```
# Study Title - Functional Outcome Prediction in Ischemic Stroke: A Comparison of Machine Learning (ML) Algorithms and Regression Models


#Study Objective - To examine the predictive performance of ML algorithms for predicting a 90-day functional impairment risk after acute ischemic stroke


#Description of Train and Test Datasets


# Train

# PROVE-IT is a prospective multi-center hospital-based cohort study of 614 patients with acute ischemic stroke presenting within 12 hours

# of stroke symptom onset with evidence of intracranial occlusion on routine computed tomography angiography CTA over 3 years


#Test

# INTERRSeCT is a prospective multi-center hospital-based cohort of patients treated with

# intravenous alteplase comparing the rates of early recanalization in 684 patients.



#Study Predictors

age, sex, stystolic blood pressure, diastolic blood pressure, glucose, NIHSS, hypertension, diabetes, treatment

history of heart disease, history of congestive heart failure, history of atrial fibrillation


#Outcome  - mRS02_D90

##############Internal Validation#############################

#                                                                                   #

##############################Random Forest###################

============================RF===================================

library(dplyr)     # for data manipulation
```

```r
library(caret)     # for model-building
library(DMwR)      # for smote implementation
library(purrr)     # for functional programming (map)
library(pROC)      # for AUC calculations
library(ROSE)      # for re-sampling
library(Zelig)     # for rare event logistic regresson
library(randomForest)   #for random forest
library(e1071)     # for svm;
library(MASS)
library(tidyverse)
library(skimr)
library(knitr)
library(party)
library(rpart)                  #for decision tree
library(C50)                    #for decision tree
library(ada)                    #for adaptive boost machine
library(partykit) #for classification and regression trees
library(compositions)
library(naivebayes)        #for naive bayes
library(psych)
library(MLmetrics)         #for F1 Score calculations
library(verification)  #for calibration brierscore
library(epiR)                   #for sensistivity and specificity calculaions
library(mccr)
shakiru<- read.csv(file.choose(), header = T)
#Normalize the train and test data
temp <- scale(shakiru)
means_s <- attr(temp, "scaled:center")
stds <- attr(temp, "scaled:scale")
```

```
scaled_train =  (shakiru - means) / standard_deviations


temp1<- scale(test)

scaled_test =  (test - means_s) / stds


#under sampling the outcome in training set

shakiru2 <- ovun.sample(mrs02_D90 ~ ., data = scaled_train, method = "under", N = 500)$data

table(shakiru2$mrs02_D90)


#under sampling the outcome in test

test2 <- ovun.sample(mrs02_D90 ~ ., data = scaled_test, method = "under", N = 476)$data

table(test2$mrs02_D90)




set.seed(123)

ind <- sample(2, nrow(norm), replace = TRUE, prob = c(0.6, 0.4))

train <- norm[ind==1,]

dim(train)

test <- norm[ind==2,]

dim(test)

str(test)

#===========Custom Control Parameters================

custom<- trainControl(method = "repeatedcv",

                      number = 3,

                      repeats = 500,

                      verboseIter = T)

#==================random forest============

mtry <- 2000

tunegrid <- expand.grid(.mtry=mtry)
```

```
lm <- train(mrs02_D90~.,
            train,
            method = "rf",
            trControl = custom,
            tuneGrid=tunegrid,
            na.action = na.exclude)
ch<- varImp(lm)
ch1<- plot(ch, main = "Variables Importance in Random Forest")

#======Prediction and Confusion Matrix in train data =========
train<- na.omit(train)
preds<-predict(lm, train)
confusionMatrix(preds,train$mrs02_D90)
Precision(preds, train$mrs02_D90)
F1_Score(preds, train$mrs02_D90)
auc.rf=auc(train$mrs02_D90, as.numeric(as.character(preds)))
auc.rf

#============Prediction and Confucion Matrix in test data========
library(klaR)
test<- na.omit(test)
str(test)
preds.rft<-predict(lm, test)
conf_matrix.rf<-table(preds.rft, test$mrs02_D90)
sensitivity(conf_matrix.rf)
epi.tests(conf_matrix.rf)
Precision(preds.rft, test$mrs02_D90)
F1_Score(preds.rft, test$mrs02_D90)
mccr(preds.rft, test$mrs02_D90)
```

```
auc.rf=auc(test$mrs02_D90, as.numeric(as.character(preds.rft)))
auc.rf
auc.rf.ci=ci.auc(test$mrs02_D90, as.numeric(as.character(preds.rft)))
test$mrs02_D90 <- as.numeric(as.character(test$mrs02_D90))
preds.rft <- as.numeric(as.character(preds.rft))
is.numeric(test$mrs02_D90)
is.numeric(preds.rft)
aa<- round(test$mrs02_D90, preds.rft)
a<- verify(test$mrs02_D90, preds.rft)
summary(a)


#Predictive Accuracy Metric
epi.tests(conf_matrix.rf)
mccr(preds.rft, test$mrs02_D90)
auc.rf
auc.rf.ci
summary(a)


###MCC 95% CI
mcc <- function (actual, predicted)
{
  TP <- sum(actual == 1 & predicted == 1)
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
```

```
  return(mcc)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(preds.rft, test$mrs02_D90, mcc))

get_boot_est_mod <- function(test, metric) {
    metric(preds.rft, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, mcc))
res <- rbind(data.frame(brier_score = reps_pred,
                      approach = 'predictions'),
            data.frame(brier_score = reps_model,
                      approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')

#####Brier Score 95%CI
brier_score <- function(preds, obs) {
  mean((obs - preds)^2)
}
N <- 1000
```

```
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(preds.rft, test$mrs02_D90, brier_score))

get_boot_est_mod <- function(test, metric) {
    metric(preds.rft, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                        approach = 'predictions'),
             data.frame(brier_score = reps_model,
                        approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')

################################ADAPTIVE BOOST############################
set.seed(1234)
lm1 <- train(mrs02_D90~.,
             train,
             method = "ada",
             trControl = custom,
             na.action = na.exclude,
             sub = c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25)))
```

```
summary(lm1)

print(lm1)

plot(lm1$imp[order(lm1$imp, decreasing = TRUE)],

+ ylim = c(0, 100), main = "Variables Ranking in Adaptive Boost Machine")

#======Prediction and Confusion Matrix in train data =========

preds<-predict(lm1, train)

confusionMatrix(preds,train$mrs02_D90)

Precision(preds, train$mrs02_D90)

F1_Score(preds, train$mrs02_D90)

auc.ad=auc(train$mrs02_D90, as.numeric(as.character(preds)))

auc.ad


#============Prediction and Confucion Matrix in test data========

library(klaR)

preds.adt<-predict(lm1, test)

conf_matrix.ad<-table(preds.adt, test$mrs02_D90)

sensitivity(conf_matrix.ad)

epi.tests(conf_matrix.ad)

Precision(preds.adt, test$mrs02_D90)

F1_Score(preds.adt, test$mrs02_D90)

mccr(preds.adt, test$mrs02_D90)

auc.ad=auc(test$mrs02_D90, as.numeric(as.character(preds.adt)))

auc.ad

auc.ad.ci=ci.auc(test$mrs02_D90, as.numeric(as.character(preds.adt)))

test$mrs02_D90 <- as.numeric(as.character(test$mrs02_D90))

preds.adt <- as.numeric(as.character(preds.adt))

is.numeric(test$mrs02_D90)

is.numeric(preds.adt)

bb<- round(test$mrs02_D90, preds.adt)
```

```
b<- verify(test$mrs02_D90, preds.adt)
summary(b)


#Predictive Accuracy Metric
epi.tests(conf_matrix.ad)
mccr(preds.adt, test$mrs02_D90)
auc.ad
auc.ad.ci
summary(b)


###MCC 95%CI
mcc <- function (actual, predicted)
{
  TP <- sum(actual == 1 & predicted == 1)
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
  return(mcc)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(preds.adt, test$mrs02_D90, mcc))
```

```
get_boot_est_mod <- function(test, metric) {
   metric(preds.adt, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                        approach = 'predictions'),
             data.frame(mcc = reps_model,
                        approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')

#####Brier Score 95% CI
reps_pred <- replicate(N, get_boot_est_preds(preds.adt, test$mrs02_D90, brier_score))


get_boot_est_mod <- function(test, metric) {
   metric(preds.adt, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                        approach = 'predictions'),
             data.frame(brier_score = reps_model,
                        approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
```

```
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')

##############################Logistic Regression############
set.seed(1234)
lm2 <- train(mrs02_D90~.,
              train,
              method = "glm",
              trControl = custom,
              family = "binomial",
              maxit = 10000,
              na.action = na.exclude)
summary(lm2)
cy<- varImp(lm2)
plot(cy, main = "" Variables Ranking in Logistic Regression)
#======Prediction and Confusion Matrix in train data =========
library(klaR)
preds<-predict(lm2, train)
confusionMatrix(preds,train$mrs02_D90)
Precision(preds, train$mrs02_D90)
F1_Score(preds, train$mrs02_D90)
auc.lr=auc(train$mrs02_D90, as.numeric(as.character(preds)))
auc.lr
#============Prediction and Confucion Matrix in test data========
library(klaR)
preds.lrt<-predict(lm2, test)
conf_matrix.lr<-table(preds.lrt, test$mrs02_D90)
```

```
sensitivity(conf_matrix.lr)

epi.tests(conf_matrix.lr)

Precision(preds.lrt, test$mrs02_D90)

F1_Score(preds.lrt, test$mrs02_D90)

mccr(preds.lrt, test$mrs02_D90)

auc.lr=auc(test$mrs02_D90, as.numeric(as.character(preds.lrt)))

auc.lr

auc.lr.ci=ci.auc(test$mrs02_D90, as.numeric(as.character(preds.lrt)))

test$mrs02_D90 <- as.numeric(as.character(test$mrs02_D90))

preds.lrt <- as.numeric(as.character(preds.lrt))

is.numeric(test$mrs02_D90)

is.numeric(preds.lrt)

cc<- round(test$mrs02_D90, preds.lrt)

c<- verify(test$mrs02_D90, preds.lrt)

summary(c)



#Predictive Accuracy Metric

epi.tests(conf_matrix.lr)

auc.lr

auc.lr.ci

mccr(preds.lrt, test$mrs02_D90)

summary(c)



###MCC 95% CI

mcc <- function (actual, predicted)

{

  TP <- sum(actual == 1 & predicted == 1)
```

```
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
  return(mcc)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(preds.lrt, test$mrs02_D90, mcc))

get_boot_est_mod <- function(test, metric) {
   metric(preds.lrt, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                  approach = 'predictions'),
            data.frame(mcc = reps_model,
                  approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')
```

```
#####Brier Score 95% CI
reps_pred <- replicate(N, get_boot_est_preds(preds.lrt, test$mrs02_D90, brier_score))


get_boot_est_mod <- function(test, metric) {
   metric(preds.lrt, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                   approach = 'predictions'),
           data.frame(brier_score = reps_model,
                   approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')


#################Classification and regression tree###################
set.seed(1234)
lm3 <- train(mrs02_D90~.,
                    train,
                    method = "rpart",
                    trControl = custom,
                    na.action = na.exclude,
                    control = rpart.control(cp = 0, maxdepth = 8,minsplit = 100))
summary(lm3)
cd<- varImp(lm3)
cd1<- plot(cd, main  = "Variables Ranking in Classification and Regression Tree")
```

```
#======Prediction and Confusion Matrix in train data =========
library(klaR)
preds<-predict(lm3, train)
confusionMatrix(preds,train$mrs02_D90)
Precision(preds, train$mrs02_D90)
F1_Score(preds, train$mrs02_D90)
auc.dt=auc(train$mrs02_D90, as.numeric(as.character(preds)))
auc.dt

#============Prediction and Confucion Matrix in test data========
preds.dtt<-predict(lm3, test)
conf_matrix.dt<-table(preds.dtt, test$mrs02_D90)
sensitivity(conf_matrix.dt)
epi.tests(conf_matrix.dt)
Precision(preds.dtt, test$mrs02_D90)
F1_Score(preds.dtt, test$mrs02_D90)
mccr(preds.dtt, test$mrs02_D90)
auc.dt=auc(test$mrs02_D90, as.numeric(as.character(preds.dtt)))
auc.dt
auc.dt.ci=ci.auc(test$mrs02_D90, as.numeric(as.character(preds.dtt)))
test$mrs02_D90 <- as.numeric(as.character(test$mrs02_D90))
preds.dtt <- as.numeric(as.character(preds.dtt))
is.numeric(test$mrs02_D90)
is.numeric(preds.dtt)
dd<- round(test$mrs02_D90, preds.dtt)
d<- verify(test$mrs02_D90, preds.dtt)
summary(d)

#Predictive Accuracy Metric
```

```
epi.tests(conf_matrix.dt)

auc.dt

auc.dt.ci

mccr(preds.dtt, test$mrs02_D90)

summary(d)


###MCC CI

mcc <- function (actual, predicted)

{

  TP <- sum(actual == 1 & predicted == 1)

  TN <- sum(actual == 0 & predicted == 0)

  FP <- sum(actual == 0 & predicted == 1)

  FN <- sum(actual == 1 & predicted == 0)


  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

  return(mcc)

}

N <- 1000

get_boot_est_preds <- function(preds, obs, metric) {

  idx <- sample(length(preds), replace = TRUE)

  metric(preds[idx], obs[idx])

}

reps_pred <- replicate(N, get_boot_est_preds(preds.dtt, test$mrs02_D90, mcc))


get_boot_est_mod <- function(test, metric) {

    metric(preds.dtt, test$mrs02_D90)

}

reps_model <- replicate(N, get_boot_est_mod(test, mcc))

res <- rbind(data.frame(mcc = reps_pred,
```

```
                    approach = 'predictions'),
          data.frame(mcc = reps_model,
                    approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')


#####Brier Score 95%CI
reps_pred <- replicate(N, get_boot_est_preds(preds.dtt, test$mrs02_D90, brier_score))


get_boot_est_mod <- function(test, metric) {
    metric(preds.dtt, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                    approach = 'predictions'),
          data.frame(brier_score = reps_model,
                    approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')


############################ C50 decion tree###
```

```
set.seed(1234)

lm4 <- train(mrs02_D90~.,

             train,

             method = "C5.0",

             trControl = custom,

             na.action = na.exclude,

             ctrls = ctree_control(minsplit=9, minbucket=2))

plot(lm4)

cj<- varImp(lm4)

plot(cj, main = "Variables Importance in C5.0 Decision Tree")

#======Prediction and Confusion Matrix in train data =========

train<- na.omit(train)

preds<-predict(lm4, train)

confusionMatrix(preds,train$mrs02_D90)

Precision(preds, train$mrs02_D90)

F1_Score(preds, train$mrs02_D90)

auc.cart=auc(train$mrs02_D90, as.numeric(as.character(preds)))

auc.cart


#============Prediction and Confucion Matrix in test data========

library(klaR)

test<- na.omit(test)

preds.cat<-predict(lm4, test)

conf_matrix.cart<-table(preds.cat, test$mrs02_D90)

sensitivity(conf_matrix.cart)

epi.tests(conf_matrix.cart)

test$mrs02_D90 <- as.factor(test$mrs02_D90)

levels(test$mrs02_D90) <- levels(preds)

Precision(preds.cat, test$mrs02_D90)
```

```
F1_Score(preds.cat, test$mrs02_D90)

mccr(preds.cat, test$mrs02_D90)

auc.cart=auc(test$mrs02_D90, as.numeric(as.character(preds.cat)))

auc.cart

auc.cart.ci=ci.auc(test$mrs02_D90, as.numeric(as.character(preds.cat)))

test$mrs02_D90 <- as.numeric(as.character(test$mrs02_D90))

preds.cat <- as.numeric(as.character(preds.cat))

is.numeric(test$mrs02_D90)

is.numeric(preds.cat)

ee<- round(test$mrs02_D90, preds.cat)

e<- verify(test$mrs02_D90, preds.cat)

summary(e)




#Predictive Accuracy Metric

epi.tests(conf_matrix.cart)

auc.cart

auc.cart.ci

mccr(preds.cat, test$mrs02_D90)

summary(e)



###MCC CI

mcc <- function (actual, predicted)

{

  TP <- sum(actual == 1 & predicted == 1)

  TN <- sum(actual == 0 & predicted == 0)

  FP <- sum(actual == 0 & predicted == 1)
```

```
  FN <- sum(actual == 1 & predicted == 0)


  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
  return(mcc)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(preds.cat, test$mrs02_D90, mcc))


get_boot_est_mod <- function(test, metric) {
    metric(preds.cat, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                        approach = 'predictions'),
             data.frame(mcc = reps_model,
                        approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')


####Brier Score CI
reps_pred <- replicate(N, get_boot_est_preds(preds.cat, test$mrs02_D90, brier_score))
```

```
get_boot_est_mod <- function(test, metric) {
   metric(preds.cat, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                  approach = 'predictions'),
            data.frame(brier_score = reps_model,
                  approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')
#######################Support Vector machine#################
lm5 <- train(mrs02_D90~.,
                  train,
                  method = "svmRadial",
                  trControl = custom,
                  na.action = na.exclude,
                  ranges=list(cost=10^(-2:2), gamma=c(.25,.5,1,2)))
summary(lm5)
cr<- varImp(lm5)
plot(cr, main = "Variables Ranking in Support Vector Machine")
#======Prediction and Confusion Matrix in train data ========
preds<-predict(lm5, train)
train$mrs02_D90 <- as.factor(train$mrs02_D90)
levels(train$mrs02_D90) <- levels(preds)
```

```
confusionMatrix(preds,train$mrs02_D90)

Precision(preds, train$mrs02_D90)

F1_Score(preds, train$mrs02_D90)

auc.svm=auc(train$mrs02_D90, as.numeric(as.character(preds)))

auc.svm

#============Prediction and Confucion Matrix in test data========

test<- na.omit(test)

preds.svm<-predict(lm5, test)

conf_matrix.svm<-table(preds.svm, test$mrs02_D90)

epi.tests(conf_matrix.svm)

Precision(preds.svm, test$mrs02_D90)

F1_Score(preds.svm, test$mrs02_D90)

mccr(preds.svm, test$mrs02_D90)

auc.svm=auc(test$mrs02_D90, as.numeric(as.character(preds.svm)))

auc.svm

auc.svm.ci=ci.auc(test$mrs02_D90, as.numeric(as.character(preds.svm)))

test$mrs02_D90 <- as.numeric(as.character(test$mrs02_D90))

preds.svm <- as.numeric(as.character(preds.svm))

is.numeric(test$mrs02_D90)

is.numeric(preds.svm)

ff<- round(test$mrs02_D90, preds.svm)

f<- verify(test$mrs02_D90, preds.svm)

summary(f)




#Predictive Accuracy Metric

epi.tests(conf_matrix.svm)
```

```
auc.svm
auc.svm.ci
mccr(preds.svm, test$mrs02_D90)
summary(f)

###MCC CI
mcc <- function (actual, predicted)
{
  TP <- sum(actual == 1 & predicted == 1)
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
  return(mcc)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(preds.svm, test$mrs02_D90, mcc))

get_boot_est_mod <- function(test, metric) {
    metric(preds.svm, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                        approach = 'predictions'),
```

```
            data.frame(mcc = reps_model,
                       approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')

####Brier Score CI
reps_pred <- replicate(N, get_boot_est_preds(preds.svm, test$mrs02_D90, brier_score))

get_boot_est_mod <- function(test, metric) {
    metric(preds.svm, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                       approach = 'predictions'),
            data.frame(brier_score = reps_model,
                       approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')
####################LASSO LOgistic regression#############
#===========LASSO Logistic Regression=============
set.seed(1234)
```

```
lasso<- train(mrs02_D90~.,
                    train,
                    family = "binomial",
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = 1,
                    lambda = seq(0.0001, 1, length = 5)),
                    trControl = custom,
                    na.action = na.exclude)
print(lasso)
ce<- varImp(lasso)
plot(ce, main = "Variables Importance in LASSO Logistic Regression")
#================Prediction and Confusion matrix in ttrain==============
library(klaR)
p1<- predict(lasso, train)
confusionMatrix(p1, train$mrs02_D90)
Precision(p1, train$mrs02_D90)
F1_Score(p1, train$mrs02_D90)
auc.las=auc(train$mrs02_D90, as.numeric(as.character(p1)))
auc.las
#=============Prediction and confusion matrix in test=========
p2<- predict(lasso, test)
conf_matrix.p2<-table(p2, test$mrs02_D90)
sensitivity(conf_matrix.p2)
epi.tests(conf_matrix.p2)
Precision(p2, test$mrs02_D90)
F1_Score(p2, test$mrs02_D90)
mccr(p2, test$mrs02_D90)
auc.las=auc(test$mrs02_D90, as.numeric(as.character(p2)))
auc.las
```

```
auc.las=ci.auc(test$mrs02_D90, as.numeric(as.character(p2)))

test$mrs02_D90 <- as.numeric(as.character(test$mrs02_D90))

p2 <- as.numeric(as.character(p2))

is.numeric(test$mrs02_D90)

is.numeric(p2)

gg<- round(test$mrs02_D90, p2)

g<- verify(test$mrs02_D90, p2)

summary(g)



#Predictive Accuracy Metric

epi.tests(conf_matrix.p2)

auc.p2

auc.p2.ci

mccr(preds.p2, test$mrs02_D90)

summary(g)




###MCC CI

mcc <- function (actual, predicted)

{

  TP <- sum(actual == 1 & predicted == 1)

  TN <- sum(actual == 0 & predicted == 0)

  FP <- sum(actual == 0 & predicted == 1)

  FN <- sum(actual == 1 & predicted == 0)


  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

  return(mcc)
```

```
}
N <- 1000
reps_pred <- replicate(N, get_boot_est_preds(p2, test$mrs02_D90, mcc))


get_boot_est_mod <- function(test, metric) {
   metric(p2, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                  approach = 'predictions'),
            data.frame(mcc = reps_model,
                  approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')


########Brier Score CI
reps_pred <- replicate(N, get_boot_est_preds(p2, test$mrs02_D90, brier_score))


get_boot_est_mod <- function(test, metric) {
   metric(p2, test$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                  approach = 'predictions'),
            data.frame(brier_score = reps_model,
```

```
                    approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')



##############################External Validation###################
#                                                                  ##
############################# External Validation###################
test<- read.csv(file.choose(), header = T)

test2$age<- as.numeric(test2$age)
test2$sbp<- as.numeric(test2$sbp)
test2$dbp<- as.numeric(test2$dbp)
test2$glucose<- as.numeric(test2$glucose)
test2$nihss<- as.numeric(test2$nihss)
test2$mrs02_D90<- as.factor(test2$mrs02_D90)
test2$hypertension<- as.factor(test2$hypertension)
test2$diabetes<- as.factor(test2$diabetes)
test2$treatment<- as.factor(test2$treatment)
test2$mrs02_D90<- as.factor(test2$mrs02_D90)


##################Decision Tree#####################
set.seed(123)
```

```
dt <- rpart(mrs02_D90~., data = shakiru2)

print(dt)

set.seed(123)


####=================Prediction and Confusion Matrix in train data=========

preds <- predict(dt, shakiru2, type = "class")

confusionMatrix(preds, shakiru2$mrs02_D90)

Precision(preds, shakiru2$mrs02_D90)

F1_Score(preds, shakiru2$mrs02_D90)

auc.dt=auc(shakiru2$mrs02_D90, as.numeric(as.character(preds)))

auc.dt


###========Prediction and Confusion Matrix in test data=============

pred.dt<-predict(dt, test2, type = "class")

conf_matrix.dt<-table(pred.dt, test2$mrs02_D90)

epi.tests(conf_matrix.dt)

confusionMatrix(pred.dt, test2$mrs02_D90)

Precision(pred.dt, test2$mrs02_D90)

F1_Score(pred.dt, test2$mrs02_D90)

mccr(pred.dt, test2$mrs02_D90)

auc.dt=auc(test2$mrs02_D90, as.numeric(as.character(pred.dt)))

auc.dt

auc.dt.ci=ci.auc(test2$mrs02_D90, as.numeric(as.character(pred.dt)))

a=verify(test2$mrs02_D90, as.numeric(as.character(pred.dt)))

summary(a)



#Predictive Accuracy Metric

epi.tests(conf_matrix.dt)
```

```
auc.dt

auc.dt.ci

mccr(pred.dt, test2$mrs02_D90)



###MCC 95%CI

mcc <- function (actual, predicted)

{

  TP <- sum(actual == 1 & predicted == 1)

  TN <- sum(actual == 0 & predicted == 0)

  FP <- sum(actual == 0 & predicted == 1)

  FN <- sum(actual == 1 & predicted == 0)


  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

  return(mcc)

}

N <- 1000

get_boot_est_preds <- function(preds, obs, metric) {

  idx <- sample(length(preds), replace = TRUE)

  metric(preds[idx], obs[idx])

}

reps_pred <- replicate(N, get_boot_est_preds(pred.dt, test2$mrs02_D90, mcc))


get_boot_est_mod <- function(test2, metric) {

    metric(pred.dt, test2$mrs02_D90)

}

reps_model <- replicate(N, get_boot_est_mod(test2, mcc))

res <- rbind(data.frame(mcc = reps_pred,

                        approach = 'predictions'),
```

```
            data.frame(mcc = reps_model,
                       approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')

####Brier Score 95%CI
reps_pred <- replicate(N, get_boot_est_preds(preds.dt, test2$mrs02_D90, brier_score))

get_boot_est_mod <- function(test2, metric) {
    metric(preds.dt, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, brier_score))
res <- rbind(data.frame(mcc = reps_pred,
                       approach = 'predictions'),
            data.frame(mcc = reps_model,
                       approach = 'refit_model'))
ggplot(res, aes(brier_score, color = approach)) +
  geom_density() +
  theme_bw()
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')
```

```
####################Adaptive Boost#####################

ad <- ada(mrs02_D90~., data = shakiru2)

print(ad)

set.seed(123)

####=================Prediction and Confusion Matrix in train data=========

preds <- predict(ad, shakiru2)

confusionMatrix(preds, shakiru$mrs02_D90)

Precision(preds, shakiru$mrs02_D90)

F1_Score(preds, shakiru2$mrs02_D90)

auc.ad=auc(shakiru2$mrs02_D90, as.numeric(as.character(preds)))

auc.ad

###========Prediction and Confusion Matrix in test data==============

pred.ad<-predict(ad, test2)

conf_matrix.ad<-table(pred.ad, test2$mrs02_D90)

epi.tests(conf_matrix.ad)

confusionMatrix(pred.ad, test2$mrs02_D90)

Precision(pred.ad, test2$mrs02_D90)

F1_Score(pred.ad, test2$mrs02_D90)

mccr(pred.ad, test2$mrs02_D90)

auc.ad=auc(test2$mrs02_D90, as.numeric(as.character(pred.ad)))

auc.ad

auc.ad.ci=ci.auc(test2$mrs02_D90, as.numeric(as.character(pred.ad)))

b=verify(test2$mrs02_D90, as.numeric(as.character(pred.ad)))

summary(b)
```

```
#Predictive Accuracy Metric
epi.tests(conf_matrix.ad)
auc.ad
mccr(pred.ad, test2$mrs02_D90)
summary(b)


###MCC 95%CI
mcc <- function (actual, predicted)
{
  TP <- sum(actual == 1 & predicted == 1)
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
  return(mcc)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(pred.ad, test2$mrs02_D90, mcc))

get_boot_est_mod <- function(test2, metric) {
    metric(pred.ad, test2$mrs02_D90)
}
```

```
reps_model <- replicate(N, get_boot_est_mod(test2, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                        approach = 'predictions'),
             data.frame(mcc = reps_model,
                        approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')

####Brier Score 95%CI
reps_pred <- replicate(N, get_boot_est_preds(preds.ad, test2$mrs02_D90, brier_score))

get_boot_est_mod <- function(test2, metric) {
    metric(preds.ad, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, brier_score))
res <- rbind(data.frame(mcc = reps_pred,
                        approach = 'predictions'),
             data.frame(mcc = reps_model,
                        approach = 'refit_model'))
ggplot(res, aes(brier_score, color = approach)) +
  geom_density() +
  theme_bw()
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
```

```
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')

##########################Support vector machine###########
sv <- svm(mrs02_D90~., data = shakiru2)
print(sv)
set.seed(123)

####=================Prediction and Confusion Matrix in train data=========
preds <- predict(sv, shakiru2)
confusionMatrix(preds, shakiru2$mrs02_D90)
Precision(preds, shakiru2$mrs02_D90)
F1_Score(preds, shakiru2$mrs02_D90)
auc.sv=auc(shakiru2$mrs02_D90, as.numeric(as.character(preds)))
auc.sv

###========Prediction and Confusion Matrix in test data=============
pred.sv<-predict(sv, test2)
conf_matrix.sv<-table(pred.sv, test2$mrs02_D90)
epi.tests(conf_matrix.sv)
confusionMatrix(pred.sv, test2$mrs02_D90)
Precision(pred.sv, test2$mrs02_D90)
F1_Score(pred.sv, test2$mrs02_D90)
mccr(pred.sv, test2$mrs02_D90)
auc.sv=auc(test2$mrs02_D90, as.numeric(as.character(pred.sv)))
auc.sv
auc.sv.ci=ci.auc(test2$mrs02_D90, as.numeric(as.character(pred.sv)))
c=verify(test2$mrs02_D90, as.numeric(as.character(pred.sv)))
```

```
summary(c)


#Predictive Accuracy Metric
epi.tests(conf_matrix.sv)
mccr(pred.sv, test2$mrs02_D90)
auc.sv
auc.sv.ci
summary(c)


###MCC 95% CI
mcc <- function (actual, predicted)
{
  TP <- sum(actual == 1 & predicted == 1)
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
  return(mcc)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(pred.sv, test2$mrs02_D90, mcc))
```

```
get_boot_est_mod <- function(test2, metric) {
   metric(pred.sv, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                    approach = 'predictions'),
            data.frame(mcc = reps_model,
                    approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')

####Brier Score 95% CI
reps_pred <- replicate(N, get_boot_est_preds(preds.sv, test2$mrs02_D90, brier_score))

get_boot_est_mod <- function(test2, metric) {
   metric(preds.sv, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                    approach = 'predictions'),
            data.frame(brier_score = reps_model,
                    approach = 'refit_model'))
ggplot(res, aes(brier_score, color = approach)) +
  geom_density() +
  theme_bw()
```

```
calc_ci_95 <- function(v) {
 q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
 paste0('(',q[1],' to ',q[2],')')
 }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')

######################C50 decision tree#############
library(C50)
ca <- C5.0(mrs02_D90~., data = shakiru2[,1:14], rules = TRUE)
print(ca)
set.seed(123)

####=================Prediction and Confusion Matrix in train data=========
preds <- predict(ca, shakiru2)
confusionMatrix(preds, shakiru2$mrs02_D90)
Precision(preds, shakiru2$mrs02_D90)
F1_Score(preds, shakiru2$mrs02_D90)
auc.ca=auc(shakiru2$mrs02_D90, as.numeric(as.character(preds)))
auc.ca

###========Prediction and Confusion Matrix in test data=============
pred.ca<-predict(ca, test2)
conf_matrix.ca<-table(pred.ca, test2$mrs02_D90)
epi.tests(conf_matrix.ca)
confusionMatrix(pred.ca, test2$mrs02_D90)
Precision(pred.ca, test2$mrs02_D90)
F1_Score(pred.ca, test2$mrs02_D90)
mccr(pred.ca, test2$mrs02_D90)
```

```
auc.ca=auc(test2$mrs02_D90, as.numeric(as.character(pred.ca)))
auc.ca
auc.ca.ci=ci.auc(test2$mrs02_D90, as.numeric(as.character(pred.ca)))
d=verify(test2$mrs02_D90, as.numeric(as.character(pred.ca)))
summary(d)


#Predcitive Accuracy Metric
epi.tests(conf_matrix.ca)
mccr(pred.ca, test2$mrs02_D90)
auc.ca
auc.ca.ci
summary(d)


###MCC 95%CI
mcc <- function (actual, predicted)
{
  TP <- sum(actual == 1 & predicted == 1)
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
  return(mcc)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
```

```
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(pred.ca, test2$mrs02_D90, mcc))

get_boot_est_mod <- function(test2, metric) {
    metric(pred.ca, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                        approach = 'predictions'),
             data.frame(mcc = reps_model,
                        approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')

####Brier Score 95%CI
reps_pred <- replicate(N, get_boot_est_preds(preds.ca, test2$mrs02_D90, brier_score))

get_boot_est_mod <- function(test2, metric) {
    metric(preds.ca, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                        approach = 'predictions'),
             data.frame(brier_score = reps_model,
```

```
                    approach = 'refit_model'))
ggplot(res, aes(brier_score, color = approach)) +
  geom_density() +
  theme_bw()
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')


##################Random Forest ######################
rf <- randomForest(mrs02_D90~., data = shakiru2)
print(rf)
set.seed(123)
#=================Prediction and Confusion Matrix in train data=========
preds <- predict(rf, shakiru2)
confusionMatrix(preds, shakiru2$mrs02_D90)
Precision(preds, shakiru2$mrs02_D90)
F1_Score(preds, shakiru2$mrs02_D90)
auc.rf=auc(shakiru2$mrs02_D90, as.numeric(as.character(preds)))
auc.rf

#========Prediction and Confusion Matrix in test data=============
pred.rf<-predict(rf, test2)
conf_matrix.rf<-table(pred.rf, test2$mrs02_D90)
epi.tests(conf_matrix.rf)
confusionMatrix(pred.rf, test2$mrs02_D90)
```

```
Precision(pred.rf, test2$mrs02_D90)
F1_Score(pred.rf, test2$mrs02_D90)
mccr(pred.rf, test2$mrs02_D90)
auc.rf=auc(test2$mrs02_D90, as.numeric(as.character(pred.rf)))
auc.rf
auc.rf.ci=ci.auc(test2$mrs02_D90, as.numeric(as.character(pred.rf)))
e=verify(test2$mrs02_D90, as.numeric(as.character(pred.rf)))
summary(e)

#Predictive Accuracy Metric
epi.tests(conf_matrix.rf)
mccr(pred.rf, test2$mrs02_D90)
auc.rf
auc.rf.ci
summary(e)


###MCC 95%CI
mcc <- function (actual, predicted)
{
  TP <- sum(actual == 1 & predicted == 1)
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))
  return(mcc)
}
N <- 1000
```

```
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
}
reps_pred <- replicate(N, get_boot_est_preds(pred.rf, test2$mrs02_D90, mcc))

get_boot_est_mod <- function(test2, metric) {
    metric(pred.rf, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, mcc))
res <- rbind(data.frame(mcc = reps_pred,
                        approach = 'predictions'),
             data.frame(mcc = reps_model,
                        approach = 'refit_model'))
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')

####Brier Score 95%CI
brier_score <- function(preds, obs) {
  mean((obs - preds)^2)
}
N <- 1000
get_boot_est_preds <- function(preds, obs, metric) {
  idx <- sample(length(preds), replace = TRUE)
  metric(preds[idx], obs[idx])
```

```
}
reps_pred <- replicate(N, get_boot_est_preds(preds.rf, test2$mrs02_D90, brier_score))


get_boot_est_mod <- function(test2, metric) {
   metric(preds.rf, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                  approach = 'predictions'),
            data.frame(brier_score = reps_model,
                  approach = 'refit_model'))
ggplot(res, aes(brier_score, color = approach)) +
  geom_density() +
  theme_bw()
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')

###################Logistic Regression####################
lr <- glm(as.factor(mrs02_D90)~., data = shakiru2, family = "binomial")
print(lr)
set.seed(123)

####==================Prediction and Confusion Matrix in train data=========
preds <- predict(lr, shakiru2, type = "response")
pred2<- ifelse(preds>0.5, 1, 0)
```

```
confusionMatrix(table(pred2, shakiru2$mrs02_D90))

Precision(pred2, shakiru2$mrs02_D90)

F1_Score(pred2, shakiru$mrs02_D90)

auc.lr=auc(shakiru2$mrs02_D90, as.numeric(as.character(pred2)))

auc.lr

###========Prediction and Confusion Matrix in test data==============

pred.lr<-predict(lr, test2)

pred3<- ifelse(pred.lr>0.5, 1, 0)

conf_matrix.lr<-table(pred3, test2$mrs02_D90)

epi.tests(conf_matrix.lr)

Precision(pred3, test2$mrs02_D90)

F1_Score(pred3, test2$mrs02_D90)

mccr(pred3, test2$mrs02_D90)

auc.rf=auc(test2$mrs02_D90, as.numeric(as.character(pred3)))

auc.rf

auc.lr.ci=ci.auc(test2$mrs02_D90, as.numeric(as.character(pred3)))

f=verify(test2$mrs02_D90, as.numeric(as.character(pred3)))

summary(f)



#Predictive Accuracy Metric

epi.tests(conf_matrix.lr)

mccr(pred3, test2$mrs02_D90)

auc.rf

auc.lr.ci

summary(f)
```

```
###MCC 95%CI

mcc <- function (actual, predicted)

{

  TP <- sum(actual == 1 & predicted == 1)

  TN <- sum(actual == 0 & predicted == 0)

  FP <- sum(actual == 0 & predicted == 1)

  FN <- sum(actual == 1 & predicted == 0)


  mcc <- ((TP*TN)-(FP*FN)) / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

  return(mcc)

}

N <- 1000

get_boot_est_preds <- function(preds, obs, metric) {

  idx <- sample(length(preds), replace = TRUE)

  metric(preds[idx], obs[idx])

}

reps_pred <- replicate(N, get_boot_est_preds(pred3, test2$mrs02_D90, mcc))


get_boot_est_mod <- function(test2, metric) {

    metric(pred3, test2$mrs02_D90)

}

reps_model <- replicate(N, get_boot_est_mod(test2, mcc))

res <- rbind(data.frame(brier_score = reps_pred,

                        approach = 'predictions'),

             data.frame(brier_score = reps_model,

                        approach = 'refit_model'))

calc_ci_95 <- function(v) {

  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)

  paste0('(',q[1],' to ',q[2],')')
```

```
 }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')

####Brier Score 95%CI
reps_pred <- replicate(N, get_boot_est_preds(pred3, test2$mrs02_D90, brier_score))

get_boot_est_mod <- function(test2, metric) {
   metric(pred3, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, brier_score))
res <- rbind(data.frame(brier_score = reps_pred,
                  approach = 'predictions'),
          data.frame(brier_score = reps_model,
                  approach = 'refit_model'))
ggplot(res, aes(brier_score, color = approach)) +
  geom_density() +
  theme_bw()
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
   calc_ci_95(reps_pred),'\n')

######################Lasso Logistic regression####################
##==========LASSO Logistic Regression=============
set.seed(1234)
lasso<- train(mrs02_D90~.,
```

```
                    shakiru2,
                    family = "binomial",
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = 1,
                    lambda = seq(0.0001, 1, length = 5)))

##=================Prediction and Confusion matrix in ttrain===============
p1<- predict(lasso, shakiru2)
shakiru2$mrs02_D90 <- as.factor(shakiru2$mrs02_D90)
levels(norm$mrs02_D90) <- levels(p1)
confusionMatrix(p1, shakiru2$mrs02_D90)
Precision(p1, shakiru2$mrs02_D90)
F1_Score(p1, shakiru2$mrs02_D90)
auc.las=auc(shakiru2$mrs02_D90, as.numeric(as.character(p1)))
auc.las

##==============Prediction and confusion matrix in test==========
p2.v<- predict(lasso, test2)
conf_matrix.las<-table(p2.v, test2$mrs02_D90)
epi.tests(conf_matrix.las)
test2$mrs02_D90 <- as.factor(test2$mrs02_D90)
levels(test2$mrs02_D90) <- levels(p2.v)
confusionMatrix(p2.v, test2$mrs02_D90)
Precision(p2.v, test2$mrs02_D90)
F1_Score(p2.v, test2$mrs02_D90)
mccr(p2.v, test2$mrs02_D90)
auc.las=auc(test2$mrs02_D90, as.numeric(as.character(p2.v)))
auc.las
auc.las.ci=ci.auc(test2$mrs02_D90, as.numeric(as.character(p2.v)))
```

```
g=verify(test2$mrs02_D90, as.numeric(as.character(p2.v)))
summary(g)


#Predictive Accuracy Metric
epi.tests(conf_matrix.las)
mccr(p2.v, test2$mrs02_D90)
auc.las
auc.las.ci
summary(g)


####Brier Score 95% CI
reps_pred <- replicate(N, get_boot_est_preds(p2.v, test2$mrs02_D90, brier_score))

get_boot_est_mod <- function(test2, metric) {
   metric(p2.v, test2$mrs02_D90)
}
reps_model <- replicate(N, get_boot_est_mod(test2, brier_score))

res <- rbind(data.frame(brier_score = reps_pred,
                        approach = 'predictions'),
             data.frame(brier_score = reps_model,
                        approach = 'refit_model'))
ggplot(res, aes(brier_score, color = approach)) +
  geom_density() +
  theme_bw()
calc_ci_95 <- function(v) {
  q <- format(quantile(v, probs = c(0.025, 0.975)), digits = 5)
```

```
  paste0('(',q[1],' to ',q[2],')')
  }
cat('CI using bootstrapped estimates from predictions only:',
    calc_ci_95(reps_pred),'\n')


##Calibration Plots

rf_lift <- train(mrs02_D90 ~ ., data = train,
                 method = "rf",
                 trControl = custom)
set.seed(1045)
C50_lift <- train(mrs02_D90 ~ ., data = train,
                 method = "C5.0",
                trControl = custom)

set.seed(1045)
ctree_lift <- train(mrs02_D90 ~ ., data = train,
                method = "ctree",
               trControl = custom)

svm_lift <- train(mrs02_D90 ~ ., data = train,
                method = "svmRadial",
                trControl = custom)


ada_lift <- train(mrs02_D90 ~ ., data = train,
                method = "ada",
               trControl = custom)
```

```
lr_lift <- train(mrs02_D90 ~ ., data = train,
              method = "glm",
                                    family = "binomial",
             trControl = custom)

lasso_lift <- train(mrs02_90 ~ ., data = train,
              method = "glmnet",
                                    family = "binomial",
                                    tuneGrid = expand.grid(alpha = 1,
                                    lambda = seq(0.0001, 1, length = 5)),
              trControl = custom)


## Generate the test set results
lift_results <- data.frame(mrs02_90 = test$mrs02_90)
lift_results$RF <- predict(rf_lift, test, type = "prob")[,"mrs02_D901"[1]]
lift_results$C50 <- predict(C50_lift, test, type = "prob")[,"mrs02_D90"[1]]
lift_results$CART <- predict(cart_lift, test, type = "prob")[,"mrs02_901"[1]]
lift_results$SVM <- predict(svm_lift, test, type = "prob")[,"mrs02_D901"[1]]
lift_results$ADA <- predict(ada_lift, test, type = "prob")[,"mrs02_D901"[1]]
lift_results$LR <- predict(lr_lift, test, type = "prob")[,"mrs02_D901"[1]]
lift_results$LASSO <- predict(lasso_lift, test, type = "prob")[,"mrs02_D901"[1]]
head(lift_results)


cal_obj <- calibration(mrs02_D90 ~ RF + DT,
                   data = lift_results,
                   cuts = 4)
```

```
i<- plot(cal_obj, type = "l", col = c("blue4", "cyan4"),
key = list(rows = 2, text = list(c("RF", "DT")),
lines = T, col = c("blue4", "cyan4")))


cal_obj1 <- calibration(mrs02_D90 ~ CART + SVM + ADA,
                    data = lift_results,
                    cuts = 4)

ii<- plot(cal_obj1, type = "l", col = c("orange", "blue", "magenta"),
key = list(rows = 3, text = list(c("SVM", "CART", "ADA")),
lines = T, col = c("orange", "blue", "magenta")))


cal_obj3 <- calibration(mrs02_D90 ~ LASSO + LR,
                    data = lift_results,
                    cuts = 4)

iii<- plot(cal_obj3, type = "l", col = c("black", "red"),
key = list(rows = 2, text = list(c("LR", "LASSO")),
lines = T, col = c("black", "red")))


#############

require(gridExtra)
grid.arrange(i, ii, iii, ncol=3)
```