STAT 538 pROJECT

Dataset: UCI default of credit card clients Data Set

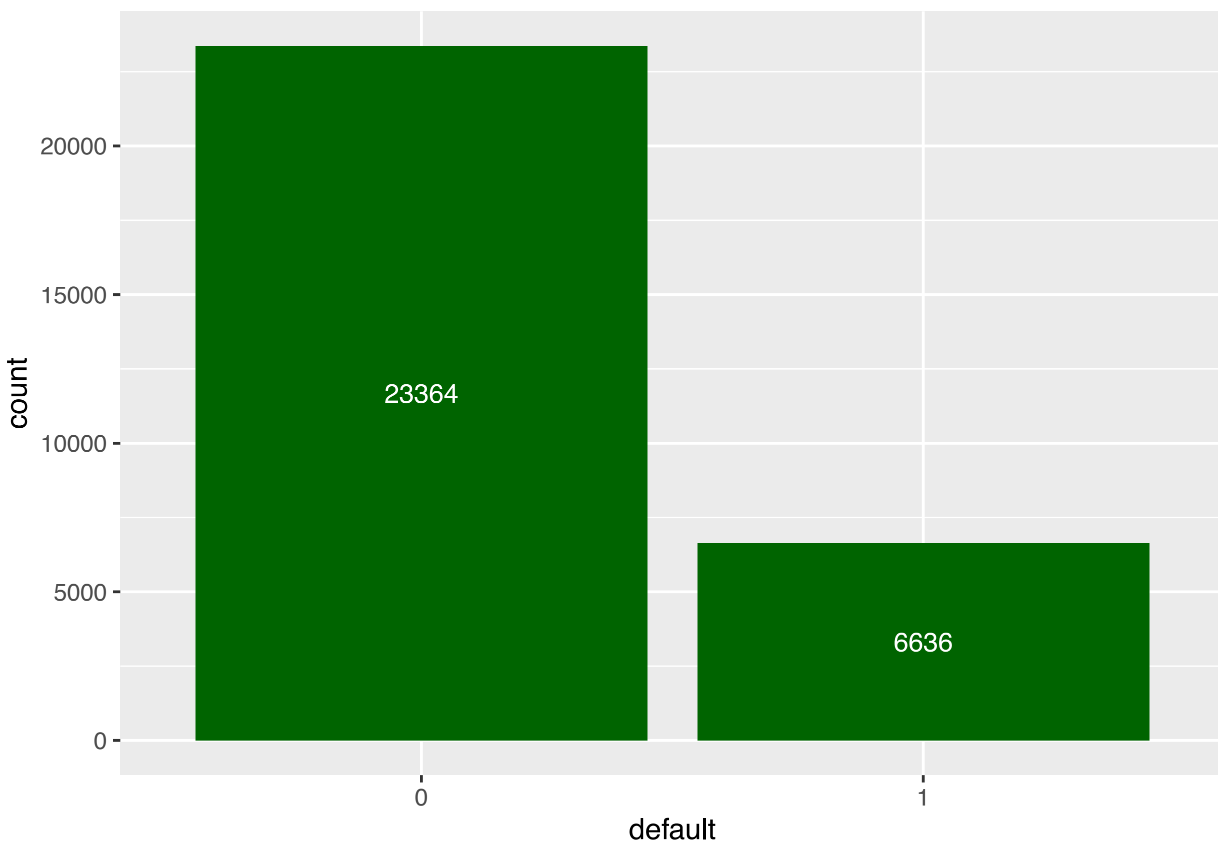
There are 25 variables: - ID: ID of each client
- LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)
- AGE: Age in years
- PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY_2: Repayment status in August, 2005 (scale same as above)
- PAY_3: Repayment status in July, 2005 (scale same as above)
- PAY_4: Repayment status in June, 2005 (scale same as above)
- PAY_5: Repayment status in May, 2005 (scale same as above)
- PAY_6: Repayment status in April, 2005 (scale same as above)
- BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)
- BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)
- BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)
- BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)
- BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)
- vBILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)
- PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)
- PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)
- PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)
- PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)
- PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)
- PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)
- default.payment.next.month: Default payment (1=yes, 0=no)

```r
library(reshape2)
library(ggplot2)
library(dplyr)
library(wesanderson)
library(gridExtra)
library(caret)
library(mlbench)
library(MASS)
library(lmtest)
library(e1071)
library(RcmdrMisc)
library(yardstick)
library(ROCR)
library(klaR)
library(pROC)
library(tidyr)
library(grid)
```

```r
df <- readxl::read_excel("~/Downloads/default _of_credit_card_clients.xls")
df = subset(df, select = -c(ID) )

# Rename the response variable
names(df)[24] <- "default"

# Dealing with categorical variables
df$SEX<-factor(df$SEX, levels=1:2, labels=c("male", "female"))
df$EDUCATION<-factor(df$EDUCATION,levels=0:6,
                     labels=c("Graduate School", "University","High school",
                              "Unknown 1","Others","Unknown 2","Unknown 3"))

df$MARRIAGE<-factor(df$MARRIAGE,levels=0:3,labels=c("Others","Unmarried","Married","Unkown"))
df$default <- as.factor(df$default)
```

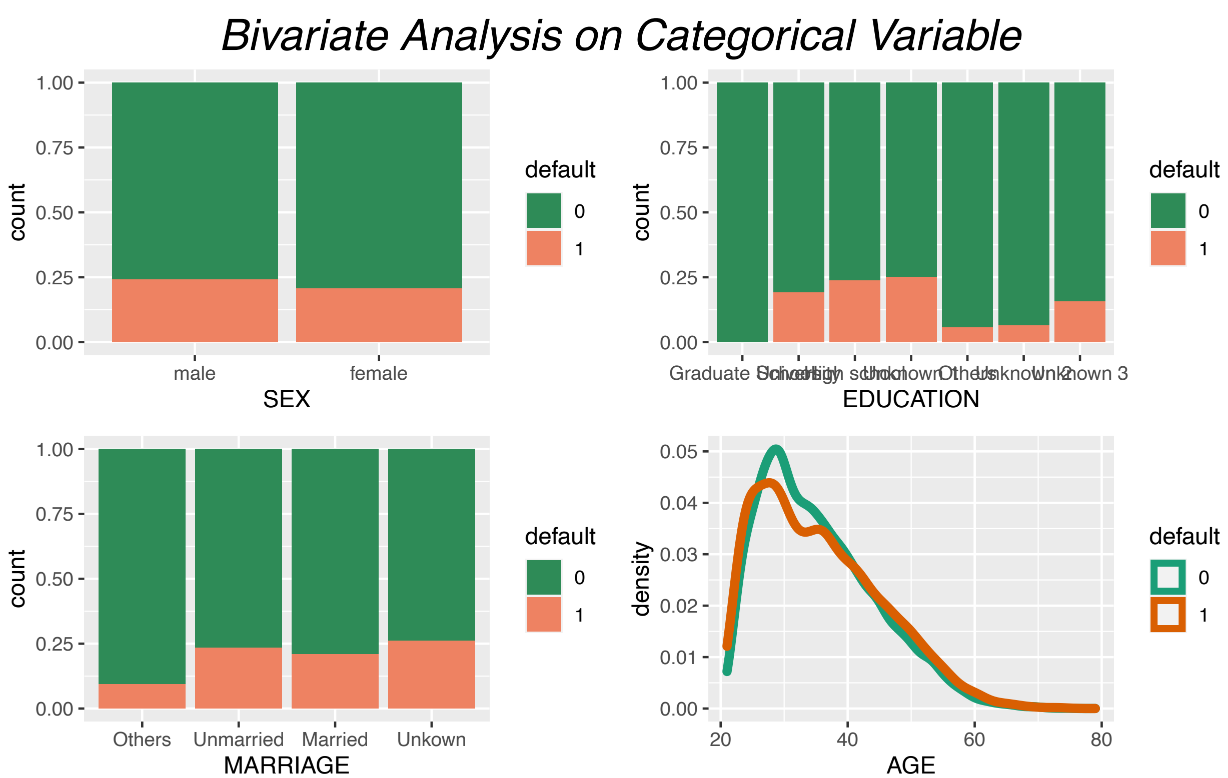
Exploratory Data Analysis

```r
ggplot(data = df, aes(x = default)) +
geom_bar(stat = "count",fill = "darkgreen") +
stat_count(geom = "text", colour = "white", size = 3.5,
aes(label = ..count..),position=position_stack(vjust=0.5))
```



Bivariate Analysis for categorical variables

```
p1 <- df %>% ggplot(aes(x = SEX, fill = default)) +
        geom_bar(position = "fill") +
        scale_fill_manual(values=c('seagreen4','salmon2'))

p2 <- df %>% ggplot(aes(x=EDUCATION, fill = default)) +
        geom_bar(position = "fill") +
        scale_fill_manual(values=c('seagreen4','salmon2'))

p3 <- df %>% filter(!is.na(MARRIAGE)) %>%
        ggplot(aes(x=MARRIAGE, fill = default)) +
        geom_bar(position = "fill") +
        scale_fill_manual(values=c('seagreen4','salmon2'))

p4 <- df %>% ggplot(aes(x=AGE, color=default)) +geom_density(size =2) + scale_color_brewer(palette="Dar


grid.arrange(ggplotGrob(p1),ggplotGrob(p2),ggplotGrob(p3),ggplotGrob(p4),
             ncol=2,top = textGrob("Bivariate Analysis on Categorical Variable",gp=gpar(fontsize=20,fon
```

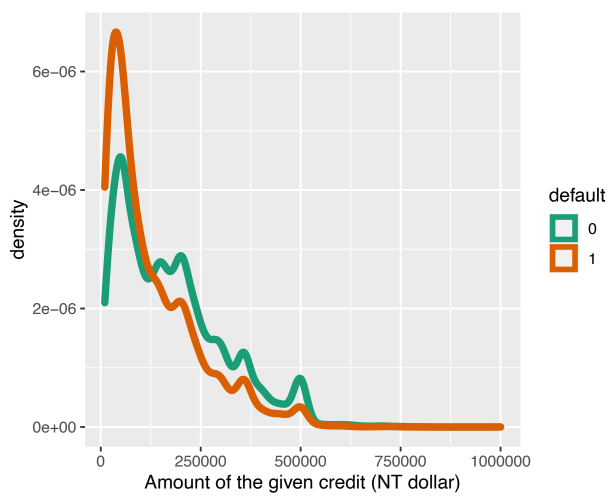


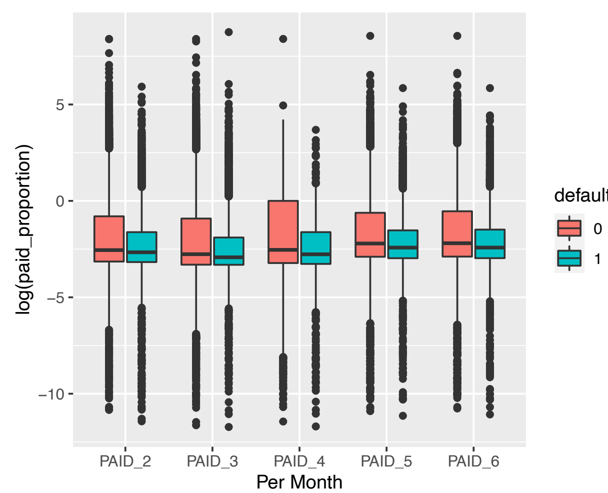
Exploring for Numerical variables

```
p5 <- df %>% ggplot(aes(x=LIMIT_BAL, color=default)) +
  geom_density(size =2) + scale_color_brewer(palette="Dark2") +
  labs(x = "Amount of the given credit (NT dollar)")
```

```r
df <- df %>% mutate(PAID_1 = ifelse(BILL_AMT1 != 0,PAY_AMT1/BILL_AMT1,0),
                    PAID_2 = ifelse(BILL_AMT2 != 0, PAY_AMT2/BILL_AMT2,0),
                    PAID_3 = ifelse(BILL_AMT3!= 0, PAY_AMT3/BILL_AMT3,0),
                    PAID_4 = ifelse(BILL_AMT4!= 0, PAY_AMT3/BILL_AMT4,0),
                    PAID_5 = ifelse(BILL_AMT5!= 0, PAY_AMT1/BILL_AMT5,0),
                    PAID_6 = ifelse(BILL_AMT6!= 0, PAY_AMT1/BILL_AMT6,0))

# Log-transformed PAID variables for better visualizations
paids <- df[,26:30]
paids <- log(paids)
default <- df$default
df_tem <- cbind(paids,default)
df.m <- melt(df_tem, id.var = "default")

p6 <- ggplot(data = df.m, aes(x=variable, y=value)) +
      geom_boxplot(aes(fill=default)) +
      labs(y="log(paid_proportion)", x="Per Month",main = "Log(paid_proportion) Per Month")
```

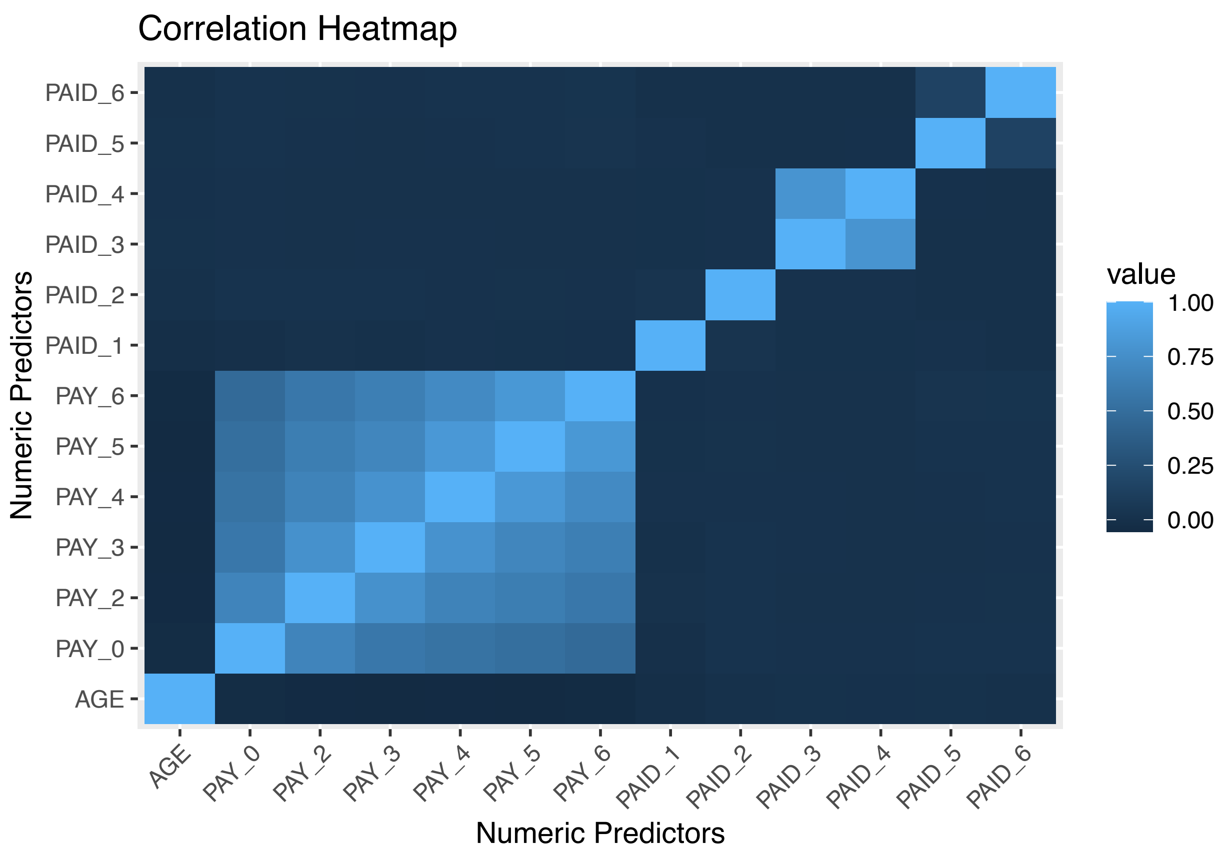
```r
grid.arrange(ggplotGrob(p5),ggplotGrob(p6),ncol=2)
```





Correlation Heatmap

```r
df_num <- df[c(5:11,25:30)]
qplot(x = Var1, y = Var2,
      data = melt(cor(df_num)),
      fill = value,
      geom = "tile") + theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
      labs(x = "Numeric Predictors",y = "Numeric Predictors") +
       ggtitle("Correlation Heatmap")
```



Initial Feature Selection

correlation matrix

Principle: generally, we want to remove attributes with an absolute correlation of 0.75 or higher.

```r
# correlation matrix
correlationMatrix <- cor(df_num)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)

# Display the name of to-be-removed variables
high_cor_col <- matrix(NA,1,4)
for (i in 1:length(highlyCorrelated)) {
  index <- highlyCorrelated[i]
  high_cor_col[1,i] <- names(df_num[,index])
}
print(high_cor_col)
```

```
##      [,1]    [,2]    [,3]    [,4]
## [1,] "PAY_4" "PAY_5" "PAY_3" "PAID_4"
```

```r
# Remove highly correlated predictcor
df_1 <- df[c(1:7,11,24:27,29:30)]
```

Spliting training set and test set

```r
train_index <- sample(1:nrow(df_1), 0.75 * nrow(df_1))
test_index <- setdiff(1:nrow(df_1), train_index)

X_train <- df_1[train_index, -9]
y_train <- df_1[train_index, "default"]

X_test <- df_1[test_index, -9]
y_test <- df_1[test_index, "default"]

train_df  <- cbind(X_train,y_train)
test_df <- cbind(X_test,y_test)
```

```r
prop.table(table(df_1$default))
```

```
##
##      0      1
## 0.7788 0.2212
```

```r
prop.table(table(train_df$default))
```

```
##
##         0         1
## 0.7789778 0.2210222
```

```r
prop.table(table(test_df$default))
```

```
##
##         0         1
## 0.7782667 0.2217333
```

Baseline model: Logistic Regression

Five models: - `full.mod`: Full models with all predict
- `for.BIC.model`: model returned by forward selection with BIC penalty - `for.AIC.model`: model returned by forward selection with AIC penalty - `back.BIC.model`: model returned by backward selection with BIC penalty - `back.AIC.model`: model returned by backward selection with AIC penalty

```r
####################################################### All models
full.mod <- glm(default~., data = train_df, family = binomial)
for.BIC.model <- stepwise(full.mod, direction = "forward", criterion = "BIC",trace = FALSE)
```

```
##
## Direction:  forward
## Criterion:  BIC
```

```r
for.AIC.model <- stepwise(full.mod, direction = "forward", criterion = "AIC",trace = FALSE)
```

```
##
## Direction:  forward
## Criterion:  AIC
```

```r
back.BIC.model <- stepwise(full.mod, direction = "backward", criterion = "BIC",trace = FALSE)
```

```
##
## Direction:  backward
## Criterion:  BIC
```

```r
back.AIC.model <- stepwise(full.mod, direction = "backward", criterion = "AIC",trace = FALSE)
```

```
##
## Direction:  backward
## Criterion:  AIC
```

```r
###################################################### Full model
# AIC and BIC
full.AIC = AIC(full.mod)
full.BIC = BIC(full.mod)

# Make predictions
probabilities <- full.mod %>% predict(test_df, type = "response")
predicted.classes <- ifelse(probabilities > 0.5,1,0)

# Prediction accuracy
observed.classes <- test_df$default
accuracy_full <- mean(predicted.classes == observed.classes,na.rm = TRUE)

#AUC
roc_obj <- roc(observed.classes, predicted.classes)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
full_AUC <- auc(roc_obj)

#ROC plot
pred <- prediction(predicted.classes, observed.classes)
perf_1 <- performance(pred,"tpr","fpr")

###################################################### AIC Forward
# AIC and BIC
for_AIC_AIC = AIC(for.AIC.model)
for_AIC_BIC = BIC(for.AIC.model)

# Make predictions
```

```r
probabilities <- predict(for.AIC.model, test_df, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
# Prediction accuracy
observed.classes <- test_df$default
accuracy_for_AIC <- mean(predicted.classes == observed.classes,na.rm = TRUE)

#AUC
roc_obj <- roc(observed.classes, predicted.classes)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
for_AIC_AUC <- auc(roc_obj)

#ROC plot
pred <- prediction(predicted.classes, observed.classes)
perf_2 <- performance(pred,"tpr","fpr")



##################################################### BIC Forward
# AIC and BIC
for_BIC_AIC = AIC(for.BIC.model)
for_BIC_BIC = BIC(for.BIC.model)

# Make predictions
probabilities <- predict(for.BIC.model, test_df, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
# Prediction accuracy
observed.classes <- test_df$default
accuracy_for_BIC <- mean(predicted.classes == observed.classes,na.rm = TRUE)

#AUC
roc_obj <- roc(observed.classes, predicted.classes)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
for_BIC_AUC <- auc(roc_obj)

#ROC plot
pred <- prediction(predicted.classes, observed.classes)
perf_3 <- performance(pred,"tpr","fpr")


##################################################### AIC Backward

# AIC and BIC
back_AIC_AIC = AIC(back.AIC.model)
back_AIC_BIC = BIC(back.AIC.model)

# Make predictions
```

```r
probabilities <- predict(back.AIC.model, test_df, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
# Prediction accuracy
observed.classes <- test_df$default
accuracy_back_AIC <- mean(predicted.classes == observed.classes,na.rm = TRUE)

#AUC
roc_obj <- roc(observed.classes, predicted.classes)
```

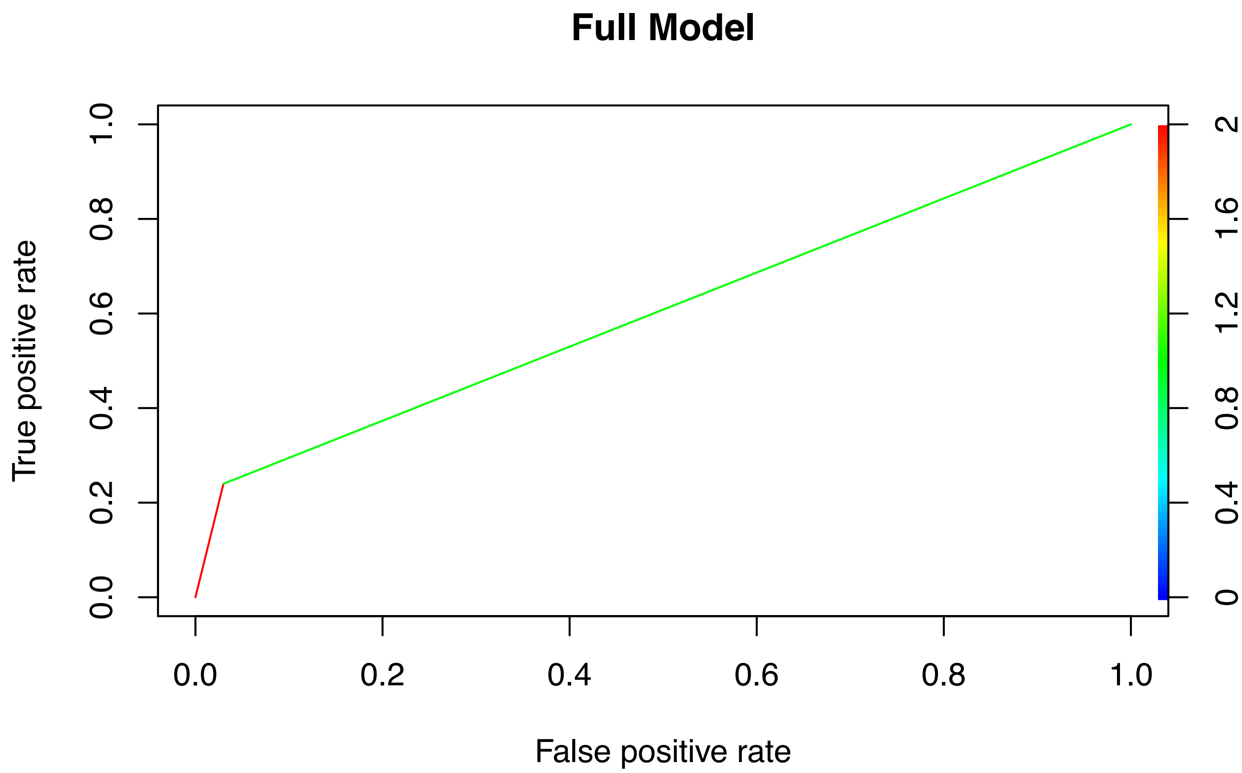
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
back_AIC_AUC <- auc(roc_obj)

#ROC plot
pred <- prediction(predicted.classes, observed.classes)
perf_4 <- performance(pred,"tpr","fpr")



###################################################### BIC Backforward
# AIC and BIC
back_BIC_AIC = AIC(back.BIC.model)
back_BIC_BIC = BIC(back.BIC.model)

# Make predictions
probabilities <- predict(back.BIC.model, test_df, type = "response")
predicted.classes <- ifelse(probabilities > 0.5, 1, 0)
# Prediction accuracy
observed.classes <- test_df$default
accuracy_back_BIC <- mean(predicted.classes == observed.classes,na.rm = TRUE)

#AUC
roc_obj <- roc(observed.classes, predicted.classes)
```

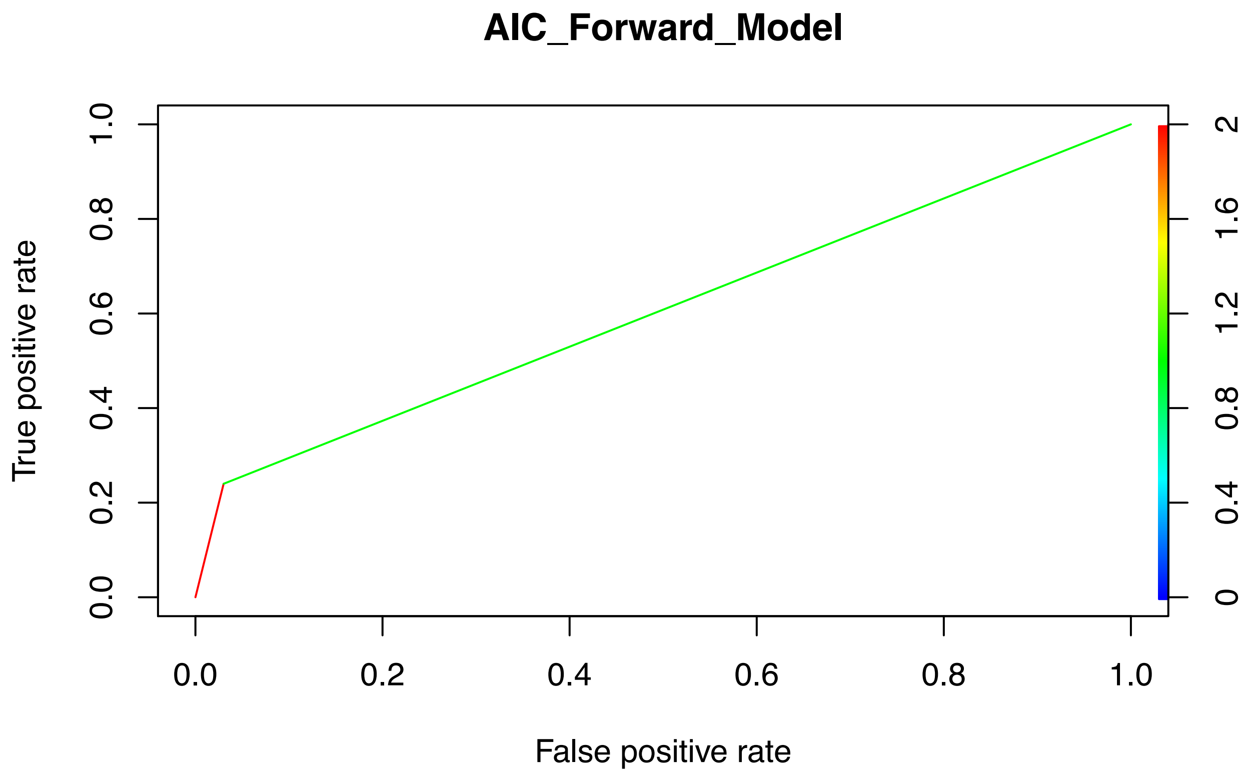
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
back_BIC_AUC <- auc(roc_obj)

#ROC plot
pred <- prediction(predicted.classes, observed.classes)
perf_5 <- performance(pred,"tpr","fpr")
```

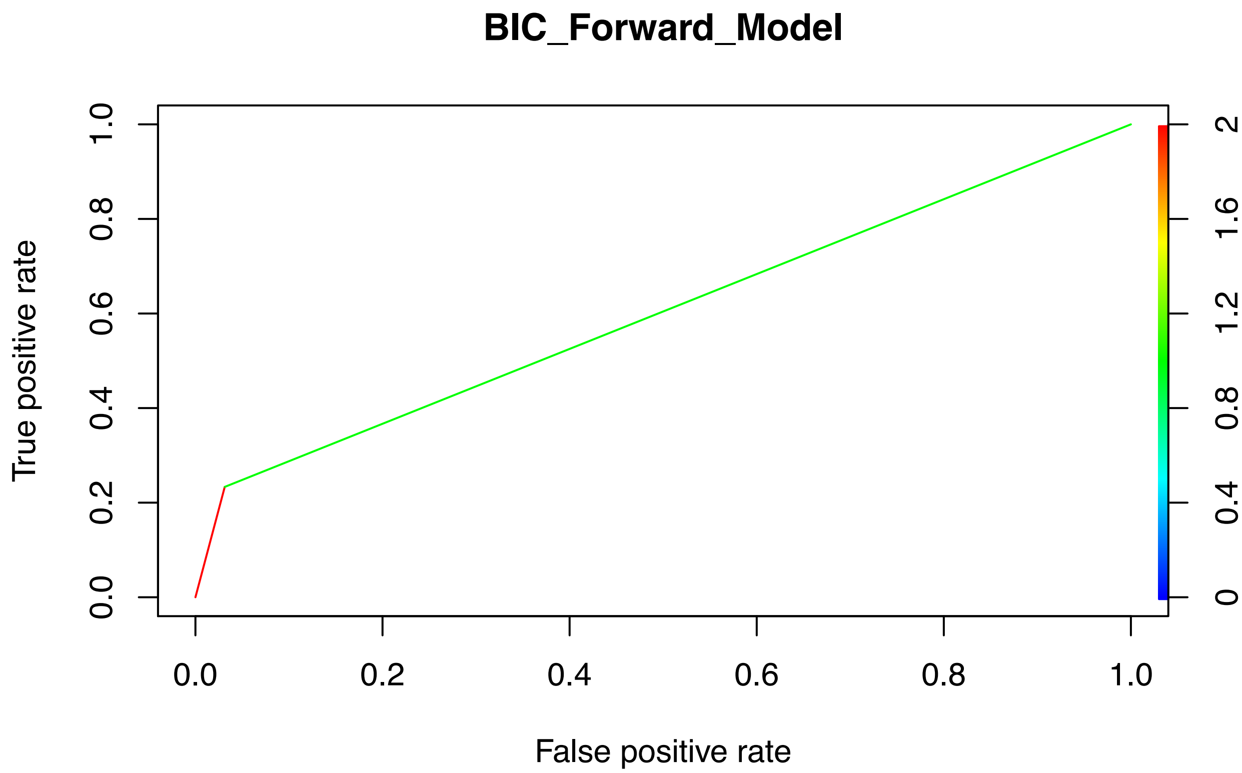
```r
#par(mfrow=c(3,2))
plot(perf_1,colorize=TRUE,main = "Full Model")
```



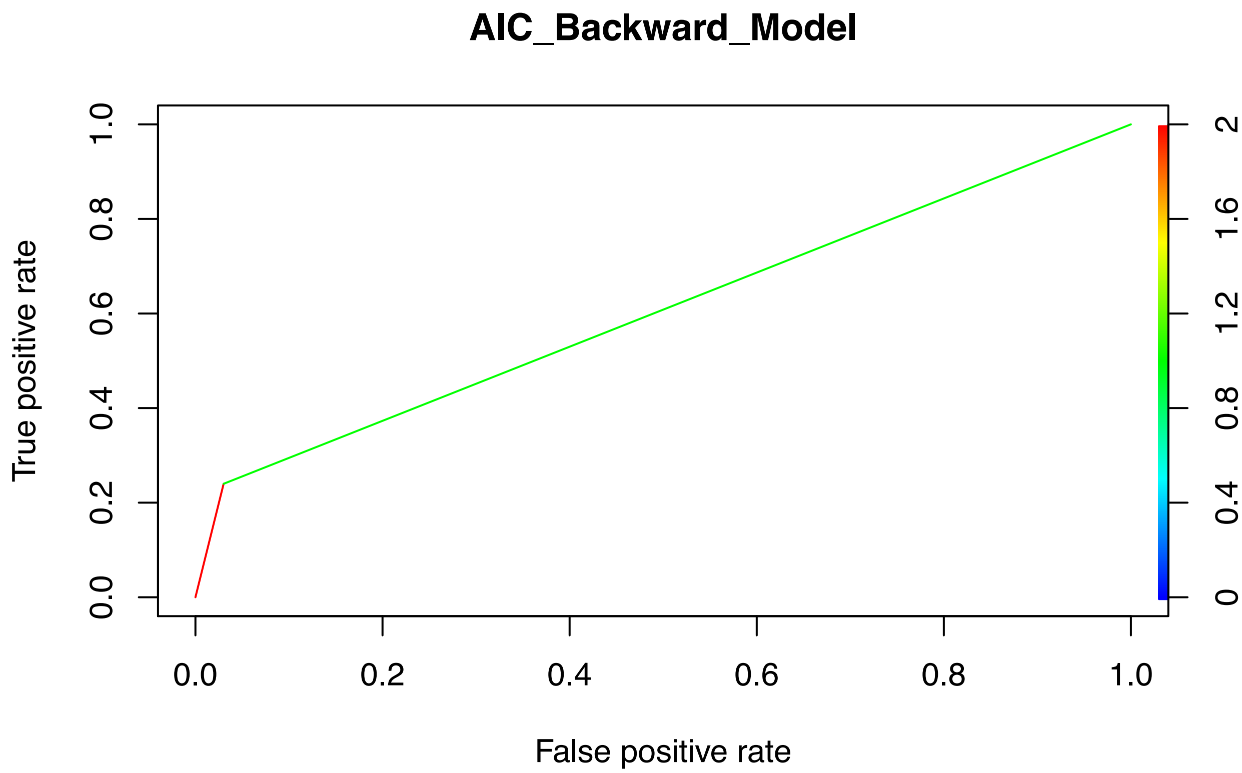

```
plot(perf_2,colorize=TRUE,main = "AIC_Forward_Model")
```



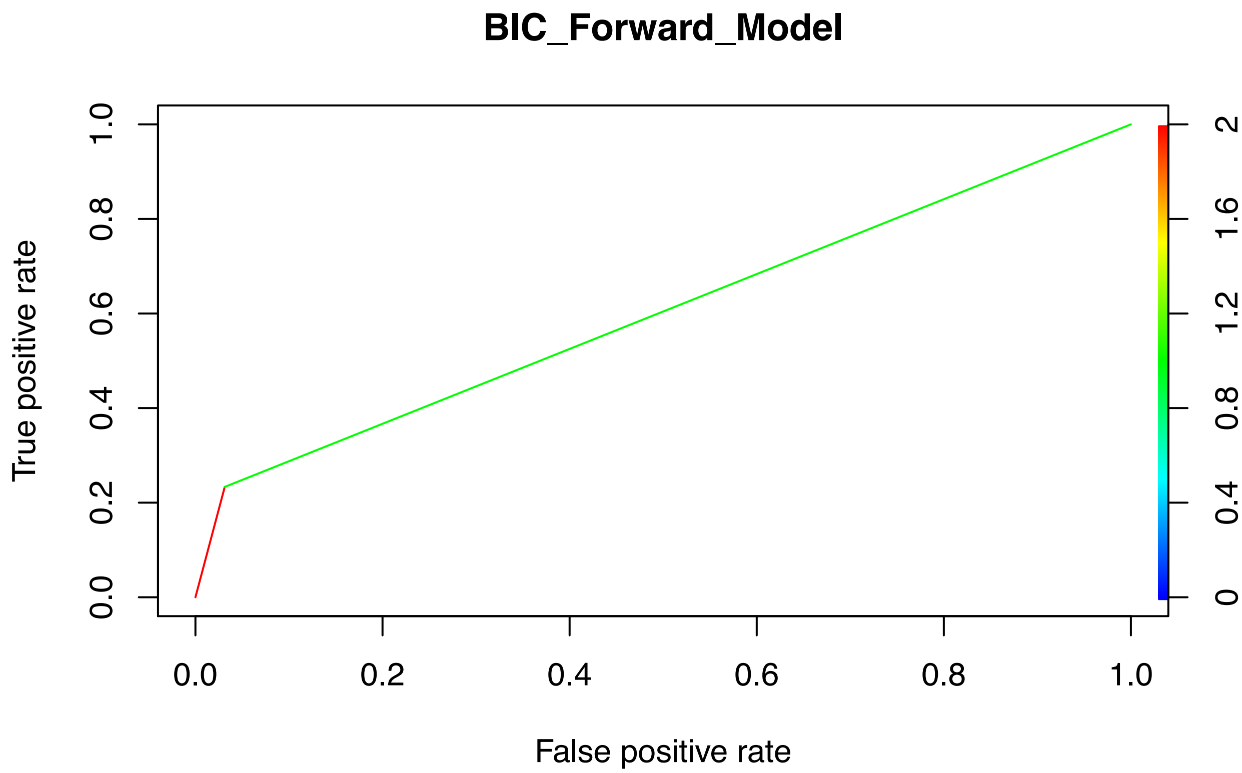

```r
plot(perf_3,colorize=TRUE,main = "BIC_Forward_Model")
```




```r
plot(perf_4,colorize=TRUE,main = "AIC_Backward_Model")
```




```
plot(perf_5,colorize=TRUE,main = "BIC_Forward_Model")
```



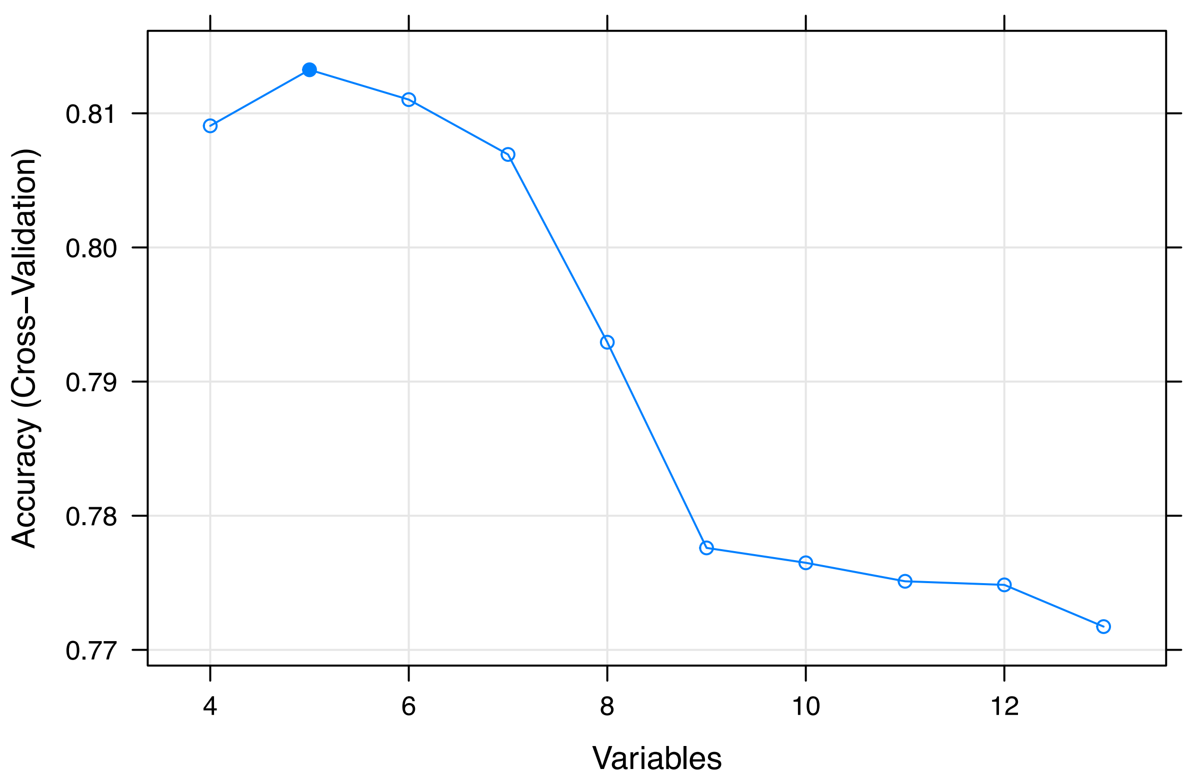
Naive Bayes Classification

```
#construct the Recursive Feature Elimination(RFE) control function: naive bayes + 2-fold cross validati
rfeControls_rf <- rfeControl(
  functions = nbFuncs,
  method = 'cv',
  repeats = 2)

prednumSeq = seq(4,16,1)

# use RFE to select features
system.time(fs_nb <- rfe(x = X_train,
            y = y_train$default,
            sizes = prednumSeq,
            rfeControl = rfeControls_rf))
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 197

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 267

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2076

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2121

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2175

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2184

## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2223

##    user  system elapsed
## 129.978   1.172 132.090
```

```
# Optimal set of predictors
plot(fs_nb, type = c('g','o'))
```



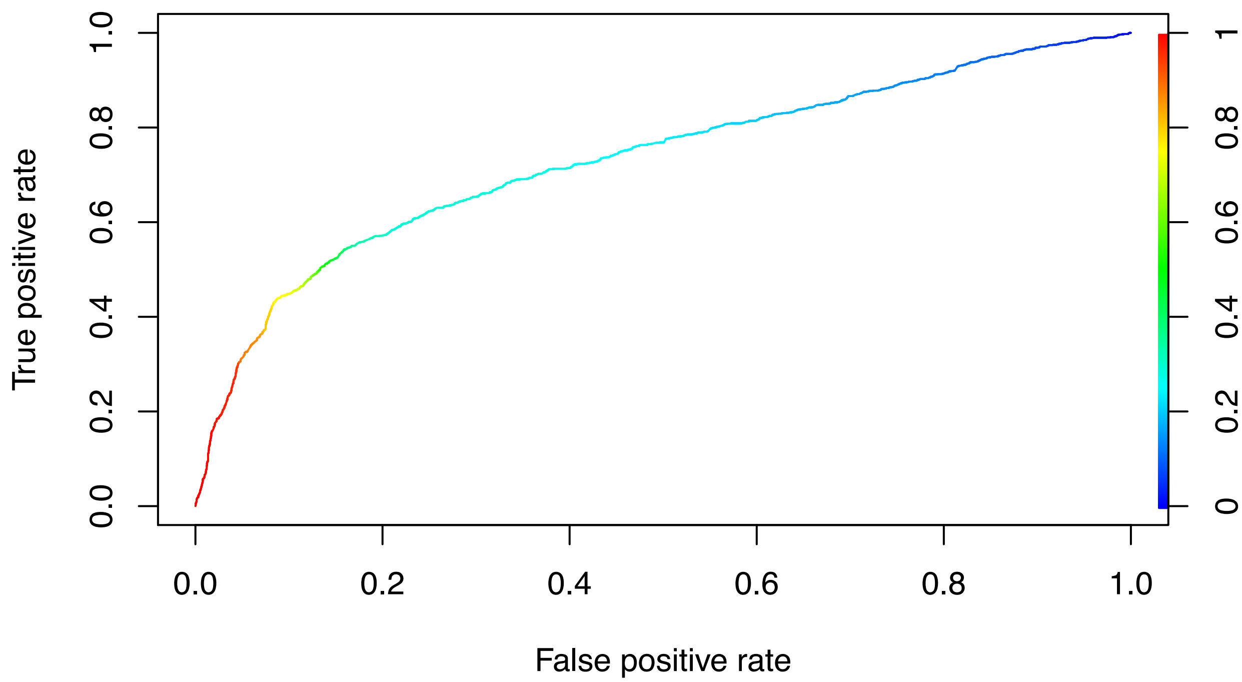

```
fs_nb$optVariables
```

```
## [1] "PAID_0"     "PAY_2"     "LIMIT_BAL" "PAID_5"    "PAY_6"
```

```r
vars <- c('default',fs_nb$optVariables)
model_naive<-naiveBayes( default ~ PAY_0 + PAY_2 + LIMIT_BAL + PAY_6 + PAID_5, train_df, laplace=1)

pred_naive<-predict(model_naive, newdata = test_df)
confusionMatrix(data=pred_naive, reference = test_df$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5025  812
##          1  812  851
##
##                Accuracy : 0.7835
##                  95% CI : (0.774, 0.7927)
##     No Information Rate : 0.7783
##     P-Value [Acc > NIR] : 0.1422
##
##                   Kappa : 0.3726
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8609
##             Specificity : 0.5117
##          Pos Pred Value : 0.8609
##          Neg Pred Value : 0.5117
##              Prevalence : 0.7783
##          Detection Rate : 0.6700
##    Detection Prevalence : 0.7783
##       Balanced Accuracy : 0.6863
##
##        'Positive' Class : 0
##
```

```r
pred_test_naive<-predict(model_naive, newdata = test_df, type="raw")
p_test_naive<-prediction(pred_test_naive[,2], test_df$default)
perf_naive<-performance(p_test_naive, "tpr", "fpr")
plot(perf_naive, colorize=T)
```




```r
performance(p_test_naive, "auc")@y.values
```

```
## [[1]]
## [1] 0.7306841
```