Technical Report: Frontier of Science in R

Some Experiments comparing logistic regression and random forests using Synthetic Data and the Interaction Miner Algorithm

Abstract
This paper uses synthetic datasets to classify the conditions in which random forest may outperform more traditional techniques such as logistic regression. We explore the theoretical implications of these experimental findings, and work towards building a theory based approach to data mining. During the course of these experiments we take the simulations where random forests dominate and add additional dimensionality to the data and run logistic regression using the additional attributes through the I* interaction miner algorithm outlined in Sharma 2011. Using the I* procedure with adequate amount of interaction terms the logistic regression can be made to match performance of random forests in the synthetic data sets where random forests dominate(Sharma,2011). This makes it seem the interaction miner algorithm along with some minimal sufficient amount of interaction and transformations allow logistic regression to match ensemble performance. This implies that without a certain amount of dimensionality in the data interaction miner and logistic regression do not benefit from the interactions. Breiman and other work shows Random Forests thrive on dimensionality that said from experiences with various data sets adding additional artificial dimensionality doesn't help forest (Breiman, 2001). There appears to be some minimum or necessary and sufficient amount of dimensionality after which more information cannot be extracted from the data. The good news is dimensionality can be created using the icreater function which add Tukey's re-expressions automatically to the data (log, negative reciprocal, and sqrt).

Dedicated to Mike Pratt, a true scientist and applied mathematician.

Acknowledgments:
Additional thanks Michael Pratt for the Idea of doing Synthetic Experiments to test out intuitions and for the structuring of the approach. He came up with the idea during discussion after watching Andrew Ng's lecture on Generative Algorithms from Andrew Ng's excellent Machine Learning course at Stanford. The online lectures on logistic regression and mixture models and distribution generating functions sparked the idea for Mike to posit this simple and elegant approach. All mistakes and confusions and aesthetics offenses are my own. Paraphrasing box we are all deluded but some delusions are more beneficial than others.

Goal: Examine simulated datasets to discern under what types of data sets random forest would outperform logistic regression. Then test out transformation and logistic regression tuning algorithm based on variable selection from random forest variable importance measures.

Metrics: Out of sample performance measured in binary prediction examples using out of sample Area under the curve.

Hypotheses:

- Logistic regression should outperform for exponential families: Gaussian, 2-dimensional Gaussian.
- Random forest should outperform when there are interaction effects in the data and for non Gaussian distributions.
    - For data sets with little interaction effects, random forests tend not to outperform logistic regression. While adding a few variables with interactions or interrelationships in the data such as affordability metrics and behavior variables random forest performance increases dramatically over logistic regression. This leads me to believe the performance of random forest is due to its ability to detect interactions in the data. In addition the fact that adding interaction terms in an intelligent manner to logistic regression yields equal performance supports this hypothesis as well. I find this explanation of random forest performance more compelling than vague hypotheses on bias and variance or margins.
- Adding interaction terms and variable transformations to logistic regression will allow logistic regression to yield equivalent or better performance as random forests
    - Reasons to believe this:
        - In practice on every credit related data set I have used this has been true from credit scoring to behavior scoring models. 6-7% performance advantage of random forests in application origination scoring and 15% advantage for behavior scoring can be added to logistic regression models by judicious addition of interaction terms. The more variables and richer the data set the greater the advantage random forests has over logistic regression. The majority of skill in predictive modeling comes in selecting appropriate variables and transformations. Over time theory, expertise, judgment and information theory have all been used to perform variable selection. Automating variable selection and using that to guide search of interaction terms reduces the search space to a manageable number. By pain-stakingly adding interaction terms using random forest variable importance I end up with logistic regression models equal to or better than random forests.

Experimental Design: For each simulation 10000 synthetic observations will be generated using simulation.

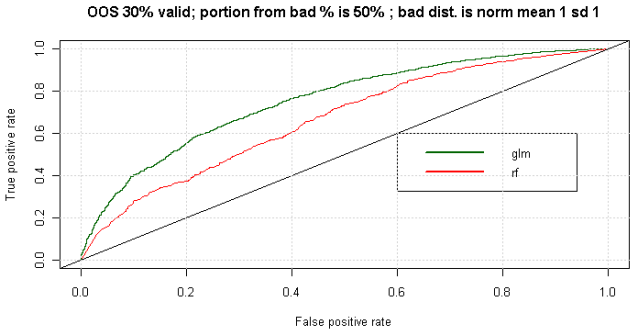
Synthetic Data set 1: (both exponential distributions)

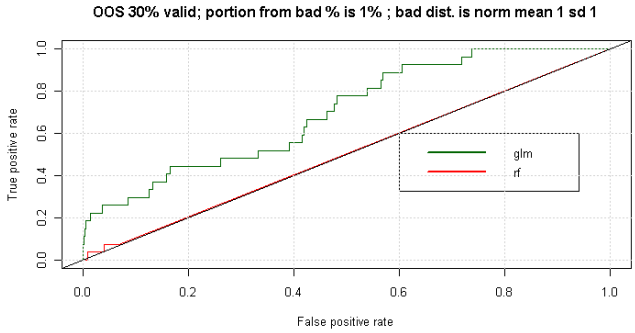
Gaussian distribution of with mean 0 and 1 with standard deviation of 1.
1 means success or good and 0 means bad or failure.

Prior expectation: For this simple simulation I would expect logistic regression to outperform random forest because there are no interaction terms and the logistic regression fits the decision boundary of an exponential distribution.

Setup: 2 versions of this scenario are tested one where randomly bads and goods are sampled from 2 gaussian distributions: one with mean 0 and standard deviation 1 and another with mean 1 and standard deviation 1. The probability of choosing the outcome of each distribution is set to 50% and in the second trial it is set to sampling bads only 1 % of the time.

Results:





Area Under Curve Out of Sample

% BAD	50%	1%
Model	AUC	
Logistic Regression	0.75	0.70
Random Forest	0.67	0.50

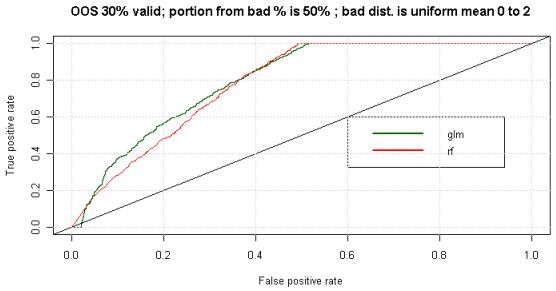
As expected logistic regression outperforms random forests in this scenario for both 50% bad sample and 1% bad sample.

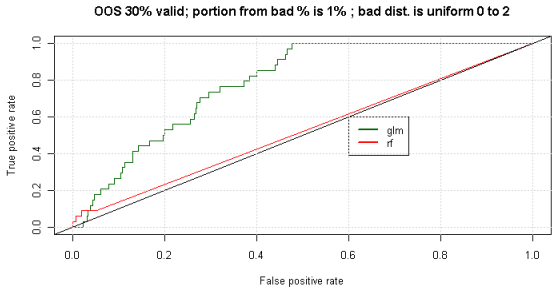
Synthetic data set 2: (one exponential distribution and one non exponential)

Gaussian and uniform distribution

Prior expectation: For this simple simulation I would expect logistic regression to outperform random forest because there are no interaction terms and the logistic regression fits the decision boundary of an exponential distribution.

Setup: 2 versions of this scenario are tested one where randomly bads and goods are sampled from 1 gaussian distributions: one with mean 0 and standard deviation 1 and another with uniform probability distribution between 0 and 2. The probability of choosing the outcome of each distribution is set to 50% and in the second trial it is set to sampling bads onely 1 % of the time.





Area Under Curve Out of Sample

% BAD	50%	1%
Model	AUC	
Logistic Regression	0.80	0.78
Random Forest	0.78	0.52

For the 50% bad simulation logistic regression and random forests perform similarly and for the 1% bad scenario random forest performs poorly.

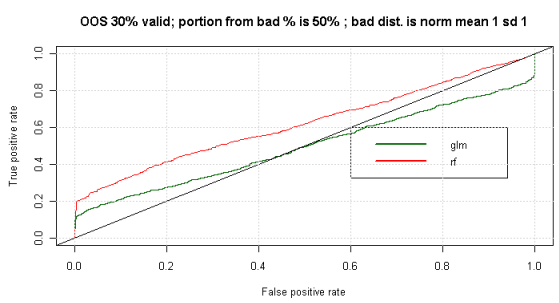
Synthetic data set 3:

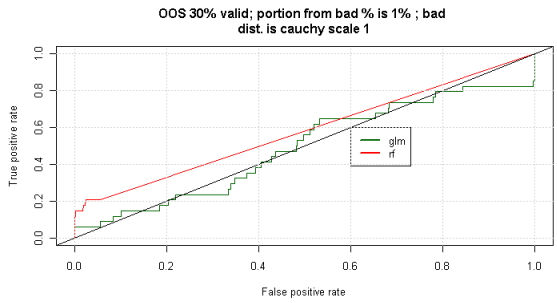
Gaussian and cauchy distribution

Prior expectation: For this simple simulation I would expect random forests to outperform logistic regression as the Cauchy distribution is not exponential and logistic regression might not fit it as well.

Setup: 2 versions of this scenario are tested one where randomly bads and goods are sampled from 1 gaussian distributions: one with mean 0 and standard deviation 1 and another with Cauchy distribution with scale =1. The probability of choosing the outcome of each distribution is set to 50% and in the second trial it is set to sampling bads onely 1 % of the time.

Results





Area Under Curve Out of Sample

% BAD	50%	1%
Model	AUC	
Logistic Regression	0.49	0.49
Random Forest	0.62	0.58

As expected random forests performs better in classifying the non-exponential distribution of bads.

Synthetic data set 4:

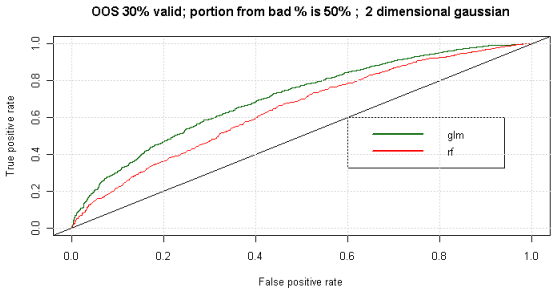
2 dimensional Gaussian with same variance/covariance in the 2 distributions

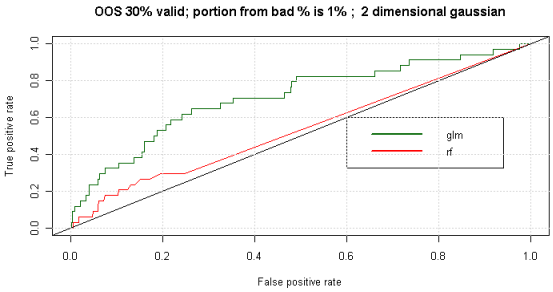
Prior expectation: For this simple simulation logistic regression to outperform as both distributions are from the exponential families.

Setup: 2 versions of this scenario are tested one where randomly bads and goods are sampled from 2 gaussian distributions: one with mean x,y with mean 0 and another 2 dimensional Gaussian with mean x,y of 1. Both variance and covariance between x,y for both 2-dimensional Gaussians is the same and is:

```
     [,1] [,2]
[1,]   10    3
[2,]    3    2
```

The probability of choosing the outcome of each distribution is set to 50% and in the second trial it is set to sampling bads only 1 % of the time.





Area Under Curve Out of Sample

% BAD	50%	1%
Model	AUC	
Logistic Regression	0.70	0.71
Random Forest	0.64	0.54

As expected logistic regression outperforms random forest with 2-dimensional Gaussians with the same variance/ covariance.

Synthetic data set 5:

2 dimensional Gaussian with different variance/covariance in the 2 distributions

Prior expectation: For this simple simulation logistic regression to outperform as both distributions are from the exponential families.

Setup: 2 versions of this scenario are tested one where randomly bads and goods are sampled from 2 gaussian distributions: one with mean x,y with mean 0 and another 2 dimensional Gaussian with mean x,y of 1. The variance and covariance between x,y for both 2-dimensional Gaussians is the different:
For goods the variance/covariance matrix is:

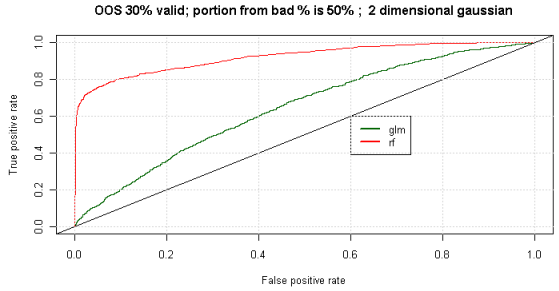
```
     [,1] [,2]
[1,]   10    3
[2,]    3    2
```

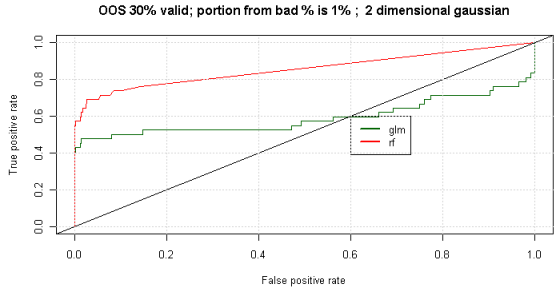
For bads the variance/covariance matrix is:

```
     [,1] [,2]
[1,]    1    6
[2,]    2    5
```

The probability of choosing the outcome of each distribution is set to 50% and in the second trial it is set to sampling bads onely 1 % of the time.

Results:



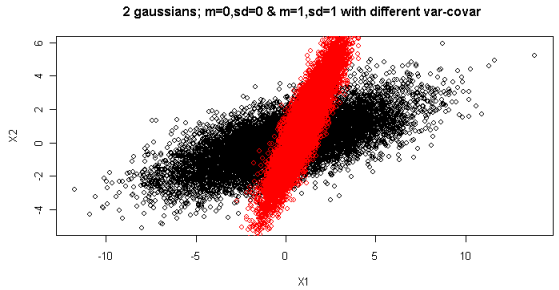


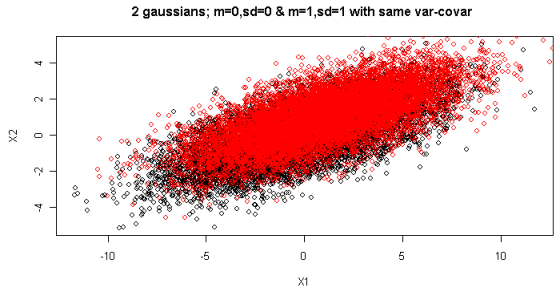
Area Under Curve Out of Sample

% BAD	50%	1%
Model	AUC	
Logistic Regression	0.64	0.59
Random Forest	0.92	0.86

Surprisingly random forest greatly outperforms logistic regression even though both distributions are exponential. The random forest is able to exploit the different variance and covariance of the good and bad distributions while logistic regression is not.

Looking at XY plot of the 2 different simulations of 2 dimensional Gaussian with same variance-covariance and different variance-covariance.





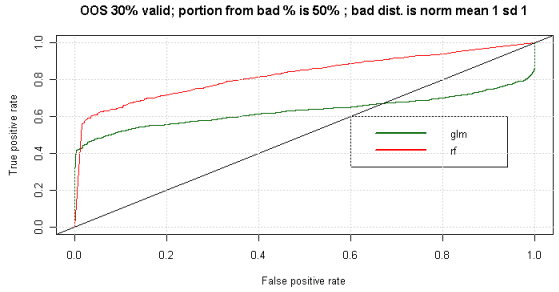
Synthetic data set 6:

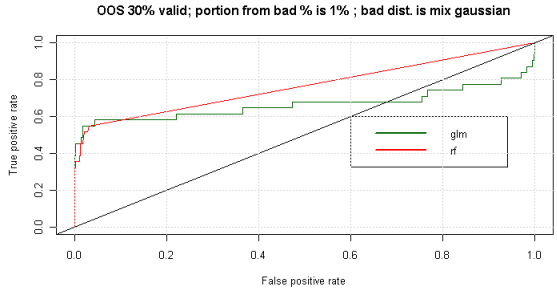
Gaussian vs. sum of Gaussians

Prior expectation: For this simple simulation I random forest to outperform as the sum of Gaussian is non-exponential and random forest might fit the more complex decision boundary better.

Setup: 2 versions of this scenario are tested one where randomly bads and goods are sampled from 2 gaussian distributions: one with mean 0 and standard deviation 1 and another with sum of 2 gaussians one with mean 1 and standard deviation 4 and another with mean .5 and standard deviation of 1. The probability of choosing the outcome of each distribution is set to 50% and in the second trial it is set to sampling bads onely 1 % of the time.

Results





Area Under Curve Out of Sample

% BAD	50%	1%
Model	AUC	
Logistic Regression	0.63	0.67
Random Forest	0.83	0.76

As expected random forests dominates logistic regression for sum of Gaussians distribution of bads.

Synthetic data set 7:

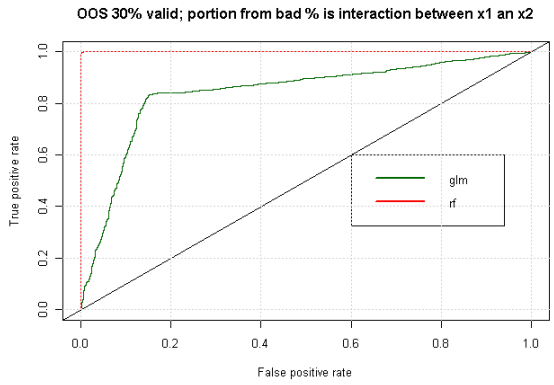
Data with interaction effect

Create random variables and an interaction variable and condition good vs. bad using interaction term.

Area Under Curve Out of Sample

Model	AUC
Logistic Regression	0.89
Random Forest	0.99

As expected the random forest outperforms logistic regression when there is an interaction between the variables.



Area Under Curve Out of Sample

Model	AUC
Logistic Regression	0.84
Random Forest	0.99

Round 2 of experiments using transformed input variables

We are going to take the simulations in which random forest dominated and add transformations and tune logistic regression by adding interaction terms and validating against a test data set and sort using random forest variable interaction to search for a logistic regression model with interaction terms and test against a validation data set.

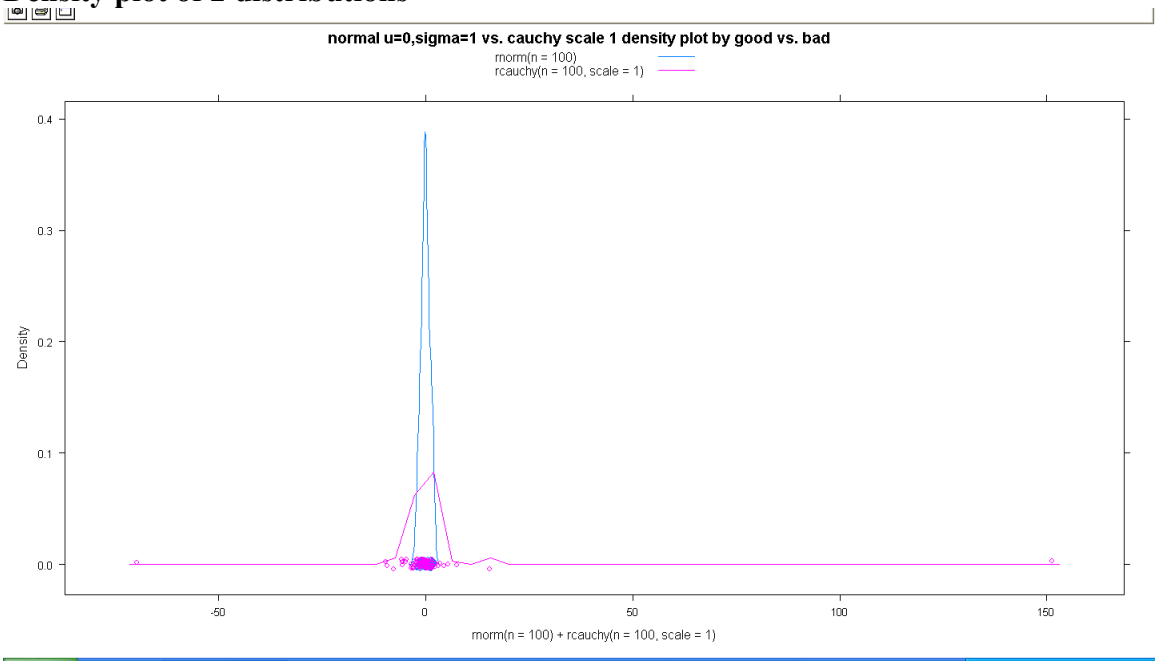
Simulations 3,5,6,7 are where random forest dominated logistic regression.

Icreater Function

Before proceeding it seems worthwhile describing the icreater function in R, it takes a data set and adds to it transformations or re-expressions based on Tukey's seminal work Exploratory Data Analysis (Tukey,1977, p54). As suggested by Tukey the log, square root, and negative reciprocals are computed. As regression will optimize + and – signs via coefficients combing this with the I* algorithm of testing interactions should make it possible to discover useful and predictive terms. Some of the most successful credit scoring attributes like the debt to income ratio (monthly expenses divided by income), loan to value (loan amount to value), and months liquid reserves (liquid assets divided by monthly payments) are all interaction terms. The idea behind I* and icreater is to have tools to be able to generate and discover useful interactions effectively and scientifically. In addition the rule posed by Maindonald was used to reduce the number of attributes created which suggest to only create log transformations when 'the ratio of largest to smallest data value is greater than 10' (Maindonald & Braun, 2003, p164). Random forests do not seem to need transformations or re-expressions added as the process of generating random trees and splits generates sufficient dimensionality to exploit latent and predictive interactions in data. It appears however for logistic regression transformations and interactions need to be added and searched through efficiently, which the I* algorithm allows for, when sufficient dimensionality is needed. That said icreater can creates thousands of columns and is best to use on a small random data set of 20000 or so and once random forests identify the salient transformations one can reduce the additional and un-necessary interactions to a smaller manageable number. In real data sets using icreater is usually not needed as there is sufficient dimensionality and interactions are present and random forests and I* algorithm to search through interactions is sufficient to match ensemble performance (Sharma, 2011). This experiment was sort of an extreme case setup with so few input variables and thus using the icreater function seemed like a good idea and it seems to work and show I* works even in the simple setting.

Simulation 3: Gaussian and cauchy distribution

Density plot of 2 distributions

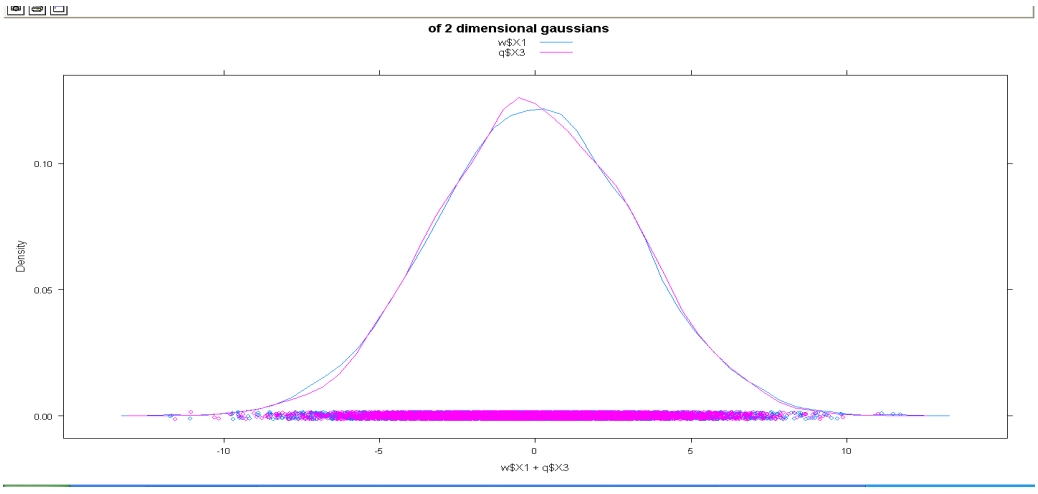


d1<-icreater(d1,"y");tuneGlmToRf(d1,"y");

" perf of orig glm no interactions oos 15% : 0.64353518821604"
" perf of logit tuned w interactions guided by rf oos 15% : 0.695908346972177"
" perf of orig rf oos 15% : 0.704909983633388"
"y~.+x:x.xrecip+xrecip:x.xrecip"

Simulation 5: 2 dimensional Gaussian with different variance/covariance in the 2 distributions

Density plot of 2 distributions

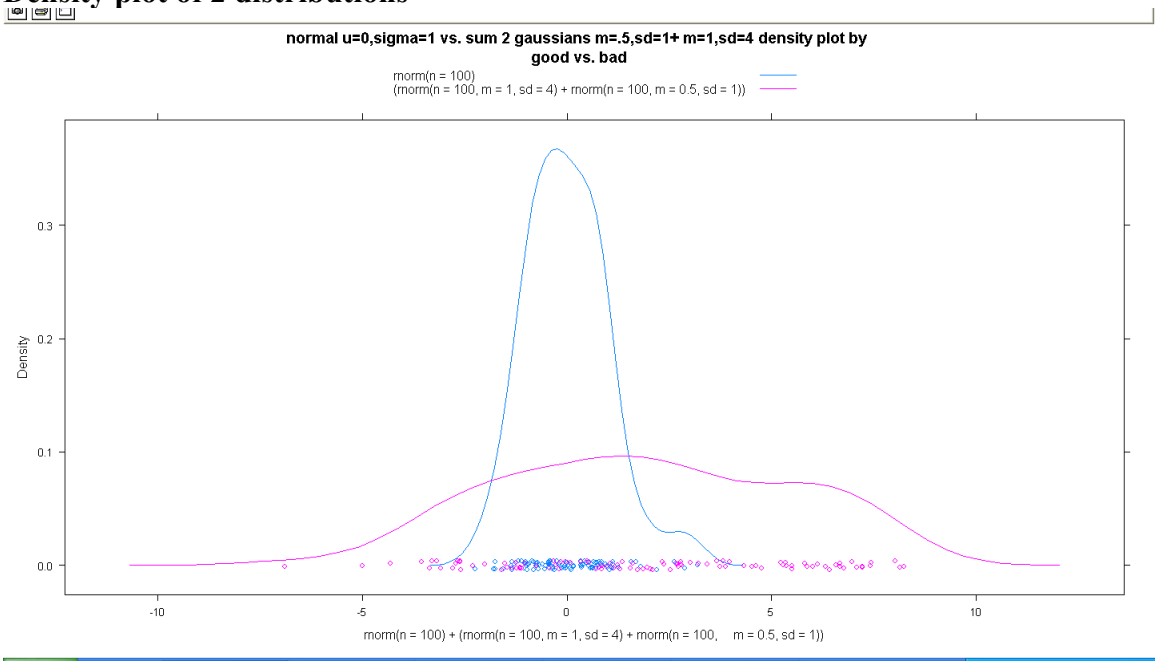


d2<-icreater(d2,"y");tuneGlmToRf(d2,"y");

" perf of orig glm no interactions oos 15% : 0.887924016282225"
" perf of logit tuned w interactions guided by rf oos 15% : 0.96336499321574"
" perf of orig rf oos 15% : 0.933785617367707"
"y~.
+x1:x1recip+x1recip:x1.x1recip+x1recip:x1.x2recip+x1recip:x2.x1recip+x1recip:x2.x2recip+x1recip:x1recip.x2recip+x1:x2.x1recip+x1.x2:x2.x1recip+x1:x1.x2recip+x2:x1.x2recip+x2recip:x1.x2recip+x1.x2:x1.x2recip+x1.x2recip:x2.x1recip+x1.x2recip:x1recip.x2recip+x1:x2recip+x1.x1recip:x2.x2recip+x1.x1recip:x1recip.x2recip+x1:x2+x2:x2recip+x2:x1.x1recip+x2:x2.x1recip+x2:x2.x2recip+x2recip:x1.x2+x2recip:x2.x1recip+x2recip:x2.x2recip"

Simulation 6: Gaussian vs. sum of Gaussians

Density plot of 2 distributions

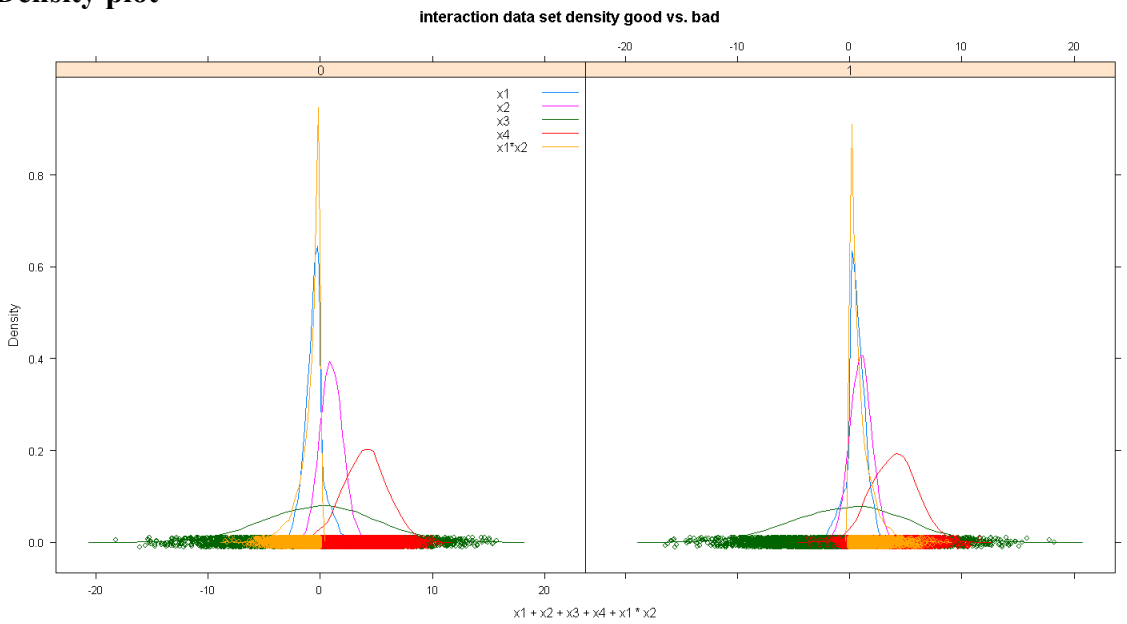


d3<-icreater(d3,"y");tuneGlmToRf(d3,"y");

" perf of orig glm no interactions oos 15% : 0.739625949736996"
" perf of logit tuned w interactions guided by rf oos 15% : 0.799532437171245"
" perf of orig rf oos 15% : 0.747516072472238"
"y~.+x:xrecip+xrecip:x.xrecip"

Simulation 7: interaction terms data

Density plot



a1<-icreater(a,"y");tuneGlmToRf(a1,"y")
Number of Fisher Scoring iterations: 15

" perf of orig glm no interactions oos 15% : 0.992629716981132"
" perf of logit tuned w interactions guided by rf oos 15% : 1"
" perf of orig rf oos 15% : 1"
"y~.
+x1:x1.x2+x3:x1.x2+x3recip:x1.x2+x1.x2:x2.x1recip+x1.x2:x2.x2recip+x1.x2:x4.x4recip+x2:x3recip+x2:x1.x3"

Conclusions

- Random forests out of the box outperform logistic regression when there is relationship or structure in data e.g. non exponential distribution of data or interaction effects present in the data. Random forests are able to create trees and using this mechanism generate dimensionality which they exploit to extract more information in the data effectively.
- Logistic regression can be tuned to perform equally well or sometimes better than ensemble methods like random forest using judicious use of interaction terms and variable transformations.
- A certain amount of dimensionality as Breiman posits is helpful in improving models and random forests benefit from it and create dimensionality via trees and beyond a certain threshold additional dimensionality does not help (Breiman, 2001). I conclude this because adding additional interactions does not seem to help random forests. Random forests do not seem to need transformations or re-expressions added as the process of generating random trees and splits generates sufficient dimensionality to exploit latent and predictive interactions in data.

Using random forests to tune variable selection for regression seems to be very effective. Every credit scoring and behavioral data set I have seen shows random forests to outperform logistic regression. Using variable selection derived from variable importance from the random forest allows for intelligent theory and information informed intelligent search for interactions. The addition of variable transformations does not improve random forest performance as random forests are able to discover structure and relationships in the data that traditional regression methods fail to. The value of a variable is not in and of itself but its overall prediction performance in relation to other relevant variables which it has interactions with. Creating automated transformations added dimensionality to the data set which can be exploited by intelligent allows regression models to perform at the same level as ensemble methods.

This union is important as we live in an era of growing models but a lack of theory to understand how to unify models and extract knowledge from them.

Automated intelligent expert systems to automate model development is an idea whose time has finally come. To make it a reality we need more tinkering and exploring and understanding of many models and work to unify them. The development of an expert system to perform model development, analysis and modeling is the next step in the evolution of predictive modeling. Variable selection from random forest variable importance measures allows for efficient and effective automated search for optimizing logistic regression models.

This is important because it allows us to see that the performance advantage of random forests is derived from its ability to accurately provide knowledge about variable predictive power which includes predictive power from interactions between variables. This is possible because ensembles methods overcome the hurdle of greedy optimization done by both regression and recursive partitioning.

References

Breiman L. (2001) Statistical modeling: the two cultures. Stat Sci;16:199-231.

Maindonald J, Braun J: (2003) Data Analysis and Graphics Using R. Cambridge: Cambridge University Press;

Ng,Andrew (2011) Generative Algorithms Lecture Notes. http://www.stanford.edu/class/cs229/notes/cs229-notes2.pdf and http://www.stanford.edu/class/cs229/materials.html

Sharma, D. (2011). Experiments comparing logistic regression and random forests using Synthetic Data and the Interaction Miner Algorithm. Technical Report: Frontier of Science in R

Tukey, J. W. (1977). Exploratory data analysis. Reading, MA: Addison-Wesley

Appendix of R Code

```
# Function takes data set and returns transformed data set
# with greater dimensionality using log, square root and
#inverse reciprocal negative
icreater<- function (model, target)

{
cc<-model

pos<-which(names(model)==target);

y<- cc[,pos]
x<-subset(cc,select=- c(pos))
counter<- length(names(x))
 for (i in 1:counter)
 {
 if (is.numeric(x[,i]) )

 {

     if  (length(levels(as.factor(x[,i]))) >=12)
     {
     #create one var per transform
     x$log1<- log( abs(x[,i] +1))
     x$root1<- sqrt( abs(x[,i]))
     x$negrec<- -1/(x[,i]+1)

     names(x)[length(names(x))-2] <- paste( names(x)[i],
"log", sep="")
     names(x)[length(names(x))-1] <- paste( names(x)[i],
"sqrt", sep="")
     names(x)[length(names(x))]  <- paste( names(x)[i],
"recip", sep="")

     if (max(na.omit(x[,i]))/(1+min(na.omit(x[,i]))) <10)
     {
          x<-x[-match(paste( names(x)[i], "log", sep=""),
names(x))]
          x<-x[-match(paste( names(x)[i], "sqrt", sep=""),
names(x))]
     }

     }
}
```

```
}



data1<-cbind(x,y)

d = sort(sample(nrow(data1), min(nrow(data1)* 1,1600)))

data1<-data1[d,]


rm(x)
rm(y)
gc()

library(fSeries)
data1<- data.frame(substituteNA(data1, type ="zeros"))


for (i in 1:(length(names(data1))-1))

 {
 {
     data1[,i] <- as.numeric( data1[,i])
}
}

y1<- data1$y

ff <- y~ .^2
utils::str(m <- model.frame(ff, data1))
mat <- model.matrix(ff, m)

mat<-data.frame(mat)


datafinal<-cbind(mat,y1)

names(datafinal)[which(names(datafinal)=="y1")]<-target
gc()
return (datafinal)

}
```

```
#too slow and perf worse for interaciton terms for cc
ctreePerf  <-function(data, origdata, formula, tsize=.8)
{
set.seed(79)

data<- cdata
origdata<- c
formula <- good_bad~.
tsize=.8

library(ROCR)
library(party)

d = sort(sample(nrow(data), nrow(data)*tsize))
train<-data[d,]
test<-data[-d,]

data<-train
#,parms=list(prior=c(.8,.2))


m1<-ctree(formula,data=data[,c(2:(length(names(origdata))-
1),length(data)) ])
m2<-ctree(formula,data=data[,c(2:length(data)) ])



par(mfrow=c(2,2))


y<- test[,length(test)]

resultdfr <- as.data.frame(do.call("rbind",
treeresponse(m1, newdata = test)))
test$score1<-resultdfr[,2]
pred1<-prediction(test$score1,y)
perf1 <- performance(pred1,"tpr","fpr")

resultdfr <- as.data.frame(do.call("rbind",
treeresponse(m2, newdata = test)))
test$score2<-resultdfr[,2]
pred2<-prediction(test$score2,y)
perf2 <- performance(pred2,"tpr","fpr")

plot(perf1, col='red', main="OOS 80% based on 20% training
set model with interactions vs. original model ROC")
plot(perf2, col='blue',add=TRUE);
```

```
legend(0.6,0.6,c('orig data' ,'data
+interactions'),col=c('red','blue'),lwd=2)

gc()
print(paste('OOS AUC for model using original data
is :',attributes(performance(pred1, "auc"))$y.values[[1]]
[1]))
print(paste('OOS AUC for model using transformations and
interactions data is :',attributes(performance(pred2,
"auc"))$y.values[[1]][1]))

barplot(cbind(attributes(performance(pred1, "auc"))
$y.values[[1]][1],attributes(performance(pred2, "auc"))
$y.values[[1]][1]),names.arg=c("original
data","data+interactions"), main="OOS AUC by
model",col=c("red","darkblue"),beside=TRUE)


gc()
#print(m1)
#print(m2)
plot(m1)
plot(m2)

}


return_RF_Formula <-function (arf1,n=7)
{
imp <- importance(arf1, class = null, scale = TRUE, type =
NULL)
e<-imp <- imp[, -(1:(ncol(imp) - 2))]
#sort(e[,1], dec=TRUE)

#only keep variables with info length(names(e[,1][e[,1]>0]))

a<-""
for (i in 1: min(n,length(sort(e[,1][e[,1]>0.025],
dec=TRUE))))
 {
 a<-paste(a,(names(sort(e[,1][e[,1]>0.025], dec=TRUE))
[i]),sep="")

 if (i< min(n,length(sort(e[,1][e[,1]>0.025], dec=TRUE))))
 {
 a<-paste(a,("+"),sep="")
```

```
 }


  }
return (a)
}









getRfTerms<-function(rf, n=10)
{




rfterms<-character(0)

imp <- importance(rf, class = null, scale = TRUE, type =
NULL)
e<-imp <- imp[, -(1:(ncol(imp) - 2))]

for (i in 1: min(n,length(sort(e[,1], dec=TRUE))))
 {
print (paste("forest: " ,substr(names(sort(e[,1],
dec=TRUE))[i],1,nchar(names(sort(e[,1], dec=TRUE))
[i])),sep=""))
rfterms[length(rfterms)+1]= substr(names(sort(e[,1],
dec=TRUE))[i],1,nchar(names(sort(e[,1], dec=TRUE))[i]))
}

return (rfterms)

}

getGlmTerms<- function (m)
{
x <- array(dim=c(length( names(summary(m)
$coefficients[,1])),3))

for (i in 1:length( names(summary(m)$coefficients[,1])))
```

```
 {
 #print(names(summary(m)$coefficients[,1])[i])
 #print(summary(m)$coefficients[i])
#print (summary(m)$coefficients[i,4])

x[i,1]<-names(summary(m)$coefficients[,1])[i]
x[i,2]<-summary(m)$coefficients[i]
x[i,3]<-summary(m)$coefficients[i,4]

 }
y<-data.frame(x,stringsAsFactors = FALSE)
return (y)
}


barplotwithtext<- function(datainplot=c(1,2,3), y=c(0,1))
{
datainplot<- c(1,2,3)
w<-barplot(datainplot,ylim=y);

text(cex=.5, x=w, y=datainplot+par("cxy")[2]/2,
round(datainplot,2), xpd=TRUE)

}



# input data, name of target classification variable ,
n=number of top most predictive variables to use for
interaction mining based on rf
# rfsample size of classes to sample from for large
dataset; usually 500 works well .
# sample a boolean flag for large datasets to sample 50k
obs for model building etc.
# 2 way interacs flag; slower not much better;
#formStat is a string model formula of a base existing
model with interactions you may want to start with as  a
starting point for model improvement
# you can run the function once and feed back the resulting
model to see if more predictive power can be added using
additional interaction terms using the
# algorithm recusively one itself.

tuneGlmToRf<-function(data, target,n=7,rfsample=0,
sample=FALSE, nway=1, formStart=NA)
{
```

```
if (nrow(data)>52000 & sample==TRUE)
{
d = sort(sample(nrow(data), 52000))
data<-data[d,]

if (rfsample==0) { rfsample=500 }
}


options(warn=-1)
library(fSeries)

data<- data.frame(substituteNA(data, type ="zeros"))
for (i in 1:(length(names(data))))

 {
 {
        if (names(data[i]) != target)
          {
               data[,i] <- as.numeric( data[,i])
               print (1)
          }
          else { data[,i] <- as.factor( data[,i]) }
}
}



formula <- paste(target,"~.",sep="")
 formularf<- as.formula(formula)

d = sort(sample(nrow(data), nrow(data)*.7))
#select training sample
train<-data[d,]
test<-data[-d,]
w= sort(sample(nrow(test), nrow(test)*.5))
test<-test[w,]
validate<-test[-w,]

library(randomForest)
library(ROCR)
set.seed(42)

if (rfsample > 0)
{
rf <- randomForest(formularf,
      data=train,
```

```
      ntree=500,
      mtry=3,
      importance=TRUE,
     sampsize=c(rfsample,rfsample),
      na.action=na.roughfix,
      replace=FALSE)
}

else
{
rf <- randomForest(formularf,
      data=train,
      ntree=500,
      mtry=3,
      importance=TRUE,
      na.action=na.roughfix,
      replace=FALSE)
}


#subset train test and validate to terms that have
predictive value
imp <- importance(rf, class = null, scale = TRUE, type =
NULL)
e<-imp <- imp[, -(1:(ncol(imp) - 2))]
data<-subset(data,select=c(names(e[,1][e[,1]>0]),target))
train<-subset(train,select=c(names(e[,1][e[,1]>0]),target))
test<-subset(test,select=c(names(e[,1][e[,1]>0]),target))
validate<-subset(validate,select=c(names(e[,1]
[e[,1]>0]),target))

rfterms<-getRfTerms(rf,n)
list_of_terms<-character(0)

form<- paste(formula, " * ",sep="")
if (!is.na(formStart))
{
form<- paste(formStart, " * ",sep="")
}


for (k in 1:length(rfterms))
{
print(paste(form,rfterms[k],sep=""))

if ( k<=nway | nway ==1)
{
```

```
print("a")

m<-
glm(paste(form,rfterms[k],sep=""),data=train,family=binomia
l)
print("b")

}

if (k>nway & nway>1)
{
f<- paste(form,rfterms[k],sep="")
for (p in 1:nway)
{
f<-paste(f, " * ", rfterms[k-p],sep="")
print(f)
}

#paste(form,rfterms[k]," * ", rfterms[k-1], " * ",
rfterms[k-2]," * ",rfterms[k-3]," * ",rfterms[k-4]," *
",rfterms[k-5],  sep="")
m<-glm(f,data=train,family=binomial)
print("c")

}
print("d")
print(summary(m))
y <- getGlmTerms(m)
y<- na.omit(y)
rm(m)
gc()

y$X2<- as.numeric(y$X2)
y$X3<- as.numeric(y$X3)

for (j in 1: nrow(y))
 {
print("Adding term")

 if ((y[j,3]<=.25 |  runif(1, 0, .61)< .16 | TRUE ) &
(regexpr(":", y[j,1]) > 0))

 {

     #print(names(y[1,j]))
     list_of_terms[length(list_of_terms)+1]= y[j,1]
 }
```

```
 }


}

############ do the same above by updating formula to
include 1 way interactions against final model



#################



maxperf<-0

morig<-glm(formula,data=train,family=binomial)

test$score1<-predict(morig,type='response',test)
pred1<-prediction(test$score1,test[target])
perf1 <- performance(pred1,"tpr","fpr")

validate$score1a<-predict(morig,type='response',validate)
pred1a<-prediction(validate$score1a,validate[target])
perf1a <- performance(pred1a,"tpr","fpr")

perforig<-attributes(performance(pred1a, "auc"))
$y.values[[1]][1]

maxperf<-attributes(performance(pred1, "auc"))
$y.values[[1]][1]

print(maxperf)

rm(morig)
gc()

list2<-character(0)
print( paste("about to try ",length(list_of_terms), "
interactions.",sep=""))
for (i in 1: length(list_of_terms))
{

if (length(list2)==0)
```

```
{
mfin<-
glm(paste(formula,list_of_terms[i],sep="+"),data=train,fami
ly=binomial)
print(paste(formula,list_of_terms[i],sep="+"))
}
else
{
finform<- formula
for (j in 1: length(list2))
{

finform<- paste(finform, list2[j],sep="+")

}
print(i)
print(finform)
print(paste(finform, list_of_terms[i],sep="+"))
mfin<-glm(paste(finform,
list_of_terms[i],sep="+"),data=train,family=binomial)
}

test$score2<-predict(mfin,type='response',test)
pred2<-prediction(test$score2,test[target])
perf2 <- performance(pred2,"tpr","fpr")

print(maxperf)
print(attributes(performance(pred2, "auc"))$y.values[[1]]
[1])

if (attributes(performance(pred2, "auc"))$y.values[[1]][1]
> maxperf)
{

 list2[length(list2)+1]= list_of_terms[i]
  maxperf<- attributes(performance(pred2, "auc"))
$y.values[[1]][1]

}



rm(mfin)
gc()

}
```

```
if (length(list2)>0)
{
mfin<-glm(finform,data=train,family=binomial)

print(paste("explored ", length(list_of_terms) ,"
interactions and ended up with following optimal
interactions : ", length(list2),sep="" ))
print("optimal glm model is ")
print(finform)
print(summary(mfin))

#compare on validation set
validate$score3<-predict(mfin,type='response',validate)
pred3<-prediction(validate$score3,validate[target])
perf3 <- performance(pred3,"tpr","fpr")
optglmperf<-attributes(performance(pred3, "auc"))
$y.values[[1]][1]

print (paste(" perf of orig glm no interactions oos 15% :
",perforig,sep=""))
print (paste(" perf of logit tuned w interactions guided by
rf oos 15% : ", optglmperf,sep=""))


validate$p4<-predict(rf,validate,type='prob')[,2]
pred4<-prediction(validate$p4,validate[target])
perf4 <- performance(pred4,"tpr","fpr")
rfperf<-attributes(performance(pred4, "auc"))$y.values[[1]]
[1]

print (paste(" perf of orig rf oos 15% : ",rfperf ,sep=""))

windows();
#pdf(varImpPlot(rf,main="random forest variable importance
(used to choose interaction terms for glm)"));
#windows();
par(mfrow=c(2,2))

plot(perf1a, col='darkgreen', main="OOS 15% validation ROC
")
plot(perf3, col='red',add=TRUE);
plot(perf4, col='blue',add=TRUE);
legend(0.6,0.6,c('orig glm','glm w interactions' ,'orig
rf'),col=c('darkgreen','red','blue'),lwd=2)
```

```
dotchart(c(rfperf,perforig,optglmperf),labels=c('oos
rf','oos orig glm','oos opt glm' ),cex=.7, main="oos area
under curve for validation data")

datainplot<-c((optglmperf/perforig)-1,(optglmperf/rfperf)-1)
q<-barplot(datainplot,names.arg=c('% improve AUC over
original glm','% improve AUC over rf'), main="OOS AUC
improvement by glm w interactions based on
rf",col=c("green","red"),beside=TRUE)
text(cex=.5, x=q, y=datainplot+par("cxy")[2]/2,
round(datainplot,2), xpd=TRUE) ;

#barplot(c(perforig,optglmperf,rfperf),names.arg=c('origina
l glm','glm with interactions tuned' ,'orig rf'), main="OOS
AUC by model",col=c("green","red","blue"),beside=TRUE)
#varImpPlot(rf, main="variable importance from rf")

imp <- importance(rf, class = null, scale = TRUE, type =
NULL)
e<-imp <- imp[, -(1:(ncol(imp) - 2))]
dotchart(sort(e[,1]),main="random forest variable
importance by  mean decrease in accuracy")


}
# run regression of list 2 terms + term

#print count and final model

#plot oss samples

return (finform)

}


Simulations:




par(mfrow=c(1,2))
```

```
compare_rf_glm(.5, rnorm(n=10000,m=0,sd=1),
rnorm(n=10000,m=1,sd=1),"bad dist. is norm mean 1 sd 1")
compare_rf_glm(.01, rnorm(n=10000,m=0,sd=1),
rnorm(n=10000,m=1,sd=1),"bad dist. is norm mean 1 sd 1")


# non exponential uniform

par(mfrow=c(2,2))
compare_rf_glm(.5, rnorm(n=10000,m=0,sd=1),
runif(10000,0,2),"bad dist. is uniform mean 0 to 2")
compare_rf_glm(.01, rnorm(n=10000,m=0,sd=1),
runif(10000,0,2),"bad dist. is uniform 0 to 2")

compare_rf_glm(.5, rnorm(n=10000,m=0,sd=1),
runif(10000,0,1),"bad dist. is uniform mean 0 to 2")
compare_rf_glm(.01, rnorm(n=10000,m=0,sd=1),
runif(10000,0,1),"bad dist. is uniform 0 to 2")



par(mfrow=c(2,2))
#non exponential cauchy
 compare_rf_glm(.5, rnorm(n=10000,m=0,sd=1),
rcauchy(n=10000,scale=1),"bad dist. is norm mean 1 sd
1");compare_rf_glm(.01, rnorm(n=10000,m=0,sd=1),
rcauchy(n=10000,scale=1),"bad dist. is cauchy scale 1")


library(MASS)

a<- mvrnorm(n=1000, c(0, 0),  matrix(c(10,3,3,2),2,2))
b<- mvrnorm(n=1000, c(1, 1),  matrix(c(1,2,6,5),2,2))
a<-data.frame(cbind(a,b))

 compare_rf_glm2(.01,a," 2 dimensional gaussian")



par(mfrow=c(2,2))
compare_rf_glm2(.5,data.frame(cbind(mvrnorm(n=10000, c(0,
0),  matrix(c(10,3,3,2),2,2)),mvrnorm(n=10000, c(1, 1),
matrix(c(10,3,3,2),2,2)))),," 2 dimensional gaussian")
compare_rf_glm2(.01,data.frame(cbind(mvrnorm(n=10000, c(0,
0),  matrix(c(10,3,3,2),2,2)),mvrnorm(n=10000, c(1, 1),
matrix(c(10,3,3,2),2,2)))),," 2 dimensional gaussian")
```

```
par(mfrow=c(2,2))
compare_rf_glm2(.5,data.frame(cbind(mvrnorm(n=10000, c(0,
0),  matrix(c(10,3,3,2),2,2)),mvrnorm(n=10000, c(1, 1),
matrix(c(1,2,6,5),2,2)))),," 2 dimensional gaussian")
compare_rf_glm2(.01,data.frame(cbind(mvrnorm(n=10000, c(0,
0),  matrix(c(10,3,3,2),2,2)),mvrnorm(n=10000, c(1, 1),
matrix(c(1,2,6,5),2,2)))),," 2 dimensional gaussian")


 compare_rf_glm2(.01,data.frame(cbind(mvrnorm(n=10000, c(0,
0),  matrix(c(10,3,3,2),2,2)),mvrnorm(n=10000, c(1, 1),
matrix(c(1,2,6,5),2,2)))),," 2 dimensional  gaussian")
[1] 0.5613465
[1] 0.8888072

 compare_rf_glm2(.5,data.frame(cbind(mvrnorm(n=10000, c(0,
0),  matrix(c(10,3,3,2),2,2)),mvrnorm(n=10000, c(1, 1),
matrix(c(10,3,3,2),2,2)))),," 2 dimensional ")
[1] 0.7091633
[1] 0.6271937
 compare_rf_glm2(.01,data.frame(cbind(mvrnorm(n=10000, c(0,
0),  matrix(c(10,3,3,2),2,2)),mvrnorm(n=10000, c(1, 1),
matrix(c(10,3,3,2),2,2)))),," 2 dimensional ")
[1] 0.6442052
[1] 0.4904769



#mixture gaussian

par(mfrow=c(2,2))
 compare_rf_glm(.5, rnorm(n=10000,m=0,sd=1),
rnorm(n=10000,m=1,sd=4)+rnorm(n=10000,m=.5,sd=1),"bad dist.
is norm mean 1 sd 1");compare_rf_glm(.01,
rnorm(n=10000,m=0,sd=1),
rnorm(n=10000,m=1,sd=4)+rnorm(n=10000,m=.5,sd=1),"bad dist.
is mix gaussian")

#gamma



e<-1
print(e)
set.seed(e)
 n <- 10000
 x1 <- rnorm(n,m=0,sd=1)
```

```
 x2 <- rnorm(n,m=1,sd=1)
x3 <- rnorm(n,m=0,sd=5)
x4 <- rnorm(n,m=4,sd=2)
 y <- ifelse(runif(1)>=.5,x1, x2) + x1*x2
y<- as.factor( ifelse( ((y> .2 ) | runif(1)<=.13) ,1,0))


a<-data.frame(x1,x2,x3,x4,y)
summary(a)


par(mfrow=c(2,2))
 compare_rf_glm3(a,"interaction between x1 an x2");


a1<-icreater(a,"y");tuneGlmToRf(a1,"y",2)

 tuneGlmToRf(a1,"y",2,0,FALSE,2)




d1<-create_data(.5, rnorm(n=10000,m=0,sd=1),
rcauchy(n=10000,scale=1))

d2<-create_data_2(.5,data.frame(cbind(mvrnorm(n=10000, c(0,
0),  matrix(c(10,3,3,2),2,2)),mvrnorm(n=10000, c(1, 1),
matrix(c(1,2,6,5),2,2)))))

d3<-create_data(.5, rnorm(n=10000,m=0,sd=1),
rnorm(n=10000,m=1,sd=4)+rnorm(n=10000,m=.5,sd=1))

d4<-a

d1<-icreater(d1,"y");tuneGlmToRf(d1,"y");

d2<-icreater(d2,"y");tuneGlmToRf(d2,"y");

d3<-icreater(d3,"y");tuneGlmToRf(d3,"y");

d4<-icreater(a,"y");tuneGlmToRf(a1,"y")



library(lattice)
```

```
densityplot(~ rnorm(n=100) + rcauchy(n=100,scale=1),
auto.key = TRUE, main="normal u=0,sigma=1 vs. cauchy scale
1 density plot by good vs. bad") ;

densityplot(~ rnorm(n=100) +
(rnorm(n=100,m=1,sd=4)+rnorm(n=100,m=.5,sd=1)), auto.key =
TRUE,main="normal u=0,sigma=1 vs. sum 2 gaussians
m=.5,sd=1+ m=1,sd=4 density plot by

good vs. bad");

densityplot( ~ x1+x2+x3+x4+x1*x2 | y, data = e,
main="interaction data set density good vs. bad")
+layer(panel.key(c("x1","x2","x3","x4","x1*x2"), corner =
c(1,.98), lines =

TRUE, points = FALSE), packets = 1)



http://cran.r-
project.org/web/packages/HSAUR/vignettes/Ch_density_estimat
ion.pdf

2-d gaussians


par(mfrow=c(2,2))
 w<-data.frame(mvrnorm(n=10000, c(0, 0),
matrix(c(10,3,3,2),2,2)))
 q<-data.frame(mvrnorm(n=10000, c(1, 1),
matrix(c(1,2,6,5),2,2)))

 plot(X2~X1,data=w, main="2 gaussians; m=0,sd=0 & m=1,sd=1
with different var-covar "); points(X2~X1,data=q,col='red')

 w<-data.frame(mvrnorm(n=10000, c(0, 0),
matrix(c(10,3,3,2),2,2)))
 q<-data.frame(mvrnorm(n=10000, c(1, 1),
matrix(c(10,3,3,2),2,2)))

 plot(X2~X1,data=w, main="2 gaussians; m=0,sd=0 & m=1,sd=1
with same var-covar "); points(X2~X1,data=q,col='red')

densityplot(~ w$X1+q$X3, auto.key = TRUE, main="of 2
dimensional gaussians") ;
```