CUNY School of Professional Studies

SPS.CUNY.EDU

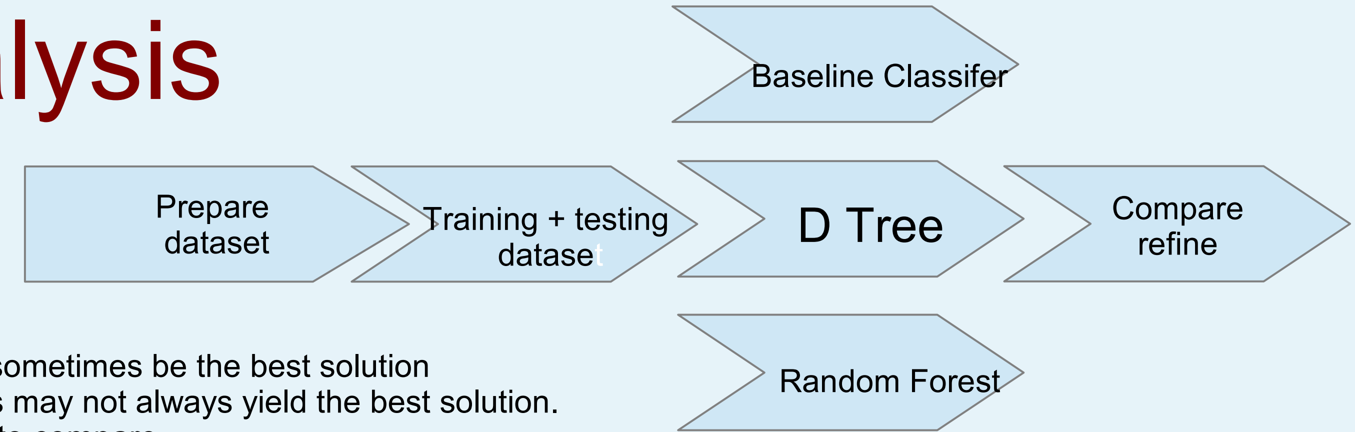
L13 Classifier Performance
2020 Spring Data-622
Comparative Analysis

Raman Kannan

Instructor Email Address: Raman.Kannan@sps.cuny.edu

Acknowledgements:
Generous support from IBM Power Systems Academic Initiative
IBM PSAI provides computing infrastructure for free

Performance Analysis



Simplest strategy may sometimes be the best solution
Sophisticated strategies may not always yield the best solution.
Setup objective means to compare
- Same training set and testing set
- Same measurement criteria

Improving Classifier Performance

Process

Preparing the Data – train and testing data set

Baseline Classifier – GLM

Tree – Refine and fine tune,all features on entire dataset

Random Forest – Aggregating Weak Learners to make a strong Learner

- Random subset of features, bootstrapped dataset

Compare Performance – ROC Curve, AUC

Dataset Preparation

```
titanic<-
read.csv("http://christianherta.de/lehre/dataScien
ce/machineLearning/data/titanic-
train.csv",header=T)
sm_titantic_3<-titanic[,c(2,3,5,6,10)]
#dont need all the columns
sm_titanic_3<-
sm_titantic_3[complete.cases(sm_titantic_3),]
set.seed(43)
tst_idx<-sample(714,200,replace=FALSE)
tstdata<-sm_titanic_3[tst_idx,]
trdata<-sm_titanic_3[-tst_idx,]
```

Running GLM

Create a model using logistic regression

```
glm_sm_titanic_3<-
glm(Survived~.,data=trdata,family=binomial())
```

Predict using the test dataset and glm model created

```
glm_predicted<-
predict(glm_sm_titanic_3,tstdata[,2:5],type="response");
```

How well did GLM perform?

require(ROCR) # use ROCR package

True/False Positives, True/False Negatives

We will use Precision, Recall, and derived measures
Precision (related to false positives)
Recall (related to false negatives)

https://en.wikipedia.org/wiki/Precision_and_recall

Binary Classifier

Precision → (TRUE POSITIVES)/(TP+FP)

Recall → (TP)/(All positives in the sample)

https://uberpython.wordpress.com/2012/01/01/precision-recall-sensitivity-and-specificity/

Standardized equations

- sensitivity = recall = tp / t = tp / (tp + fn)
- specificity = tn / n = tn / (tn + fp)
- precision = tp / p = tp / (tp + fp)

Calculate

```
glm_sm_titanic_3<-glm(Survived~.,data=trdata,family=binomial())
glm_predicted<-predict(glm_sm_titanic_3,tstdata[,2:5],type="response");
```

Model with training and Predict with test

```
require(ROCR)
```

Extract fp and tp using ROCR package

```
glm_auc_1<-prediction(glm_predicted,tstdata$Survived)
glm_prf<-performance(glm_auc_1, measure="tpr", x.measure="fpr")
glm_slot_fp<-slot(glm_auc_1,"fp")
glm_slot_tp<-slot(glm_auc_1,"tp")
```

```
glm_fpr3<-unlist(glm_slot_fp)/unlist(slot(glm_auc_1,"n.neg"))
glm_tpr3<-unlist(glm_slot_tp)/unlist(slot(glm_auc_1,"n.pos"))
```

Calculate fpr and tpr vectors

```
glm_perf_AUC=performance(glm_auc_1,"auc")
glm_AUC=glm_perf_AUC@y.values[[1]]
```

Calculate Area Under Curve – ROC

Plot the results for GLM

```
plot(glm_fpr3,glm_tpr3,
main="ROC Curve from first principles -- raw counts",
xlab="FPR",ylab="TPR")
```

Plot RoC

```
points(glm_fpr3,glm_fpr3,cex=0.3) # will generate a diagonal plot
```

Let us draw a diagonal and see the lift

Let us place AUC on the plot

```
text(0.4,0.6,
paste("GLM AUC = ",format(glm_AUC, digits=5, scientific=FALSE)))
```

Plot the results for GLM

```
plot(glm_fpr3,glm_tpr3,
main="ROC Curve from first principles -- raw counts",
xlab="FPR",ylab="TPR")
```

Plot RoC

Let us draw a diagonal and see the lift

```
points(glm_fpr3,glm_fpr3,cex=0.3) # will generate a diagonal plot
```

```
text(0.4,0.6,
paste("GLM AUC = ",format(glm_AUC, digits=5, scientific=FALSE)))
```

Let us place AUC on the plot

Let us repeat the steps for individual trees

We will use recursive partition package – there are many other packages

tree<-rpart(Survived~.,data=trdata)
tree_predicted_prob_03<-predict(pruned.tree.03,tstdata[,2:5])

Model with training and Predict with test

#Use prune and printcp/plotcp to fine tune the model
as shown in the video

Extract measures using ROCR package

```
tree_predicted_class_03<-round(tree_predicted_prob_03)
tree_prediction_rocr_03<-prediction(tree_predicted_class_03,tstdata$Survived)
tree_prf_rocr_03<-performance(tree_prediction_rocr_03, measure="tpr", x.measure="fpr"

tree_perf_AUC_03=
performance(tree_prediction_rocr_03,"auc")
```

Visualize

```
plot(tree_prf_rocr_03,main="ROC plot cp=0.03(DTREE using rpart)")
text(0.5,0.5,paste("AUC=",format(tree_perf_AUC_03@y.values[[1]],digits=5, scientific=FALSE)))
```

Motivation for Ensemble Methods

Tree while non-parametric, includes all features and all observations, and consequently can result in over-fitting.

Bootstrap aggregation (aka Bagging).

- Construct multiple individual trees
  - At each split while constructing the tree
    - Randomly select a subset of features
    - Bootstrap dataset (with REPLACEMENT)
- Aggregate results from multiple trees
  - Using any strategy
    - Allow voting and pick most voted
    - Simply average over all the trees

Let us repeat the steps for Ensemble

We will use randomForest package (which uses CART)– there are many other packages

Model with training and Predict with test

fac.titanic.rf<-
randomForest(as.factor(Survived)~.,data=trdata,keep.inbag=TRUE,
type=classification,importance=TRUE,keep.forest=TRUE,ntree=193)

predicted.rf <- predict(fac.titanic.rf, tstdata[,-1], type='response')

#Use prune and printcp/plotcp to fine tune the model
as shown in the video

Extract measures using ROCR package

```
confusionTable  <- table(predicted.rf, tstdata[,1],dnn = c('Predicted','Observed'))
table( predicted.rf==tstdata[,1])
pred.rf<-prediction(as.numeric(predicted.rf),as.numeric(tstdata[,1]))
perf.rf<-performance(pred.rf,measure="tpr",x.measure="fpr")

auc_rf<-performance(pred.rf,measure="auc")
```

Visualize

```
plot(perf.rf, col=rainbow(7), main="ROC curve Titanic (Random Forest)",
xlab="FPR", ylab="TPR")
text(0.5,0.5,paste("AUC=",format(auc_rf@y.values[[1]],digits=5, scientific=FALSE)))
```

Calculate - LDA

lda.titanic<-lda(Survived~.,data=trdata)

lda.predicted<-predict(object=lda.titanic,newdata=tstdata[,-1]);

Model with training and Predict with test

---

require(ROCR)

Extract fp and tp using ROCR package

```
lda_pred<-prediction(as.numeric(lda.predicted$class),as.numeric(tstdata[,1]))
lda_perf<-performance(lda_pred,measure="tpr",x.measure="fpr")
```

```
lda_auc_rf<-performance(lda_pred,measure="auc")
```

Calculate Area Under Curve – ROC

Visualize

```
plot(lda_perf, col=rainbow(7), main="ROC curve (LDA)",  xlab="FPR",ylab="TPR")
points(seq(0.01,1.0,0.01),seq(0.01,1.0,0.01),cex=0.3)
text(0.5,0.5,paste("AUC=",format(lda_auc_rf@y.values[[1]],digits=5, scientific=FALSE)))
```

Calculate - QDA

```
qda.titanic<-qda(Survived~.,data=trdata)
qda.predicted<-predict(object=qda.titanic,newdata=tstdata[,-1]);
```

Model with training and Predict with test

```
require(ROCR)

qda_pred<-prediction(as.numeric(qda.predicted$class),as.numeric(tstdata[,1]))
qda_perf<-performance(qda_pred,measure="tpr",x.measure="fpr")
```

Extract fp and tp using ROCR package

```
qda_auc_rf<-performance(qda_pred,measure="auc")
```

Calculate Area Under Curve – ROC

```
plot(qda_perf, col=rainbow(7), main="ROC curve (QDA)",  xlab="FPR",ylab="TPR")
points(seq(0.01,1.0,0.01),seq(0.01,1.0,0.01),cex=0.3)
text(0.5,0.5,paste("AUC=",format(qda_auc_rf@y.values[[1]],digits=5, scientific=FALSE)))
```

Visualize

Calculate – Naive Bayes (package e1071)

```
model.nb<-naiveBayes(trdata[,-1],trdata[,1])

predict.nb<-predict(object=model.nb,newdata=tstdata[,-1],type="raw")
```

Model with training and Predict with test

```
require(ROCR)
predict.nb.class<-apply(round(predict.nb),1,FUN=function(x){x[1]!=1})
predict.nb.class<-as.numeric(predict.nb.class)
nb.prediction.rocr<-prediction(predict.nb.class,tstdata$Survived)
nb.perf<-performance( nb.prediction.rocr, measure="tpr", x.measure="fpr")
```

Extract fp and tp using ROCR package

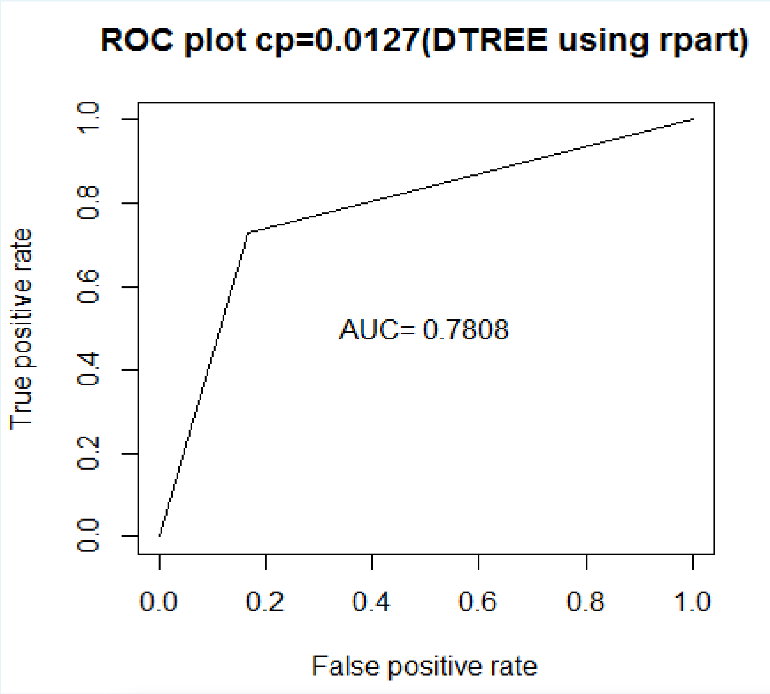
```
nb.perf.AUC<-performance( nb.prediction.rocr,"auc")
```

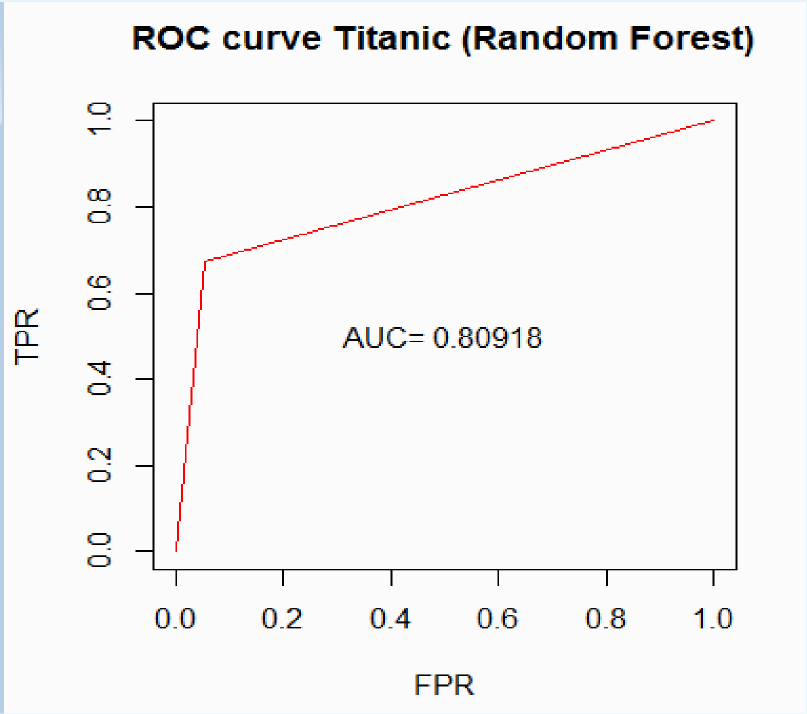
Calculate Area Under Curve – ROC

```
plot(nb.perf,main="NB ROC Titanic",xlab="FPR", ylab="TPR",cex=0.3)
abline(0,1)
text(0.5,0.5,paste("AUC=",format(nb.perf.AUC@y.values[[1]],digits=5, scientific=FALSE)))
```

Visualize

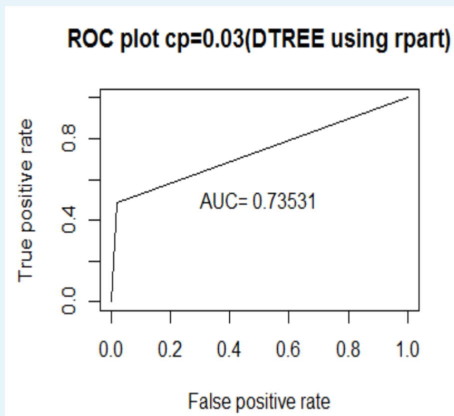
Comparing Models

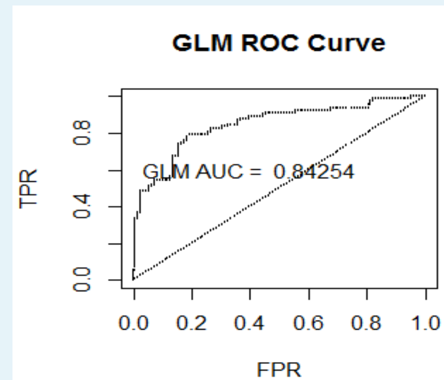




Random Forests always outperforms Individual Trees
In this experiment, GLM outperforms Random Forest.

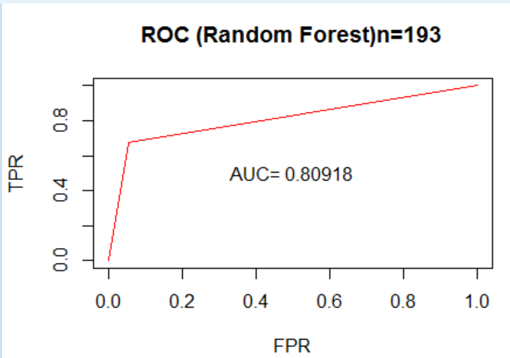
Comparing Models

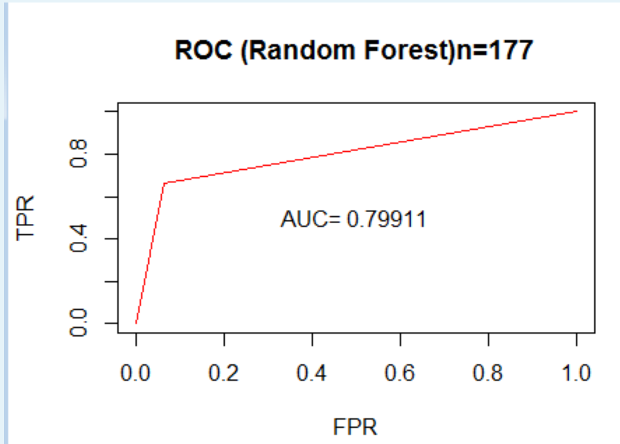


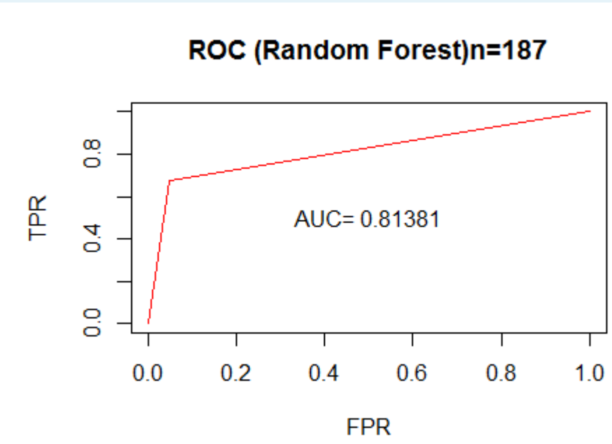


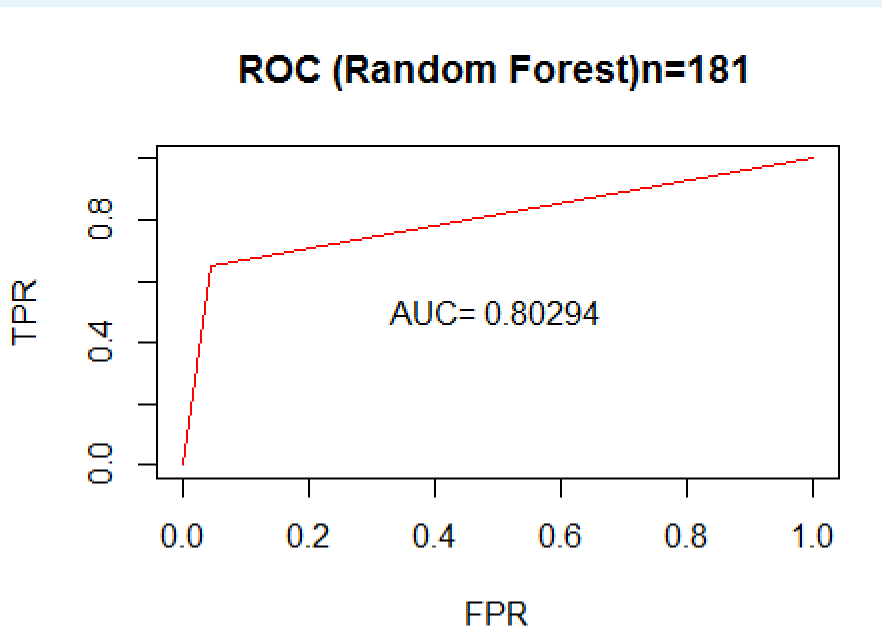
By AUC, GLM is by far the best, RF is second best followed by DecisionTree and then LDA.

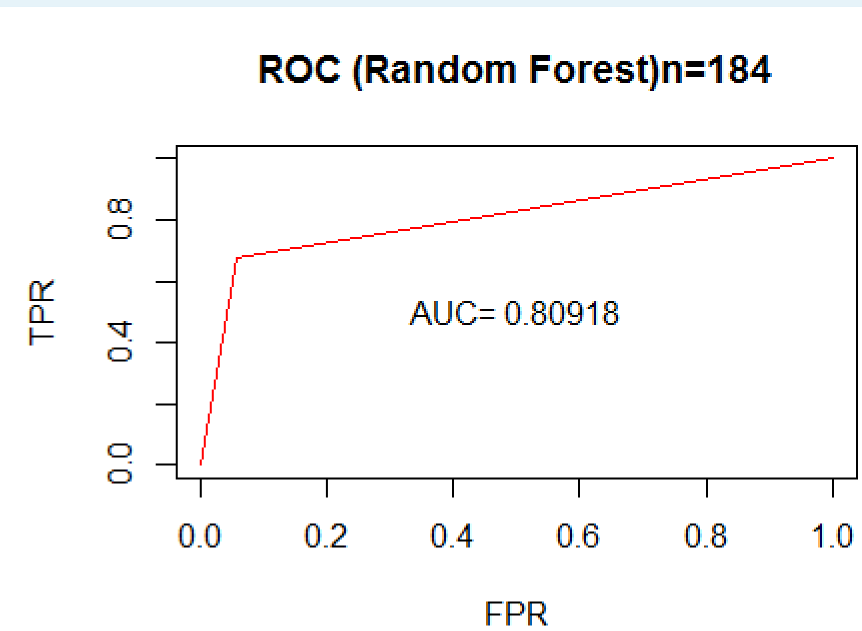
Random Forests by varying n (number of Trees)



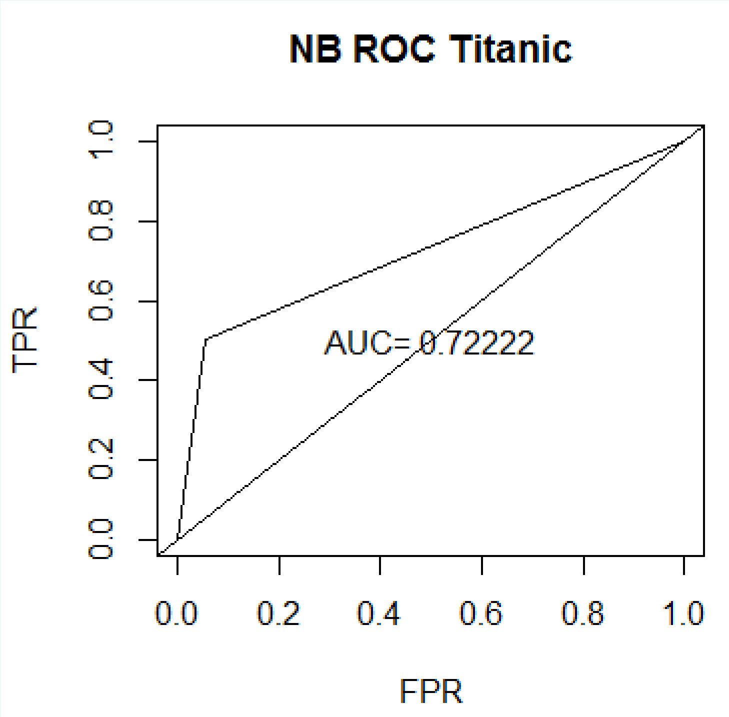








NB ROC Curve



AUC for GLM → 0.84
AUC for DT → 0.78
AUC for RF → 0.82
AUC for LDA → 0.77
AUC for QDA → 0.74
AUC for NB → 0.72

Calculate – NN (package nnet)

```
nn.model<-nnet(Survived ~ ., data
      =z.trdata,size=20,maxit=10000,decay=.001, linout=F, trace = F)

predict.nn<-predict(nn.model,newdata=z.tstdata[,-1])
```

Model with training and Predict with test

```
require(ROCR)
predict.nn.class<-round(predict.nn)
 nn.pred<-prediction(pred.nn.class,z.tstdata[,1])

nn.perf<-performance( predict.nn.class, measure="tpr", x.measure="fpr")
```

Extract fp and tp using ROCR package

```
plot(nn.perf,main="NN ROC Titanic",xlab="FPR", ylab="TPR",cex=0.3)
abline(0,1)
```

Calculate Area Under Curve – ROC

Visualize

```
nn.perf.AUC<-performance( nn.pred,measure="auc")
text(0.5,0.5,paste("AUC=",format(nn.perf.AUC@y.values[[1]],digits=5,scientific=FALSE)))
```

Preparing data for NN

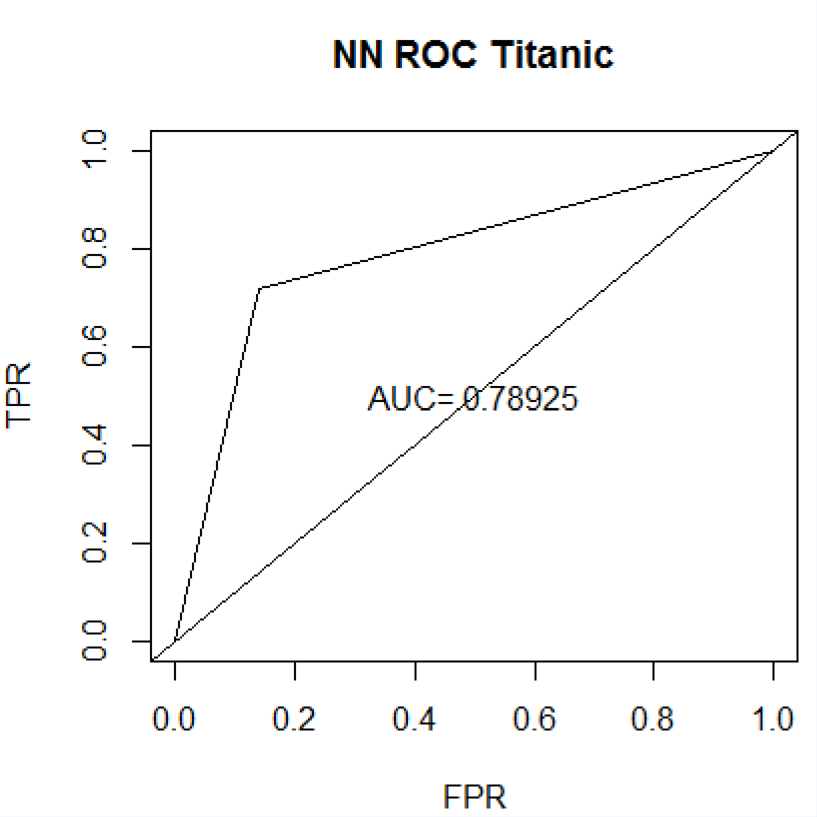
NN requires numerical data and the labels to be factors
Data must be normalized (x-u)/sd

One must normalize before-partitioning the data.
Recall sm_titanic_3 has all complete cases and tst_idx has the test sample indices.

This is the R code
to prepare data for NN

```
head(sm_titanic_3)
ynum.titanic<-cbind(sm_titanic_3,sm_titanic_3$Sex=="male")
head(ynum.titanic)
ynum.titanic<-ynum.titanic[,c(1,2,4,5,6)]
head(ynum.titanic)
names(ynum.titanic)<-c(names(ynum.titanic)[1:4],"Sex")
head(ynum.titanic)
ynum.titanic$Sex<-as.numeric(ynum.titanic$Sex)
head(ynum.titanic)
z.titanic<-as.data.frame(apply(ynum.titanic,2,FUN=function(x){ (x-mean(x))/sd(x) }))
z.titanic$Survived<-ynum.titanic$Survived
z.tstdata<-z.titanic[tst_idx,] # tst_idx is the sample after set.seed(43)
z.trdata<-z.titanic[-tst_idx,]
```

NN ROC Curve



AUC for GLM → 0.84
AUC for RF → 0.82
AUC for NN → 0.79
AUC for DT → 0.78
AUC for LDA → 0.77
AUC for QDA → 0.74
AUC for NB → 0.72

GLM still rules...

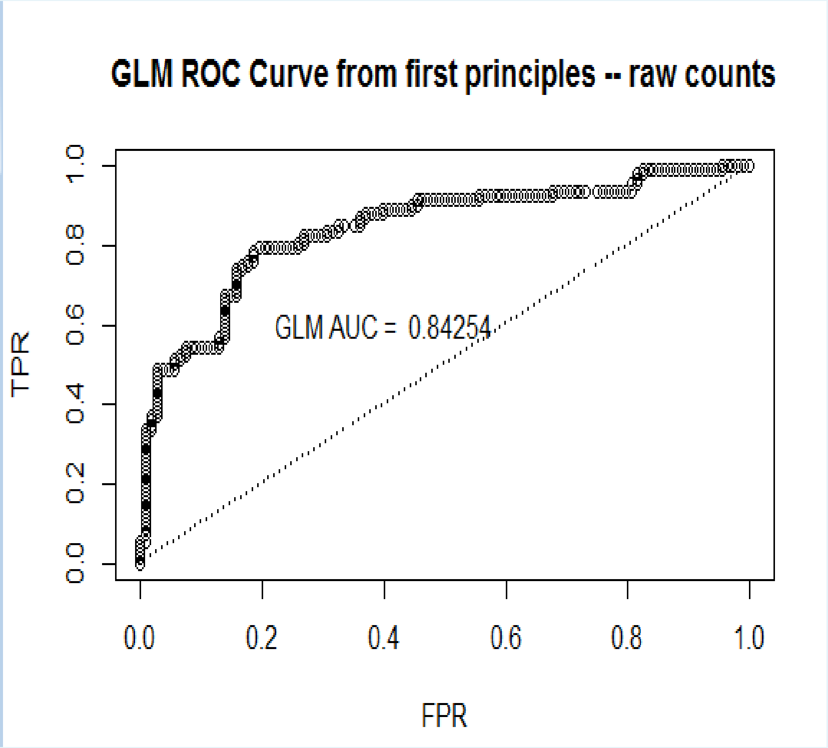
AUC for GLM → 0.84
AUC for DT → 0.78
AUC for RF → 0.82

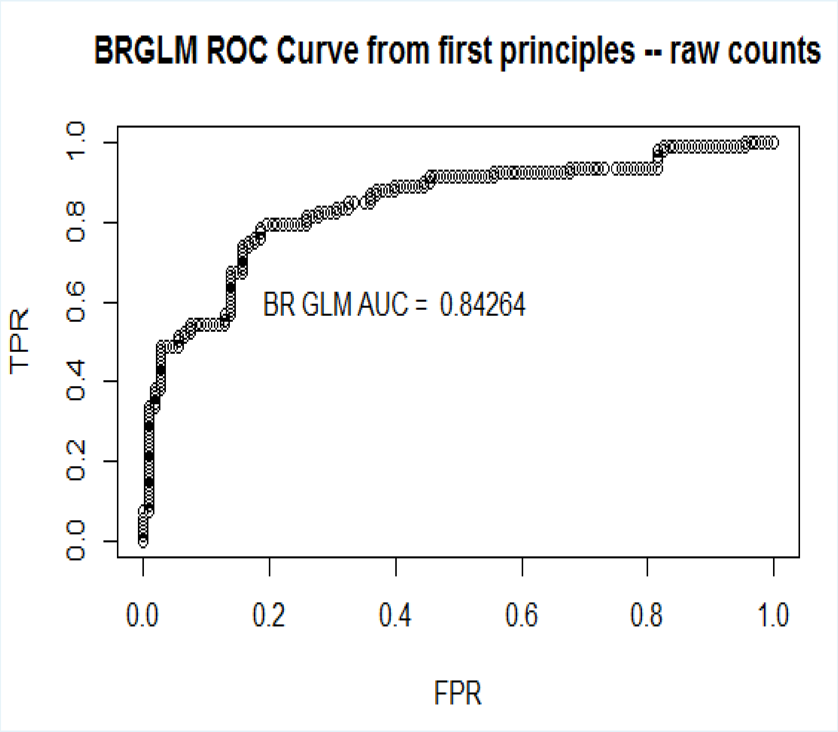
AUC (more area) → Improvement

Research Directions: Random Forest

R Exercise : extract and reconstruct individual trees for further investigation

ROC Curves


GLM ROC Curve from first principles -- raw counts
GLM AUC = 0.84254
TPR
0.0
0.2
0.4
0.6
0.8
1.0
FPR


BRGLM ROC Curve from first principles -- raw counts
BR GLM AUC = 0.84264
TPR
0.0
0.2
0.4
0.6
0.8
1.0
FPR

GLMNET

```
ynum.titanic<-cbind(sm_titanic_3,sm_titanic_3$Sex=="male")
ynum.titanic<-ynum.titanic[,c(1,2,4,5,6)]
names(ynum.titanic)<-c(names(ynum.titanic)[1:4],"Sex")
ynum.titanic$Sex<-as.numeric(ynum.titanic$Sex)
lars.trdata<-ynum.titanic[-tst_idx,]
lars.tstdata<-ynum.titanic[tst_idx,]
```

```
glmnet.model<-glmnet(as.matrix(lars.trdata[, 2:5]), y = lars.trdata$Survived,family = "binomial")
```

```
glmnet.model.predicted<-predict(glmnet.model,as.matrix(lars.tstdata[,2:5]),s=0.001,type='response')
glmnet.predicted.numeric<-as.numeric(glmnet.model.predicted>0.50)
glmnet.pred<-prediction(glmnet.predicted.numeric,lars.tstdata$Survived)
glmnet.perf<-performance(glmnet.pred,measure="tpr", x.measure="fpr")
glmnet.AUC<-performance(glmnet.pred,measure="auc")
```

```
plot(glmnet.perf,main="GLMNET ROC Titanic",xlab="FPR", ylab="TPR",cex=0.3)
abline(0,1)
text(0.5,0.5,paste("GLMNET AUC =",format(glmnet.AUC@y.values[[1]],digits=5,scientific=F)))
history(30)
```

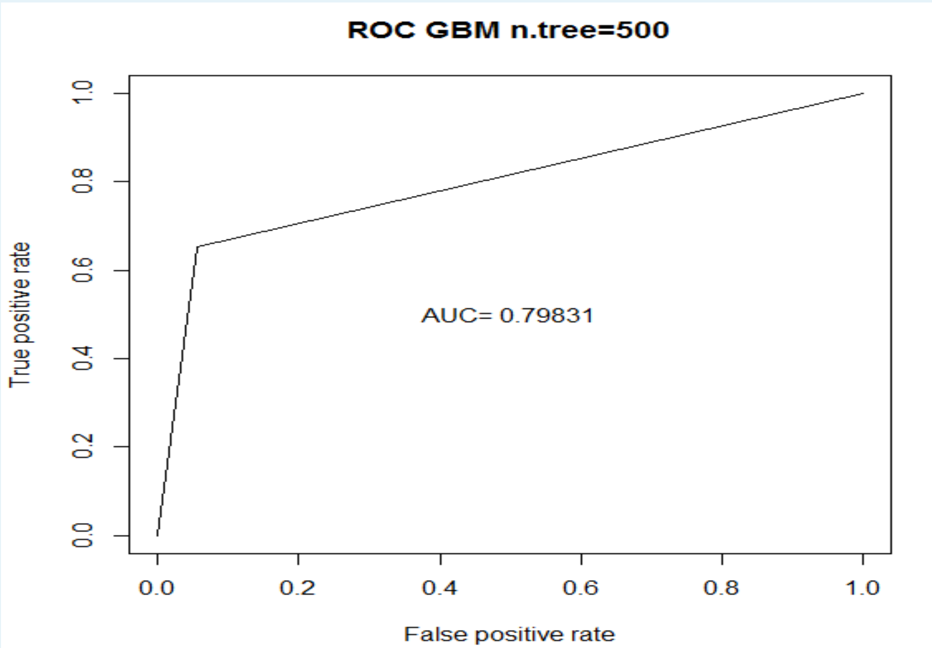
GLMNET

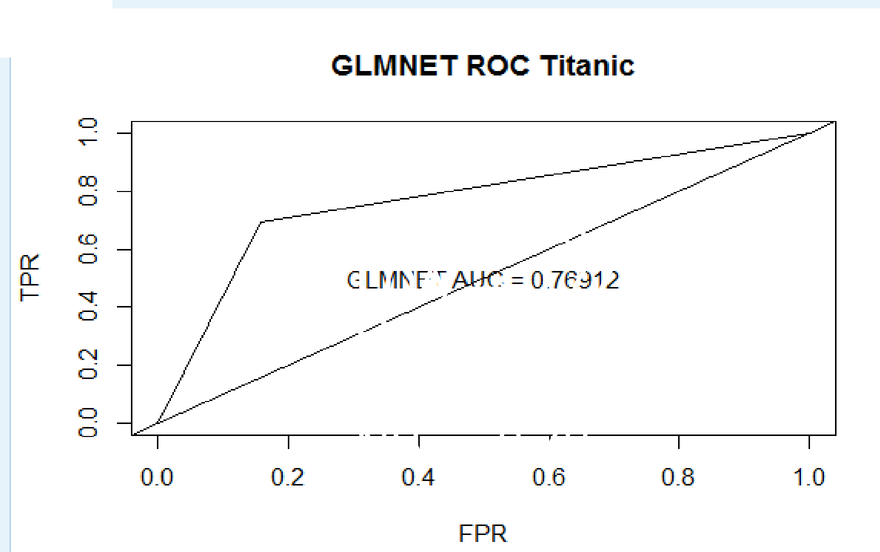
```
ynum.titanic<-cbind(sm_titanic_3,sm_titanic_3$Sex=="male")
ynum.titanic<-ynum.titanic[,c(1,2,4,5,6)]
names(ynum.titanic)<-c(names(ynum.titanic)[1:4],"Sex")
ynum.titanic$Sex<-as.numeric(ynum.titanic$Sex)
lars.trdata<-ynum.titanic[-tst_idx,]
lars.tstdata<-ynum.titanic[tst_idx,]

glmnet.model<-glmnet(as.matrix(lars.trdata[, 2:5]), y = lars.trdata$Survived,family = "binomial")

glmnet.model.predicted<-predict(glmnet.model,as.matrix(lars.tstdata[,2:5]),s=0.001,type='response')
glmnet.predicted.numeric<-as.numeric(glmnet.model.predicted>0.50)
glmnet.pred<-prediction(glmnet.predicted.numeric,lars.tstdata$Survived)
glmnet.perf<-performance(glmnet.pred,measure="tpr", x.measure="fpr")
glmnet.AUC<-performance(glmnet.pred,measure="auc")

plot(glmnet.perf,main="GLMNET ROC Titanic",xlab="FPR", ylab="TPR",cex=0.3)
abline(0,1)
text(0.5,0.5,paste("GLMNET AUC =",format(glmnet.AUC@y.values[[1]],digits=5,scientific=F)))
history(30)
```

GBM n.trees=500 (gbm package)

```
gbm_sm_titanic_3<-gbm(Survived~.,data=trdata,distribution="bernoulli",n.trees=500,
shrinkage=0.01,interaction.depth=3,n.minobsinnode=10,verbose=T, keep.data=T)

  gbm_predict<-predict(gbm_sm_titanic_3,tstdata[,2:5],
  type="response",gbm_sm_titanic_3$n.trees)
  gbm_predicted<-round(gbm_predict)

  library(ROCR)
  require(ROCR)

 gbm_prediction<-prediction(gbm_predicted,tstdata$Survived)
 gbm_perf<-performance(gbm_prediction,measure="tpr",x.measure="fpr")
 plot(gbm_perf,main="ROC GBM n.tree=500")
 gbm_auc<-performance(gbm_prediction,measure="auc")

text(0.5,0.5,paste("AUC=",format(gbm_auc@y.values[[1]],digits=5, scientific=FALSE)))
```

ROC and AUC for GBM and GLMNET





GLM → 0.84
RF → 0.82
GBM → 0.80
NN → 0.79

xgboost

```
# prepare data as we did for NN – numerical and scaled/normalized
xgb_model <- xgboost(data = xgb_data_mx[,2:5],label=xgb_data_mx[,1], max.depth = 2,
eta = 1, nthread = 2, nround = 2, subsample = 0.5, colsample_bytree = 0.5, objective = "binary:logistic")
```

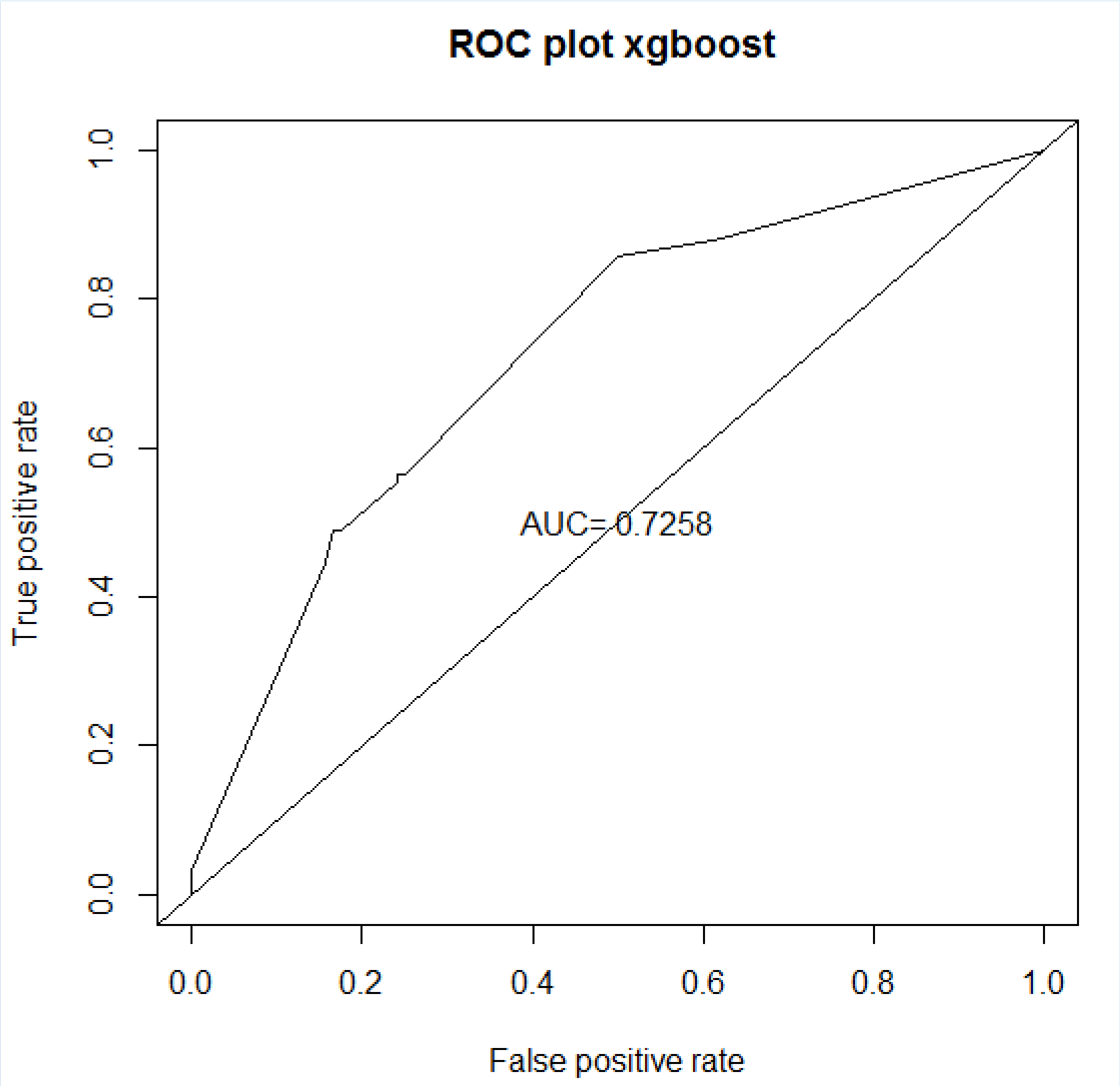
```
xgb_predicted<-predict(xgb_model,as.matrix(z.tstdata[,2:5]))
```

```
xgb_prediction<-prediction(xgb_predicted,z.tstdata$Survived)
xgb_prf<-performance(xgb_prediction, measure="tpr", x.measure="fpr")
xgb_slot_fp<-slot(xgb_prediction,"fp")
xgb_slot_tp<-slot(xgb_prediction,"tp")

xgb_perf_AUC=performance(xgb_prediction,"auc")
xgb_AUC=xgb_perf_AUC@y.values[[1]]

plot(xgb_prf,main="ROC plot xgboost ")
text(0.5,0.5,paste("AUC=",format(xgb_perf_AUC@y.values[[1]],
digits=5, scientific=FALSE)))
abline(0,1)
```

Preparing data for xgboost (similar to NN and LARS)

```
ynum.titanic<-cbind(sm_titanic_3,sm_titanic_3$Sex=="male")

ynum.titanic<-ynum.titanic[,c(1,2,4,5,6)]

names(ynum.titanic)<-c(names(ynum.titanic)[1:4],"Sex")

ynum.titanic$Sex<-as.numeric(ynum.titanic$Sex)

z.titanic<-as.data.frame(apply(ynum.titanic,2,FUN=function(x){ (x-mean(x))/sd(x) }))
z.titanic$Survived<-ynum.titanic$Survived
z.tstdata<-z.titanic[tst_idx,] # tst_idx is the sample after set.seed(43)
z.trdata<-z.titanic[-tst_idx,]

xgb_data_mx<-as.matrix(z.trdata)
```

xgboost



GLM → 0.84
RF → 0.82
GBM → 0.80
NN → 0.79
DT → 0.78

LDA → 0.77
GLMNET → 0.77
QDA → 0.74
XGB → 0.73
NB → 0.72