CUNY School of Professional Studies

SPS.CUNY.EDU

Lecture 15
2020 Spring Data-622
Ensemble: RandomForest

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

Script for Algorithms

Develop the Intuition
Understand the assumptions
Develop the mathematics
Run the algorithms
Learn to interpret the result/output
Predict using the model
Learn to determine the performance
Distinguish training/testing error
Differentiate between overfitting/underfitting
Techniques to improve performance

Intuition (Ensemble)

So far we have seen single algorithm strategies at work.

We have been trying to minimize variance and bias, recognizing there is an irreducible error – we cannot achieve lower error rate.
Error in regression is RMSE and for classification can be one of accuracy, precision, AUC or other measures based on TPR/FPR, TNR,FNR. Averaging is a proven smoothing technique.

Now we want to incorporate techniques where we will seek to leverage multiple algorithms.

We have the option of creating random disjoint datasets, run the same model and then take the average (CV,Bagging, Boosting L12).
OR
Generate numerous models using random attribute sets and then take the average. This is called RandomForest which combine multiple trees.
OR
As a third option run different types of algorithms and combine the results. This strategy is called Stacking.

Generate a Model (RandomForest) Load Data

```
require(randomForest)
require(ROCR)
path<-"C:/Users/rk215/cuny/L11-tree/binary.csv"
admit_data<-read.csv(path,head=TRUE);
head(admit_data)

#make some columns factors
fad<-data.frame(as.factor(admit_data$admit),admit_data$gre,
admit_data$gpa,as.factor(admit_data$rank))
names(fad)<-names(admit_data)

# create test and training set
set.seed(43)
tstset<-sample(400,120,replace=FALSE) # 30% hold out test set
admit_trdata<-fad[-tstset,]
admit_tstdata<-fad[tstset,]
```

Let us create a model and compute Performance

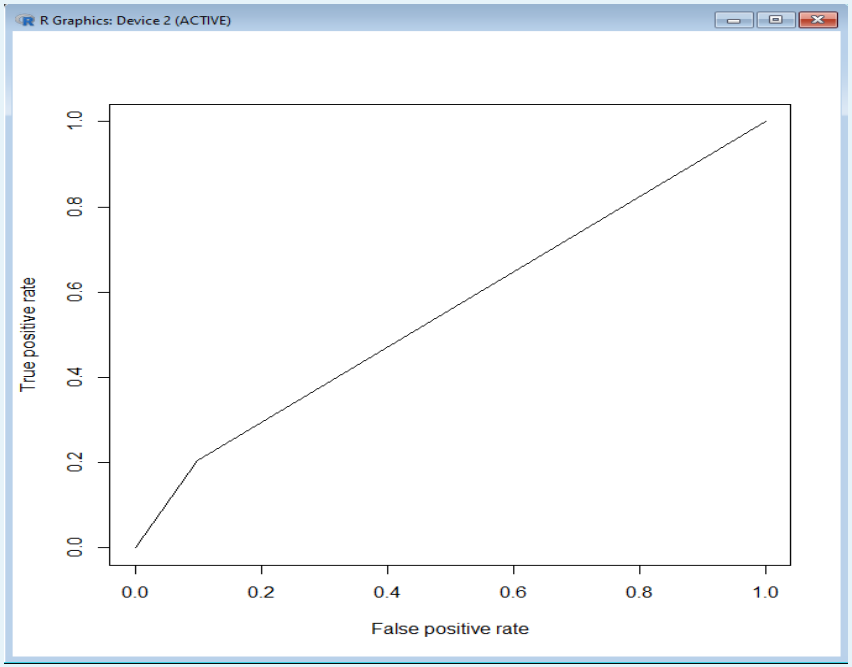
```
admit.rf<-
randomForest(admit~.,data=admit_trdata,ntree=1000,importance=T)
admit.rf.pr<-predict(admit.rf,admit_tstdata,type='response')

confusionMX.admit.rf<-
table(admit.rf.pr, admit_tstdata$admit,
 dnn=c('Predicted','Actual'))

admit.rf.pred<-prediction(as.numeric(admit.rf.pr),admit_tstdata$admit)
admit.rf.perf<-performance(admit.rf.pred,measure = "tpr", x.measure =
"fpr")
admit.rf.perf.auc<-performance(admit.rf.pred,measure = "auc")

plot(admit.rf.perf)
table(as.numeric(admit.rf.pr),admit_tstdata$admit)
admit.rf.auc<-admit.rf.perf.auc@y.values[[1]]
admit.rf.auc
```

Visualizing

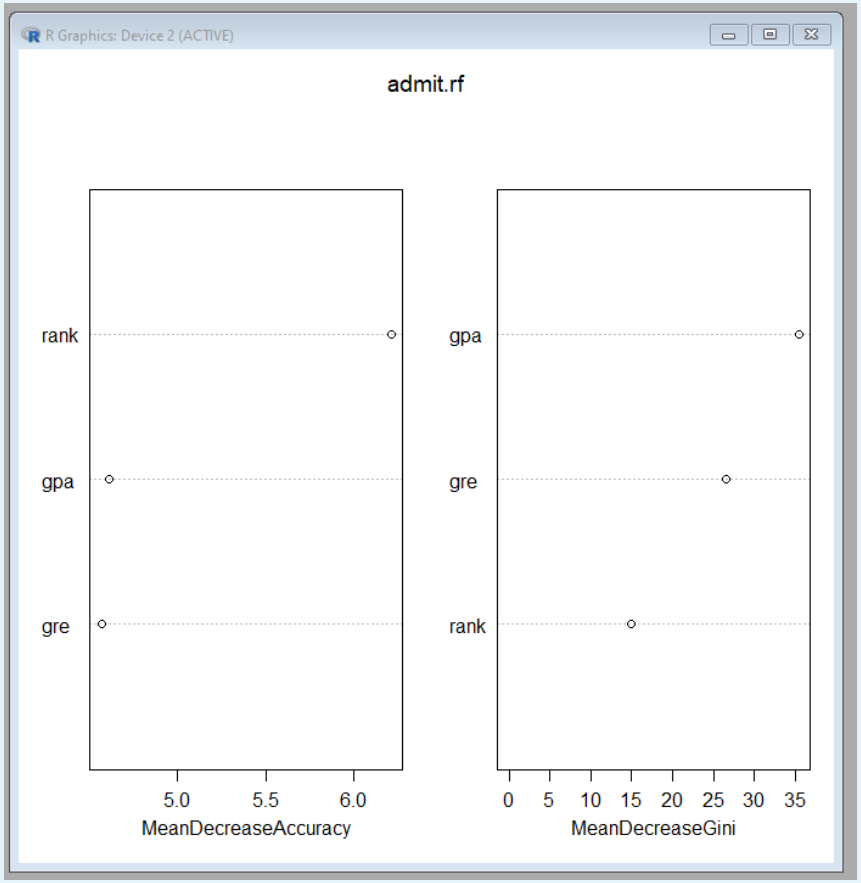



```
> confusionMX.admit.rf<-table(admit.rf.pr,admit_tstdata$admit,
+  dnn=c('Predicted','Actual'))
>
> admit.rf.pred<-prediction(as.numeric(admit.rf.pr),admit_tstdata$admit)
> admit.rf.perf<-performance(admit.rf.pred,measure = "tpr", x.measure = "fpr")
> admit.rf.perf.auc<-performance(admit.rf.pred,measure = "auc")
>
> plot(admit.rf.perf)
> table(as.numeric(admit.rf.pr),admit_tstdata$admit)

     0  1
  1 73 31
  2  8  8
> admit.rf.auc<-admit.rf.perf.auc@y.values[[1]]
> admit.rf.auc
[1] 0.5531814
>
```

Diagnostic plots

varImpPlot(admit.rf)



plot(admit.rf)