1. (a) Develop boosted tree models (using either gbm or xgBoost) to predict loan\_status. Experiment

with different parameters using a grid of parameter values. Use cross-validation. Explain the rationale

for your experimentation. How does performance vary with parameters, and which parameter setting

you use for the 'best' model.

Model performance should be evaluated through use of same set of criteria as for the earlier models -

confusion matrix based, ROC analyses and AUC, cost-based performance.

Provide a table with comparative evaluation of all the best models from each methods; show their ROC

curves in a combined plot. Also provide profit-curves and 'best' profit' and associated cutoff. At this

cutoff, what are the accuracy values for the different models?

We have used xgboost boosted tree model for binary classification. The xgboost models with manipulating various parameters, is determined wherein eta(learning rate ), max\_depth(maximum depth of a tree), subsample (subsample ratio of the training instance. Setting it to 0.5 means that xgboost randomly collected half of the data instances to grow trees and this will prevent overfitting), lambda(regularization term). Each model is trained and by making various changes in its parameters, auc performance is measured best prediction is estimated and plotted.

The Model1 has the highest accuracy is 86.35% and it has the best iteration at 174th Sample. So it can be considered as the ‘best’ model

#Clean Data

---

title: "Clean\_Data"

output: html\_document

---

library(tidyverse)

library(lubridate)

library(ggplot2)

library(dplyr)

library(rpart)

library(rpart.plot)

library(caret)

library(C50)

library(ROCR)

library(ranger)

library(lift)

#library(CRAN)

library(dplyr,warn.conflicts = FALSE)

options(dplyr.summarise.inform = FALSE)

```

```{r}

###Are there missing values? What is the proportion of missing values in different variables? Explain how you will handle missing values for different variables. You should consider what the variable is about, and what missing values may arise from – for example, a variable monthsSinceLastDeliquency may have no value for someone who has not yet had a delinquency; what is a sensible value to replace the missing values in this case? Are there some variables you will exclude from your model due to missing values?

############# Missing values######################

### R code to plot NA Values

###Proportion of na's in different variables####

lcdf <- read.csv("lcData100K.csv")

dim(lcdf)

###### Drop variables with 100% NA values

lcdf <- lcdf %>% select\_if(function(x){!all(is.na(x))})

dim(lcdf)

################columns where there are missing values

colMeans(is.na(lcdf))[colMeans(is.na(lcdf))>0]

dim(lcdf)

###remove variables which have more than 60% missing values

colMeans(is.na(lcdf))>0.6

finalnona<-names(lcdf)[colMeans(is.na(lcdf))>0.6]

final\_lcdf <- lcdf %>% select(-finalnona)

dim(final\_lcdf)

################### columns with remaining missing values

colMeans(is.na(final\_lcdf))[colMeans(is.na(final\_lcdf))>0]

#summary of data in these columns final\_lcdf

nm<- names(final\_lcdf)[colSums(is.na(final\_lcdf))>0]

summary(final\_lcdf[, nm])

######Replace missing values with some value###

NoNAlcdf <- final\_lcdf %>% replace\_na(list(mths\_since\_last\_delinq=500, revol\_util=median(final\_lcdf$revol\_util, na.rm=TRUE), bc\_open\_to\_buy=median(final\_lcdf$bc\_open\_to\_buy, na.rm=TRUE), mo\_sin\_old\_il\_acct=1000, mths\_since\_recent\_bc=1000, mths\_since\_recent\_inq=50, num\_tl\_120dpd\_2m = median(lcdf$num\_tl\_120dpd\_2m, na.rm=TRUE),percent\_bc\_gt\_75 = median(final\_lcdf$percent\_bc\_gt\_75, na.rm=TRUE), bc\_util=median(final\_lcdf$bc\_util, na.rm=TRUE), avg\_cur\_bal=median(final\_lcdf$avg\_cur\_bal,na.rm = TRUE), num\_rev\_accts=mean(final\_lcdf$num\_rev\_accts,na.rm = TRUE), emp\_length=median(final\_lcdf$emp\_length,na.rm = TRUE), pct\_tl\_nvr\_dlq=mean(final\_lcdf$pct\_tl\_nvr\_dlq, na.rm = TRUE)))

#####To check if we have no more NA values #######

colMeans(is.na(NoNAlcdf))[colMeans(is.na(NoNAlcdf))>0]

dim(NoNAlcdf)

# Charged off loans will not have a last payment date - so we are excluding this one and this can cause data leakage

```

# dropping variables which cause data leakage

```{r}

varsOmit <- c('issue\_d','last\_pymnt\_d',

'zip\_code',

'emp\_title',

'last\_credit\_pull\_d',

'pymnt\_plan',

'addr\_state',

'policy\_code',

'disbursement\_method',

'title',

'term',

'funded\_amnt\_inv',

'out\_prncp',

'out\_prncp\_inv',

'total\_pymnt\_inv',

'total\_rec\_prncp',

'total\_rec\_int',

'debt\_settlement\_flag',

'hardship\_flag',

'application\_type',

'last\_pymnt\_amnt',

'last\_pymnt\_d',

'funded\_amnt\_inv',

'mths\_since\_last\_delinq',

'last\_pymnt\_amnt',

'total\_pymnt',

'issue\_d',

'funded\_amnt',

'last\_pymnt\_d',

'recoveries',

'num\_tl\_op\_past\_12m',

'collection\_recovery\_fee',

'total\_rec\_late\_fee',

'num\_tl\_120dpd\_2m',

'num\_tl\_30dpd',

'num\_tl\_90g\_dpd\_24m',

'earliest\_cr\_line',

'num\_tl\_op\_past\_12m',

'earliest\_cr\_line'

) #are there others?

mydata <- NoNAlcdf %>% select(-varsOmit)

#change chr to factors:

mydata$grade <- factor(mydata$grade, levels=c("A", "B","C","D", "E","F","G"))

mydata$sub\_grade <- factor(mydata$sub\_grade, levels=c("A1", "A2", "A3", "A4", "A5", "B1", "B2", "B3", "B4", "B5", "C1", "C2", "C3", "C4", "C5", "D1", "D2", "D3", "D4", "D5", "E1", "E2", "E3", "E4", "E5", "F1", "F2", "F3", "F4", "F5", "G1", "G2", "G3", "G4", "G5"))

mydata$initial\_list\_status <- factor(mydata$initial\_list\_status, levels=c("w", "f"))

mydata$loan\_status <- factor(mydata$loan\_status, levels=c("Fully Paid", "Charged Off"))

mydata$emp\_length <- factor(mydata$emp\_length, levels=c("n/a", "< 1 year","1 year","2 years", "3 years" , "4 years", "5 years", "6 years", "7 years" , "8 years", "9 years", "10+ years" ))

mydata$purpose <- fct\_recode(mydata$purpose)

mydata$home\_ownership <- as.factor((mydata$home\_ownership))

mydata$verification\_status<- as.factor(mydata$verification\_status)

#mydata$earliest\_cr\_line<- as.factor(mydata$earliest\_cr\_line)

# str(mydata)

dim(mydata)

```

# Split Train and Test Data

```{r }

#split the data into trn, tst subsets

nr=nrow(mydata)

mydata

trnIndex = sample(1:nr, size = round(0.7\*nr), replace=FALSE)

lcdfTrn=mydata[trnIndex,]

lcdfTst = mydata[-trnIndex,]

dim(lcdfTrn)

dim(lcdfTst)

str(lcdfTrn)

```

---

title: "Sid\_assgn2"

output: html\_document

---

```{r setup, include=FALSE}

knitr::opts\_chunk$set(echo = TRUE)

```

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the \*\*Knit\*\* button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r}

set.seed(12345)

#proportion of examples in the training sample

```

```{r}

#Needs all data to be numeric -- so we convert categorical (i.e. factor) variables using one-hot encoding – multiple ways to do this

# use the dummyVars function in the 'caret' package to convert factor variables to # dummy-variables

library(xgboost)

mydata2<-mydata

fdum<-dummyVars(~.,data=mydata2 %>% select(-loan\_status)) #do not include loan\_status for this

dxlcdf <- predict(fdum, mydata2)

dylcdf <- class2ind(mydata2$loan\_status, drop2nd = FALSE)

fplcdf <- dylcdf [ , 1] # or,

colcdf <- dylcdf [ , 2]

#Training subsets

dxlcdfTrn <- dxlcdf[trnIndex,]

colcdfTrn <- colcdf[trnIndex]

dxlcdfTst <- dxlcdf[-trnIndex,]

colcdfTst <- colcdf[-trnIndex]

dxTrn <- xgb.DMatrix( dxlcdfTrn, label=colcdfTrn)

dxTst <- xgb.DMatrix(dxlcdfTst, label=colcdfTst)

#Model 1(xg\_lsm1)

# add watchlist

xgbWatchlist <- list(train = dxTrn,eval = dxTst)

#we can watch the progress of learning by observing performance on these datasets

#training the model and getting predictions

#make a list of parameters

xgbParam <- list (max\_depth = 5, eta = 0.01,

objective = "binary:logistic",

eval\_metric="error", eval\_metric = "auc")

#iterative predictions

require(xgboost)

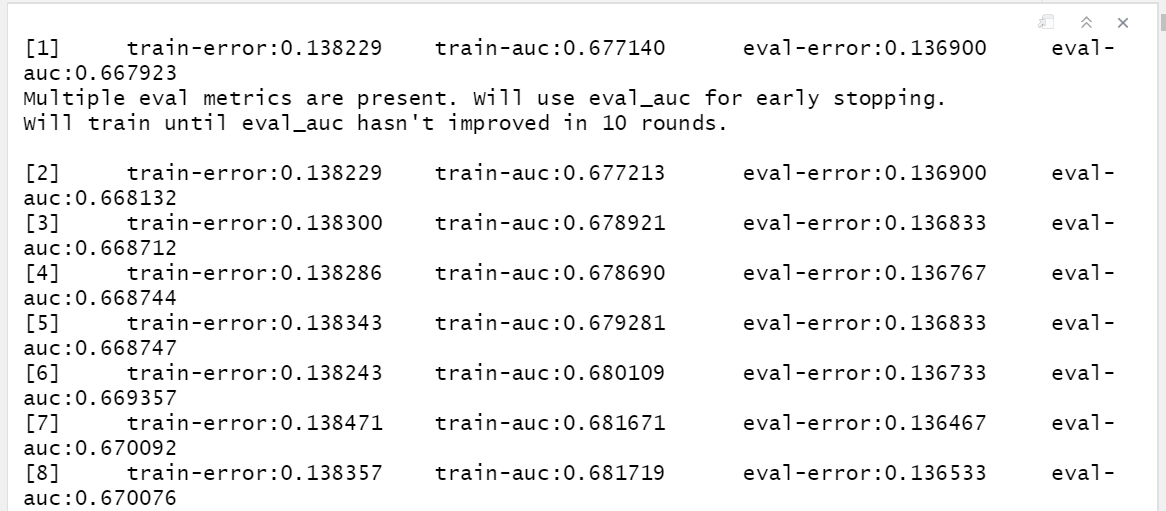
xgb\_lsM1 <- xgb.train ( xgbParam, dxTrn, nrounds = 500, xgbWatchlist, early\_stopping\_rounds = 10)

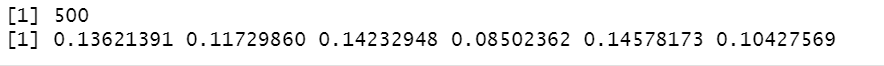
xgb\_lsM1$best\_iteration

xpredTrg1<-predict(xgb\_lsM1, dxTst) # best\_iteration is used

head(xpredTrg1)

```





```{r}

#cross-validation

xgbParam <- list (

max\_depth = 3, eta = 0.1,

objective = "binary:logistic",

eval\_metric="error", eval\_metric = "auc")

xgb\_lscv <- xgb.cv( xgbParam, dxTrn, nrounds = 500, nfold=5, early\_stopping\_rounds = 10 )

#best iteration

xgb\_lscv$best\_iteration

# or for the best iteration based on performance measure (among those specified in xgbParam)

best\_cvIter <- which.max(xgb\_lscv$evaluation\_log$test\_auc\_mean)

#learn the best model without eval

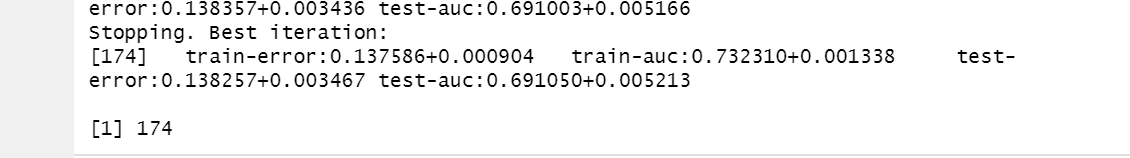
xgb\_lsbest <- xgb.train( xgbParam, dxTrn, nrounds = xgb\_lscv$best\_iteration )

#variable importance

xgb.importance(model = xgb\_lsbest) %>% view()

```

```{r}



#perfomance on test data for model 1

require(ROCR)

pred\_xgb\_lsM1=prediction(xpredTrg1,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

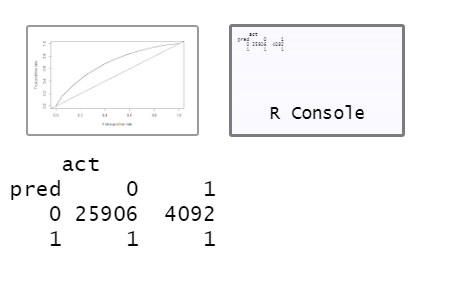
aucPerf\_xgb\_lsM1=performance(pred\_xgb\_lsM1, "tpr", "fpr")

plot(aucPerf\_xgb\_lsM1)

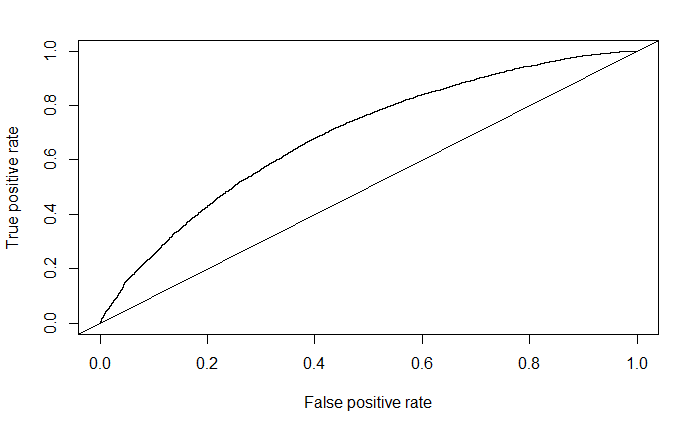
abline(a=0, b= 1)

#confusion matrix

table(pred=as.numeric(xpredTrg1>0.5), act=colcdfTst)



**Accuracy = 86.35%**



```

```{r}

xgbParam <- list (

max\_depth = 4, #eta = 0.01,

objective = "binary:logistic",

eval\_metric="error", eval\_metric = "auc")

#model 2 with eta = 1

xgb\_lsM2 <- xgb.train( xgbParam, dxTrn, nrounds = 500,xgbWatchlist, early\_stopping\_rounds = 10, eta=1 )

xgb\_lsM2$best\_iteration

xpredTrg2<-predict(xgb\_lsM2, dxTst)

#perfomance on test data for model 2

pred\_xgb\_lsM2=prediction(xpredTrg2,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

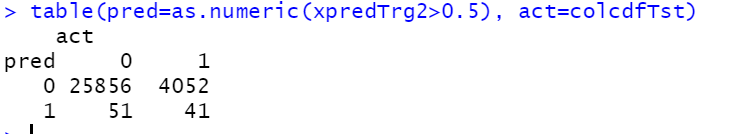
aucPerf\_xgb\_lsM2=performance(pred\_xgb\_lsM2, "tpr", "fpr")

plot(aucPerf\_xgb\_lsM2)

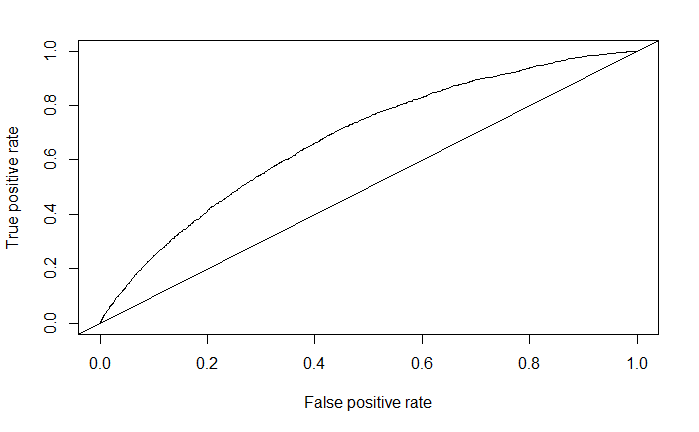
abline(a=0, b= 1)

#confusion matrix

table(pred=as.numeric(xpredTrg2>0.5), act=colcdfTst)



**Accuracy : 86.18%**



#model 3 with eta = 0.1

xgb\_lsM3 <- xgb.train( xgbParam, dxTrn, nrounds = 500,

xgbWatchlist, early\_stopping\_rounds = 10, eta=0.1 )

xgb\_lsM3$best\_iteration

xpredTrg3<-predict(xgb\_lsM3, dxTst)

#perfomance on test data for model 3

pred\_xgb\_lsM3=prediction(xpredTrg3,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

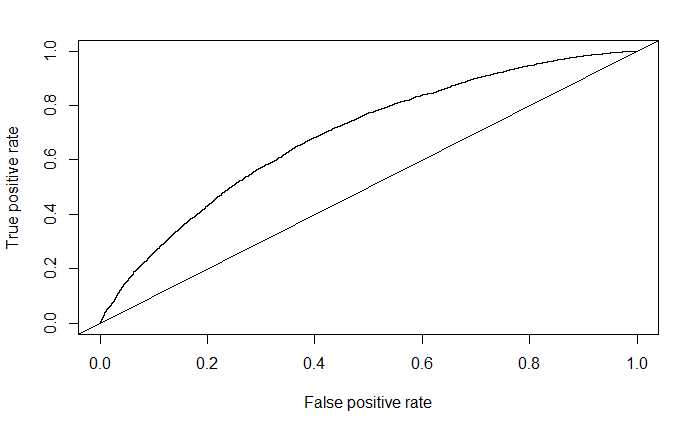
aucPerf\_xgb\_lsM3=performance(pred\_xgb\_lsM3, "tpr", "fpr")

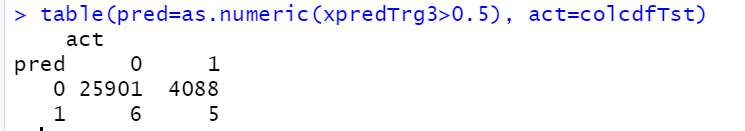
plot(aucPerf\_xgb\_lsM3)

abline(a=0, b= 1)

#confusion matrix

table(pred=as.numeric(xpredTrg3>0.5), act=colcdfTst)





**Accuracy = 86.33**

#model 4 with eta = 0.5

xgb\_lsM4 <- xgb.train( xgbParam, dxTrn, nrounds = 500,

xgbWatchlist, early\_stopping\_rounds = 10, eta=0.5 )

xgb\_lsM4$best\_iteration

xpredTrg4<-predict(xgb\_lsM4, dxTst)

#perfomance on test data for model 4

pred\_xgb\_lsM4=prediction(xpredTrg4,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

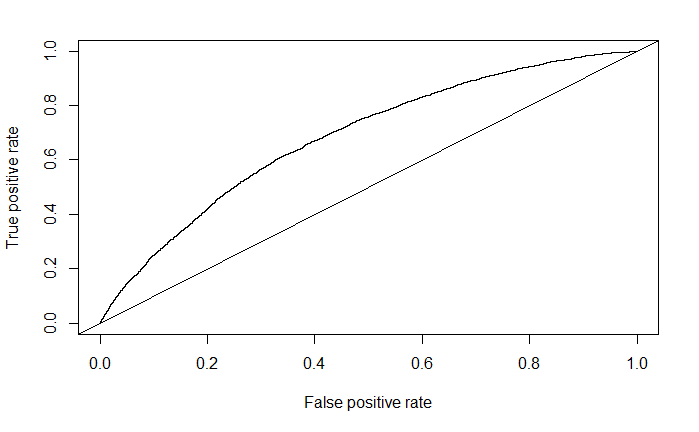
aucPerf\_xgb\_lsM4=performance(pred\_xgb\_lsM4, "tpr", "fpr")

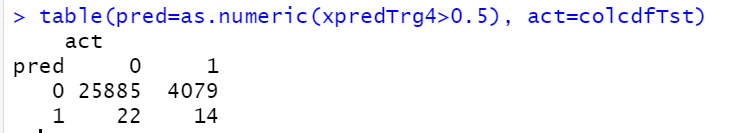
plot(aucPerf\_xgb\_lsM4)

abline(a=0, b= 1)

#confusion matrix

table(pred=as.numeric(xpredTrg4>0.5), act=colcdfTst)





**Accuracy = 86.28**

#model 5 with eta = 0.01

xgb\_lsM5 <- xgb.train( xgbParam, dxTrn, nrounds = 500,

xgbWatchlist, early\_stopping\_rounds = 10, eta=0.01 )

xgb\_lsM5$best\_iteration

xpredTrg5<-predict(xgb\_lsM5, dxTst)

#perfomance on test data for model 5

pred\_xgb\_lsM5=prediction(xpredTrg5,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

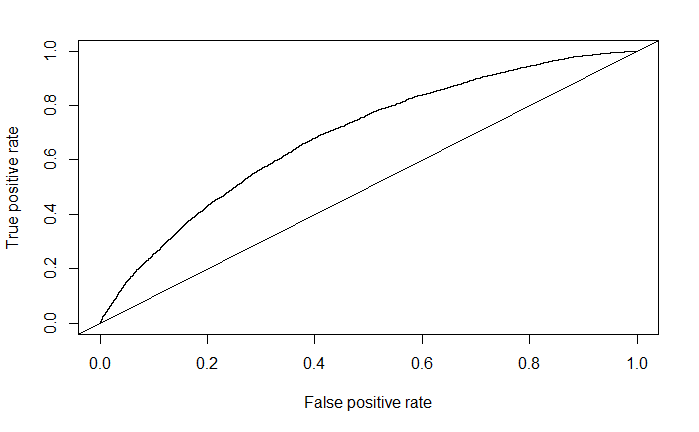
aucPerf\_xgb\_lsM5=performance(pred\_xgb\_lsM5, "tpr", "fpr")

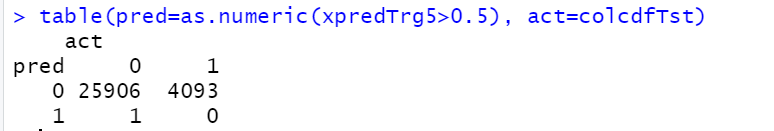
plot(aucPerf\_xgb\_lsM5)

abline(a=0, b= 1)

#confusion matrix

table(pred=as.numeric(xpredTrg5>0.5), act=colcdfTst)





**Accuracy = 86.35**

#model 6 with eta = 0.1, max\_depth=0.6

xgbParam <- list (

max\_depth = 6,

objective = "binary:logistic",

eval\_metric="error", eval\_metric = "auc")

xgb\_lsM6 <- xgb.train( xgbParam, dxTrn, nrounds = 500, xgbWatchlist,

early\_stopping\_rounds = 10, eta=0.1 )

xgb\_lsM6$best\_iteration

xpredTrg6<-predict(xgb\_lsM6, dxTst)

#perfomance on test data for model 6

pred\_xgb\_lsM6=prediction(xpredTrg6,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

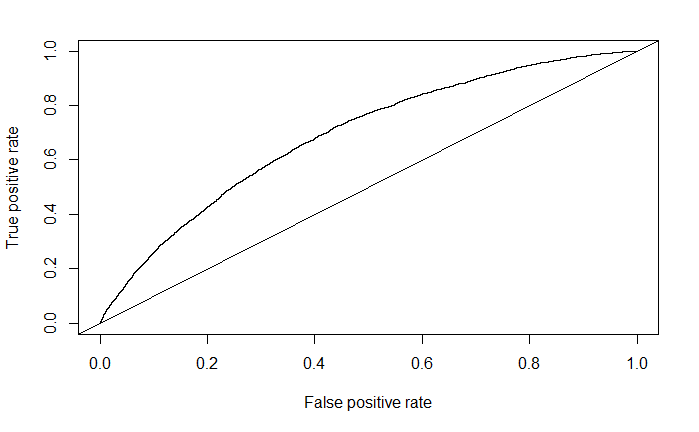
aucPerf\_xgb\_lsM6=performance(pred\_xgb\_lsM6, "tpr", "fpr")

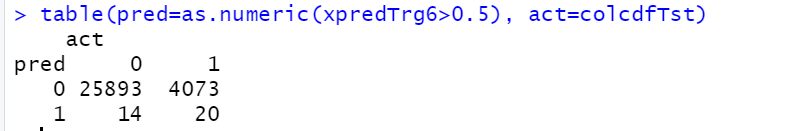
plot(aucPerf\_xgb\_lsM6)

abline(a=0, b= 1)

#confusion matrix

table(pred=as.numeric(xpredTrg6>0.5), act=colcdfTst)





**Accuracy = 86.31**

#model 7 same as 6 but with nrounds = 1000

xgb\_lsM7 <- xgb.train( xgbParam, dxTrn, nrounds = 1000, xgbWatchlist,

early\_stopping\_rounds = 10, eta=0.1)

xgb\_lsM7$best\_iteration

xpredTrg7<-predict(xgb\_lsM7, dxTst)

#perfomance on test data for model 7

pred\_xgb\_lsM7=prediction(xpredTrg7,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

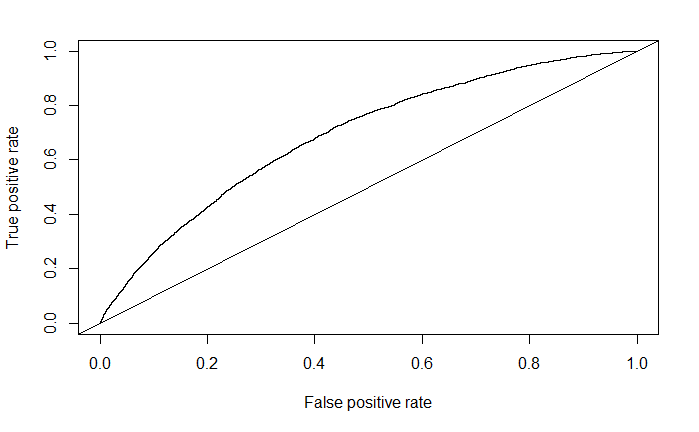
aucPerf\_xgb\_lsM7=performance(pred\_xgb\_lsM7, "tpr", "fpr")

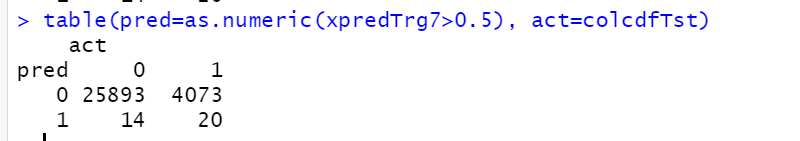
plot(aucPerf\_xgb\_lsM7)

abline(a=0, b= 1)

#confusion matrix

table(pred=as.numeric(xpredTrg7>0.5), act=colcdfTst)





**Accuracy = 86.31**

#model 8 same as 7 but with lambda=0.05, subsample=0.7, colsample\_bytree=0.5

xgb\_lsM8 <- xgb.train( xgbParam, dxTrn, nrounds = 1000, xgbWatchlist, early\_stopping\_rounds = 10,

eta=0.1, lambda=0.05, subsample=0.7, colsample\_bytree=0.5 )

xgb\_lsM8$best\_iteration

xpredTrg8<-predict(xgb\_lsM8, dxTst)

#perfomance on test data for model 8

pred\_xgb\_lsM8=prediction(xpredTrg8,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

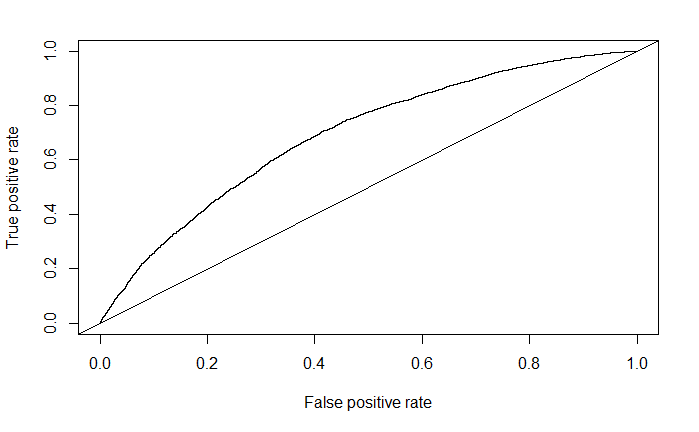
aucPerf\_xgb\_lsM8=performance(pred\_xgb\_lsM8, "tpr", "fpr")

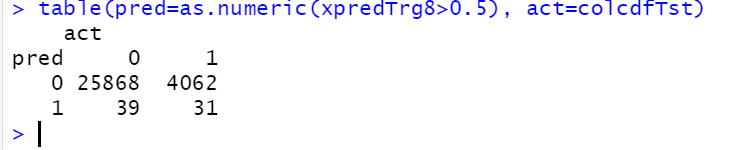
plot(aucPerf\_xgb\_lsM8)

abline(a=0, b= 1)

#confusion matrix

table(pred=as.numeric(xpredTrg8>0.5), act=colcdfTst)





**Accuracy = 86.22**

#model 9 same as 8 but with eta=0.01

xgb\_lsM9 <- xgb.train( xgbParam, dxTrn, nrounds = 1000, xgbWatchlist, early\_stopping\_rounds

= 10, eta=0.01, subsample=0.7, colsample\_bytree=0.5 )

xgb\_lsM9$best\_iteration

xpredTrg9<-predict(xgb\_lsM9, dxTst)

#perfomance on test data for model 9

pred\_xgb\_lsM9=prediction(xpredTrg9,lcdfTst$loan\_status,label.ordering = c("Fully Paid", "Charged Off"))

aucPerf\_xgb\_lsM9=performance(pred\_xgb\_lsM9, "tpr", "fpr")

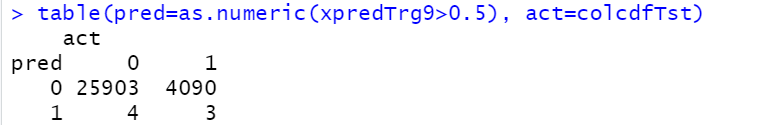
plot(aucPerf\_xgb\_lsM9)

abline(a=0, b= 1)

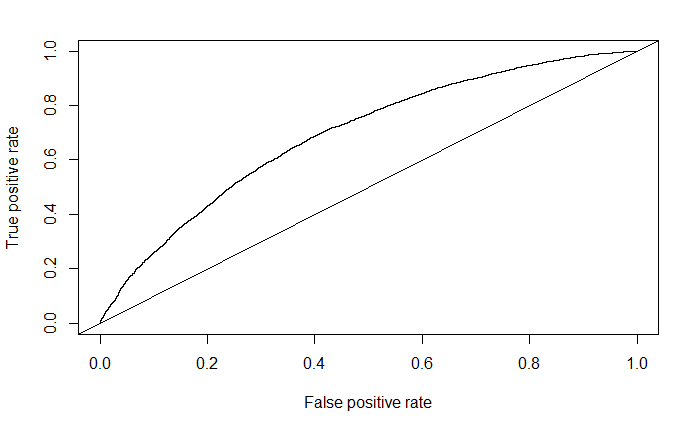
#confusion matrix

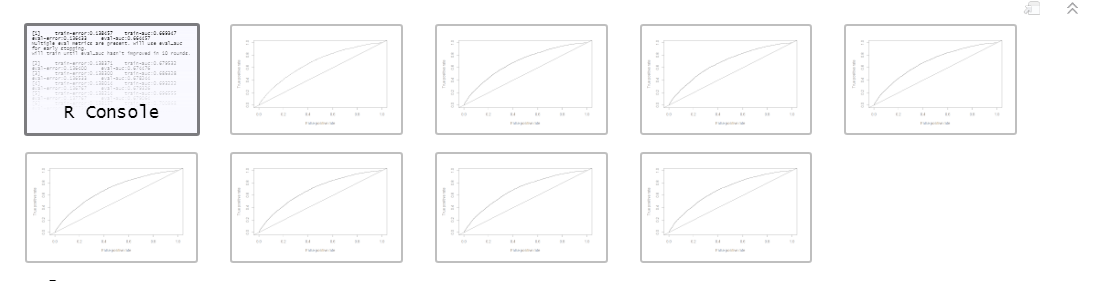
table(pred=as.numeric(xpredTrg9>0.5), act=colcdfTst)

```



**Accuracy = 86.34**





#Combination of all Plots

```{r}

plot(aucPerf\_xgb\_lsM9, col='red', main = "Consolidated AUC for model9 & model8", cex = 0.6)

plot(aucPerf\_xgb\_lsM8, col='green', add=TRUE)

plot(aucPerf\_xgb\_lsM6, col='blue', add=TRUE)

plot(aucPerf\_xgb\_lsM5, col='yellow', add=TRUE)

plot(aucPerf\_xgb\_lsM4, col='orange', add=TRUE)

plot(aucPerf\_xgb\_lsM3, col='pink', add=TRUE)

plot(aucPerf\_xgb\_lsM2, col='black', add=TRUE)

plot(aucPerf\_xgb\_lsM1, col='brown', add=TRUE)

legend('bottomright', c('model9','model8','model7','model6','model5','model4','model3','model2','model1' ), lty=1, col=c('red','green','blue','yellow','orange','pink','black','brown'), cex = 0.8)

```

