# need to predict if a person has diabetes or not

require(rpart)

df <- read.csv("diabetes.csv")

summary(df)

> summary(df)

Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome

Min. : 0.000 Min. : 0.0 Min. : 0.00 Min. : 0.00 Min. : 0.0 Min. : 0.00 Min. :0.0780 Min. :21.00 0:500

1st Qu.: 1.000 1st Qu.: 99.0 1st Qu.: 62.00 1st Qu.: 0.00 1st Qu.: 0.0 1st Qu.:27.30 1st Qu.:0.2437 1st Qu.:24.00 1:268

Median : 3.000 Median :117.0 Median : 72.00 Median :23.00 Median : 30.5 Median :32.00 Median :0.3725 Median :29.00

Mean : 3.845 Mean :120.9 Mean : 69.11 Mean :20.54 Mean : 79.8 Mean :31.99 Mean :0.4719 Mean :33.24

3rd Qu.: 6.000 3rd Qu.:140.2 3rd Qu.: 80.00 3rd Qu.:32.00 3rd Qu.:127.2 3rd Qu.:36.60 3rd Qu.:0.6262 3rd Qu.:41.00

Max. :17.000 Max. :199.0 Max. :122.00 Max. :99.00 Max. :846.0 Max. :67.10 Max. :2.4200 Max. :81.00

> str(df)

'data.frame': 768 obs. of 9 variables:

$ Pregnancies : int 6 1 8 1 0 5 3 10 2 8 ...

$ Glucose : int 148 85 183 89 137 116 78 115 197 125 ...

$ BloodPressure : int 72 66 64 66 40 74 50 0 70 96 ...

$ SkinThickness : int 35 29 0 23 35 0 32 0 45 0 ...

$ Insulin : int 0 0 0 94 168 0 88 0 543 0 ...

$ BMI : num 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...

$ DiabetesPedigreeFunction: num 0.627 0.351 0.672 0.167 2.288 ...

$ Age : int 50 31 32 21 33 30 26 29 53 54 ...

$ Outcome : int 1 0 1 0 1 0 1 0 1 1 ...

## Since outcome is the one to be predicted , we need to convert Outcome as factor variable.

## Pregnancies are also values ranging from 1 to 17, it is a descrete variable with less range, so we can convert pregnancies also as a factor.

> df$Pregnancies = as.factor(df$Pregnancies )

> df$Outcome = as.factor(df$Outcome )

## Since this is a prediction of binomial variable, start modelling using a simple classifier.

## Splitting data into training and testing datasets

# taking sample of 600 elements for training data

df\_train = df[sample(1:nrow(df),600),]

## considering remaining elements as test data

df\_test = df[-(list(as.integer(rownames(df\_train)))[[1]]),]

#Using Rpart (recursive partitoning of trees ) for intial analysis

#Using all the variables for prediction.

> model <- rpart(Outcome~., data = df\_train)

> model$cptable

CP nsplit rel error xerror xstd

1 0.23762376 0 1.0000000 1.0000000 0.05730470

2 0.09900990 1 0.7623762 0.7871287 0.05351689

3 0.01980198 2 0.6633663 0.7227723 0.05203282

4 0.01732673 7 0.5495050 0.7475248 0.05262410

5 0.01237624 12 0.4603960 0.7970297 0.05373017

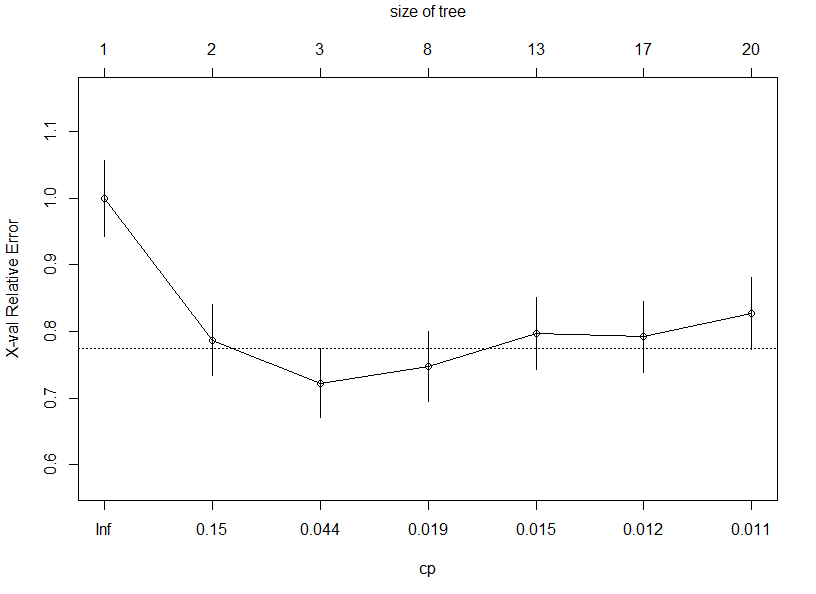
6 0.01155116 16 0.4059406 0.7920792 0.05362402

7 0.01000000 19 0.3712871 0.8267327 0.05434695

# The model is too complex. We need to prune the model with correct CP value . For this we need to plot the CP value vs relative Xerror.

> plotcp(model)

# We can see from the plot that there is drastic drop in the error rate when CP = 0.044 .So we will use this value to prune the model.



> pfit <- prune(model , cp = 0.044)

> table(predict(pfit, df\_test[,-9], type = "class"), df\_test[,9])

0 1

0 88 30

1 14 36

## Accuracy of prediction on test dataset = 124/168 ~ 74 %

# Checking with ensemble method Random forest on the same dataset to check if there is any increase in the prediction

> require(randomForest)

> model2 <- randomForest(Outcome~.,data =df\_train , ntree = 100 )

> model2

Call:

randomForest(formula = Outcome ~ ., data = df\_train, ntree = 100)

Type of random forest: classification

Number of trees: 100

No. of variables tried at each split: 2

OOB estimate of error rate: 23.33%

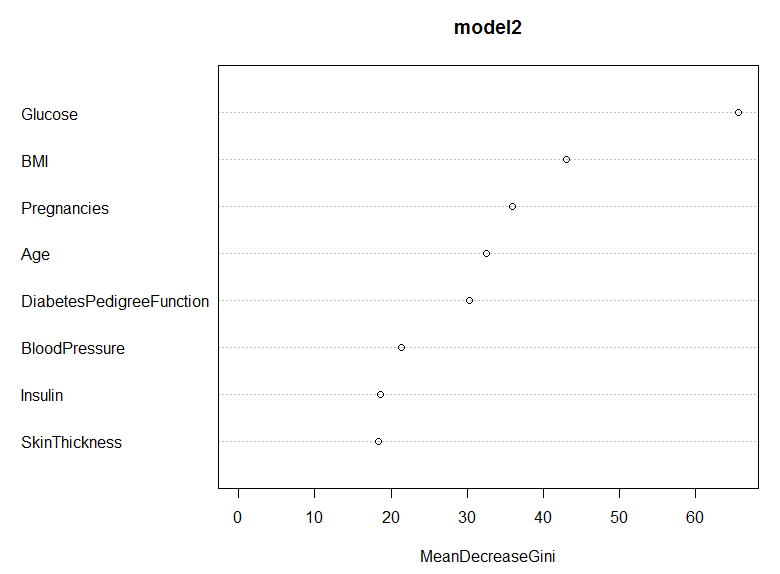
Confusion matrix:

0 1 class.error

0 337 61 0.1532663

1 79 123 0.3910891

# we can see that OOB error rate = 23.33 % . Let’s consider varImpPlot with respect to this model.



# we see that MeanDecreaseGini has not changed significantly after the variable Blood pressure. So , we will remove the bottom 2 variables(Insulin and Skin Thickness) that contributed to the model to check its performance.

> model3 <- randomForest(Outcome~Glucose + BMI + Pregnancies + Age + DiabetesPedigreeFunction + BloodPressure,data =df\_train , ntree = 100 )

> model3

Call:

randomForest(formula = Outcome ~ Glucose + BMI + Pregnancies + Age + DiabetesPedigreeFunction + BloodPressure, data = df\_train, ntree = 100)

Type of random forest: classification

Number of trees: 100

No. of variables tried at each split: 2

OOB estimate of error rate: 22.67%

Confusion matrix:

0 1 class.error

0 340 58 0.1457286

1 78 124 0.3861386

## We now see that model performance increased a little bit. OOB decreased from 23.33 to 22.67

## We now use this model to test the training dataset.

> table(predict(model3, df\_test[,-9], type = "class"), df\_test[,9])

0 1

0 82 24

1 20 42

## We are getting same accuracy when we used CART ~ 74 %.

## Lets use Naïve bayes classifier to check the performance with the above independent variables used for randomForest.

> model4 <- naiveBayes(Outcome~Glucose + BMI + Pregnancies + Age + DiabetesPedigreeFunction + BloodPressure,data =df\_train )

> table(predict(model4, df\_test[,-9], type = "class"), df\_test[,9])

0 1

0 87 25

1 15 41

## We see that this model gave us better accuracy over the tree models 76 % (approx.).

So we will use this model to consider our predictions.