Business Data analytics Group 6- Show/No Show classification

Import data and set objective

```
df.train <- read.csv("NS.TRAIN.csv")
df.test <- read.csv("NS.TEST.csv")
```

```
dim(df.train)
```

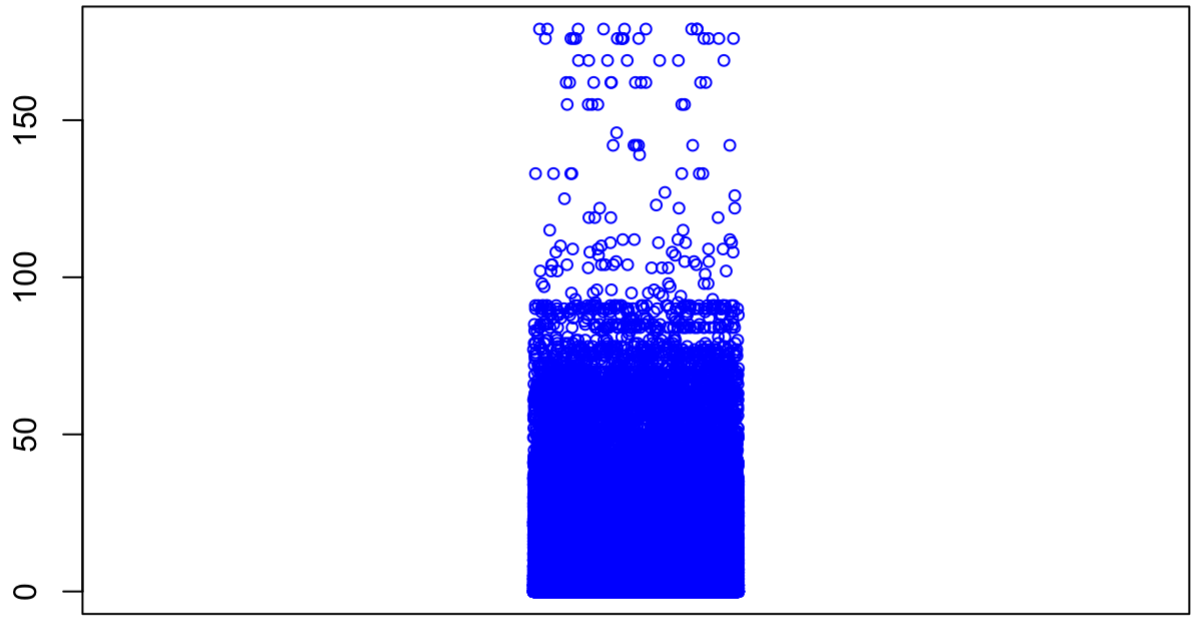
```
## [1] 88416    35
```

The train DF holds 88416x35 rows and attributes(columns)

High importance variables

Review waiting time in the test data

```
stripchart(df.train$waiting_time, method = "jitter",  vertical = TRUE, col ="blue",cex = 0.75, pch= 1)
```

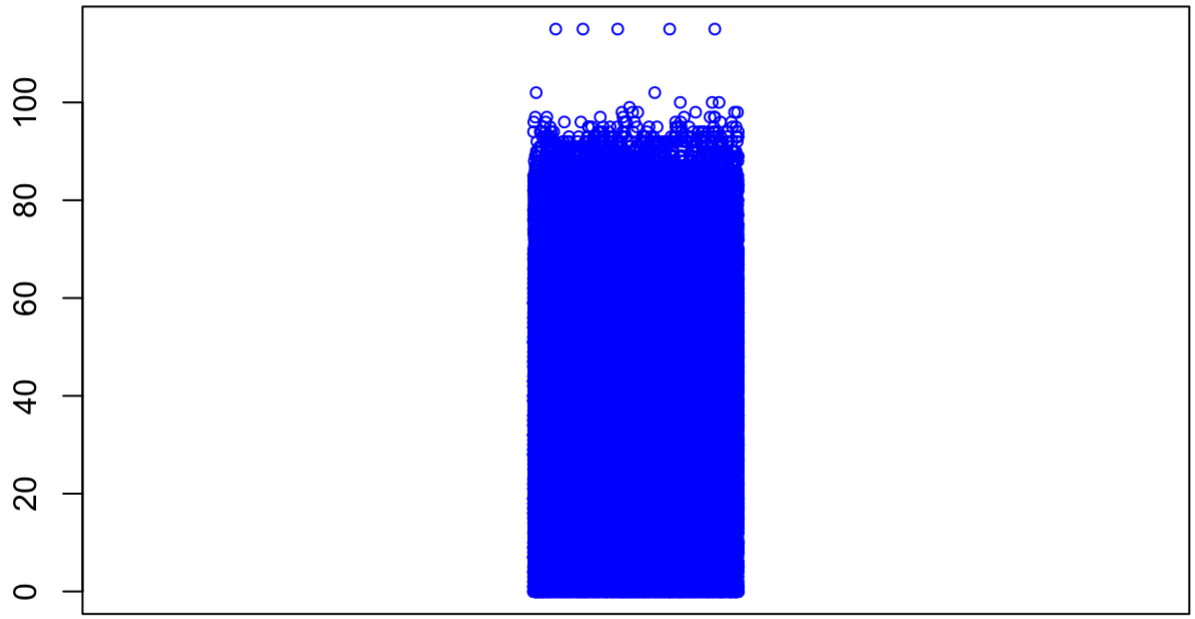


We can see that we have some outliers in the waiting time feature. Hence, we'll compress the over 90 days waiting tome to group of 90 days waiting time.

Review ages in the test data

Outliers for Age feature. we can see that we have several ages that are beyond 85 y.o and the number of occurrence is not high, hence, we'll compress them to the age of 85

```
stripchart(df.train$age, method = "jitter",  vertical = TRUE, col ="blue",cex = 0.75, pch= 1)
```

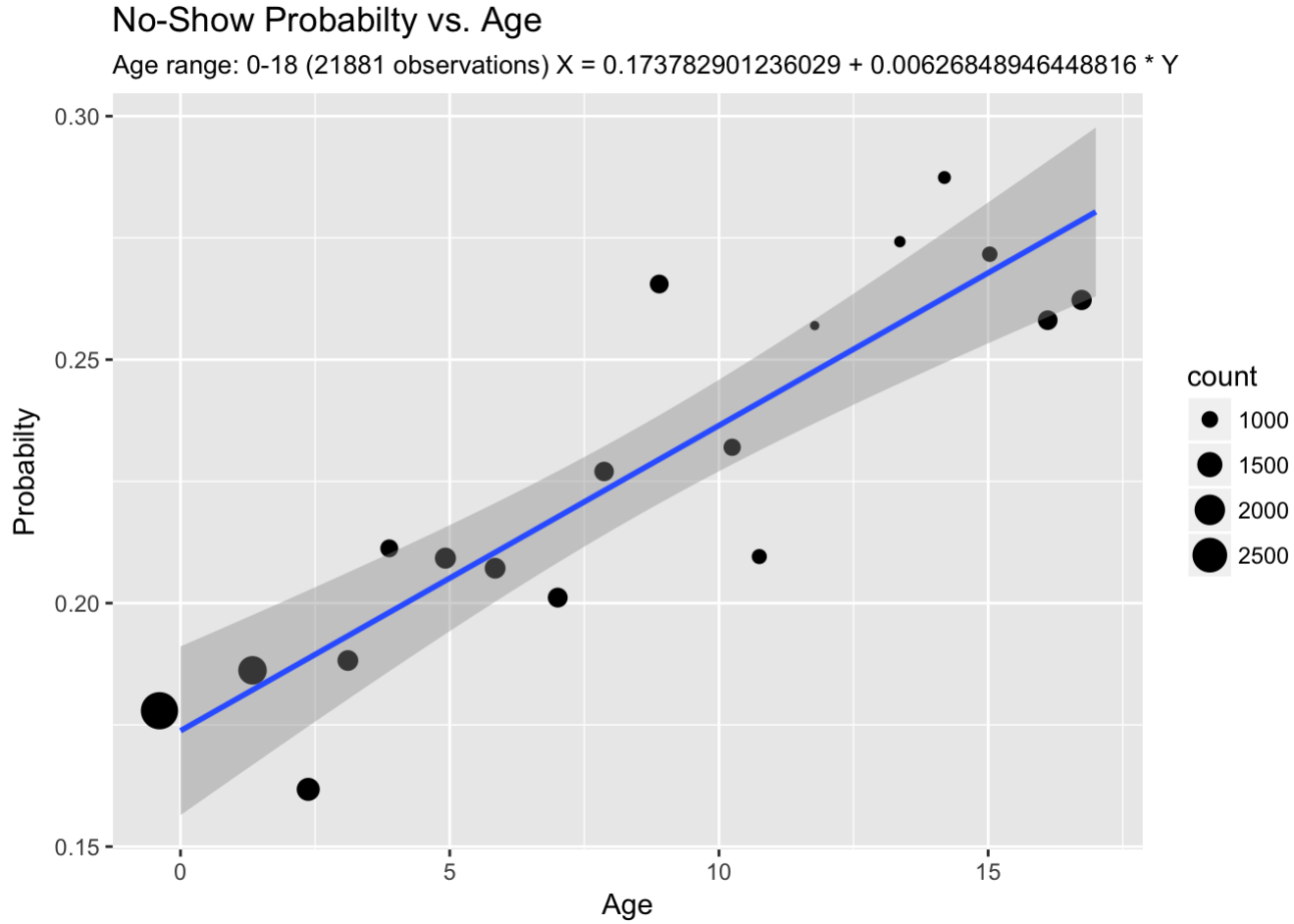



```
df.train$age[df.train$age>85] <- 85
```

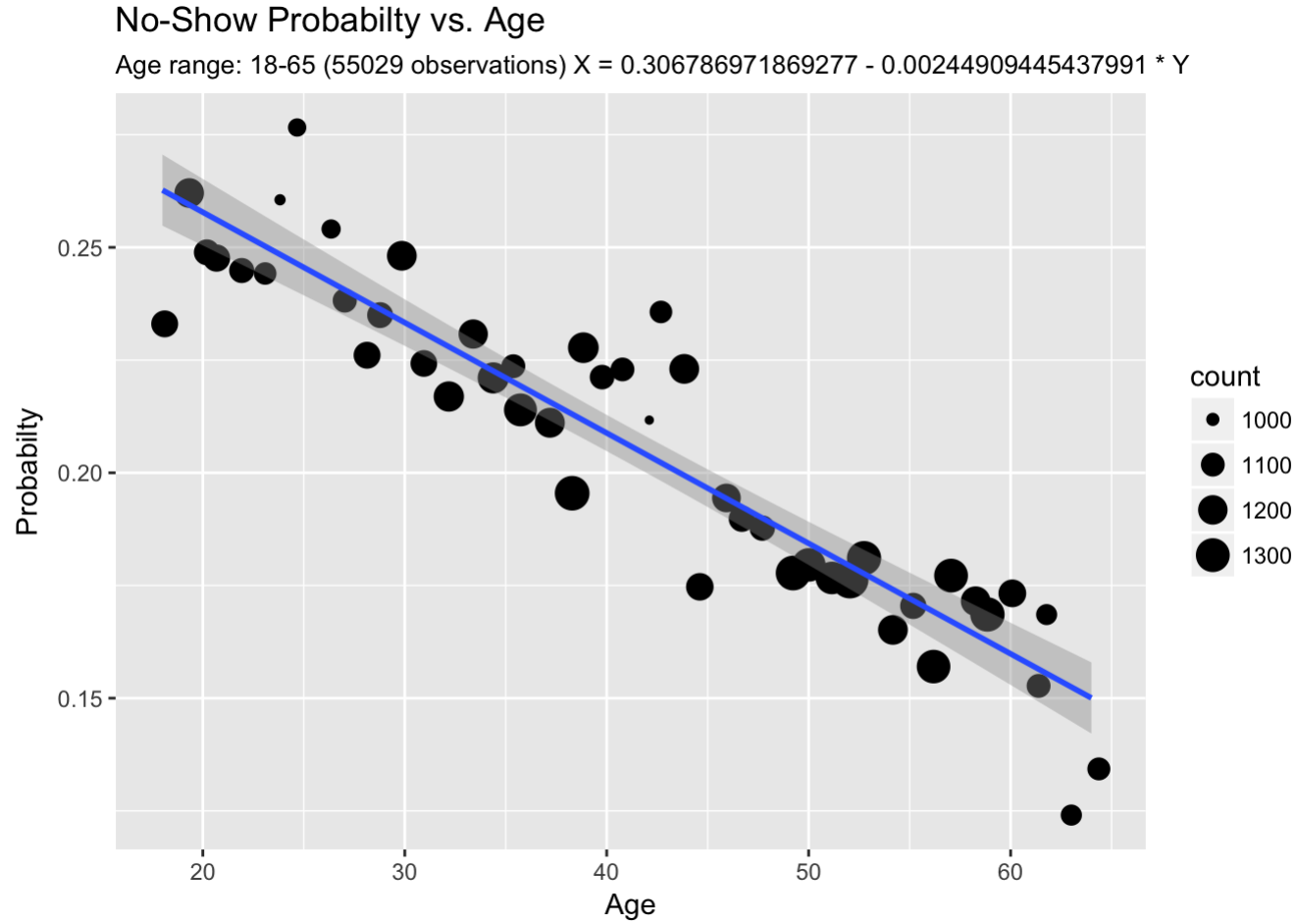
removing the outliers for age

Let's examine the Age variable and see if we have a linear correlation to NS probability

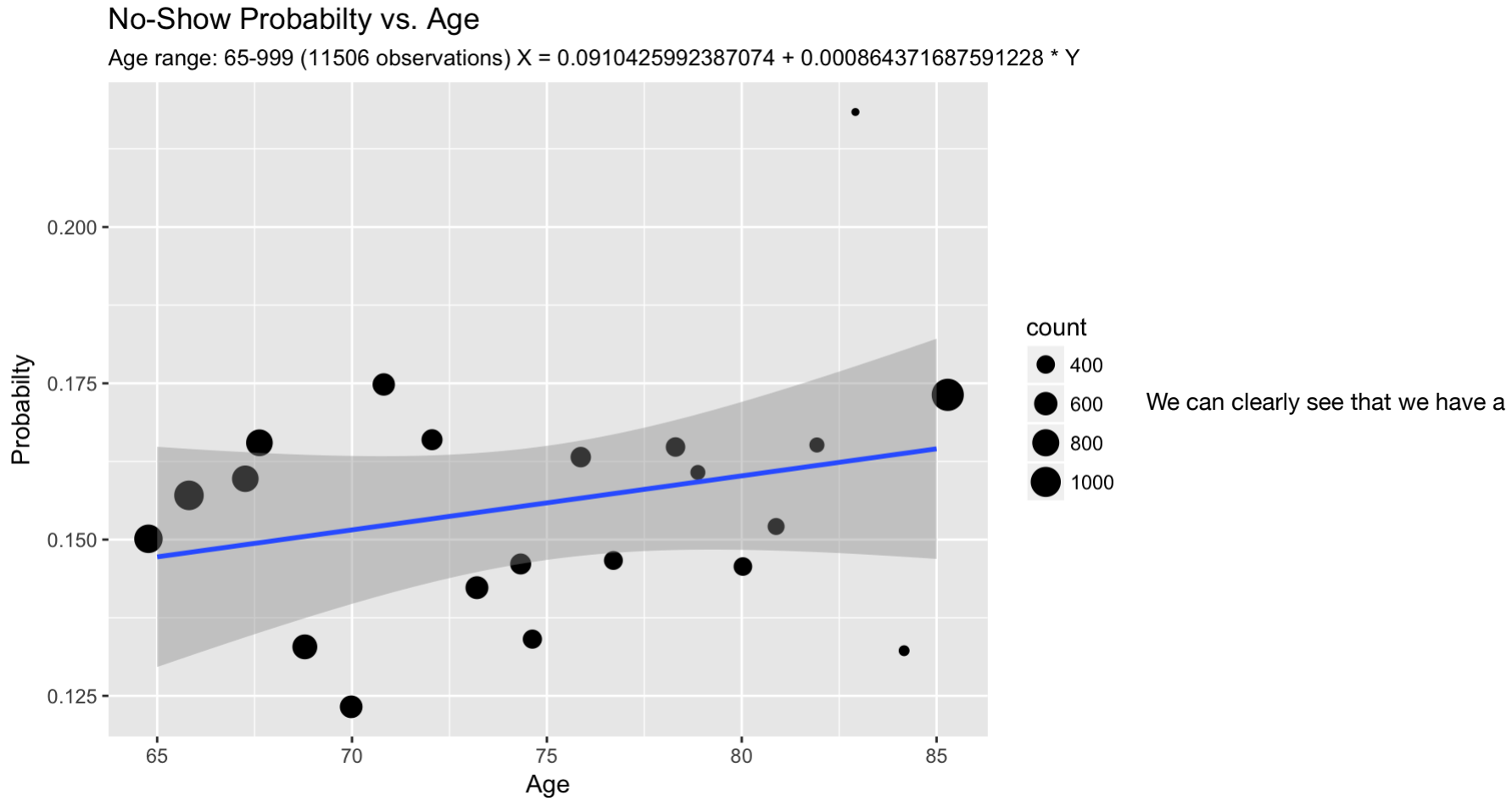
```
#Draw
draw_age_gg(0,18)
```




```
draw_age_gg(18,65)
```




```
draw_age_gg(65,999)
```



We can clearly see that we have a linear correlation.

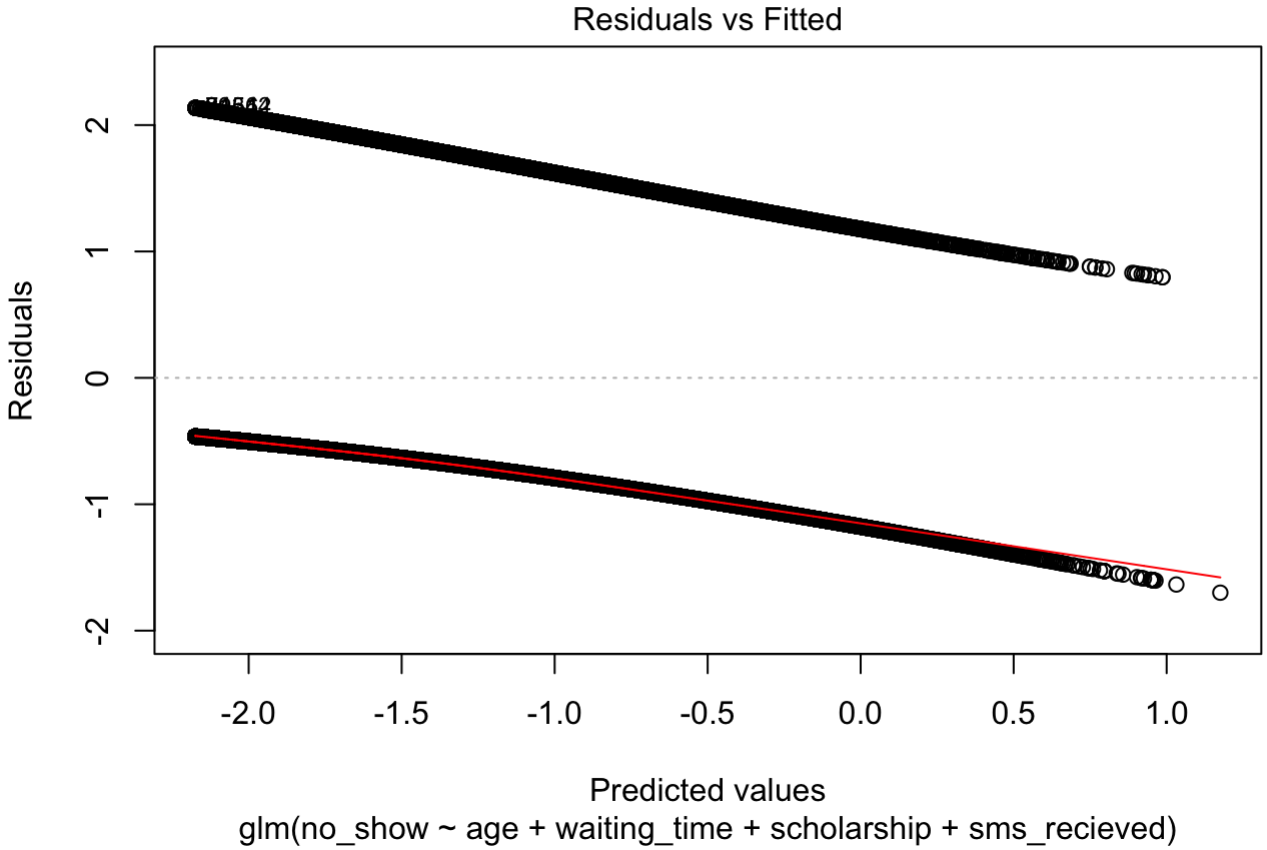
Trainning models on train dataset

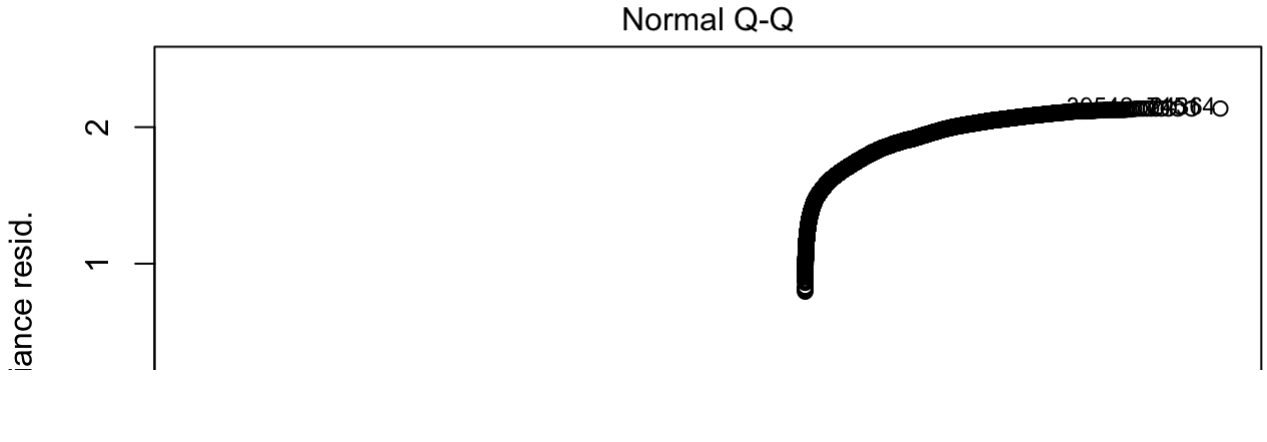
Logistic Model

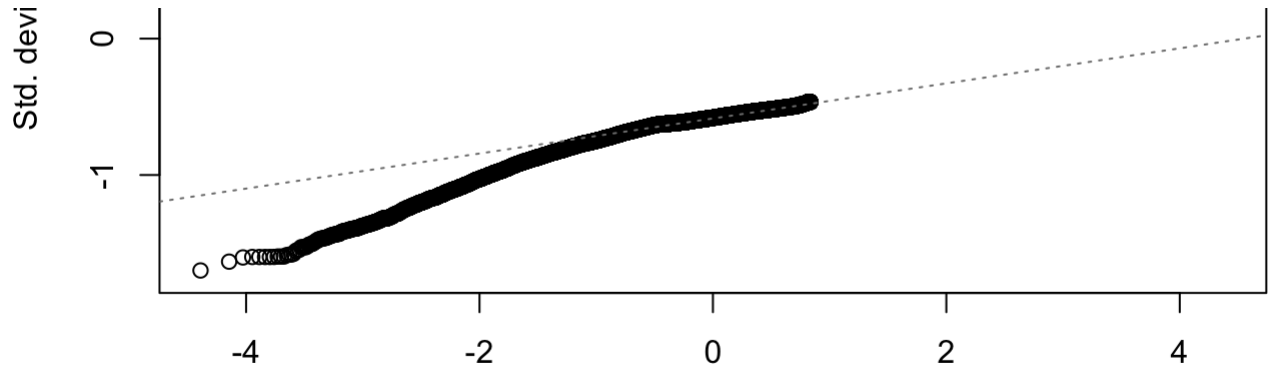
```
noshow.LM <- glm(no_show ~ age+
                         waiting_time+
                         scholarship+
                         sms_recieved, data = df.train, family = binomial)
summary (noshow.LM)
```

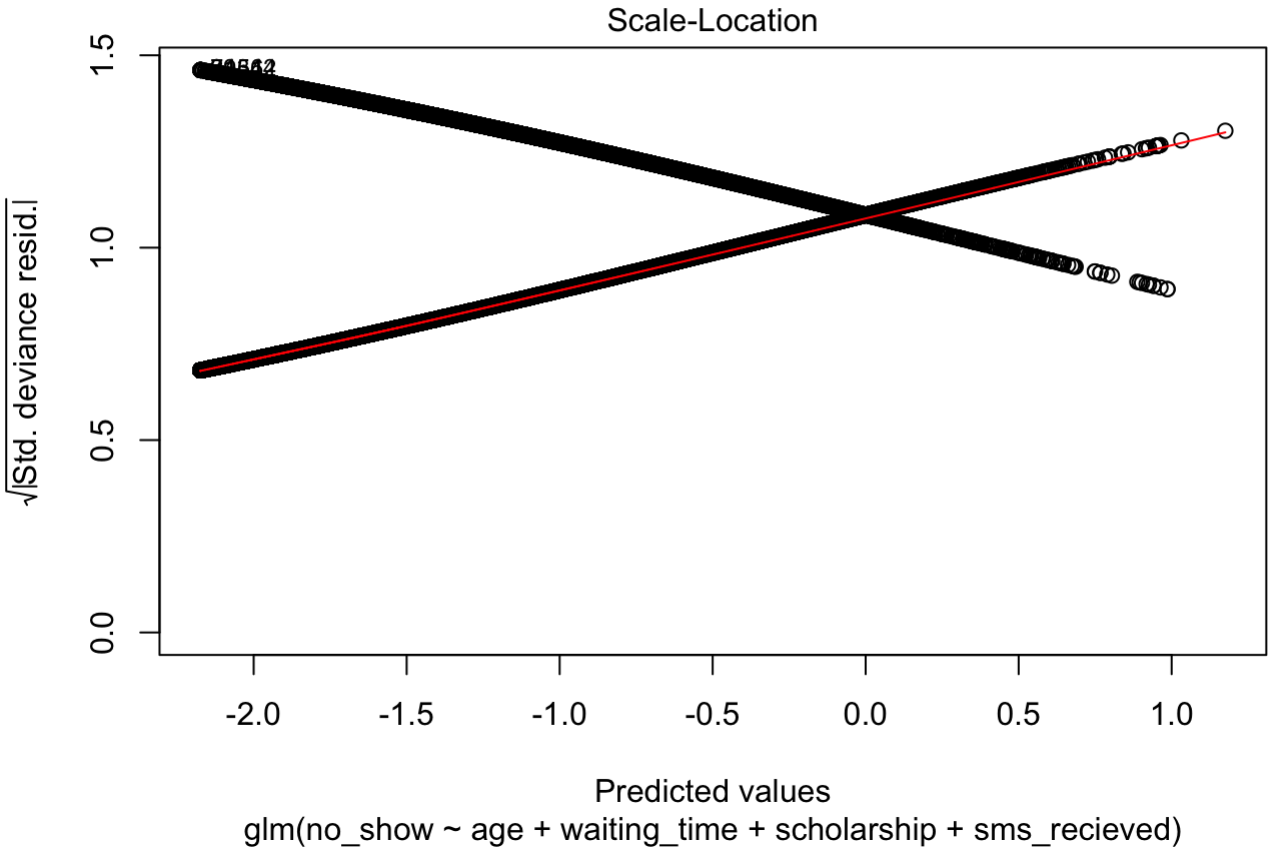
```
##
## Call:
## glm(formula = no_show ~ age + waiting_time + scholarship + sms_recieved,
##     family = binomial, data = df.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6997  -0.6720  -0.5800  -0.4986   2.1361
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.5307220  0.0179009 -85.511   <2e-16 ***
## age          -0.0075646  0.0003794 -19.937   <2e-16 ***
## waiting_time  0.0238551  0.0005481  43.521   <2e-16 ***
## scholarship   0.2270561  0.0273954   8.288   <2e-16 ***
## sms_recieved  0.3472715  0.0188400  18.433   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 89074  on 88415  degrees of freedom
## Residual deviance: 85459  on 88411  degrees of freedom
## AIC: 85469
##
## Number of Fisher Scoring iterations: 4
```

```
plot(noshow.LM)
```


Residuals vs Fitted
Residuals
2
1
0
-1
-2
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
Predicted values
glm(no_show ~ age + waiting_time + scholarship + sms_recieved)


Normal Q-Q
iance resid.
2
1

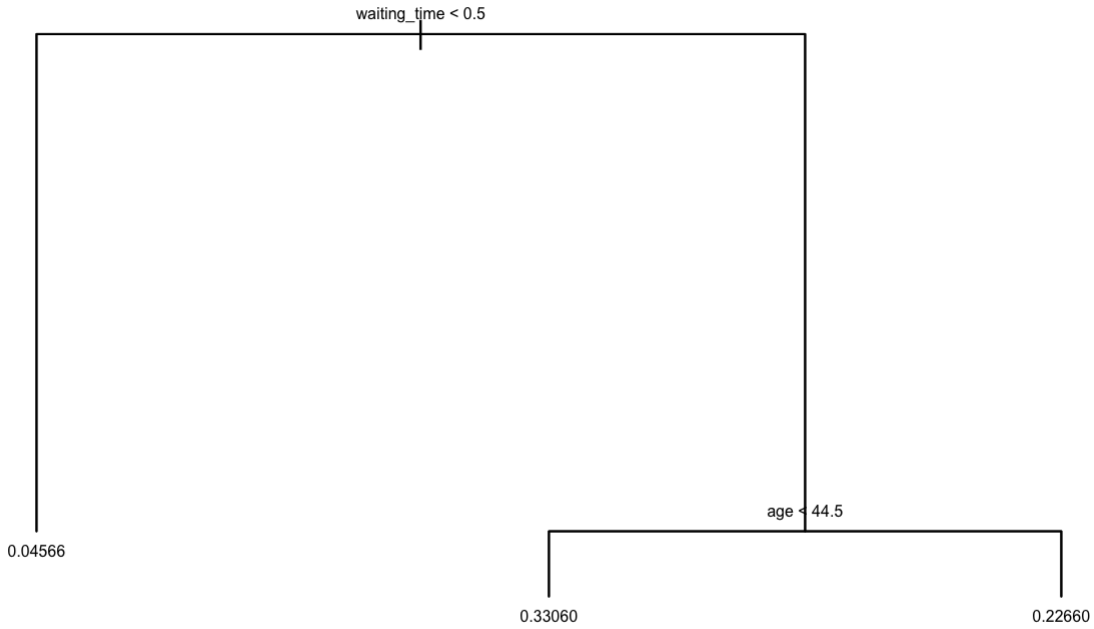

Std. devi
0
-1
-4
-2
0
2
4
Theoretical Quantiles
glm(no_show ~ age + waiting_time + scholarship + sms_recieved)


Scale-Location
√|Std. deviance resid.|
1.5
1.0
0.5
0.0
-2.0
-1.5
-1.0
-0.5
0.0
0.5
1.0
Predicted values
glm(no_show ~ age + waiting_time + scholarship + sms_recieved)


Residuals vs Leverage
Std. Pearson resid.
3
2
1
0
-1
-2
0e+00
2e-04
4e-04
6e-04
Leverage
glm(no_show ~ age + waiting_time + scholarship + sms_recieved)
Cook's distance
66911

CART

```
pacman::p_load("tree")
noshow.CART <- tree(no_show ~ week_day+
                      waiting_time+
                      age+
                      is_female+
                      scholarship+
                      hipertension+
                      diabetes+
                      alcoholism+
                      handcap+
                      sms_recieved+
                      poverty+
                      region ,data = df.train)
plot(noshow.CART)
text(noshow.CART, pretty = 0, cex=0.5)
```




```
summary(noshow.CART)
```

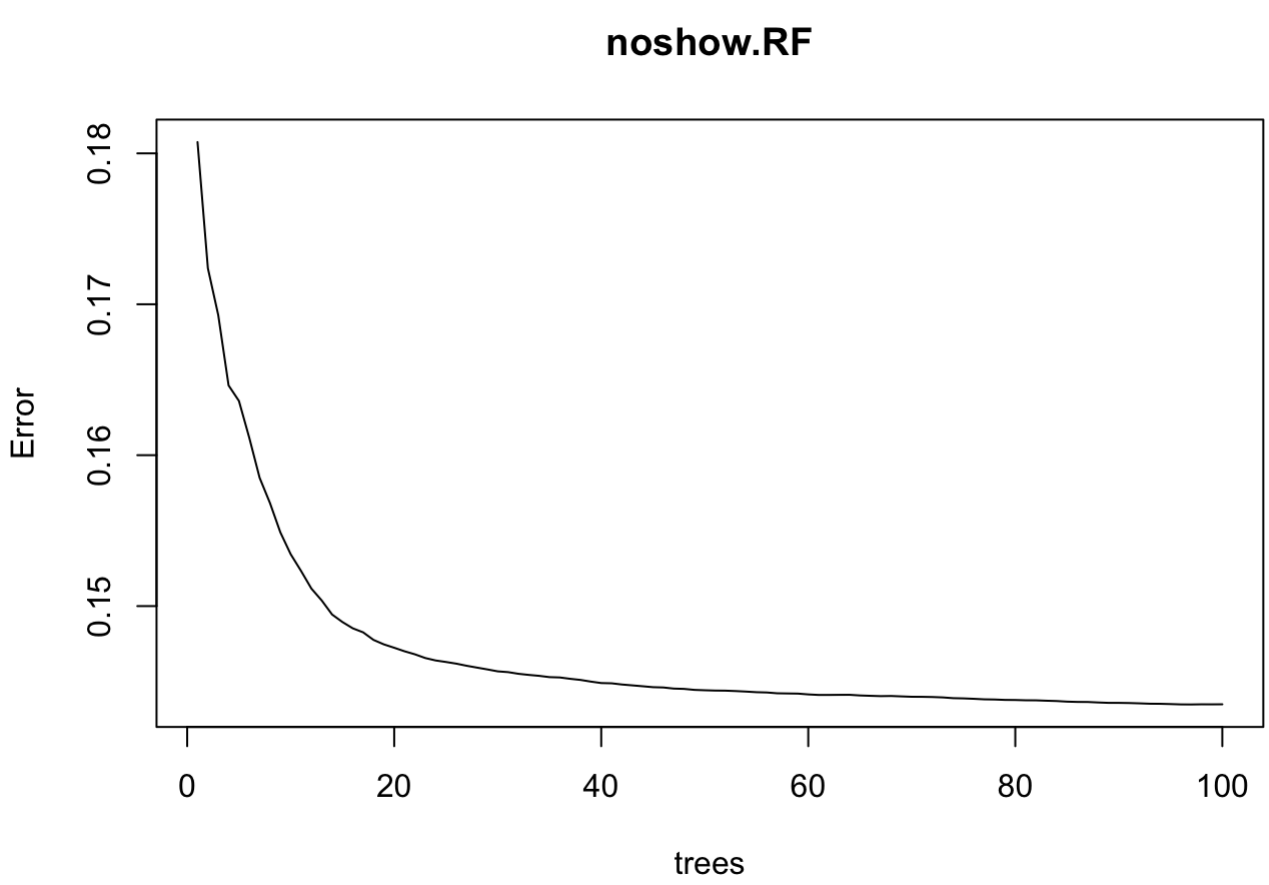
```
##
## Regression tree:
## tree(formula = no_show ~ week_day + waiting_time + age + is_female +
##     scholarship + hipertension + diabetes + alcoholism + handcap +
##     sms_recieved + poverty + region, data = df.train)
## Variables actually used in tree construction:
## [1] "waiting_time" "age"
## Number of terminal nodes:  3
## Residual mean deviance:  0.1466 = 12960 / 88410
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.33060 -0.33060 -0.04566  0.00000 -0.04566  0.95430
```

Random Forest

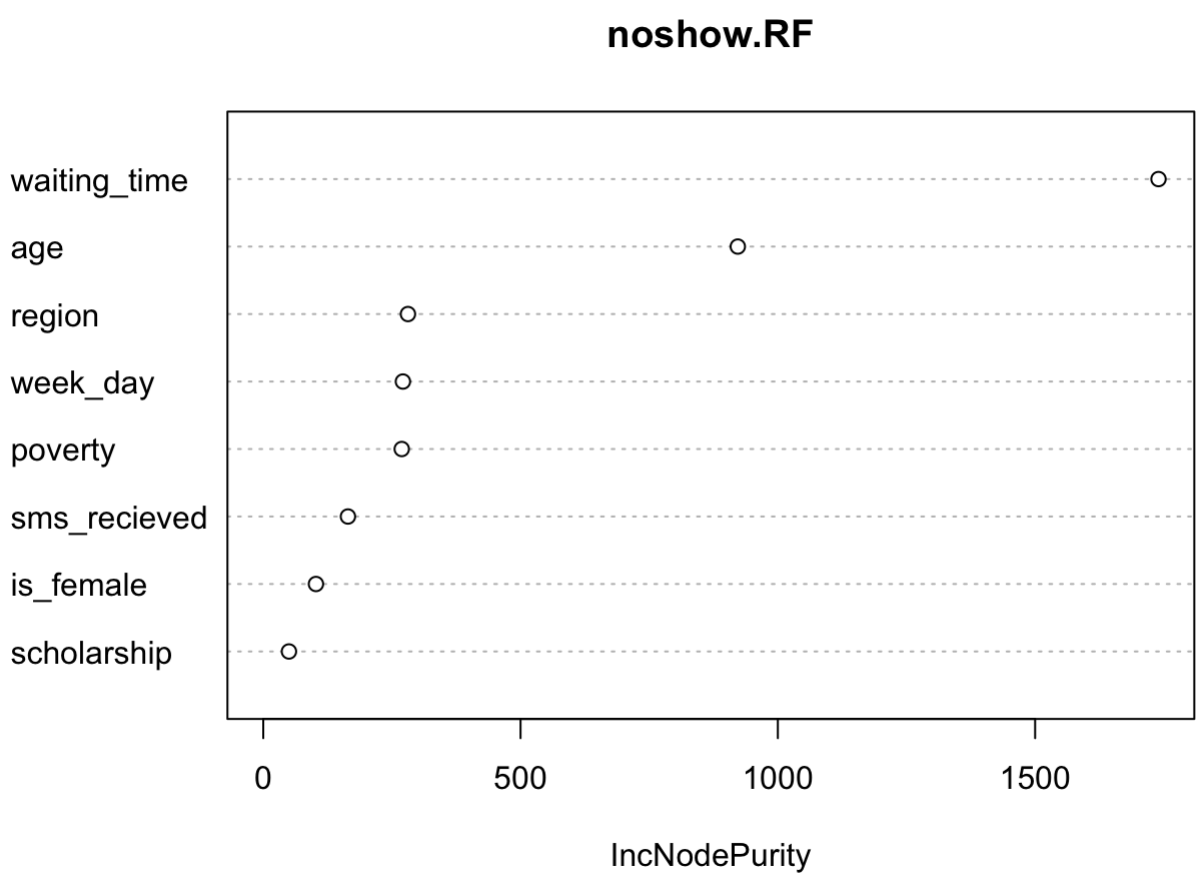
```
pacman::p_load("randomForest")
set.seed(7)
noshow.RF <- randomForest(no_show ~ week_day
                          +waiting_time
                          +age
                          +is_female
                          +scholarship
                          +sms_recieved
                          +poverty
                          +region
                          , data = df.train, na.action=na.omit, type="classification", ntree=100)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
plot(noshow.RF)
```




```
#importance(noshow.RF)
varImpPlot(noshow.RF)
```



Gradient Boosting Machine

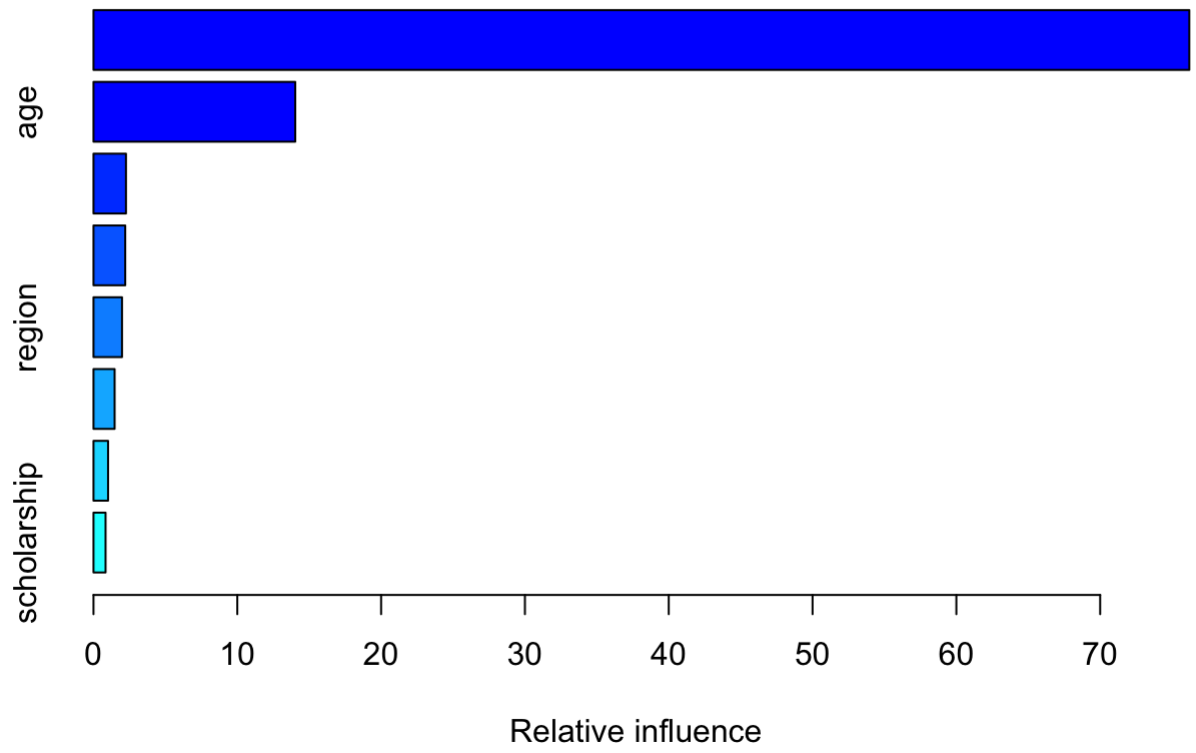
```
# install.packages("gbm",repos = "http://cran.us.r-project.org")
#library("gbm")
pacman::p_load("gbm")
set.seed(7) #same seed to repeat the RF
no_show.GBM <- gbm (no_show ~ week_day+
                         waiting_time+
                         age+
                         is_female+
                         scholarship+
                         sms_recieved+
                         poverty+
                         region ,data = df.train, n.trees = 100, interaction.depth = 4, shrinkage = 0.2, verbose
 = F)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
no_show.GBM
```

```
## gbm(formula = no_show ~ week_day + waiting_time + age + is_female +
##     scholarship + sms_recieved + poverty + region, data = df.train,
##     n.trees = 100, interaction.depth = 4, shrinkage = 0.2, verbose = F)
## A gradient boosted model with bernoulli loss function.
## 100 iterations were performed.
## There were 8 predictors of which 8 had non-zero influence.
```

```
summary(no_show.GBM)
```




```
##                       var    rel.inf
## waiting_time waiting_time 76.1985652
## age                  age 14.0289750
## sms_recieved sms_recieved  2.2593918
## poverty           poverty  2.2071554
## region             region  1.9833846
## week_day         week_day  1.4691339
## is_female       is_female  1.0137334
## scholarship   scholarship  0.8396609
```

Model evaluation

Logistic Model

```
threshold = 0.3
fitted.lm.results <- predict(noshow.LM,df.test,type='response')
lm.prediction <- ifelse(fitted.lm.results > threshold,1,0)
lm.accuracy <- mean(lm.prediction == df.test$no_show)
lm.accuracy
```

```
## [1] 0.7597376
```

```
pacman::p_load("caret")
pacman::p_load("e1071")
confusionMatrix(data = lm.prediction, reference =  df.test$no_show)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15904  3528
##          1  1783   890
##
##                Accuracy : 0.7597
##                  95% CI : (0.754, 0.7654)
##     No Information Rate : 0.8001
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1181
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8992
##             Specificity : 0.2014
##          Pos Pred Value : 0.8184
##          Neg Pred Value : 0.3330
##              Prevalence : 0.8001
##          Detection Rate : 0.7195
##    Detection Prevalence : 0.8791
##       Balanced Accuracy : 0.5503
##
##        'Positive' Class : 0
##
```

```
cross.table <- table(lm.prediction, df.test$no_show)
l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }
accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.759737615923999"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.332959221848111"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.201448619284744"
```

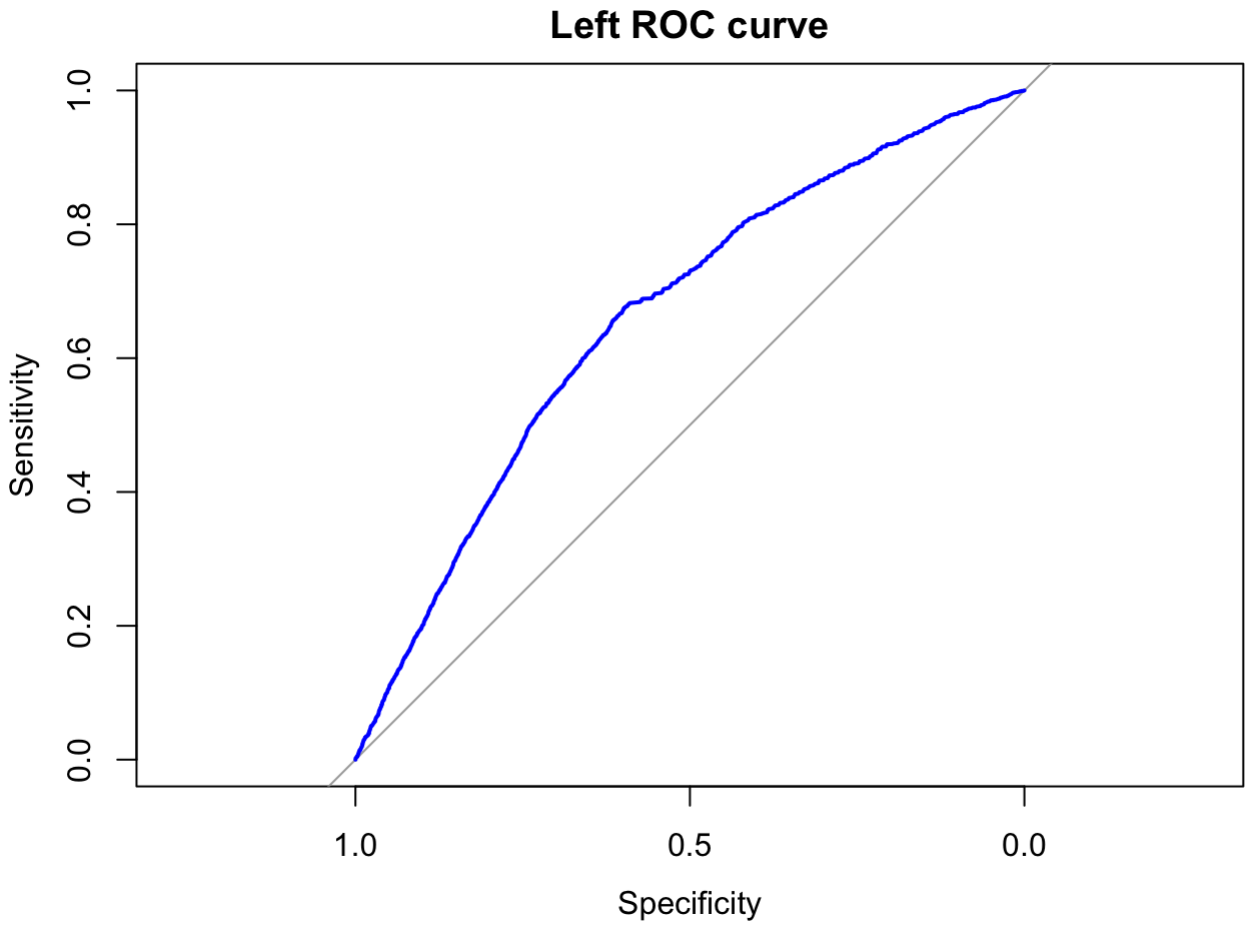
```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.251022422789451"
```

```
model <- "LM"

lm_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

```
#install.packages("pROC",repos = "http://cran.us.r-project.org")
pacman::p_load("pROC")
plot(roc(df.test$no_show, fitted.lm.results, direction="<"), col="blue", main="Left ROC curve")
```



CART

```
threshold = 0.3
fitted.cart.results <- predict(noshow.CART,df.test)
summary(fitted.cart.results)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.04566 0.04566 0.22663 0.20236 0.33062 0.33062
```

```
cart.prediction <- ifelse(fitted.cart.results > threshold,1,0)
summary(cart.prediction)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.3766  1.0000  1.0000
```

```
cross.table <- table(cart.prediction, df.test$no_show)
cross.table
```

```
##
## cart.prediction     0     1
##               0 12016  1765
##               1  5671  2653
```

```
l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }

accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.663605519113323"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.318716962998558"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.600497962879131"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.416418144718255"
```

```
model <- "CART"

cart_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

RF

```
threshold <- 0.4
fitted.rf.results <- predict(noshow.RF,df.test)
summary(fitted.rf.results)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02339 0.06164 0.21297 0.20387 0.30357 0.66058
```

```
rf.prediction <- ifelse(fitted.rf.results > threshold,1,0)
summary(rf.prediction)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.06125 0.00000 1.00000
```

```
cross.table <- table(rf.prediction, df.test$no_show)

l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }

accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.792716579959285"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.43943870014771"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.134676324128565"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.206167706167706"
```

```
model <- "RF"

rf_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

GBM

```
threshold <- 0.6
fitted.gbm.results <- predict(no_show.GBM,df.test, n.trees = 100)
summary(fitted.gbm.results)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.5307 -2.9895 -1.2782 -1.6894 -0.8313  0.6897
```

```
gbm.prediction <- ifelse(fitted.gbm.results > threshold,1,0)
summary(gbm.prediction)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.00e+00 0.00e+00 0.00e+00 4.52e-05 0.00e+00 1.00e+00
```

```
cross.table <- table(gbm.prediction, df.test$no_show)

l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }
accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.800180954535173"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 1"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.000226346763241286"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.000452591083955646"
```

```
model <- "GBM"

gbm_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

```
evaluate <- rbind(lm_results, cart_results, rf_results, gbm_results)
colnames(evaluate) <- c("Model","Accuracy","Precision","Recall","F1")
kable(evaluate, caption = "Resuls Evaluation")
```

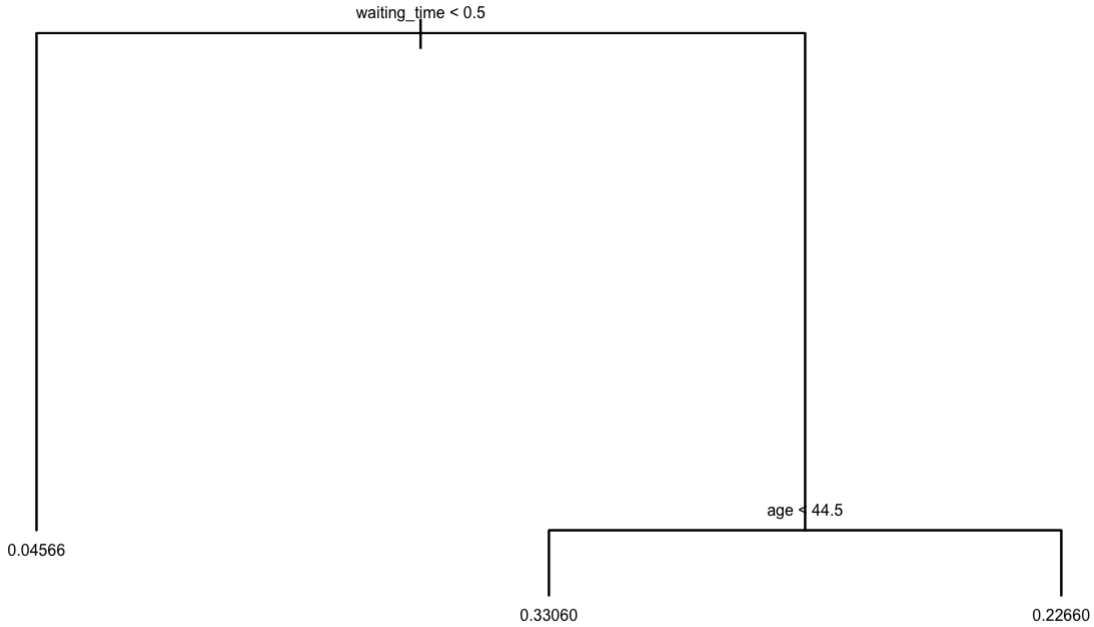
Resuls Evaluation

	Model	Accuracy	Precision	Recall	F1
lm_results	LM	0.7597	0.333	0.2014	0.251
cart_results	CART	0.6636	0.3187	0.6005	0.4164
rf_results	RF	0.7927	0.4394	0.1347	0.2062
gbm_results	GBM	0.8002	1	2e-04	5e-04

Conclusion

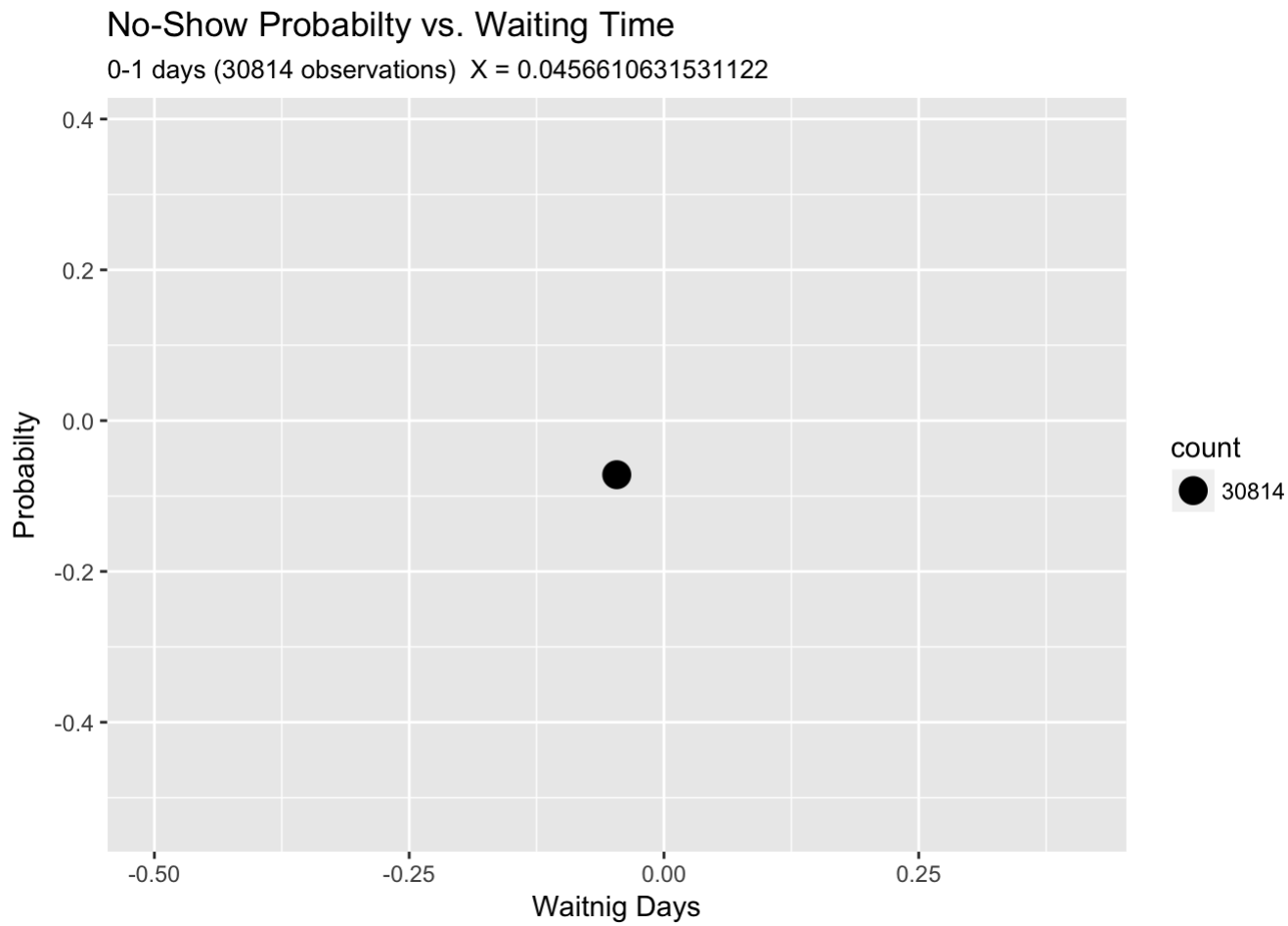
Not good enough.. Recall and Precision are low although accuracy is high. It looks like something is biasing the data. Lets take onther look at the tree generated by CART alg

```
plot(noshow.CART)
text(noshow.CART, pretty = 0, cex=0.5)
```

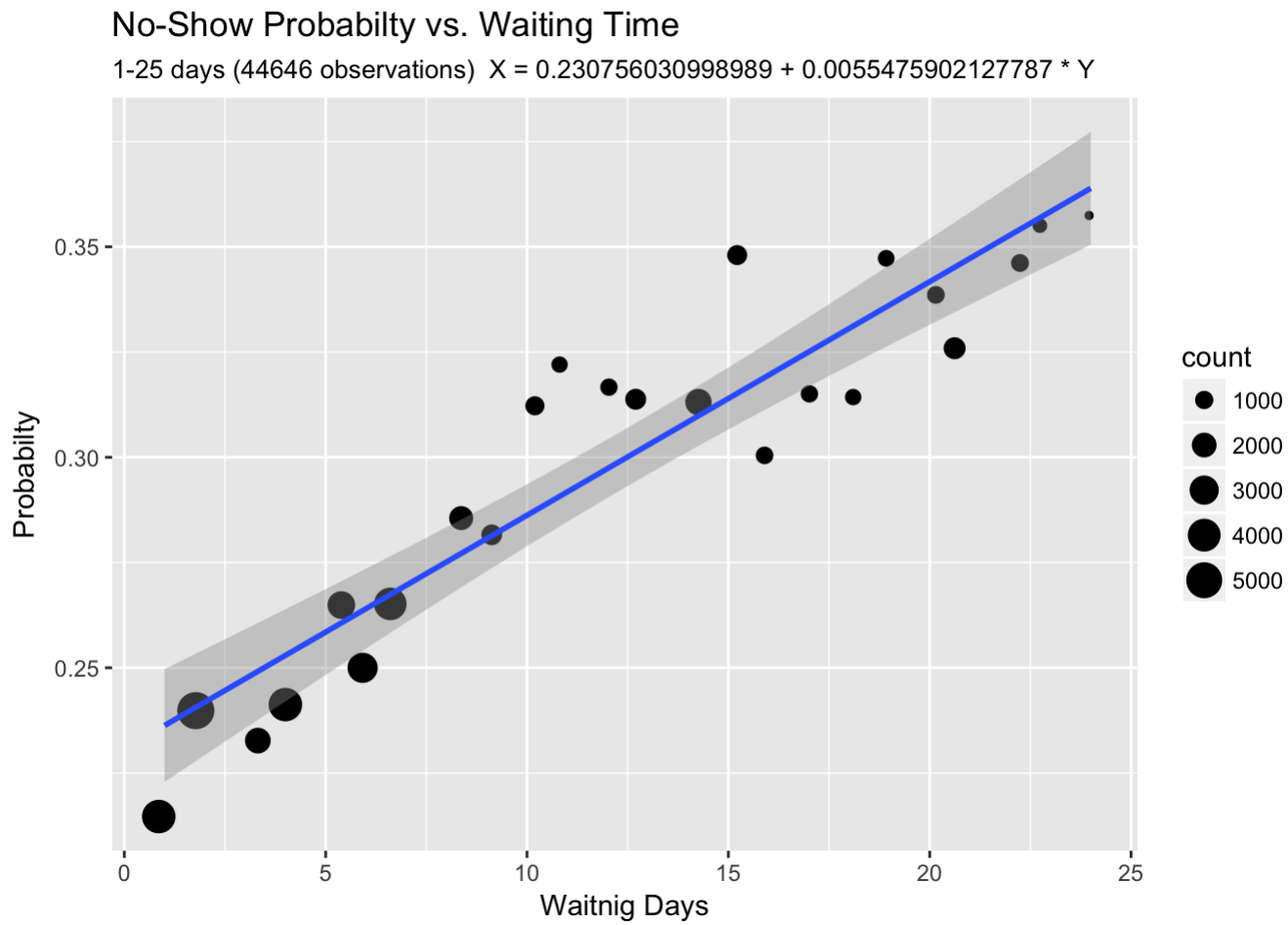


Let’s Examine the waiting time varaible as well

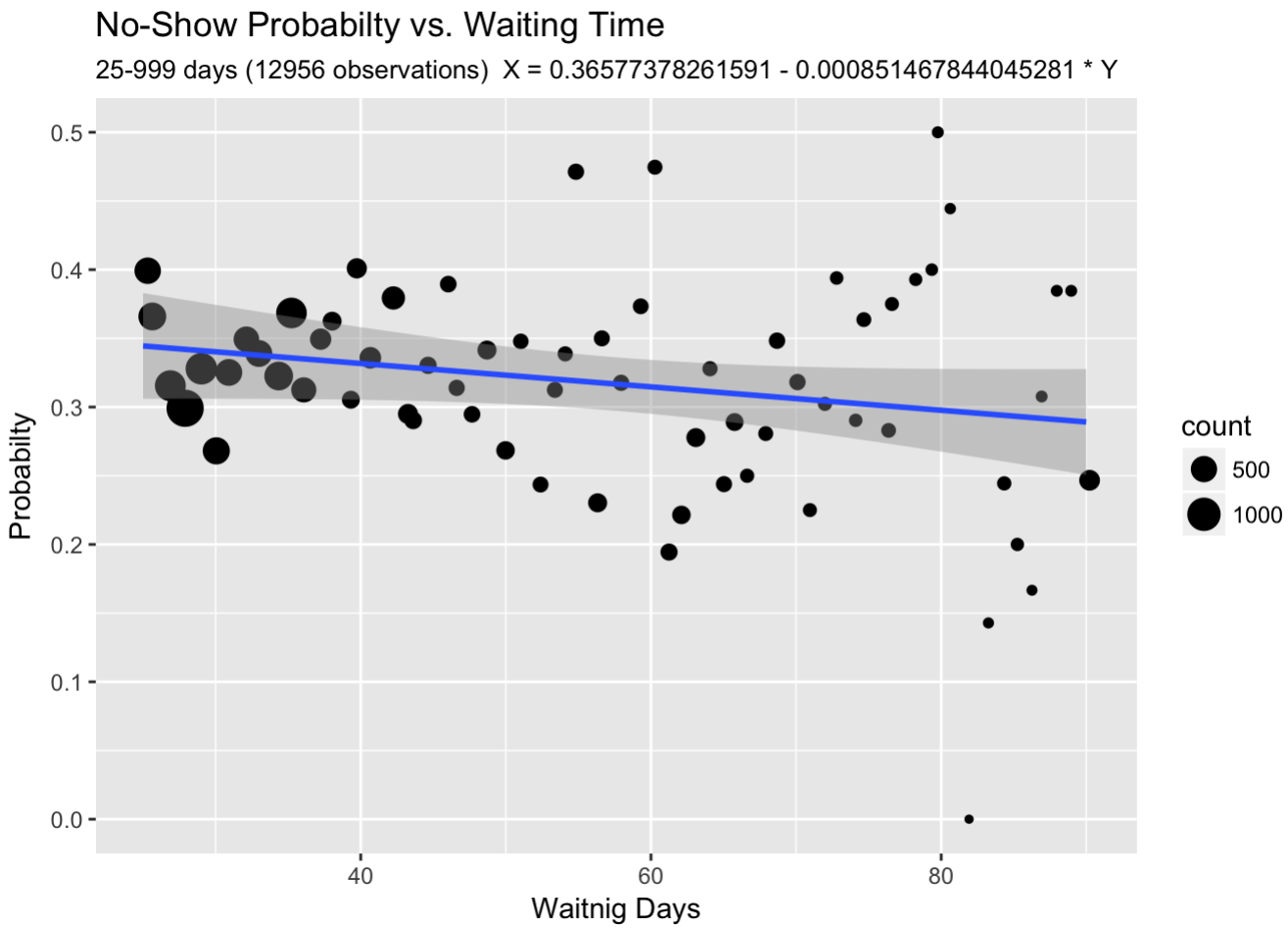
```
#Draw
draw_waiting_days_gg(0,1)
```




```
draw_waiting_days_gg(1,25)
```




```
draw_waiting_days_gg(25,999)
```



Note that the CART tree is not splitting as we would like since we only have 2 major variables and the other variables are screened by those two. Let's "cut" the tree and remove the waiting_time < 0.5.

** *Current model's assumption **- if the patient's waiting_time is eq or less than 0.5 days, he has high probability to** show** to its appointment.

Split the data set to include only waiting time > 0.5

```
df.train.0.5 <- df.train[df.train$waiting_time>0.5,]
df.test.0.5 <- df.test[df.test$waiting_time >0.5,]
dim(df.train.0.5)
```

```
## [1] 57602    35
```

```
dim(df.test.0.5)
```

```
## [1] 14357    35
```

Train Again on the above 0.5 train dataset

```
noshow.LM.2 <- glm(no_show ~ age+
                           waiting_time+
                           scholarship+
                           sms_recieved, data = df.train.0.5, family = binomial)

noshow.CART.2 <- tree(no_show ~ week_day+
                        waiting_time+
                        age+
                        is_female+
                        scholarship+
                        hipertension+
                        diabetes+
                        alcoholism+
                        handcap+
                        sms_recieved+
                        poverty+
                        region ,data = df.train.0.5)

set.seed(7)
noshow.RF.2 <- randomForest(no_show ~ week_day
                            +waiting_time
                            +age
                            +is_female
                            +scholarship
                            +sms_recieved
                            +poverty
                            +region
                            , data = df.train.0.5, na.action=na.omit, type="classification", ntree=100)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
no_show.GBM.2 <- gbm (no_show ~ week_day+
                          waiting_time+
                          age+
                          is_female+
                          scholarship+
                          sms_recieved+
                          poverty+
                          region ,data = df.train.0.5, n.trees = 100, interaction.depth = 4, shrinkage = 0.2, verb
ose = F)
```

```
## Distribution not specified, assuming bernoulli ...
```

Evaluate Again on test

LM

```
threshold = 0.5
fitted.lm.results <- predict(noshow.LM.2,df.test.0.5,type='response')
lm.prediction <- ifelse(fitted.lm.results > threshold,1,0)


cross.table <- table(lm.prediction, df.test.0.5$no_show)
l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }
accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.718743470084279"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.352941176470588"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.00148772625836846"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.00296296296296296"
```

```
model <- "LM"

lm_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

CART

```
threshold = 0.3
fitted.cart.results <- predict(noshow.CART.2,df.test.0.5)
cart.prediction <- ifelse(fitted.cart.results > threshold,1,0)
cross.table <- table(cart.prediction, df.test.0.5$no_show)
l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }

accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.508880685379954"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.318716962998558"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.657822960575254"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.429392247309217"
```

```
model <- "CART"

cart_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

RF

```
threshold <- 0.4
fitted.rf.results <- predict(noshow.RF.2,df.test.0.5)
rf.prediction <- ifelse(fitted.rf.results > threshold,1,0)
cross.table <- table(rf.prediction, df.test.0.5$no_show)

l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }

accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.699240788465557"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.424161788185205"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.19761963798661"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.269621109607578"
```

```
model <- "RF"

rf_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

GBM

```
threshold <- 0.4
fitted.gbm.results <- predict(no_show.GBM.2,df.test.0.5, n.trees = 100)
gbm.prediction <- ifelse(fitted.gbm.results > threshold,1,0)
cross.table <- table(gbm.prediction, df.test.0.5$no_show)
l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }
accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.719231037124748"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.75"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.00074386312918423"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.00148625216745108"
```

```
model <- "GBM"

gbm_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

```
evaluate_2 <- rbind(lm_results, cart_results, rf_results, gbm_results)
colnames(evaluate_2) <- c("Model","Accuracy","Precision","Recall","F1")

kable(evaluate_2, caption = "Resuls Evaluation wiht wating time > 0.5")
```

Resuls Evaluation wiht wating time > 0.5

	Model	Accuracy	Precision	Recall	F1
lm_results	LM	0.7187	0.3529	0.0015	0.003
cart_results	CART	0.5089	0.3187	0.6578	0.4294
rf_results	RF	0.6992	0.4242	0.1976	0.2696
gbm_results	GBM	0.7192	0.75	7e-04	0.0015

2nd conclusion

Model Recalls are still low. We're still assuming the data is biased. Let's examine our train dataset

```
dim(df.train[df.train$no_show==0,])
```

```
## [1] 70520    35
```

```
dim(df.train[df.train$no_show==1,])
```

```
## [1] 17896    35
```

We're holding 70K shows and 17K no shows. this will cause the model to be baised.. and we don't want that! so we are trying to affect the mosdels' loss function with a symetric train set (no show probabilty = 50%)

Split Train to 50/50 show/no show

```
df.no_show <- sample(df.train[df.train$no_show==1,])
df.show <- sample(df.train[df.train$no_show==0,])
df.train.balanced <- rbind(df.no_show, df.show[1:17896,])
dim(df.train.balanced)
```

```
## [1] 35792    35
```

Now we're having a balanced DF with 50%/50% shows/no shows

```
dim(df.train.balanced[df.train.balanced$no_show==0,])
```

```
## [1] 17896    35
```

```
dim(df.train.balanced[df.train.balanced$no_show==1,])
```

```
## [1] 17896    35
```

Train Again on the balanced train dataset

With balanced train set

```
noshow.LM.b <- glm(no_show ~ age+
                         waiting_time+
                         scholarship+
                         sms_recieved, data = df.train.balanced, family = binomial)

noshow.CART.b <- tree(no_show ~ week_day+
                       waiting_time+
                       age+
                       is_female+
                       scholarship+
                       hipertension+
                       diabetes+
                       alcoholism+
                       handcap+
                       sms_recieved+
                       poverty+
                       region ,data = df.train.balanced)

set.seed(7)
noshow.RF.b <- randomForest(no_show ~ week_day
                          +waiting_time
                          +age
                          +is_female
                          +scholarship
                          +sms_recieved
                          +poverty
                          +region
                          , data = df.train.balanced, na.action=na.omit, type="classification", ntree=100)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
no_show.GBM.b <- gbm (no_show ~ week_day+
                        waiting_time+
                        age+
                        is_female+
                        scholarship+
                        sms_recieved+
                        poverty+
                        region ,data = df.train.balanced, n.trees = 100, interaction.depth = 4, shrinkage = 0.2,
 verbose = F)
```

```
## Distribution not specified, assuming bernoulli ...
```

Evaluate on test dataset

LM

```
threshold = 0.3
fitted.lm.results <- predict(noshow.LM.b,df.test,type='response')
lm.prediction <- ifelse(fitted.lm.results > threshold,1,0)


cross.table <- table(lm.prediction, df.test$no_show)
l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }
accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.202261931689663"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.200263086274154"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.999320959710276"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.333660822249093"
```

```
model <- "LM"

lm_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

CART

```
threshold = 0.3
fitted.cart.results <- predict(noshow.CART.b,df.test)
cart.prediction <- ifelse(fitted.cart.results > threshold,1,0)
cross.table <- table(cart.prediction, df.test$no_show)
l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }

accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.515539470707985"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.280908267743958"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.912856496152105"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.429613848202397"
```

```
model <- "CART"

cart_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

RF

```
threshold <- 0.3
fitted.rf.results <- predict(noshow.RF.b,df.test)
rf.prediction <- ifelse(fitted.rf.results > threshold,1,0)
cross.table <- table(rf.prediction, df.test$no_show)

l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }

accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.508889391540375"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.280422919508868"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.930511543684925"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.430967606667365"
```

```
model <- "RF"

rf_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

GBM

```
threshold <- 0.3
fitted.gbm.results <- predict(no_show.GBM.b,df.test, n.trees = 100)
gbm.prediction <- ifelse(fitted.gbm.results > threshold,1,0)
cross.table <- table(gbm.prediction, df.test$no_show)
l <- nrow(cross.table)
if(l< 2)  {
          cross.table <- rbind(cross.table, c(0,0))
          }
accuracy <- (cross.table[1,1]+cross.table[2,2])/ (cross.table[1,1]+cross.table[2,2]+cross.table[1,2]+cross.table[
2,1])
precision <- cross.table[2,2]/(cross.table[2,2]+cross.table[2,1])
recall <- cross.table[2,2]/(cross.table[2,2]+cross.table[1,2])
f1 <- 2*(recall*precision)/(recall+precision)

paste("Accuracy -",accuracy)
```

```
## [1] "Accuracy - 0.661253110156073"
```

```
paste("Precision -",precision)
```

```
## [1] "Precision - 0.326318171532021"
```

```
paste("Recall -",recall)
```

```
## [1] "Recall - 0.652784065187868"
```

```
paste("F1 -",f1)
```

```
## [1] "F1 - 0.435123717561859"
```

```
model <- "GBM"

gbm_results <- c(model, round(accuracy,4),round(precision,4),round(recall,4),round(f1,4))
```

Resuls Evaluation whn training on balanced train set

	Model	Accuracy	Precision	Recall	F1
lm_results	LM	0.2023	0.2003	0.9993	0.3337
cart_results	CART	0.5155	0.2809	0.9129	0.4296
rf_results	RF	0.5089	0.2804	0.9305	0.431
gbm_results	GBM	0.6613	0.3263	0.6528	0.4351

Conlclusion

Random Forest will be a suitable model for the balanced train set and the test set.

Depending the business question, we'll assume the following - 1. Using high model threshold (>0.5), we'll get high precision - Since high precision tells us that the ratio of correctly predicted positive observations of the total predicted positive observations, we're sure that the predicted patients will not show up to the appointment. in this case, we'll recommand to **perform a double booking** to another patient to avoid no shows. 2. Using high model threshold (<0.4), we'll get high recall - Since high recall tells us that the ratio of correctly predicted positive observations of the total observations, we're positive that the predicted patients belong to the class of no show. in this case, we'll recommand to **perform a phone call** to that patient remind him his appointment. This way, we can reduce the probability to no show.