K2 Analytics
Building Skills, Building Individuals

Random Forest

About K2 Analytics

At K2 Analytics, we believe that skill development is very important for the growth of an individual, which in turn leads to the growth of Society & Industry and ultimately the Nation as a whole. For this it is important that access to knowledge and skill development trainings should be made available easily and economically to every individual.

Our Vision: *"To be the preferred partner for training and skill development"*

Our Mission: *"To provide training and skill development training to individuals, make them skilled & industry ready and create a pool of skilled resources readily available for the industry"*

*We have chosen Business Intelligence and Analytics as our focus area. With this endeavour we make this presentation on "**Random Forest**" accessible to all those who wish to learn Analytics. We hope it is of help to you. For any feedback / suggestion or if you are looking for job in analytics then feel free to write back to us at ar.jakhotia@k2analytics.co.in*

Welcome to Analytics Training !!!

Learning Objectives

- What is Ensemble Modeling?
- What is Bagging?
- Random Forest Algorithm
- Out of Bag Error Rate
- Finding Optimal Number of Trees
- Finding Optimal Number of Variables to Select

Some Concepts

- **Ensemble** : use of ***multiple learning algorithms*** to obtain better ***predictive performance*** than could be obtained from any of the constituent learning algorithms

- **Bootstrap aggregating**, also called **bagging:** Given a standard training set *D* of size *n*, bagging generates *m* new training sets D_i, each of size *n′*, by sampling from D uniformly with replacement. By sampling with replacement, some observations may be repeated in each D_i. The kind of sample is called Bootstrap. The *m* models are fitted using the above *m* bootstrap samples and combined (aggregated) by averaging the output (for regression) or voting (for classification).

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

Random Forest

- Ensemble Technique
- Involves constructing multitude of decision trees at training time
- Prediction is based on mode for classification tree and mean for regression tree
- Help reduce over-fitting
  - Note: there is possibility of high over-fitting at individual tree level but averaging removes the bias

RF Algorithm

- Step 1: Random Sampling with replacement

ID	A1	A2	A3	...	A(M)	Class
R1						
R2						
R3						
.						
.						
.						
Rn						

D_1

ID	A1	A2	A3	...	A(M)	Class
R10						
R3						
R15						
R4						
.						
.						

D_2

ID	A1	A2	A3	...	A(M)	Class
R1						
R1						
R100						
R500						
.						
.						

$D_{\ldots}$

ID	A1	A2	A3	...	A(M)	Class
R1						
R1						
R100						
R500						
.						
.						

D_n

ID	A1	A2	A3	...	A(M)	Class
R1						
R1						
R100						
R500						
.						
.						

RF Algorithm… contd

- Step 2: Building the tree for each sample with only partial set of ‘m’ variable being considered at each node
- m << M where M is total number of predictor variables

ID	A1	A5	A7	Class
R10				
R3				
R15				
R4				
.				
.				

Only a partial list of variables are considered for splitting based on the best variable from the partial list

ID	A2	A6	A9	Class
R10				
R3				
R15				
R4				
.				
.				

A different set of partial list of variables considered

ID	A1	A3	A4	Class
R10				
R3				
R15				
R4				
.				
.				

A different set of partial list of variables considered

RF Algorithm… contd

Step 3: Classifying

- Based on ‘n’ samples… ‘n’ tree are built
- Each records is classified based on the n tree
- Final class for each record is decided based on voting

Note: We do not have the pruning step in RF

Some original papers on RF proved that the RF error rate depends on two factors

1. The *correlation* between any two trees in the forest. Increasing the correlation increases the forest error rate.

2. The *strength* of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

3. Reducing m reduces both the correlation and the strength. Increasing it increases both. Somewhere in between is an "optimal" range of m - usually quite wide

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Building Random Forest in R

```
## Building the model using Random Forest

## importing the data
RFDF.dev <- read.table("datafile/DEV_SAMPLE.csv", sep = ",", header = T)
RFDF.holdout <- read.table("datafile/HOLDOUT_SAMPLE.csv", sep = ",", header = T)
c(nrow(RFDF.dev), nrow(RFDF.holdout))

##install.packages("randomForest")
library(randomForest)

## Calling syntax to build the Random Forest
RF <- randomForest(as.factor(Target) ~ ., data = RFDF.dev[,-1],
    ntree=100,            ## number of trees to be built
    mtry = 3,             ## number of variables randomly sampled as candidate at each split
    nodesize = 10,        ## minimum number of records in terminal node
    importance=TRUE )     ## should importance of predictors be assessed
print(RF)
```

OOB Estimate of error rate

```
> print(RF)

Call:
 randomForest(formula = as.factor(Target) ~ ., data = RFDF.dev[,      -1],
ntree = 100, mtry = 3, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 3

        OOB estimate of  error rate: 8.29%
Confusion matrix:
      0   1 class.error
0 12667  98 0.007677242
1  1063 172 0.860728745
```

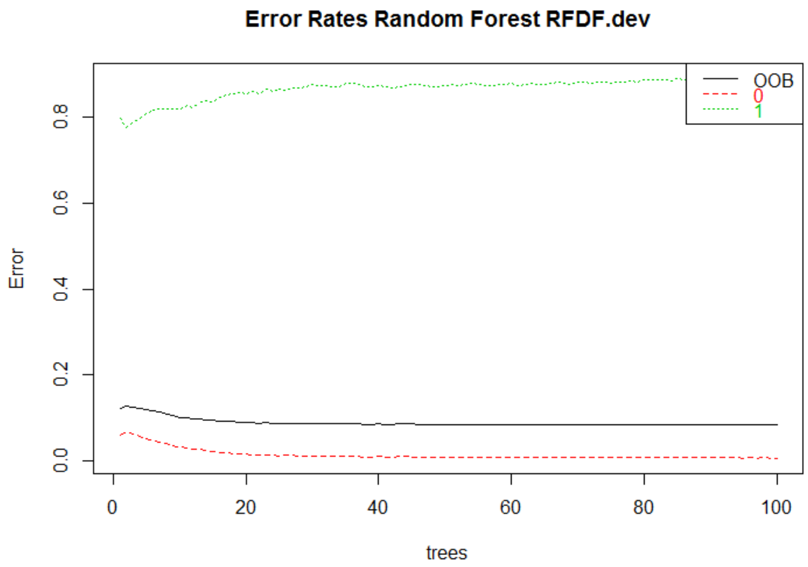
OOB Error Rate Computation Steps

- Sample left out (out-of-bag) in K^{th} tree is classified using the K^{th} tree
- Assume j cases are mis-classified
- Proportion of time that j is not equal to true class averaged over all cases is the oob estimate of error rate

OOB Error Rate … contd

- OOB Estimate of Error Rate is dependent on two key factors
  - nTree
  - Mtry

```
plot(RF, main="")
legend("topright", c("OOB", "0", "1"), text.col=1:6, lty=1:3, col=1:3)
title(main="Error Rates Random Forest RFDF.dev")
```



Variable Importance

List the importance of the variables.

```
impVar <- round(randomForest::importance(RF), 2)
impVar[order(impVar[,3], decreasing=TRUE),]
```

	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
Occupation	63.56	23.53	69.45	115.16
No_OF_CR_TXNS	62.10	35.10	67.64	200.48
Holding_Period	17.11	63.47	42.70	216.26
Gender	43.21	-12.98	41.54	37.41
Balance	21.34	33.06	33.21	275.39
Age	24.29	9.52	27.95	128.85
AGE_BKT	18.28	13.70	22.19	94.17
SCR	-0.10	21.25	9.99	264.89

Variable Importance

- Random Forest computes two measures of Variable Importance
  - Mean Decrease in Accuracy
  - Mean Decrease in Gini

- Mean Decrease in Accuracy is based on permutation
  - Randomly permute values of a variable for which importance is to be computed in the OOB sample
  - Compute the Error Rate with permuted values
  - Compute decrease in OOB Error rate (Permuted - Not permuted)
  - Average the decrease over all the trees

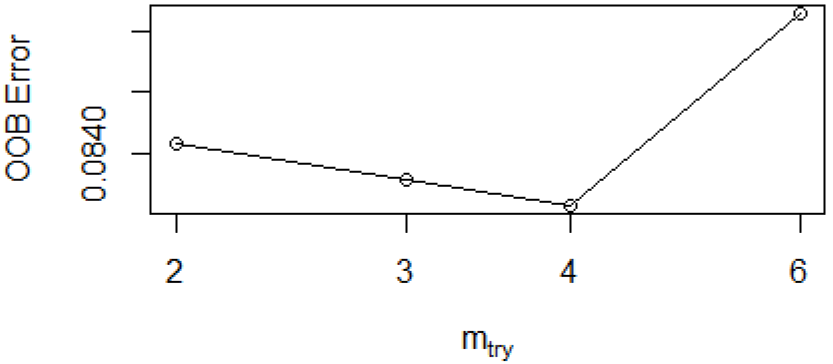
- Mean Decrease in Gini is computed as "**total decrease in node impurities from splitting on the variable, averaged over all trees**"

Finding optimal mtry value

```
## Tuning Random Forest
tRF <- tuneRF(x = RFDF.dev[,-c(1,2)],
          y=as.factor(RFDF.dev$Target),
          mtryStart = 3,
          ntreeTry=100,
          stepFactor = 1.5,
          improve = 0.0001,
          trace=TRUE,
          plot = TRUE,
          doBest = TRUE,
          nodesize = 10,
          importance=TRUE
          )
```

Parameter Explanation
x – predictor variables
y – Target Variable
mtryStart – starting value of mtry
ntreeTry – No of tree used for tuning
stepFactor – steps to increase (deflate) mtry
improve – the relative oob by atleast this much
trace – print the trace or not
plot – plot OOB vs mtry graph or not
doBest – Finally build the RF using optimal mtry
nodesize – min terminal node size
importance – compute variable importance or not

```
mtry = 3  OOB error = 8.38%
Searching left ...
mtry = 2        OOB error = 8.41%
-0.00341006 0.0001
Searching right ...
mtry = 4        OOB error = 8.36%
0.002557545 0.0001
mtry = 6        OOB error = 8.51%
-0.01880342 0.0001
```



Measuring RF Model performance

Syntax remains same as for the earlier model

```
## Scoring syntax
RFDF.dev$predict.class <- predict(tRF, RFDF.dev, type="class")
RFDF.dev$predict.score <- predict(tRF, RFDF.dev, type=“prob")

## deciling
RFDF.dev$deciles <- decile(RFDF.dev$predict.score[,2])

## Ranking code
library(data.table)
tmp_DT = data.table(RFDF.dev)
rank <- tmp_DT[, list(
  cnt = length(Target),
  cnt_resp = sum(Target),
  cnt_non_resp = sum(Target == 0)) ,
  by=deciles][order(deciles)]
rank$rrate <- rank$cnt_resp * 100 / rank$cnt;
rank$cum_resp <- cumsum(rank$cnt_resp)
rank$cum_non_resp <- cumsum(rank$cnt_non_resp)
rank$cum_rel_resp <- rank$cum_resp / sum(rank$cnt_resp);
rank$cum_rel_non_resp <- rank$cum_non_resp /
sum(rank$cnt_non_resp);
rank$ks <- abs(rank$cum_rel_resp - rank$cum_rel_non_resp);
rank
```

…contd

AUC Computation
```
library(ROCR)
pred <- prediction(RFDF.dev$predict.score[,2],
RFDF.dev$Target)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])
auc <- performance(pred,"auc");
auc <- as.numeric(auc@y.values)
```

Gini Compuation
```
library(ineq)
gini = ineq(RFDF.dev$predict.score[,2], type="Gini")
```

Printing the model performance statistics
```
with(RFDF.dev, table(Target, predict.class))
auc
KS
gini
```

Compare RF Model Performance with CART Model

```
> with(RFDF.dev, table(Target, predict.class))
      predict.class
Target     0     1
     0 12758     7
     1   797   438
> auc
[1] 0.9949836
> KS
[1] 0.9571465
> gini
[1] 0.7276404
```

Mis-Class = 5.7%

Hold Out Sample Testing

Scoring syntax

RFDF.holdout$predict.class <- predict(tRF, RFDF.holdout, type="class")

RFDF.holdout$predict.score <- predict(tRF, RFDF.holdout, type="prob")

with(RFDF.holdout, table(Target, predict.class))

```
> with(RFDF.holdout, table(Target, predict.class))
      predict.class
Target    0    1
     0 5474   28
     1  437   61
```

Mis-Class = 7.8%

Comparing Mis-Classification Rate of Dev & Hold Out we can say that there is Over-Fitting… however, this mis-classification would have been low if we would have many predictor variables

Why I like RF technique?

- … very good technique to pacify Business Users

Variable Category	Variable Name	Variable Description	Variable Name	Variable Description
TxnMode	no_of_csh_dep_txns_in_mth_	Number of cash deposit transactions	tot_csh_dep_amt_in_mth_	Total cash deposit amount
	no_of_l_u_non_csh_cr_txns_in_mth_	Number of all user inititated non-cash credit (deposit) transactions	tot_u_non_csh_cr_amt_in_mth_	Total cheque deposit amount
	no_of_chq_cr_txns_in_mth_	Number of cheque deposit transactions	tot_chq_cr_amt_in_mth_	Total user inititated non-cash credit (deposit) amount
	no_of_csh_wdl_txns_in_mth_	Number of cash withdrawal transactions	tot_csh_wdl_amt_in_mth_	Total cash withdrawal amount
	no_of_l_u_non_csh_dr_txns_in_mth_	Number of all user inititated non-cash debit transactions	tot_u_non_csh_dr_amt_in_mth_	Total cheque issued amount
	no_of_chq_dr_txns_in_mth_	Number of cheque issued transactions	tot_chq_dr_amt_in_mth_	Total user inititated non-cash debit amount
Cr Dr	no_of_l_cr_txns_in_mth_	Number of all credit transactions in month	tot_cr_amt_in_mth_	Total Credit Amount in month
	no_of_l_dr_txns_in_mth_	Number of all debit transactions in month	to_dr_amt_in_mth_	Total Debit Amount in month
	no_of_l_u_cr_txns_in_mth_	Number of all user initiated credit transactions	tot_l_u_cr_amt_in_mth_	Total user initiated credit deposit
	no_of_l_u_dr_txns_in_mth_	Number of all user initiated debit transactions		Total user initiated debit amount
Channel	no_of_atm_csh_wdl_txns_in_mth_	Number of ATM cash withdrawal transactions		
	no_of_atm_csh_dep_txns_in_mth_	Number of ATM cash deposit transactions		
	no_of_br_csh_wdl_txns_in_mth_	Number of Branch cash withdrawal transactio		
	no_of_br_csh_dep_txns_in_mth_	Number of Branch cash deposit transactio		
	no_of_atm_chq_dep_txns_in_mth_	Number of ATM cheque deposit transa		
	no_of_atm_cr_txns_in_mth_	Number of deposits (Cash or cheq		
	no_of_br_cr_txns_in_mth_	Number of credit transaction d		
	no_of_net_cr_txns_in_mth_	Number of credits received		
	no_of_net_dr_txns_in_mth_	Number of transfers do		
	no_of_br_dr_txns_in_mth_	Number of debit tr		
	no_of_mb_txns_in_mth_	Number of Mob		
	no_of_pb_txns_in_mth_	Number of P		
	no_of_si_txns_in_mth_	Number		
	no_of_pos_txns_in_mth_	Num		
Purpose (Penal Charges)	no_of_aqb_chq_txns_in_mth_			
	no_of_iw_chq_bno_txns_in_mth_			
	no_of_ow_chq_bno_txns_in_mth_			
Commission & Oth Charges				
Purpose of ment				

- Typically you will have 300 – 500 variables for modeling
- With techniques like Logistic Regression you will be forced to drop variables because of multi-collinarity
- Business users will have their own hypothesis and would want collinear variables to be part of the model
- Ensemble techniques like RF helps you build models by considering multitude of predictor variable permutations

Challenges

- You do not get a Equation
- Somewhat of Black Box and hence not used in some industries like Banks for Risk Modeling

K2 Analytics
Building Skills, Building Individuals

Questions?? … Thankyou

Contact Us
ar.jakhotia@k2analytics.co.in