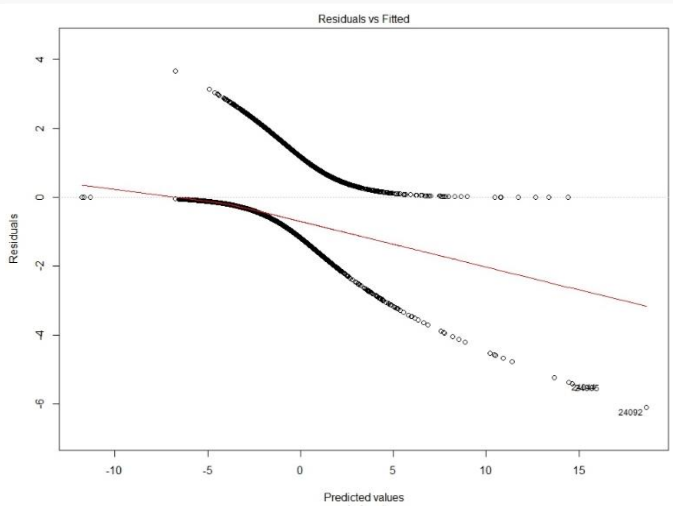
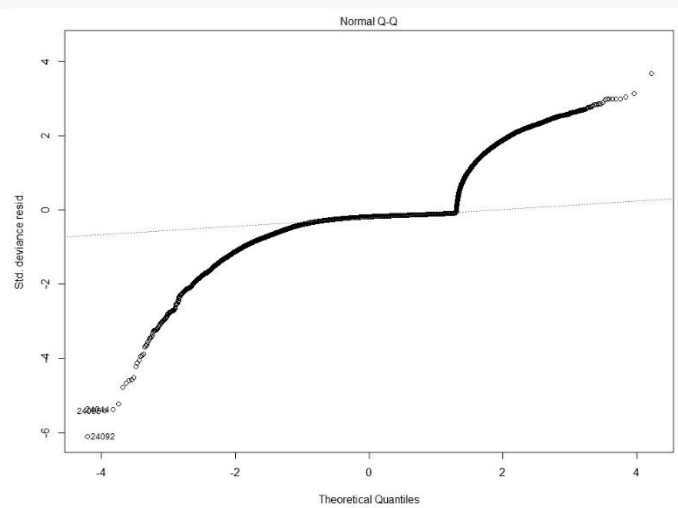
```r
#===============================================================================
#                       Using R: Success of bank telemarketing
#===============================================================================
#The bank telemarketing data is available at:
#http://archive.ics.uci.edu/ml/datasets/Bank+Marketing
#Let's perform classification modeling on this data (~ 41k records).

#Input variables:

# bank client data:
# 1 - age (numeric)
# 2 - job: type of job (categorical: "admin.", "blue-collar", "entrepreneur",
"housemaid", "management", "retired", "self-employed", "services", "student",
"technician", "unemployed", "unknown")
# 3 - marital: marital status (categorical: "divorced", "married", "single", "unknown";
#note: "divorced" means divorced or widowed)
# 4 - education (categorical: "basic.4y","basic.6y","basic.9y", "high.school",
"illiterate", "professional.course", "university.degree", "unknown")
# 5 - default: has credit in default? (categorical: "no", "yes" ,"unknown")
# 6 - housing: has housing loan? (categorical: "no" ,"yes" ,"unknown")
# 7 - loan: has personal loan? (categorical: "no" ,"yes" ,"unknown")

# related with the last contact of the current campaign:
#  8 - contact: contact communication type (categorical: "cellular", "telephone")
#  9 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov",
"dec")
# 10 - day_of_week: last contact day of the week (categorical: "mon", "tue", "wed",
"thu", "fri")
# 11 - duration: last contact duration, in seconds (numeric). Important note:  this
attribute highly affects the output target (e.g., if duration=0 then y="no"). Yet, the
duration is not known before a call is performed. Also, after the end of the call y is
obviously known. Thus, this input should only be included for benchmark purposes and
should be discarded if the intention is to have a realistic predictive model.

# other attributes:
# 12 - campaign: number of contacts performed during this campaign and for this client
(numeric, includes last contact)
# 13 - pdays: number of days that passed by after the client was last contacted from a
previous campaign (numeric; 999 means client was not previously contacted)
# 14 - previous: number of contacts performed before this campaign and for this client
(numeric)
# 15 - poutcome: outcome of the previous marketing campaign (categorical: "failure",
"nonexistent", "success")

# social and economic context attributes
# 16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
# 17 - cons.price.idx: consumer price index - monthly indicator (numeric)
# 18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
# 19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
# 20 - nr.employed: number of employees - quarterly indicator (numeric)

Bank.Marketing <- read.table("bank-additional-full.csv", header=TRUE, sep=";")
```

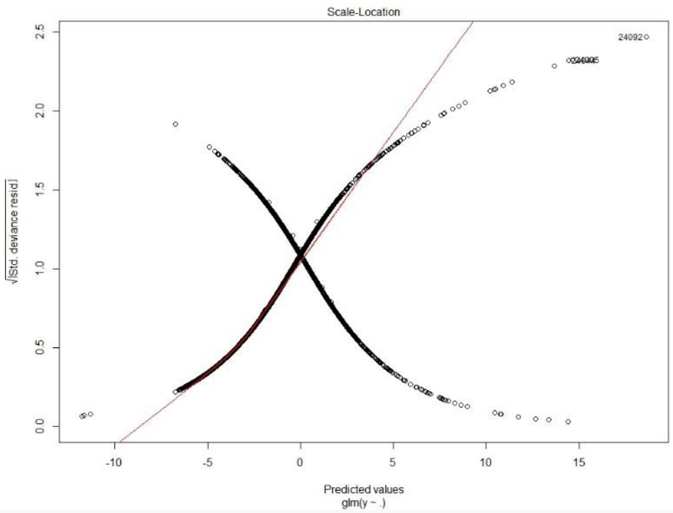
```
#Assessing Missing Data
Bank.Marketing[Bank.Marketing=="unknown"] <- NA
mean(is.na(Bank.Marketing))
aggregate(Bank.Marketing, by=list(Bank.Marketing$y), function(x) mean(is.na(x)))

#Imputing Missing Data using KNN method
library(VIM)
temp <- kNN(Bank.Marketing, k=10)
Bank.Marketing <- temp[,1:21]
Bank.Marketing$y = ifelse(Bank.Marketing$y == "no", 0, 1)
Bank.Marketing$y <- as.factor(Bank.Marketing$y)
```

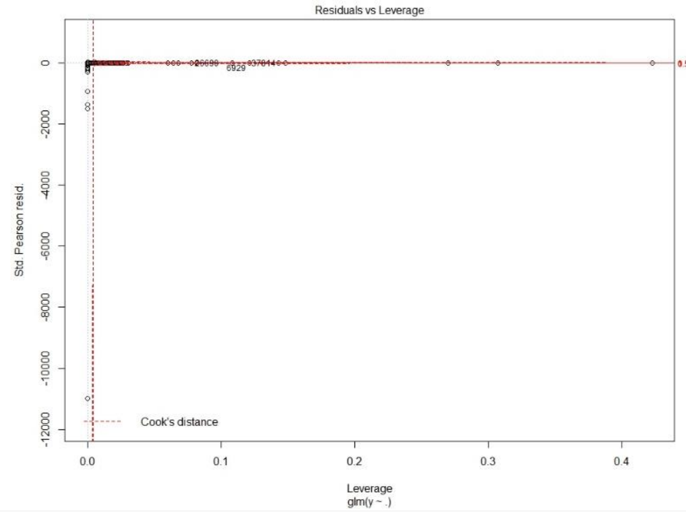
The Data has 41188 observation and one dependent variable and 20 predictors. There is a total missing of about 1.47% in the data and they are associated with “job”, “marital”, “education”, “default”, “housing” and “loan”. I impute the missing data using K Nearest Neighbor method with k=10.

```
#-------------------------------------------------------------------------------------
#(a)I used 39568 first observations associated with the data from the beginning to the
end of April 2010 for the training and the rest –1620– is kept for test purpose. The
proportion of test to training is about 4.1%. In this regard, we use old data to predict
future.

# Concerning cross validation, it is ignored for Logistic Regression. However, in Elastic
net and Decision tree I consider 10 cross validations. Also, in the random forest and
boosted tree it is also ignored due to computational difficulty (Out of memory).

Training <- Bank.Marketing[1:39568,]
Test <- Bank.Marketing[39569:length(Bank.Marketing[,1]),]

#-------------------------------------------------------------------------------------
#Using a logistic regression approach, I evaluate the influence, variance inflation, and
residual diagnostics of your model.
fit.LR <- glm(data=Training, y~., family="binomial")
summary(fit.LR)
plot(fit.LR)
```

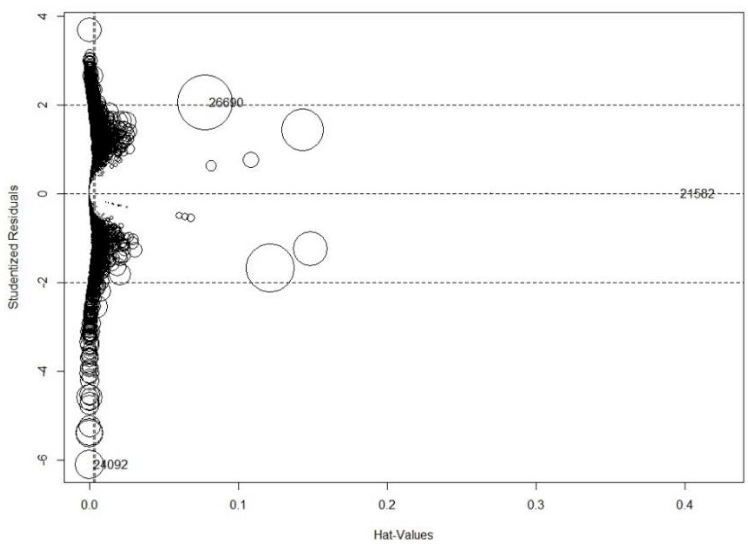









```
#influence
influence.measures(fit.LR)
influencePlot(fit.LR)
```



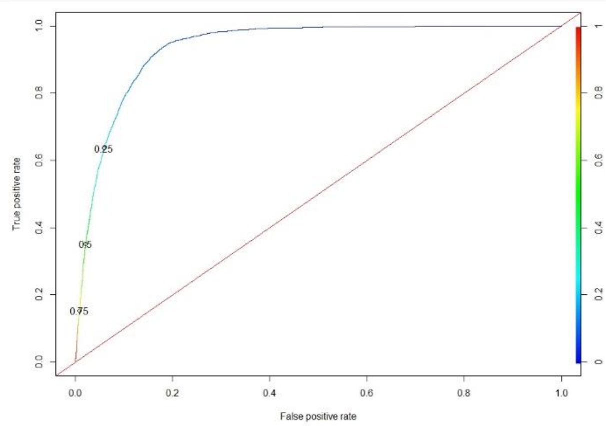

```
#As can be seen the residuals are spread in both side of the zero line. But there are
some large residuals or with large cook’s distance. They can be a potential for being
outlier or having a large leverage.

#variance inflation
library(car)
vif(fit.LR)
# Some variables have large factor (greater than 10) such as “emp.var.rate”, “cons.price.
idx”, “euribor3m” and “nr.employed”. They could be an indicative of problem in case of mu
lticollinearity. Also, all the variables have VIF greater than 1 and they are considered
to indicate a problem.
As can be seen there are big residuals in the data using this method. There may be
nonlinear relationship among them, and this method are not able to capture them. Overall,
from all the graphs it can be understand that this method is not a suitable method.

#---------------------------------------------------------------------------------------
#(c)using elastic net regularization (for logistic regression), decision tree, and either
random forest or a boosted tree, we can build the best classifier for test data.

#Elastic Net regularization (for logistic regression)
```

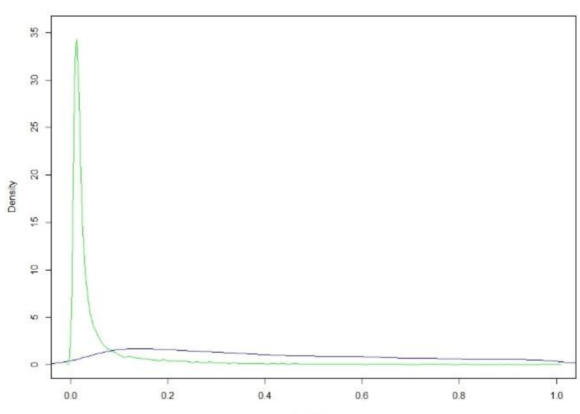
```r
library(caret)
fitControl <- trainControl(method="cv", number=10)# 10-fold CV
fit.EN <- train(y~., data=Training, method="glmnet", trControl=fitControl)
summary(fit.EN)
plot(fit.EN)

#Decision Tree
library(rpart)
fit.DT <- rpart(y~., data=Training, parms=list(split="information"),
control=rpart.control(cp=0.001), xval=20)
summary(fit.DT)
library(rattle)
fancyRpartPlot(fit.DT)
printcp(fit.DT) #the cost-parameter
plotcp(fit.DT)
cp <- fit.DT$cptable[which.min(fit.DT$cptable[,"xerror"]),"CP"]
fit.DT <- prune(fit.DT, cp=cp)#Pruning the tree
fancyRpartPlot(fit.DT)

#Random Forest
library(randomForest)
fit.RF <- randomForest(y ~ ., data=Training, importance=TRUE, ntrees=1500, mtry=3)
plot(fit.RF)

par(mfrow=c(2, 1))
barplot(fit.RF$importance[, 3], main="Importance (Dec.Accuracy)")
barplot(fit.RF$importance[, 4], main="Importance (Gini Index)")
par(mfrow=c(1, 1))
varImpPlot(fit.RF)

#Boosted Tree
library(adabag)
fit.BT <- boosting(y ~ ., data=Training, boos=FALSE, mfinal=20)

#-----------------------------------------------------------------------------------------
#(d) Using the custom function “wrapper” we can evaluate and compare the models
developed.
###################################For Training Data###################################
Tr = Training$y

#Elastic Net regularization
Prob.EN <- predict(fit.EN, newdata=Training, type="prob")[,2]
Pred.EN <- as.numeric(predict(fit.EN, newdata=Training))-1
wrapper(Tr, Pred.EN, Prob.EN)

#Decision Tree
Prob.DT <- predict(fit.DT, newdata=Training, type="prob")[,2]
Pred.DT <- as.numeric(predict(fit.DT, newdata=Training, type="class"))-1
wrapper(Tr, Pred.DT, Prob.DT)

#Random Forest
Prob.RF <- predict(fit.RF, newdata=Training[,1:20], type="prob")[,2]
pred.RF <- as.numeric(predict(fit.RF, newdata=Training, type="class"))-1
wrapper(Tr, pred.RF, Prob.RF)
```

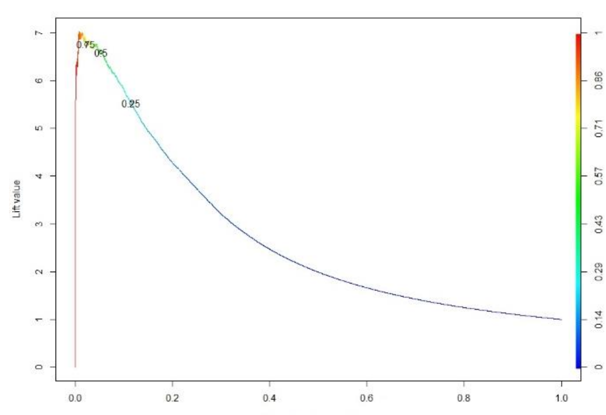
```
#Boosted Tree
Prob.BT <- predict(fit.BT, newdata=Training)$prob[,2]
pred.BT <- as.numeric(as.factor(predict(fit.BT, newdata=Training)$class))-1
wrapper(Tr, pred.BT, Prob.BT)

####################################For Test Data####################################
Tr <- Test$y

#Elastic Net regularization
Prob.EN <- predict(fit.EN, newdata=Test, type="prob")[,2]
Pred.EN <- as.numeric(predict(fit.EN, newdata=Test))-1
wrapper(Tr, Pred.EN, Prob.EN)

#Decision Tree
Prob.DT <- predict(fit.DT, newdata=Test, type="prob")[,2]
Pred.DT <- as.numeric(predict(fit.DT, newdata=Test, type="class"))-1
wrapper(Tr, Pred.DT, Prob.DT)

#Random Forest
Prob.RF <- predict(fit.RF, newdata=Test, type="prob")[,2]
pred.RF <- as.numeric(predict(fit.RF, newdata=Test, type="class"))-1
wrapper(Tr, pred.RF, Prob.RF)

#Boosted Tree
Prob.BT <- predict(fit.BT, newdata=Test)$prob[,2]
pred.BT <- as.numeric(as.factor(predict(fit.BT, newdata=Test)$class))-1
wrapper(Tr, pred.BT, Prob.BT)
```

Elastic Net: (Training Data)

AUC = 66.54249
Accuracy = 0.9181
Kappa = 0.4131
Sensitivity = 0.35239
Specificity = 0.97846

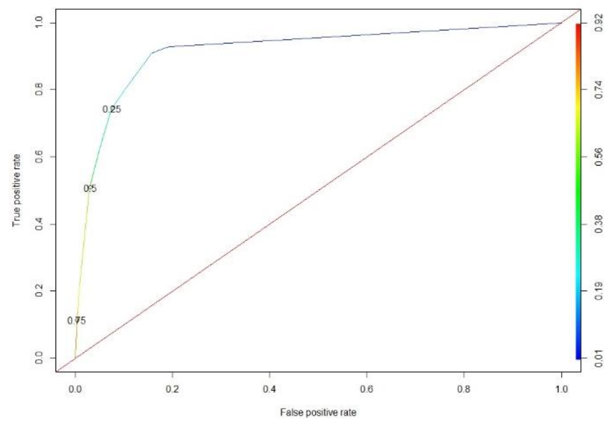


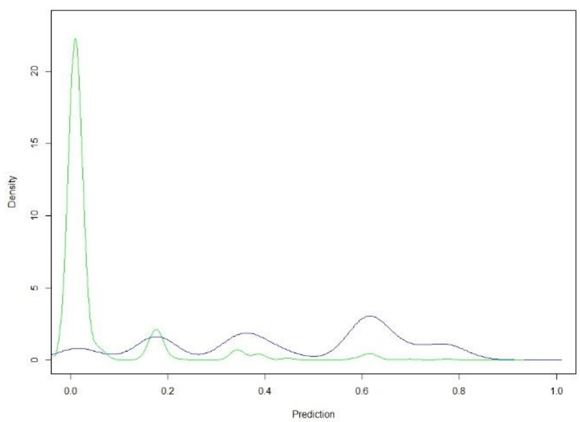


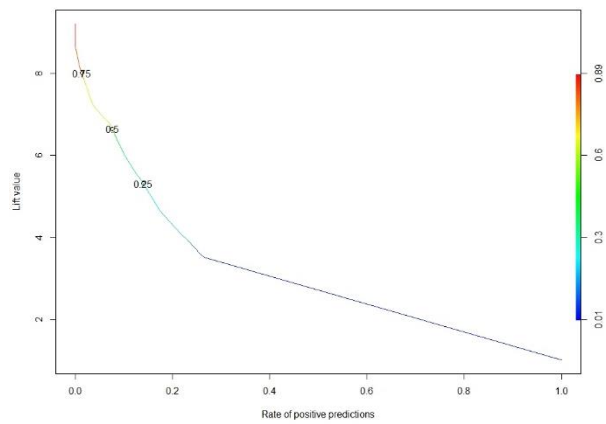


Decision Tree: (Training Data)

AUC = 73.02561
Accuracy = 0.9256
Kappa = 0.519
Sensitivity = 0.48820
Specificity = 0.97231

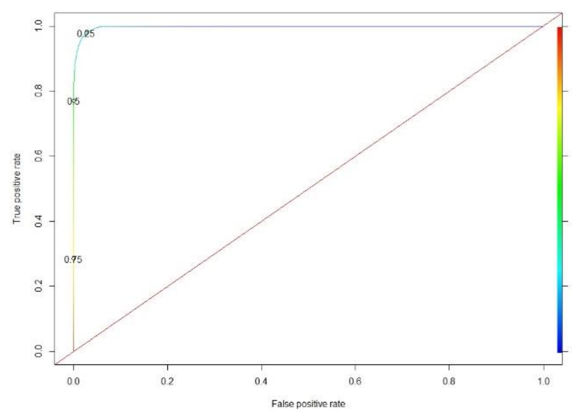


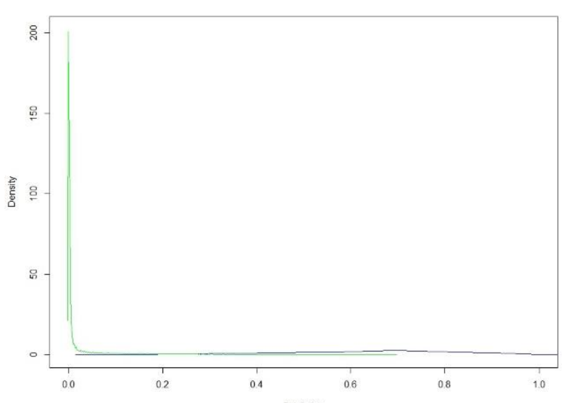


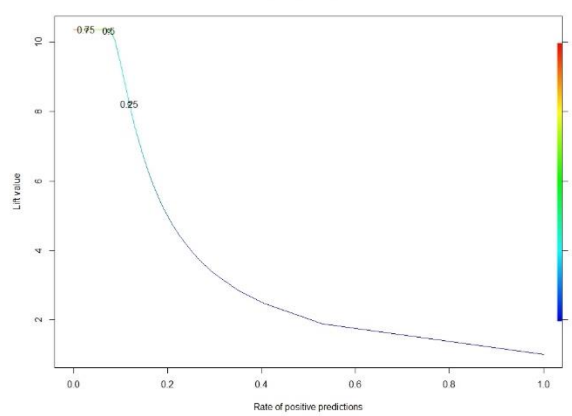


Random Forest: (Training Data)

AUC = 88.38542
Accuracy = 0.9774
Kappa = 0.8555
Sensitivity = 0.76796
Specificity = 0.99975

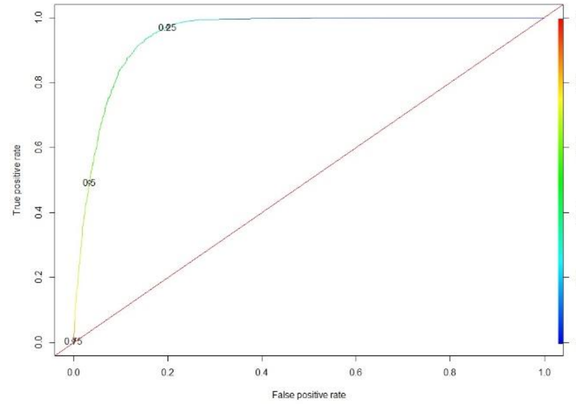


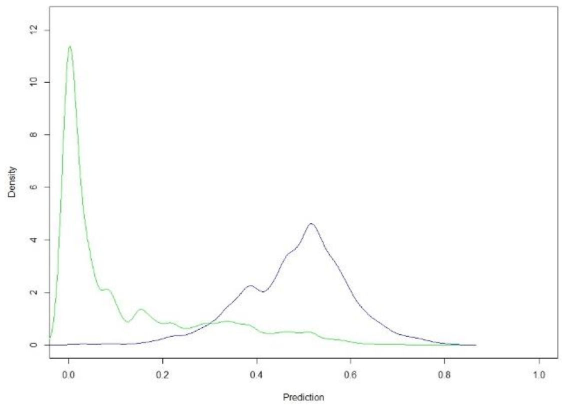


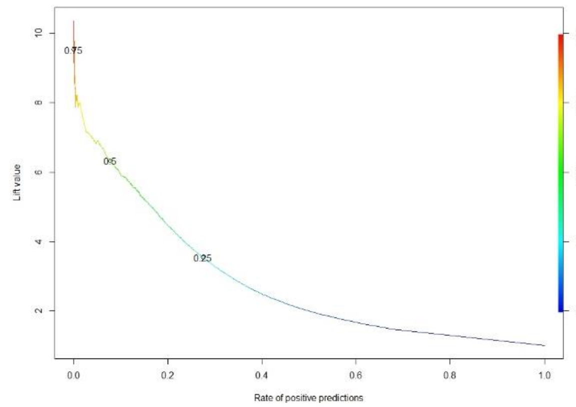


Boosted Tree: (Training Data)

AUC = 73.00024
Accuracy = 0.9207
Kappa = 0.5028
Sensitivity = 0.49371
Specificity = 0.96630

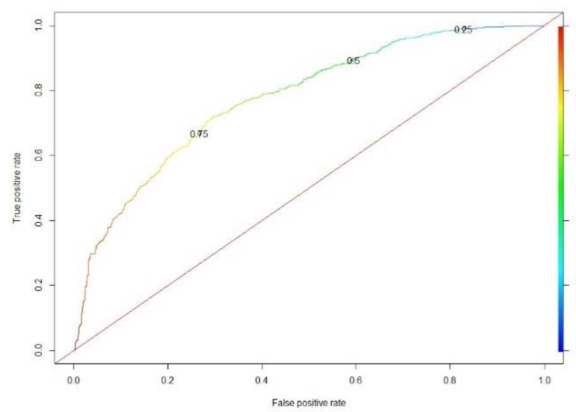


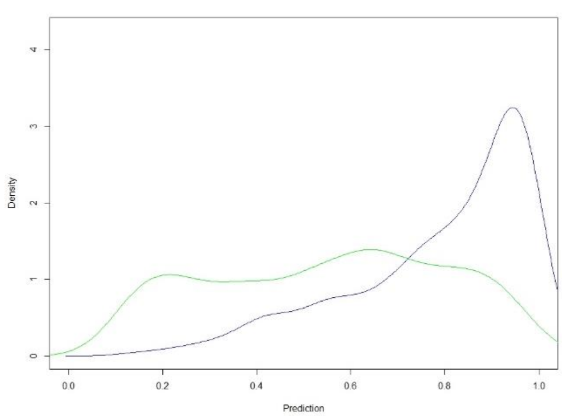


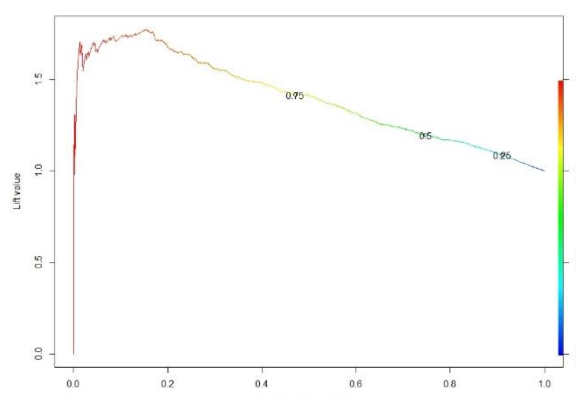


Elastic Net: (Test Data)

AUC = 65.01073
Accuracy = 0.6549
Kappa = 0.3031
Sensitivity = 0.8947
Specificity = 0.4055

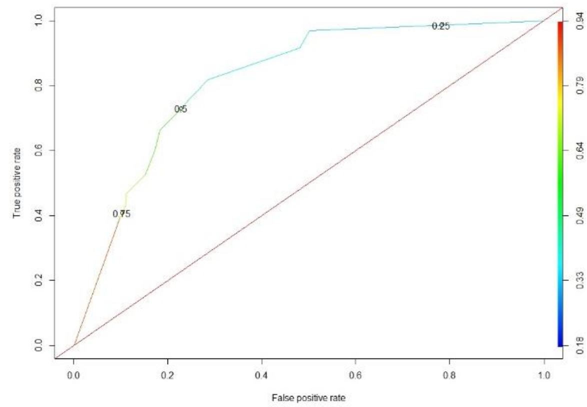


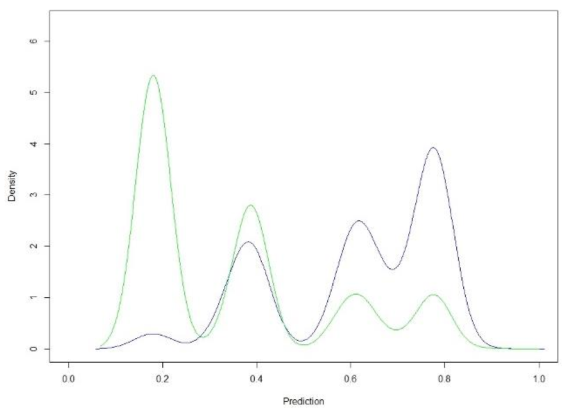


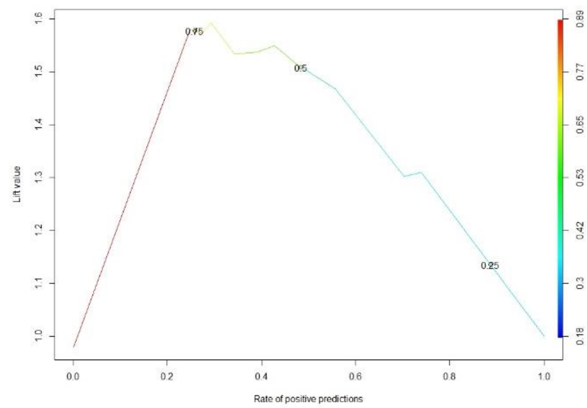


Decision Tree: (Test Data)

AUC = 75.04269
Accuracy = 0.75
Kappa = 0.5003
Sensitivity = 0.7288
Specificity = 0.7720

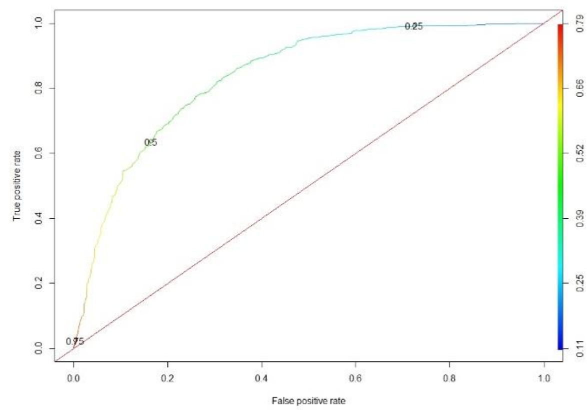


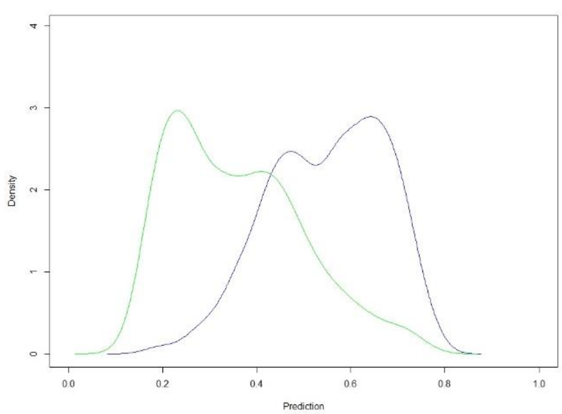


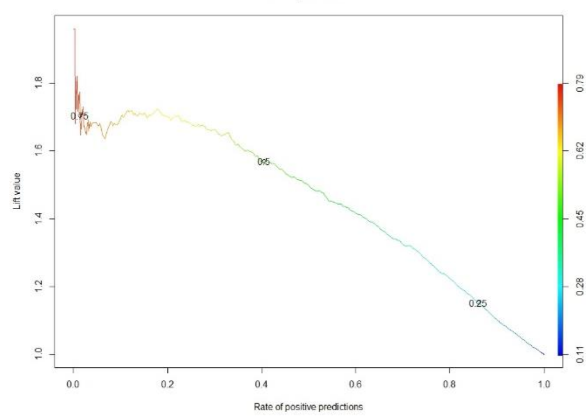


Random Forest: (Test Data)

AUC = 73.53273
Accuracy = 0.7333
Kappa = 0.4687
Sensitivity = 0.6344
Specificity = 0.8363

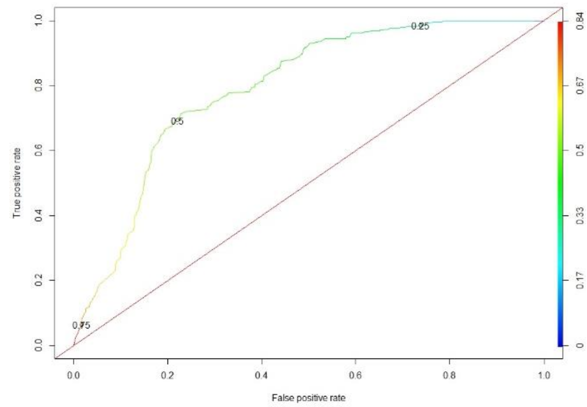


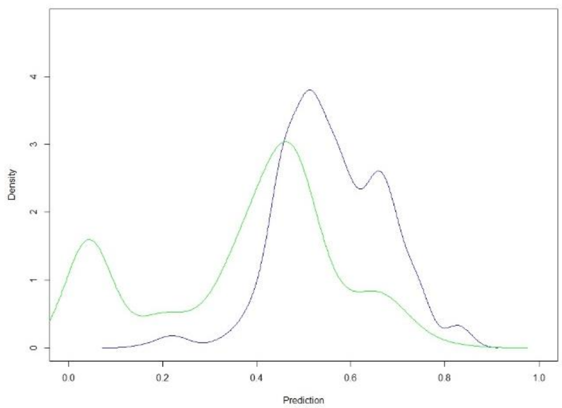


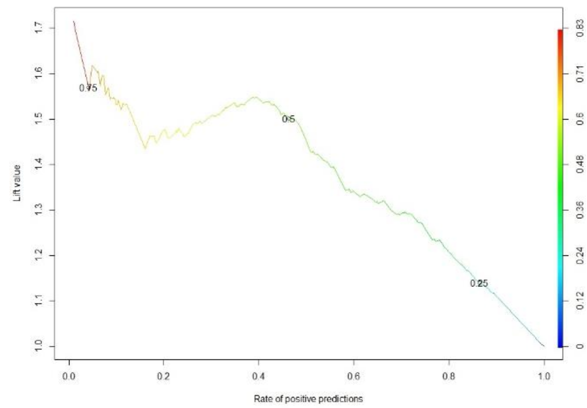


Boosted Tree: (Test Data)

AUC = 73.66508
Accuracy = 0.7358
Kappa = 0.4724
Sensitivity = 0.6937
Specificity = 0.7796







#Considering AUC as a metric for comparing different method, in the training phase the Random Forest is shown to be the most powerful method and Decision Tree, Boosted Tree and Elastic Net are in the next ranking in the order mentioned. In the testing phase, the rankings are Decision Tree, Boosted Tree, Random Forest and Elastic Net. As can be seen Elastic Net is the worst method. Random Forest is really good in the training phase but not in the testing.