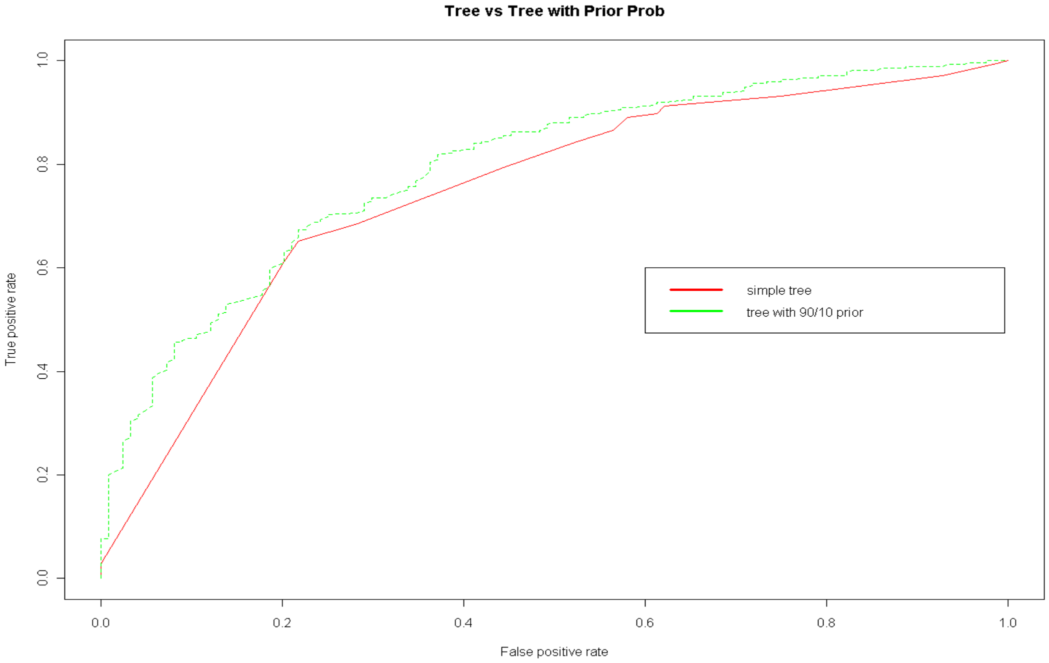
Comparing Complexity and out of Sample Error

```
#prints complexity and out of sample error
printcp(fit1)
#plots complexity vs. error
plotcp(fit1)

#prints complexity and out of sample error
printcp(fit2)
#plots complexity vs. error
plotcp(fit2)
```

For more details on tuning trees and plots see Thernau 1997 and Williams' excellent book on Data Mining. As Rpart uses error rates based on cross validation they are unbiased and accurate measures of performance.

Compare ROC Performance of Trees

```
plot(perf5,col='red',lty=1,main='Tree vs Tree with Prior
Prob');
plot(perf4, col='green',add=TRUE,lty=2);
legend(0.6,0.6,c('simple tree','tree with 90/10
prior'),col=c('red','green'),lwd=3)
```



Performance Of Tree with and Without Priors

The ROC plot shows how the tree built using prior weights outperforms the regular tree by significant degree at most points.

Converting Trees to Rules

This section uses a slightly modified version of Graham Williams' function from his excellent Desktop Data Mining Survival Guide. The function code is in the appendix of this document.

I particularly find the rule format more usable as Tree plots are confusing, counterintuitive and hard to read.

```
#print rules for all classes
list.rules.rpart(fit1)
list.rules.rpart(fit2)

#custom function to only print rules for bad loans
listrules(fit1)
listrules(fit2)
```

(See appendix for code for both functions)

<u>Sample output of select rules:</u>

```
 Rule number: 16 [yval=bad cover=220 N=121 Y=99 (37%) prob=0.04]
   checking< 2.5
   afford< 54
   history< 3.5
   coapp< 2.5

 Rule number: 34 [yval=bad cover=7 N=3 Y=4 (1%) prob=0.06]
   checking< 2.5
   afford< 54
   history< 3.5
   coapp>=2.5
   age< 27
 Rule number: 18 [yval=bad cover=50 N=16 Y=34 (8%) prob=0.09]
   checking< 2.5
   afford< 54
   history>=3.5
   job>=2.5
```

The rules show that loans with low checking, affordability, history, and no co-applicants are much riskier.

For other more robust recursive partitioning see Breiman's Random Forests and Zeleis and Hothorn's conditional inference trees and model based recursive partitioning which allows econometricians the ability to use theory to guide the development of tree logic (2007).

Bayesian Networks in Credit Scoring

The ability to understand the relationships between credit scoring variables is critical in building sound models. Bayesian Networks provide a powerful technique to understand causal relationships between variables via graphical directed graphs showing relationships among variables. The lack of causal analysis in econometric papers is an issue raised by Pearl and discussed at length in his beautiful work on causal inference (Pearl, 200). The technique treats the variables are random variables and uses markov chain monte carlo methods to assess relationships between variables. It is computationally intensive but another important tool to have in the credit scoring tool kit.

For literature on the applications of Bayesian networks to credit scoring please see Baesens et al(2001) and Chang et al (2000).

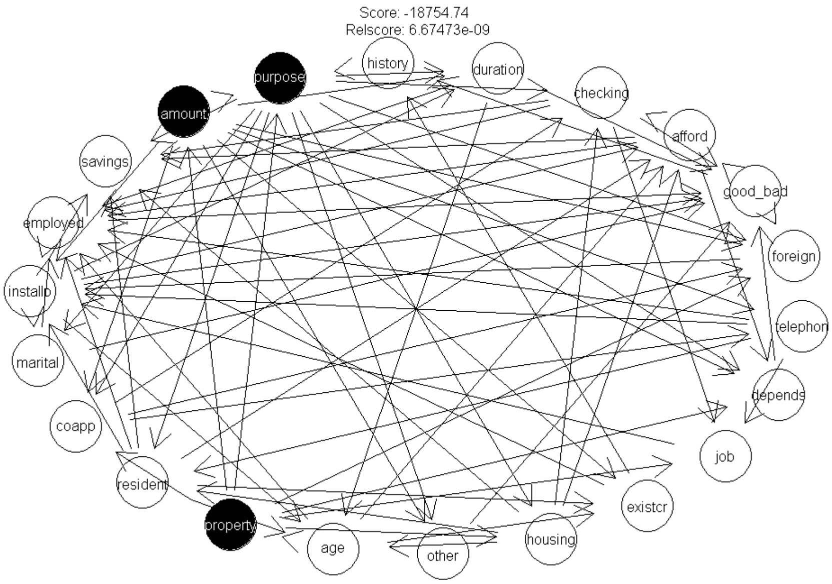
For details on Bayesian network Package see the deal package (Bøttcher & Dethlefsen, 2003).

Bayesian Network Credit Scoring in R

```
#load library
library(deal)

  #make copy of train
 ksl<-train
 #discrete cannot inherit from continuous so binary
good/bad must be converted to numeric for deal package
 ksl$good_bad<-as.numeric(train$good_bad)

#no missing values allowed so set any missing to 0
# ksl$history[is.na(ksl$history1)] <- 0
```

```
#drops empty factors
# ksl$property<-ksl$property[drop=TRUE]



ksl.nw<-network(ksl)
ksl.prior <- jointprior(ksl.nw)

#The ban list is a matrix with two columns. Each row
contains the directed edge
#that is not allowed.
#banlist <- matrix(c(5,5,6,6,7,7,9,8,9,8,9,8,9,8),ncol=2)
## ban arrows towards Sex and Year
#      [,1] [,2]
#[1,]    5    8
#[2,]    5    9
#[3,]    6    8
#[4,]    6    9
#[5,]    7    8
#[6,]    7    9
#[7,]    9    8
# note this a computationally intensive procuredure and if
you know that certain variables should have not
relationships you should specify
# the arcs between variables to exclude in the banlist
ksl.nw <- learn(ksl.nw,ksl,ksl.prior)$nw
#this step appears expensive so reset restart from 2 to 1
and degree from 10 to 1
result <-
heuristic(ksl.nw,ksl,ksl.prior,restart=1,degree=1,trace=TRU
E)
thebest <- result$nw[[1]]
savenet(thebest, "ksl.net")
print(ksl.nw,condposterior=TRUE)
```

Bayesian Network of German Credit Data



Using a Bayesian network diagram shows in a simple manner various important relationships in the data. For example one can see that affordability, dependents, marital status, and employment status all have causal effects on the likelihood of the loan defaulting as expected. The relationship between home ownership and savings also becomes clearer.

Using Traditional recursive Partitioning

Recursive Partitioning trees offer various benefits to credit scoring: quick, simple logic which can be converted into rules and credit policies, non parametric and can deal with interactions. The down side of trees is that they unstable, small changes in data can lead to large deviations in models, and can overfit if not built using cross validation and pruning. R's Rpart is one of best performing and robust tree algorithms and is comparable to J4.5 algorithm in java (Schauerhuber etal, 2007)

The fact that Rpart uses out of sample data to build and fit the tree makes it a very strong implementation (Therneau etal, 1997).

In an important study of logistic regression vs. tree algorithms Perlich etal show that high signal to noise data favors logistic regression while high speration favors tree algorithms and also 'apparent superiority of one method over another on small data sets' does not hold out over large samples' (Perlich etal, 2003). Although bagging helps improve recursive paritioning Using random forests is strongly recommended in lieu of trees or model based recursive partitioning but for simple needs the decision tree is still a powerful technique.

Trees can be used to clean variables, find splits in cut offs of other variables, break data in segments, and offer simple insights. Also it is possible to generate a large number of trees which perform equivalently but may look vastly different. They are perfect for generatin straw credit policies for rule based systems for quick and dirty needs.

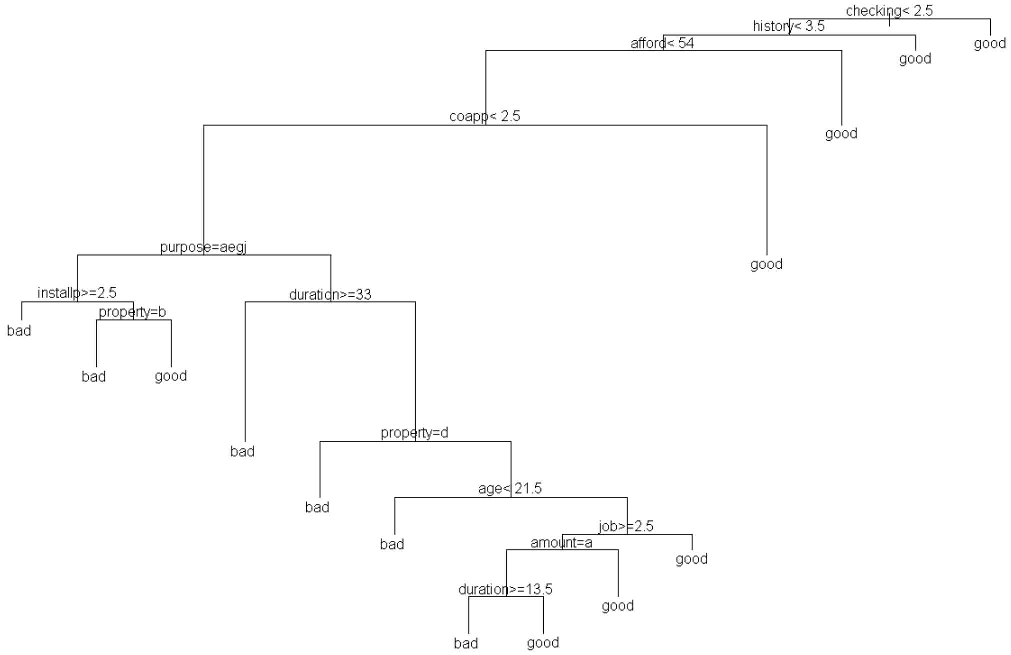
In terms of modeling rare events like fraud or low default credit portfolios using prior probabilities to configure trees can help improve performance. In particular trying 80/20, 90/10, 60/40, 50/50 type priors seems to be a quick and effective hueristic approach to getting high performing trees.

The following code builds decision trees and plots them and compares the tree with and without priors. As you can see the tree with priors performs better in this case.

The section then concludes with using Graham William's based code to convert trees into rules for rule based systems.

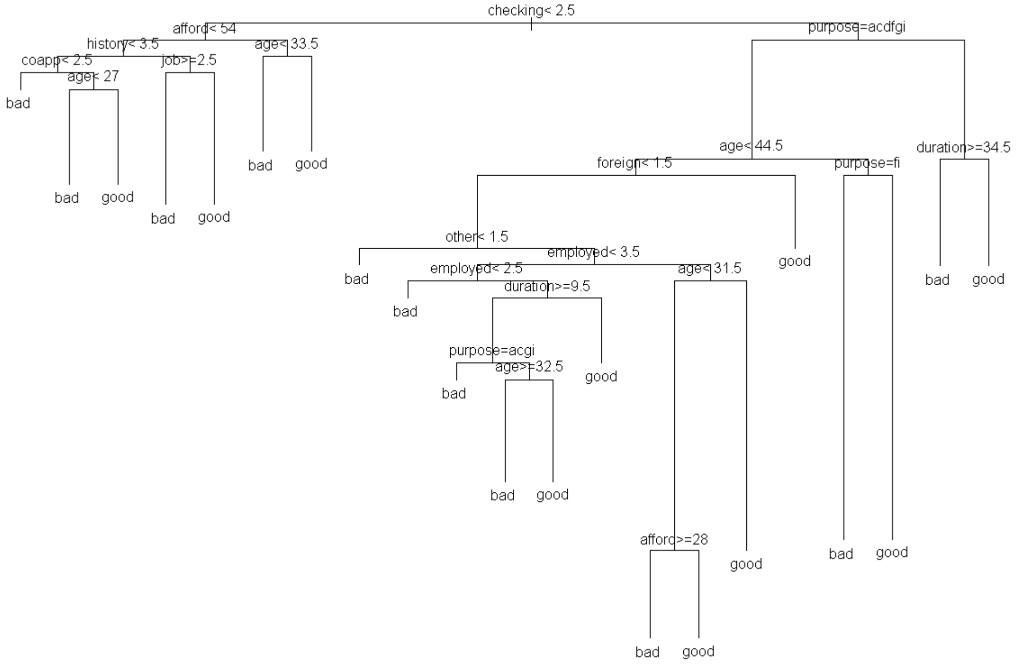
```
#load tree package
library(rpart)


fit1<-rpart(good_bad~.,data=train)
plot(fit1);text(fit1);
#test$t<-predict(fit1,type='class',test)
```

Plot of Tree without Priors



```
#score test data
test$tscore1<-predict(fit1,type='prob',test)
pred5<-prediction(test$tscore1[,2],test$good_bad)
perf5 <- performance(pred5,"tpr","fpr")

#build model using 90% 10% priors
#with smaller complexity parameter to allow more complex trees
# for tuning complexity vs. pruning see Thernau 1997
fit2<-rpart(good_bad~.,data=train,parms=list(prior=c(.9,.1)),cp=.0002)
plot(fit2);text(fit2);
```

Plot of Tree with Priors and Greater Complexity



```
test$tscore2<-predict(fit2,type='prob',test)
pred6<-prediction(test$tscore2[,2],test$good_bad)
perf6<- performance(pred6,"tpr","fpr")
```

Comparing Complexity and out of Sample Error

```
#prints complexity and out of sample error
printcp(fit1)
```

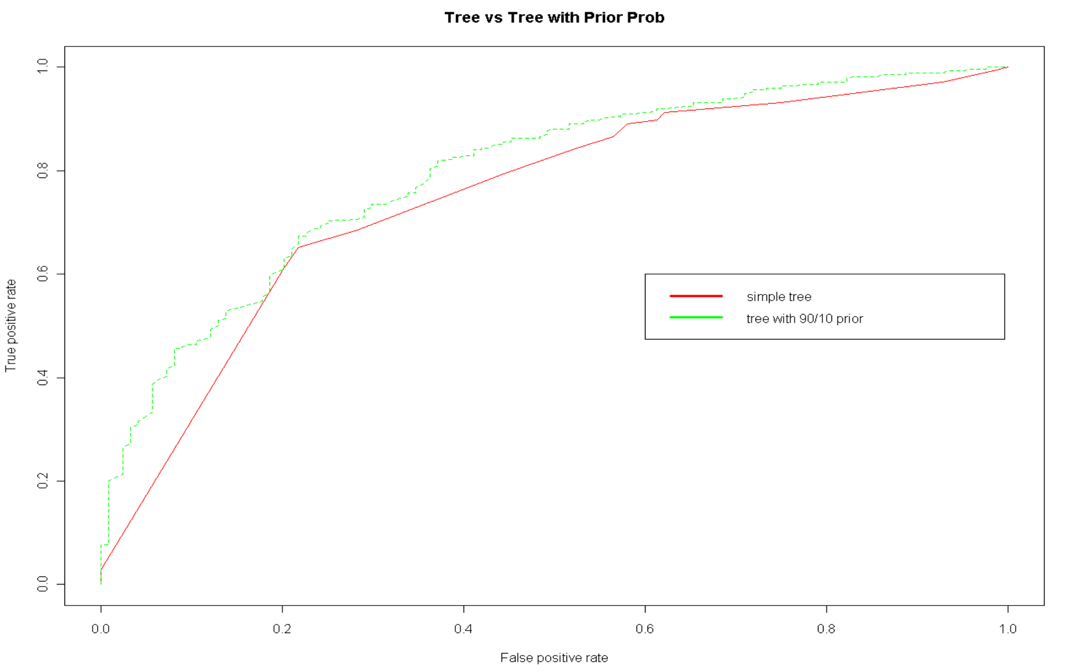
```
#plots complexity vs. error
plotcp(fit1)

#prints complexity and out of sample error
printcp(fit2)
#plots complexity vs. error
plotcp(fit2)
```

For more details on tuning trees and plots see Thernau 1997 and Williams's excellent book on Data Mining. As Rpart uses error rates based on cross validation they are unbiased and accurate measures of performance.

Compare ROC Performance of Trees

```
plot(perf5,col='red',lty=1,main='Tree vs Tree with Prior Prob');
plot(perf4, col='green',add=TRUE,lty=2);
legend(0.6,0.6,c('simple tree','tree with 90/10 prior'),col=c('red','green'),lwd=3)
```

Performance Of Tree with and Without Priors



Converting Trees to Rules

This section uses a slightly modified version of Graham William's function from his excellent Desktop Data Mining Survival Guide. The function code is in the appendix.

I particularly find the rule format more usable as Tree plots are confusing and counterintuitive and hard to read.

```
#print rules for all classes
list.rules.rpart(fit1)
list.rules.rpart(fit2)

#custom function to only print rules for bad loans
listrules(fit1)
listrules(fit2)
```

(See appendix for code for both functions)

Sample output of select rules:

```
 Rule number: 16 [yval=bad cover=220 N=121 Y=99 (37%) prob=0.04]
   checking< 2.5
   afford< 54
   history< 3.5
   coapp< 2.5

 Rule number: 34 [yval=bad cover=7 N=3 Y=4 (1%) prob=0.06]
   checking< 2.5
   afford< 54
   history< 3.5
   coapp>=2.5
   age< 27

 Rule number: 18 [yval=bad cover=50 N=16 Y=34 (8%) prob=0.09]
   checking< 2.5
   afford< 54
   history>=3.5
   job>=2.5
```

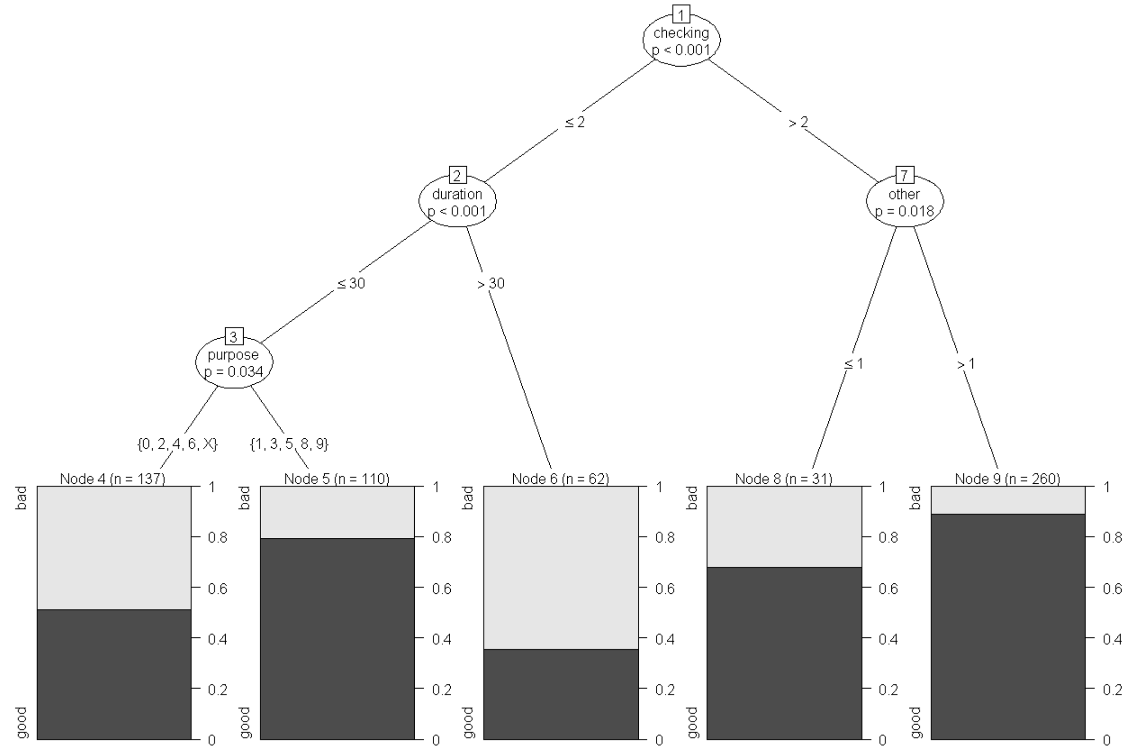
For other more robust recursive partitioning see Breiman's Random Forests and Zeleis and Hothorn's conditional inference trees and model based recursive partitioning which allows econometricians the ability to use theory to guide the development of tree logic (2007).

Conditional inference Trees

Conditional inference Trees are the next generation of Recursive Partitioning methodology and over comes the instability and biases found in traditional recursive partitioning like CARTtm and CHAID. Conditional Inference trees offer a concept of statistical significance based on bonferroni metric unlike traditional tree methods like CHAID. Conditional inference trees perform as well as Rpart and are robust and stable with statistically significant tree partitions being selected (Hothorn etal, 2007) .

```
#conditional inference trees corrects for known biases in chaid and cart
library(party)
cfit1<-ctree(good_bad~.,data=train)
plot(cfit1);
```

Conditional inference Tree Plot



ctree plot shows the distribution of classes under each branch.

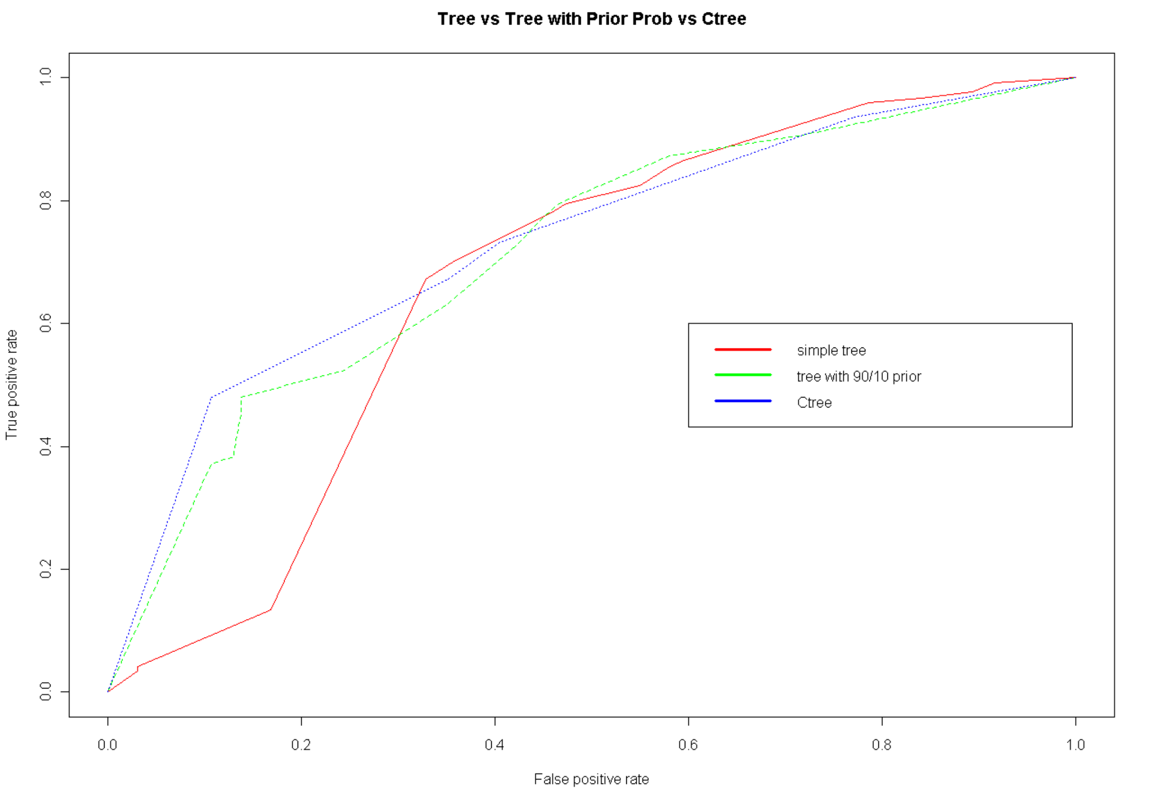
```
resultdfr <- as.data.frame(do.call("rbind", treeresponse(cfit1, newdata = test)))
```

```
test$tscore3<-resultdfr[,2]
```

```
pred9<-prediction(test$tscore3,test$good_bad)
perf9 <- performance(pred9,"tpr","fpr")
```

```
plot(perf5,col='red',lty=1,main='Tree vs Tree with Prior Prob vs Ctree');
plot(perf6, col='green',add=TRUE,lty=2);
plot(perf9, col='blue',add=TRUE,lty=3);
legend(0.6,0.6,c('simple tree','tree with 90/10
prior','Ctree'),col=c('red','green','blue'),lwd=3)
```

Performance of Trees vs. Ctrees



Using Random Forests

Given the known issues of instability of traditional recursive partitioning techniques Random Forests offer a great alternative to traditional credit scoring and offer better insight into variable interactions than traditional logistic regression

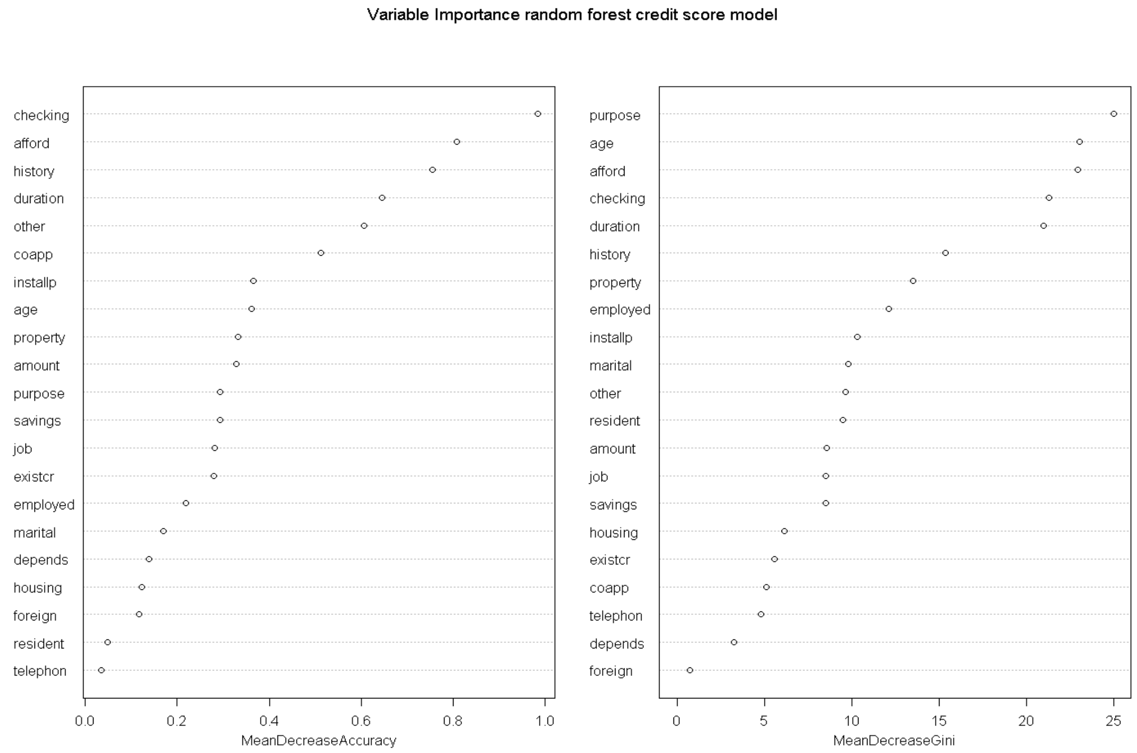
```
library(randomForest)
```

```
arf<-
randomForest(good_bad~.,data=train,importance=TRUE,proximit
y=TRUE,ntree=500, keep.forest=TRUE)
#plot variable importance
varImpPlot(arf)

testp4<-predict(arf,test,type='prob')[,2]
pred4<-prediction(testp4,test$good_bad)
perf4 <- performance(pred4,"tpr","fpr")

#plotting logistic results vs. random forest ROC
#plotting logistic results vs. random forest ROC
plot(perf,col='red',lty=1, main='ROC Logistic Vs. RF');
plot(perf4, col='blue',lty=2,add=TRUE);
legend(0.6,0.6,c('simple','RF'),col=c('red','blue'),lwd=3)
```

Using Random Forests to Improve Logistic Regression

Random forests are able detect interactions between variables which can add predictive value to credit scorecards. It is important to realize the financial affordability variables interactions are important to explore, as affordability as a construct is theoretically sound and it makes sense to see interactions with affordability data and other credit scorecard variables. Affordability in terms of free cash flows, liquid and liquefiable assets, and potential other borrowings comprise the ability of borrowers to pay back the loan (Overstreet et al, 1996). All ratios such as expense ratio, Loan to Value, and months reserves are interaction terms. Once one realizes this it makes sense to create and test interaction terms to add to the model using the affordability variables available. Using these terms in random forests and testing variable importance allows the most important interaction terms to be narrowed down and then it is a matter of testing various regression and variable groups to isolate the best performing interaction terms (Breiman, 2002). As the wholesale inclusion of interaction terms can lead to overfit (Gayler, 1995) the ability to narrow down a list of meaningful interaction terms is a valuable feature of random forests. Although this process can be automated, to date I have followed this approach manually. This would be a useful extension to have in R. For detailed analysis of issues with traditional scorecards and mixed results reported in the credit literature see Sharma etal 2009.

Using Random Forests Variable Importance Plot



```
#plot variable importance
varImpPlot(arf)
```

The variable importance plot lists variables in terms of importance using the decrease in accuracy metric, of loss of predictive power if the variable is dropped, vs. the importance in terms of Gini index, a measure of separation of classes. (See Strobl for the new and unbiased random forest variable importance metrics)

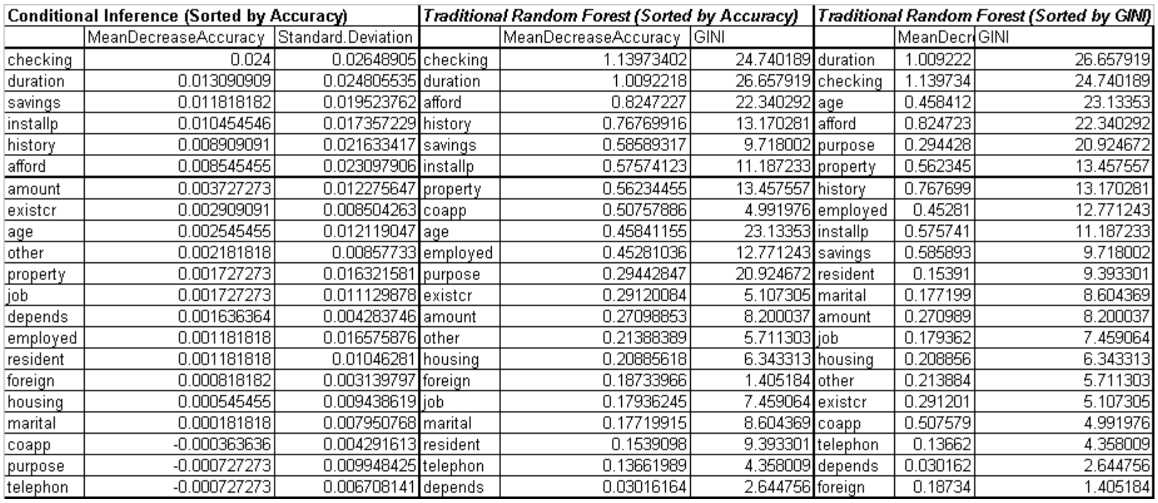
Although this process can be automated, to date I have followed this approach manually. Automating this process would be very easy in R. Using Random Forests Conditional Variable Importance

Recently Strobl etal have shown that Variable Importance measures can be biased towards certain variables and have developed an unbiased conditional Variable Importance measure (Strobl etal, 2009).

```
library(party)
set.seed(42)
crf<-cforest(good_bad~.,control = cforest_unbiased(mtry = 2, ntree = 50), data=train)
varimp(crf)
```

Cutting Edge Technique: Comparing Variable Important Sorted by Conditional Random Forests and traditional Forests Shows Which Variables Matter More

In this data for the top 5-6 variables the results stay the same. Traditional variable importance is more computationally efficient than conditional variable importance but can lead to some biases (Strobl etal, 2009). IN contrast the cforest package uses unbiased recursive partitioning based on conditional inference to produce truly unbiased variable importance, which can vary depending on the data set (Strobl etal, 2007).



Conditional Inference (Sorted by Accuracy)			*Traditional Random Forest (Sorted by Accuracy)*			*Traditional Random Forest (Sorted by GINI)*		
	MeanDecreaseAccuracy	Standard.Deviation		MeanDecreaseAccuracy	GINI		MeanDecr	GINI
checking	0.024	0.02648905	checking	1.13973402	24.740189	duration	1.009222	26.657919
duration	0.013090909	0.024805535	duration	1.0092218	26.657919	checking	1.139734	24.740189
savings	0.011818182	0.019523762	afford	0.8247227	22.340292	age	0.458412	23.13353
installp	0.010454546	0.017357229	history	0.76769916	13.170281	afford	0.824723	22.340292
history	0.008909091	0.021633417	savings	0.58589317	9.718002	purpose	0.294428	20.924672
afford	0.008545455	0.023097906	installp	0.57574123	11.187233	property	0.562345	13.457557
amount	0.003727273	0.012275647	property	0.56234455	13.457557	history	0.767699	13.170281
existcr	0.002909091	0.008504263	coapp	0.50757886	4.991976	employed	0.45281	12.771243
age	0.002545455	0.012119047	age	0.45841155	23.13353	installp	0.575741	11.187233
other	0.002181818	0.00857733	employed	0.45281036	12.771243	savings	0.585893	9.718002
property	0.001727273	0.016321581	purpose	0.29442847	20.924672	resident	0.15391	9.393301
job	0.001727273	0.011129878	existcr	0.29120084	5.107305	marital	0.177199	8.604369
depends	0.001636364	0.004283746	amount	0.27098853	8.200037	amount	0.270989	8.200037
employed	0.001181818	0.016575876	other	0.21388389	5.711303	job	0.179362	7.459064
resident	0.001181818	0.01046281	housing	0.20885618	6.343313	housing	0.208856	6.343313
foreign	0.000818182	0.003139797	foreign	0.18733966	1.405184	other	0.213884	5.711303
housing	0.000545455	0.009438619	job	0.17936245	7.459064	existcr	0.291201	5.107305
marital	0.000181818	0.007950768	marital	0.17719915	8.604369	coapp	0.507579	4.991976
coapp	-0.000363636	0.004291613	resident	0.1539098	9.393301	telephon	0.13662	4.358009
purpose	-0.000727273	0.009948425	telephon	0.13661989	4.358009	depends	0.030162	2.644756
telephon	-0.000727273	0.006708141	depends	0.03016164	2.644756	foreign	0.18734	1.405184

Strobl etal recommends the following when using variable importance in Random Forests:

> If all predictor variables are of the same type(for example: all continuous or all unordered categorical with the same number of categories), use either randomForest (randomForest) or cforest (party).
>
> While randomForest is computationally faster, cforest is safe even for variables of different types.
>
> For predictor variables of the same type, the Gini importance importance(obj,type = 2) or the permutation importance importance(obj, type = 1) available for randomForest and the permutation importance varimp(obj) available for cforest are all adequate importance measures.
>
> If the predictor variables are uncorrelated but of different types (for example: different scales of measurement, different numbers of categories), use cforest (party) with the default option controls = cforest_unbiased and the permutation importance varimp(obj).

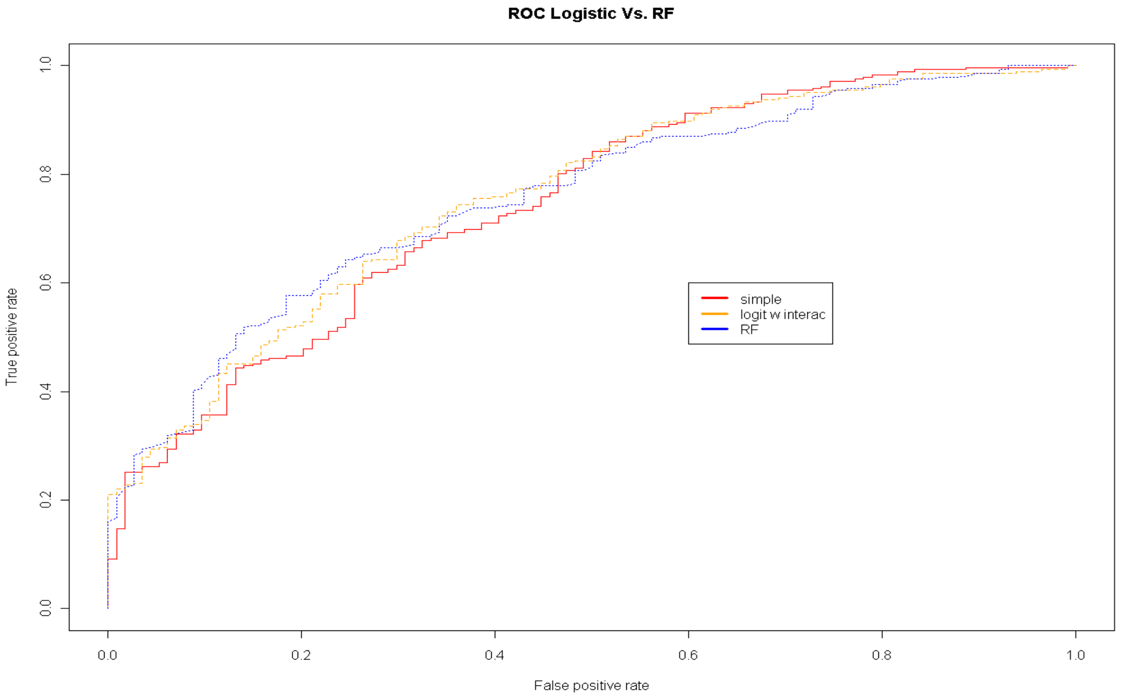
(Stroble etal, 2009).

Improving Logit Using Rand Forests

Using the approach described above of testing and trying interaction terms based on affordability and based on random forest variable importance measures one can improve logistic regression, using out of sample testing. The resulting model for the credit scoring data is as follows:

```
#model based on trial and error based on random forest
variable importance
m2<-glm(good_bad~.+history:other+history:employed
+checking:employed+checking:purpose,data=train,family=binom
ial())

test$score2<-predict(m2,type='response',test)
pred2<-prediction(test$score2,test$good_bad)
perf2 <- performance(pred2,"tpr","fpr")
plot(perf2)

#plotting logistic results vs. random forest ROC
plot(perf,col='red',lty=1, main='ROC Logistic Vs. RF');
plot(perf2, col='orange',lty=2,add=TRUE);
plot(perf4, col='blue',lty=3,add=TRUE);
legend(0.6,0.6,c('simple','logit w
interac','RF'),col=c('red','orange','blue'),lwd=3)
```

Performance of Using Random Forests to Improve Logistic regression



As you can see using the logistic regression with interaction terms improves the performance of logistic regression close to the order of Random Forests. The important thing to keep track of is trying and testing the interactions out of sample as adding too many interaction terms can lead to overfit. I would like to call the process of using affordability interactions using Random Forests and tuning logistic regression the Sharma method.

The German credit data is small but in larger commercial databases this approach will be less likely to overfit and requires judgment to oversee. Going to the trouble of using Random Forests and affordability related interactions if done properly can improve model performance by 5-10% in my experience. This can be a difference of millions of dollars on a multi-billion dollar credit risk portfolio. As such the trade off of exploring interaction terms to complexity is worthwhile, especially if you end up with a more accurate and well specified model.

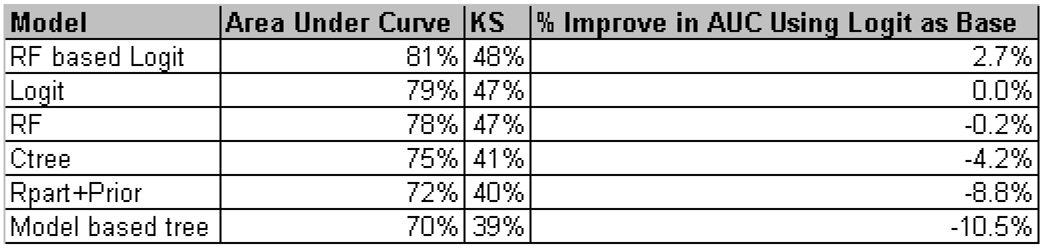
It is well known that credit variables have strong multi-collinearity and as such p values in regression can be misleading. Random Forest Variable Importance gives credit modelers an invaluable tool to safely explore important variable interactions in a well contained and manageable way.

Calculating Area under the Curve

The ROCR package has a function to calculate area under the Receiver operating curve. The AUC (area under the ROC curve) is another way to measure predictive power of models but as cautioned by David Hand it may mislead. One useful suggestion from Hand is to test out of sample and out of time and calibrate cut offs using changing date over time to calibrate scorecard cut offs.

```
# the following line computes the area under the curve for
# models
#simple model
performance(pred,"auc")
#random forest
performance(pred2,"auc")
#logit plus random forest interaction of affordability term
performance(pred4,"auc")
```

Comparing the methods discussed here shows that improving logit using random forests yielded the best results on the out of sample data. On larger data sets improvements on the order of 5-10% in model performance have been noted by the author.



Model	Area Under Curve	KS	% Improve in AUC Using Logit as Base
RF based Logit	81%	48%	2.7%
Logit	79%	47%	0.0%
RF	78%	47%	-0.2%
Ctree	75%	41%	-4.2%
Rpart+Prior	72%	40%	-8.8%
Model based tree	70%	39%	-10.5%

Cross Validation

As promulgated by John Maindonald using cross validation results based on sampling can provide more robust measures of accuracy (Maindonald, 2007).

```
#load Data Analysis And Graphics Package for R (DAAG)
library(DAAG)
#calculate accuracy over 100 random folds of data for
simple logit
h<-CVbinary(obj=m, rand=NULL, nfolds=100,
print.details=TRUE)
#calculate accuracy over 100 random folds of data for logit
+affordability interactions
g<-CVbinary(obj=m2, rand=NULL, nfolds=100,
print.details=TRUE)
```

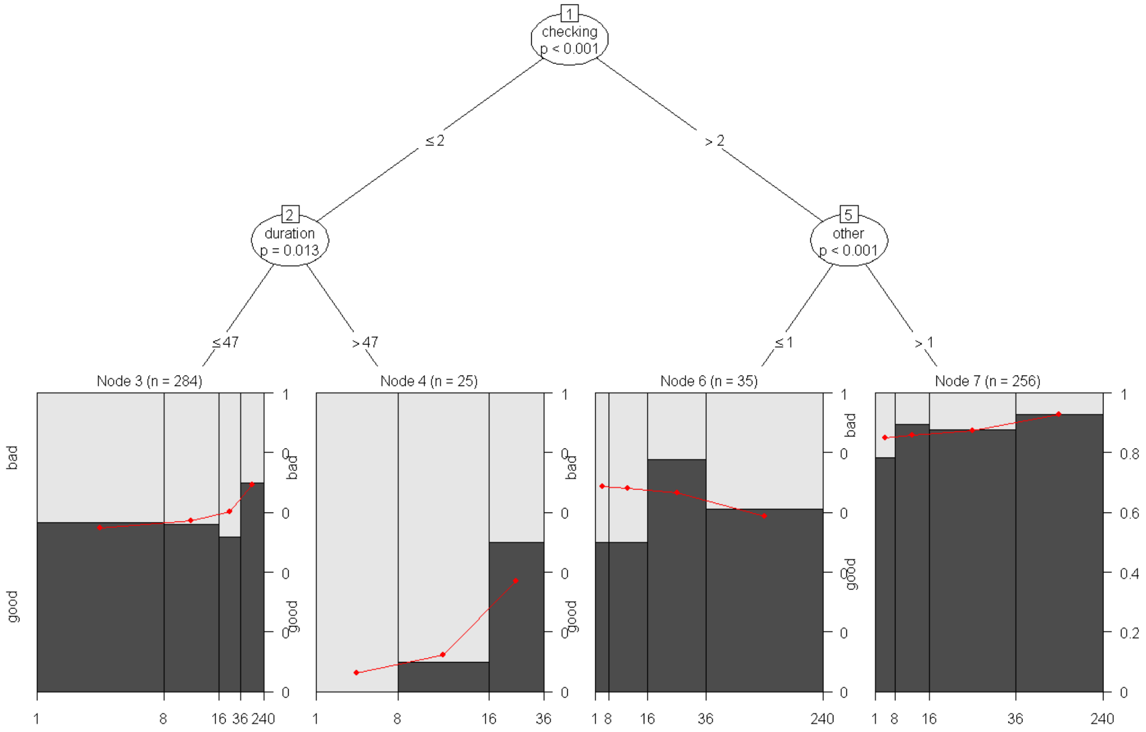
Cutting Edge techniques: Party Package(Unbiased Non parametric methods-Model Based Trees)

Strobl et al have shown that bias in random forests can be improved and corrected using conditional inference random forests using the Party package in R (Strobl et al, 2009). Also other powerful non parametric models which can use theory to drive the recursive partitioning and combining it with logistic regression and other models after segmenting data proves to be an important area which Model based Recursive Partitioning work has addressed (Zeleis, Hothorn and Hornik, 2006). This approach promises clarity and understanding expected in consumer credit scoring and adds more robust mechanisms for building recursive partitioning tree logic.

Model based trees work by 'fitting a parametric model to data, testing parameter instability over a set of partitioning variables supplied by the modeler and if there is overall parameter instability the tree model is split with the parameter with highest instability and this is repeated recursively' (Zeleis et al, 2006). The segmentation of using tree models, which are unstable without theory based segmentation, can lead to suboptimal models. As opposed to traditional segmentation (for e.g. see TransUnion White Paper, 2006 in references) the Model based trees allow econometrically based non parametric tree to be built. From a practical point of view this allows one to control the splits in more manageable way that trying a few split at a time in traditional trees to force ‘more sensible’ variables from the point of view of the client or modeler as the basis for splits and segmentation.

```
#model based recursive paritioning
library(party)
model<-mob(good_bad~afford |
amount+other+checking+duration+savings+marital+coapp+proper
ty+resident+amount,data=train,
model=glinearModel,family=binomial())
 plot(model)
```

Plot of Model based Tree with Logistic Regression

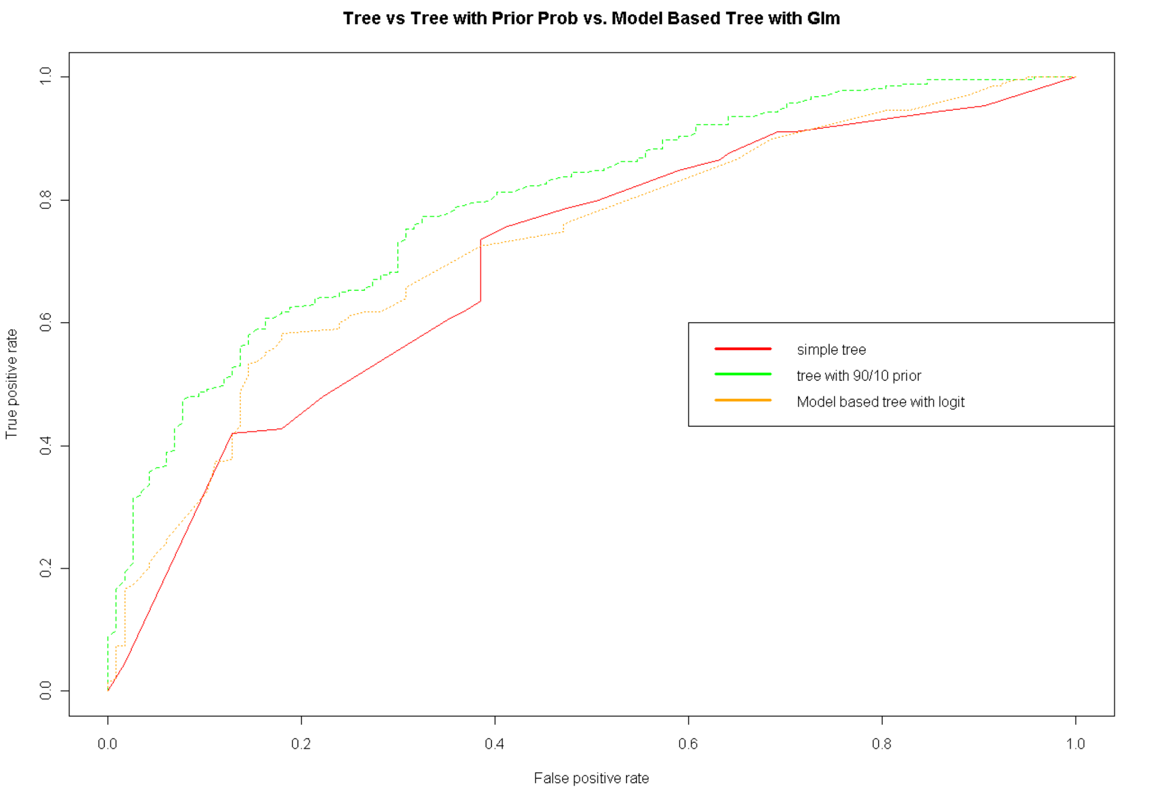


The graph at the end nodes shows how the affordability variables vary with good and bad loans based on the logistic regression built under each segment. Note how in each segment the distributions of affordability variable under the logistic regression built are different. This approach easily allows for the advantage of segment based scoring which has been heuristic and ad hoc (Kowalczyk, 2003).

```
test$mobscore<-predict(model, newdata = test, type =
c("response"))
```

```
pred7<-prediction(test$mobscore,test$good_bad)
perf7 <- performance(pred7,"tpr","fpr")

plot(perf5,col='red',lty=1,main='Tree vs Tree with Prior
Prob vs. Model Based Tree with Glm');
plot(perf4, col='green',add=TRUE,lty=2);
plot(perf7, col='orange',add=TRUE,lty=3);
legend(0.6,0.6,c('simple tree','tree with 90/10 prior',
'Model based tree with
logit'),col=c('red','green','orange'),lwd=3)
```

<u>Performance of Model based Tree vs. Other trees</u>



The Party package offers conditional inference trees, unbiased random forest variable importance and model based trees (Zelies etal, 2006). For the scope of this guide we review Model based Trees as they have benefit of combining unbiased recursive partitioning trees with logistic regression models (Zeileis, 2006). At this stage in my experience using the techniques I found them to be computationally expensive in the current form of the package but it is an important project for the future of credit scoring. Also using glm with Model trees can lead to failure to converge error common in logit as well. To understand with and play with all the options in MOB trees see the Party Package (2006).

Appendix of Useful Functions

Div

```
#function to divide to create interaction terms without divide by zero
div<-function(a,b) ifelse(b == 0, b, a/b)
```

List rules based on Rattle code (source http://www.togaware.com/datamining/survivor/Convert_Tree.html)

```
list.rules.rpart <- function(model)
{
  if (!inherits(model, "rpart")) stop("Not a legitimate rpart tree")
  #
  # Get some information.
  #
  frm     <- model$frame
  names   <- row.names(frm)
  ylevels <- attr(model, "ylevels")
  ds.size <- model$frame[1,]$n
  #
  # Print each leaf node as a rule.
  #
  for (i in 1:nrow(frm))
  {
    if (frm[i,1] == "<leaf>")
    {
      # The following [,5] is hardwired - needs work!
      cat("\n")
      cat(sprintf(" Rule number: %s ", names[i]))
      cat(sprintf("[yval=%s cover=%d (%.0f%%) prob=%0.2f]\n",
                  ylevels[frm[i,]$yval], frm[i,]$n,
                  round(100*frm[i,]$n/ds.size), frm[i,]$yval2[,5]))
      pth <- path.rpart(model, nodes=as.numeric(names[i]),
print.it=FALSE)
      cat(sprintf("   %s\n", unlist(pth)[-1]), sep="")
    }
  }
}
```

My modified version of the function needs to be tweaked depending on the data set. If the predictor variable is bad then following function will only print rules which classify bad loans. If your data has a different value then that line in the code needs to be changed for your use.

```
listrules<-function(model)
{


  if (!inherits(model, "rpart")) stop("Not a legitimate
rpart tree")
  #
  # Get some information.
  #
  frm     <- model$frame
  names   <- row.names(frm)
  ylevels <- attr(model, "ylevels")
  ds.size <- model$frame[1,]$n
  #
  # Print each leaf node as a rule.
  #
  for (i in 1:nrow(frm))
  {
    if (frm[i,1] == "<leaf>" & ylevels[frm[i,]$yval]=='bad')
    {
      # The following [,5] is hardwired - needs work!
      cat("\n")
      cat(sprintf(" Rule number: %s ", names[i]))
      cat(sprintf("[yval=%s cover=%d N=%.0f Y=%.0f (%.0f%%)
prob=%0.2f]\n",
                  ylevels[frm[i,]$yval], frm[i,]$n,
formatC(frm[i,]$yval2[,2], format = "f", digits = 2),
 formatC(frm[i,]$n-frm[i,]$yval2[,2], format = "f", digits
= 2),
                  round(100*frm[i,]$n/ds.size), frm[i,]
$yval2[,5]))
      pth <- path.rpart(model, nodes=as.numeric(names[i]),
print.it=FALSE)
      cat(sprintf("   %s\n", unlist(pth)[-1]), sep="")
    }
  }
}
```

References

Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science. Source of German Credit Data.

Baesens B; , Egmont-Petersen,M; Castelo, R, and Vanthienen, J. (2001)
Learning Bayesian network classifiers for credit scoring using Markov ChainMonte Carlo search. Retrieved from http://www.cs.uu.nl/research/techreps/repo/CS-2001/2001-58.pdf

Beling P, Covaliu Z and Oliver RM (2005). Optimal Scoring cutoff policies and efficient frontiers. J. Opl Res Soc 56: 1016–1029.

Bøttcher,SG; Dethlefsen,C (2003) Deal: A package for learning bayesian networks. Journal of Statistical Software. Retrieved from http://www.jstatsoft.org/v08/i20/paper

Breiman,L. (2002) Wald 2: Looking Inside the Black Box. Retrieved from www.stat.berkeley.edu/users/breiman/wald2002-2.pdf

Brown, Don (2005) *Linear Models* Unpublished Manuscript at University of Virginia.

Chang,KC; Fund, R.; Lucas,A; Oliver, R and Shikaloff, N (2000)
Bayesian networks applied to credit scoring. IMA Journal of Management Mathematics 2000 11(1):1-18

Gayler, R. (1995) Is the Wholesale Modeling of interactions Worthwhile? (Proceedings of Conference on CreditScoring and Credit Control, University of Edinburgh Management School, U.K.).

Gayler, R (2008) Credit Risks Analystics Occasional newsletter. Retrieved from http://r.gayler.googlepages.com/CRAON01.pdf

Gibson, J; Scherer, W.T. (2004) How to Do a Systems Analysis?

Hand, D. J. (2005). Good practice in retail credit score-card assessment. Journal of the Operational Research Society, 56, 1109–1117.

Kowalczyk, W (2003) Heuristics for building scorecard Trees.

Hothorn, T; Hornik, K & Zeilesi, A (2007) Unbiased Recursive Paritioning: A Conditional inference Framework. Retrieved from http://www.crc.man.ed.ac.uk/conference/archive/2003/abstracts/kowalkzyk.pdfhttp://statmath.wu.ac.at/~zeileis/papers/Hothorn+Hornik+Zeileis-2006.pdf

Maindonald, J.H. and Braun, W.J. (2007) "Data Analysis and Graphics Using R".
http://cran.ms.unimelb.edu.au/web/packages/DAAG/DAAG.pdf
.

Mays, Elizabeth. (2000) *Handbook of Credit Scoring*, Chicago: Glenlake,

Overstreet, GA; Bradly, E. (1996) Applicability of Generic Linear Scoring Models in the U.S credit-union environment. IMA Journal of Math Applied in Business and Industry. 7.

Pearl, J. (2000). Causality: Models, reasoning, and inference. Cambridge, England: Cambridge University Press.

Perlich, C; Provost, F; & Simonoff, J.S. (2003) Tree Induction vs. Logistic Regression: A Learning-Curve Analysis. Journal of Machine Learning Research 4 (2003) 211-255

Sharma, D; Overstreet, George; Beling, Peter (2009) Not If Affordability data adds value but how to add real value by Leveraging Affordability Data: Enhancing Predictive capability of Credit Scoring Using Affordability Data. CAS (Casualty Actuarial Society) Working Paper. Retrieved from http://www.casact.org/research/wp/index.cfm?fa=workingpapers

Sing,Tobias; Sander, Oliver; Beerenwinkel, Niko; & Lengauer, Thomas. (2005) ROCR: visualizing classifier performance in R. *Bioinformatics* 21(20):3940-3941

Schauerhuber,M; Zeileis,Achim ; Meyer, David; and Hornik, Kurt (2007) Benchmarking Open-Source Tree Learners in R/RWeka. Retrieved from http://epub.wu.ac.at/dyn/virlib/wp/eng/mediate/epub-wu-01_bd8.pdf?ID=epub-wu-01_bd8

Sing,Tobias; Sander, Oliver; Beerenwinkel, Niko; & Lengauer, Thomas. (2005) ROCR: visualizing classifier performance in R.Strobl, C., A.-L. Boulesteix, A. Zeileis, and T. Hothorn (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC *Bioinformatics* 21(20):3940-39418:25.

Strobl,C.; Hothorn,T& A. Zeileis (2009). Party on! A new, conditional variable importance measure
for random forests available in the party package. Technical report (submitted). Retrieved from http://epub.ub.uni-muenchen.de/9387/1/techreport.pdf.

Strobl,Carolin; Malley,J. and Tutz,G (2009) An Introduction to Recursive Partitioning. Retrieved from http://epub.ub.uni-muenchen.de/10589/1/partitioning.pdf

Therneau, T.M.; Atkinson, E.J.; (1997) An Introduction to Recursive Partitioning Using the RPART Routines Retrieved from www.mayo.edu/hsr/techrpt/61.pdf .

Tranunion (2006) SEGMENTATION FOR CREDIT-BASED DELINQUENCY MODELS White Paper. Retrieved from http://www.transunion.com/corporate/vantageScore/documents/segmentationr6.pdf

Velez, Jorge Ivan (2008) R-Help Support group email. http://tolstoy.newcastle.edu.au/R/e4/help/08/07/16432.html

Williams, Graham Desktop Guide to Data Mining Retrieved from http://www.togaware.com/datamining/survivor/ and http://datamining.togaware.com/survivor/Convert_Tree.html

Zeileis,A; Hothorn, T; and Hornik, K (2006) Evaluating Model-based Trees in Practice Achim Zeileis, Torsten Hothorn, Kurt Hornik. Retrieved from http://epub.wu-wien.ac.at/dyn/virlib/wp/eng/mediate/epub-wu-01_95a.pdf?ID=epub-wu-01_95a

Zeileis,A; Hothorn, T; and Hornik, K (2006) Party with the mob: Model-based Recursive Partitioning in R
Retrieved from http://cran.r-project.org/web/packages/party/vignettes/MOB.pdf For Party package in R. retrieved from http://cran.r-project.org/web/packages/party/party.pdf

Appendix: German Credit Data

http://ocw.mit.edu/NR/rdonlyres/Sloan-School-of-Management/15-062Data-MiningSpring2003/94F99F14-189D-4FBA-91A8-D648D1867149/0/GermanCredit.pdf

Variable Type Code Description

1

OBS#
Observation No.
Categorical

2

CHK_ACCT
Checking account status
Categorical
0 : < 0 DM
1: 0 < ...< 200 DM
2 : => 200 DM
3: unknown

3

DURATION
Duration of credit in months
Numerical

4

HISTORY
Credit history
Categorical
0: no credits taken
1: all credits at this bank paid back duly
2: existing credits paid back duly till now
3: delay in paying off in the past
4: critical account

5

NEW_CAR
Purpose of credit
Binary
car (new) 0: No, 1: Yes

6

USED_CAR
Purpose of credit
Binary
car (used) 0: No, 1: Yes

7

FURNITURE
Purpose of credit
Binary
furniture/equipment 0: No, 1: Yes

8

RADIO/TV
Purpose of credit

Binary
radio/television 0: No, 1: Yes

9
EDUCATION
Purpose of credit
Binary
education 0: No, 1: Yes

10
RETRAINING
Purpose of credit
Binary
retraining 0: No, 1: Yes

11
AMOUNT
Credit amount
Numerical

12
SAV_ACCT
Average balance in savings account
Categorical
0 : < 100 DM
1 : 100<= ... < 500 DM
2 : 500<= ... < 1000 DM
3 : =>1000 DM
4 : unknown

13
EMPLOYMENT Present employment since
Categorical
0 : unemployed
1: < 1 year
2 : 1 <= ... < 4 years
3 : 4 <=... < 7 years
4 : >= 7 years

14
INSTALL_RATE Installment rate as % of disposable
income
Numerical

15
MALE_DIV
Applicant is male and divorced
Binary
0: No, 1: Yes

16
MALE_SINGLE
Applicant is male and single
Binary
0: No, 1: Yes

17
MALE_MAR
Applicant is male and married or widower Binary

0: No, 1: Yes

Page 2

Var. # Variable Name

Description

Variable Type Code Description

18

CO-APPLICANT Application has a co-applicant

Binary

0: No, 1: Yes

19

GUARANTOR

Applicant has a guarantor

Binary

0: No, 1: Yes

20

TIME_RES

Present resident since - years

Categorical

0: <= 1 year

1<…<=2 years

2<…<=3 years

3:>4years

21

REAL_ESTATE

Applicant owns real estate

Binary

0: No, 1: Yes

22

PROP_NONE

Applicant owns no property (or unknown) Binary

0: No, 1: Yes

23

AGE

Age in years

Numerical

24

OTHER_INSTALL Applicant has other installment plan credit Binary

0: No, 1: Yes

25

RENT

Applicant rents

Binary

0: No, 1: Yes

26

OWN_RES

Applicant owns residence

Binary

0: No, 1: Yes

27

NUM_CREDITS Number of existing credits at this bank

Numerical

28

JOB
Nature of job
Categorical
0 : unemployed/ unskilled - non-resident
1 : unskilled - resident
2 : skilled employee / official
3 : management/ self-employed/highly
qualified employee/ officer

29

NUM_DEPEND Number of dependents
Numerical

30

TELEPHONE
Applicant has phone in his or her name Binary
0: No, 1: Yes

31

FOREIGN
Foreign worker
Binary
0: No, 1: Yes

32

RESPONSE
Fulfilled terms of credit agreement
Binary
0: No, 1: Yes
Binary
0: No, 1: Yes

Sample of Full R code in One Shot
(in case one wants to copy paste and run all the code at once)

```
data<-read.csv("C:/Documents and Settings/GermanCredit.csv")
data$afford<-data$checking*
data$savings*data$installp*data$housing

#code to convert variable to factor
data$property <-as.factor(data$property)
#code to convert to numeric
data$age <-as.numeric(data$age)
#code to convert to decimal
data$amount<-as.double(data$amount)
data$amount<-as.factor(ifelse(data$amount<=2500,'0-
2500',ifelse(data$amount<=5000,'2600-5000','5000+')))


d = sort(sample(nrow(data), nrow(data)*.6))
```

```
#select training sample
train<-data[d,]
test<-data[-d,]
train<-subset(train,select=-default)


#m<-
glm(good_bad~.*(checking+amount),data=train,family=binomial
())
#m<-step(m)


m<-glm(good_bad~.,data=train,family=binomial())
#m<-glm(good_bad~(checking)*.,data=train,family=binomial())
#m<-
glm(good_bad~checking:duration+.,data=train,family=binomial
())
#m<-glm(good_bad~.+history:other+history:employed
+checking:employed+checking:purpose,data=train,family=binom
ial())

library(ROCR)
#score test data set
test$score<-predict(m,type='response',test)
pred<-prediction(test$score,test$good_bad)
perf <- performance(pred,"tpr","fpr")
plot(perf)


max(attr(perf,'y.values')[[1]]-attr(perf,'x.values')[[1]])


#get results of terms in regression
g<-predict(m,type='terms',test)
#function to pick top 3 reasons
ftopk<- function(x,top=3){

     res=names(x)[order(x, decreasing = TRUE)][1:top]
     paste(res,collapse=";",sep="")
}

# Application of the function using the top 3 rows
topk=apply(g,1,ftopk,top=3)
# Result
#add reason list to scored tets sample
test<-cbind(test, topk)
```

```
library(randomForest)
arf<-
randomForest(good_bad~.,data=train,importance=TRUE,proximit
y=TRUE,ntree=500, keep.forest=TRUE)
#plot variable importance
varImpPlot(arf)

testp4<-predict(arf,test,type='prob')[,2]
pred4<-prediction(testp4,test$good_bad)
perf4 <- performance(pred4,"tpr","fpr")




m2<-glm(formula = good_bad ~ checking + duration + history
+ purpose +
    amount + savings + employed + installp + marital +
coapp +
    age + other + depends + telephon + foreign +
checking:amount +
    checking:duration + duration:amount + #checking:purpose
+
    purpose:amount + checking:savings + checking:employed +
checking:coapp +
    amount:age + checking:other + amount:other +
amount:depends +
    amount:telephon, family = binomial(), data = train)
#m2<-glm(good_bad~.+history:other+history:employed
+checking:employed+checking:purpose,data=train,family=binom
ial())

m2<-glm(good_bad~.+history:other+history:employed
+checking:employed+checking:purpose,data=train,family=binom
ial())

#m2<-glm(good_bad~.*afford,data=train,family=binomial())

test$score2<-predict(m2,type='response',test)
pred2<-prediction(test$score2,test$good_bad)
perf2 <- performance(pred2,"tpr","fpr")
plot(perf2)


#plotting logistic results vs. random forest ROC
plot(perf,col='red',lty=1, main='ROC Logistic Vs. RF');
plot(perf2, col='orange',lty=2,add=TRUE);
plot(perf4, col='blue',lty=3,add=TRUE);
legend(0.6,0.6,c('simple','logit w
interac','RF'),col=c('red','orange','blue'),lwd=3)


performance(pred,"auc")
performance(pred2,"auc")
performance(pred4,"auc")

library(DAAG)
h<-CVbinary(obj=m, rand=NULL, nfolds=100,
print.details=TRUE)
g<-CVbinary(obj=m2, rand=NULL, nfolds=100,
print.details=TRUE)

library(rpart)


fit1<-rpart(good_bad~.,data=train)
plot(fit1);text(fit1);
#test$t<-predict(fit1,type='class',test)

test$tscore1<-predict(fit1,type='prob',test)




pred5<-prediction(test$tscore1[,2],test$good_bad)
perf5 <- performance(pred5,"tpr","fpr")


fit2<-
rpart(good_bad~.,data=train,parms=list(prior=c(.9,.1)),cp=.
0002)
plot(fit2);text(fit2);

test$tscore2<-predict(fit2,type='prob',test)

pred6<-prediction(test$tscore2[,2],test$good_bad)
perf6<- performance(pred6,"tpr","fpr")

plot(perf5,col='red',lty=1,main='Tree vs Tree with Prior
Prob');
plot(perf6, col='green',add=TRUE,lty=2);
legend(0.6,0.6,c('simple tree','tree with 90/10
prior'),col=c('red','green'),lwd=3)
```

```
listrules<-function(model)
{


  if (!inherits(model, "rpart")) stop("Not a legitimate
rpart tree")
  #
  # Get some information.
  #
  frm     <- model$frame
  names   <- row.names(frm)
  ylevels <- attr(model, "ylevels")
  ds.size <- model$frame[1,]$n
  #
  # Print each leaf node as a rule.
  #
  for (i in 1:nrow(frm))
  {
    if (frm[i,1] == "<leaf>" & ylevels[frm[i,]$yval]=='bad')
    {
      # The following [,5] is hardwired - needs work!
      cat("\n")
      cat(sprintf(" Rule number: %s ", names[i]))
      cat(sprintf("[yval=%s cover=%d N=%.0f Y=%.0f (%.0f%%)
prob=%0.2f]\n",
                  ylevels[frm[i,]$yval], frm[i,]$n,
formatC(frm[i,]$yval2[,2], format = "f", digits = 2),
 formatC(frm[i,]$n-frm[i,]$yval2[,2], format = "f", digits
= 2),
                  round(100*frm[i,]$n/ds.size), frm[i,]
$yval2[,5]))
      pth <- path.rpart(model, nodes=as.numeric(names[i]),
print.it=FALSE)
      cat(sprintf("   %s\n", unlist(pth)[-1]), sep="")
    }
  }
}

listrules(fit1)
listrules(fit2)

library(deal)


  #make copy of train
 ksl<-train
 #discrete cnnot inherit from continuous so binary good/bad
must be converted to numeric for deal package
 ksl$good_bad<-as.numeric(train$good_bad)

#no missing values allowed so set any missing to 0
# ksl$history[is.na(ksl$history1)] <- 0

#drops empty factors
# ksl$property<-ksl$property[drop=TRUE]



ksl.nw<-network(ksl)
ksl.prior <- jointprior(ksl.nw)

#The ban list is a matrix with two columns. Each row
contains the directed edge
#that is not allowed.
#banlist <- matrix(c(5,5,6,6,7,7,9,8,9,8,9,8,9,8),ncol=2)
## ban arrows towards Sex and Year
#      [,1] [,2]
#[1,]     5    8
#[2,]     5    9
#[3,]     6    8
#[4,]     6    9
#[5,]     7    8
#[6,]     7    9
#[7,]     9    8

# note this a computationally intensive procuredure and if
you know that certain variables should have not
relationships you should specify
# the arcs between variables to exclude in the banlist

ksl.nw <- learn(ksl.nw,ksl,ksl.prior)$nw
#this step appears expensive so reset restart from 2 to 1
and degree from 10 to 1
result <-
heuristic(ksl.nw,ksl,ksl.prior,restart=1,degree=1,trace=TRU
E)
thebest <- result$nw[[1]]
savenet(thebest, "ksl.net")
print(ksl.nw,condposterior=TRUE)
```

```
#conditional inference trees corrects for known biases in chaid and cart
```

```
library(party)
cfit1<-ctree(good_bad~.,data=train)
plot(cfit1);


resultdfr <- as.data.frame(do.call("rbind", treeresponse(cfit1, newdata = test)))


test$tscore3<-resultdfr[,2]


pred9<-prediction(test$tscore3,test$good_bad)
perf9 <- performance(pred9,"tpr","fpr")

plot(perf5,col='red',lty=1,main='Tree vs Tree with Prior Prob vs Ctree');
plot(perf6, col='green',add=TRUE,lty=2);
plot(perf9, col='blue',add=TRUE,lty=3);
legend(0.6,0.6,c('simple tree','tree with 90/10
prior','Ctree'),col=c('red','green','blue'),lwd=3)


library(party)
set.seed(42)
crf<-cforest(good_bad~.,control = cforest_unbiased(mtry = 2, ntree = 50), data=train)
varimp(crf)

#note to use conditional functionality the 9.999 version of Party is needed and R .=2.9
varimp(crf, conditional=true)
# note this feature currently requires even for small data set  a lot computational
resources and memory
```

Why this is still cutting edge?
This requires a great deal of memory; for a small data set it can take up to 3 gig
Also variables with too many levels bog down conditional variable importance.
Regardless this is an important development and look to future versions of the package or more efficient and scalable implementations. If you have computing resources available then using a more accurate measure like conditional variable importance is advisable.

```
#model based recursive paritioning
library(party)
model<-mob(good_bad~afford |
amount+other+checking+duration+savings+marital+coapp+proper
ty+resident+amount,data=train,
model=glinearModel,family=binomial())
 plot(model)
```

```
test$mobscore<-predict(model, newdata = test, type =
c("response"))
pred7<-prediction(test$mobscore,test$good_bad)
perf7 <- performance(pred7,"tpr","fpr")

plot(perf5,col='red',lty=1,main='Tree vs Tree with Prior
Prob vs. Model Based Tree with Glm');
plot(perf6, col='green',add=TRUE,lty=2);
plot(perf7, col='orange',add=TRUE,lty=3);
legend(0.6,0.6,c('simple tree','tree with 90/10 prior',
'Model based tree with
logit'),col=c('red','green','orange'),lwd=3)
```