Assignment Three

Muiz

18/11/2019

```
#Necessary Packages
library(caret,quietly = TRUE)
library(dplyr,quietly = TRUE)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(pROC,quietly = TRUE)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(kableExtra,quietly = TRUE)
library(ROracle,quietly = TRUE)
```

Introduction

This report focuses on making a k nearest neighbours model to predict whether the telemarketing's customers will default or not on a loan. KNN models were used to nalayse the given data and produce models that will work on both the split of train and test data sets.

```
#Data Preparation
set.seed(321)
drv <- dbDriver("Oracle")

## Refer to Oracle Database Net Services Administator's Guide for
## details on connect string specification.
host <- "oracle.vittl.it.bond.edu.au"
port <- 1521
sid <- "inft320"
connect.string <- paste(
  "(DESCRIPTION=",
  "(ADDRESS=(PROTOCOL=tcp)(HOST=", host, ")(PORT=", port, "))",
  "(CONNECT_DATA=(SID=", sid, ")))", sep = "")

## Use username/password authentication.
con <- dbConnect(drv, username = "A13599863", password = "A13599863",
```

```r
                 dbname = connect.string)

## Run a SQL statement by creating first a resultSet object.
rs <- dbSendQuery(con, "select * from brucedba.BankMarketing")
loandata <- fetch(rs)
```

```r
#Clean DataSet And Remove Unecessary Variable(s)
clean_loan <- select(loandata,-c(DURATION))
```

```r
#Convert Factors to Numeric
factornames <- c("AGE","PDAYS","PREVIOUS","EMP_VAR_RATE","CONS_PRICE_IDX","CONS_CONF_IDX","EURIBOR3M","

clean_loan[,factornames] <- lapply(factornames, function (x) as.numeric(as.character(clean_loan[,x])))
```

As these variables are defined as characters, we have to convert these factors to categorical to enable R to interept them correctly.

```r
#Convert Factors to Categorical
factornames1 <- c("JOB","MARITAL","EDUCATION","DEFAULTCREDIT","HOUSING","LOAN","CONTACT","MONTH","DAY_O

clean_loan[,factornames1] <- lapply(factornames1, function (x) as.factor(as.character(clean_loan[,x])))
```

```r
#Split the data into 75% train and 25% test
set.seed(234)
SplitIndex <- sample(x = c("Train", "Test"),size = 150,replace = TRUE, prob = c(0.75,0.25))
TrainData <- clean_loan[SplitIndex == "Train", ]
TestData <- clean_loan[SplitIndex == "Test", ]
```

```r
# Train a model with many k values
KnnModel <- train(
  form = Y~.,
  data = TrainData,
  method = 'knn',
  preProcess=c("center","scale"),
  tuneGrid=expand.grid(k=1:25),
  metric='Kappa',
  trControl=trainControl(
    method='repeatedcv',
    number=5,
    repeats=1))

TestPred <- predict(object= KnnModel, newdata = TestData, type = "raw")
mean(TestPred == TestData$Y)
```

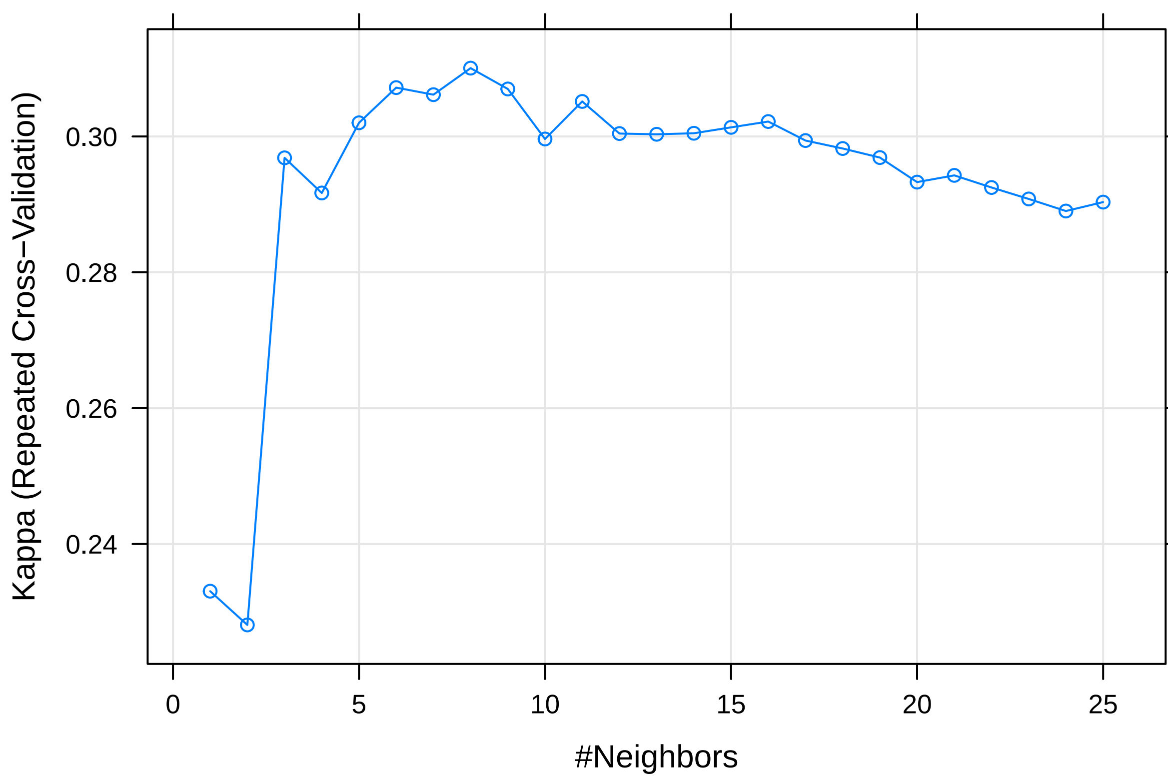
```
[1] 0.8925173
```

```r
confusionMatrix(data = TestPred, reference = TestData$Y)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  7357  695
       yes  220  241

               Accuracy : 0.8925
                 95% CI : (0.8857, 0.899)
```

```
         No Information Rate : 0.8901
         P-Value [Acc > NIR] : 0.2395

                       Kappa : 0.2938
      Mcnemar's Test P-Value : <2e-16

                 Sensitivity : 0.9710
                 Specificity : 0.2575
              Pos Pred Value : 0.9137
              Neg Pred Value : 0.5228
                  Prevalence : 0.8901
              Detection Rate : 0.8642
        Detection Prevalence : 0.9458
           Balanced Accuracy : 0.6142

            'Positive' Class : no
```

```r
plot(KnnModel)
```




```r
#Graph above shows accuracies of different Kappa values which consistently fluctuates
#Best Kappa value is 5 as it optimizes the model for all variables
```

```r
#Predictions on Train Data
TestPred <- predict(object= KnnModel, newdata = TrainData, type = "raw")
mean(TestPred == TrainData$Y)
```

```
[1] 0.905922
```

```r
confusionMatrix(data = TestPred, reference = TrainData$Y)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    no   yes
       no  28403  2506
       yes   568  1198

               Accuracy : 0.9059
                 95% CI : (0.9027, 0.9091)
    No Information Rate : 0.8866
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3936
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9804
            Specificity : 0.3234
         Pos Pred Value : 0.9189
         Neg Pred Value : 0.6784
             Prevalence : 0.8866
         Detection Rate : 0.8693
   Detection Prevalence : 0.9460
      Balanced Accuracy : 0.6519

       'Positive' Class : no
```

```r
#GLM - Model
KnnModel2 <- train(
  form = Y~JOB+EDUCATION+DEFAULTCREDIT+CONTACT+MONTH+DAY_OF_WEEK+CAMPAIGN+PDAYS+POUTCOME+EMP_VAR_RATE+C
  data = TrainData,
  method ='knn',
  preProcess=c("center","scale"),
  tuneGrid=expand.grid(k=1:25),
  metric='Kappa',
  trControl=trainControl(
    method='repeatedcv',
    number=5,
    repeats=1))

TestPred2 <- predict(object= KnnModel2, newdata = TestData, type = "raw")
mean(TestPred2 == TestData$Y)
```

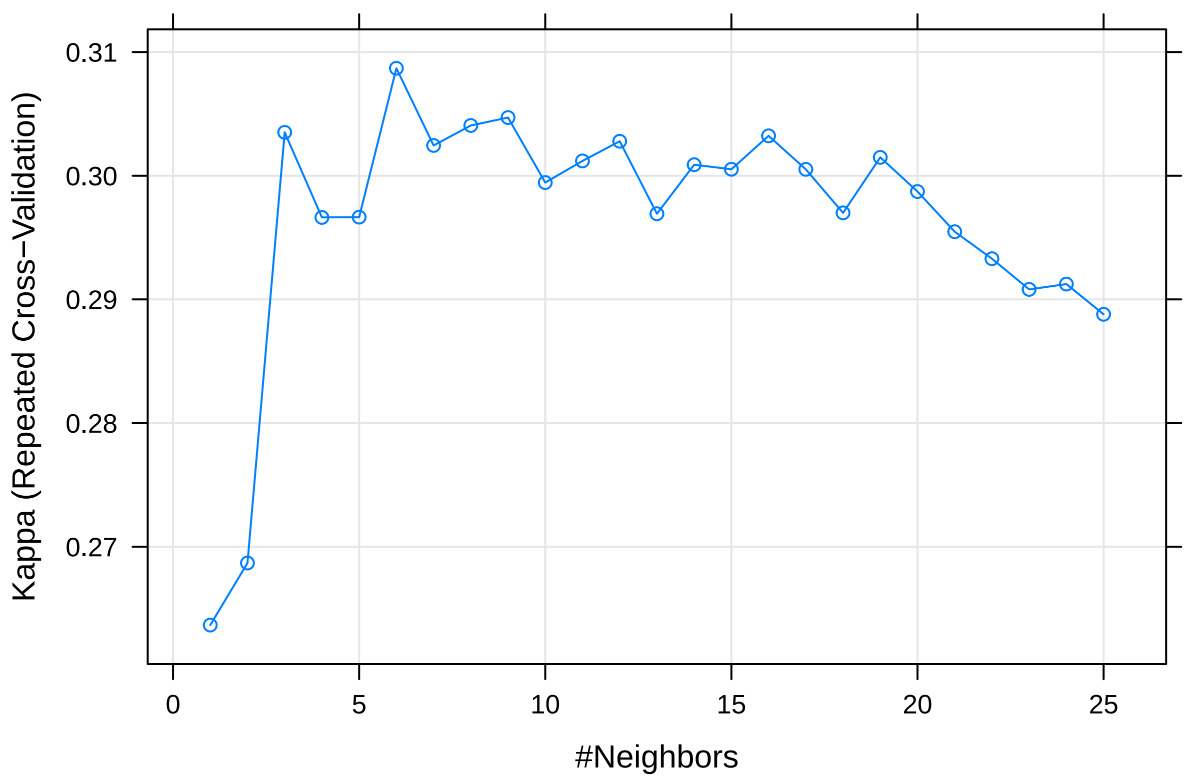
```
[1] 0.8938095
```

```r
confusionMatrix(data = TestPred2, reference = TestData$Y)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  7384  711
       yes  193  225
```

```
                Accuracy : 0.8938
                  95% CI : (0.8871, 0.9003)
     No Information Rate : 0.8901
     P-Value [Acc > NIR] : 0.1374

                   Kappa : 0.2837
  Mcnemar's Test P-Value : <2e-16

             Sensitivity : 0.9745
             Specificity : 0.2404
          Pos Pred Value : 0.9122
          Neg Pred Value : 0.5383
              Prevalence : 0.8901
          Detection Rate : 0.8674
    Detection Prevalence : 0.9509
       Balanced Accuracy : 0.6075

        'Positive' Class : no
```

```r
plot(KnnModel2)
```




```r
KnnModel3 <- KnnModel3 <- train(
  form = Y~NR_EMPLOYED+AGE+EURIBOR3M+CAMPAIGN+CONS_CONF_IDX+PDAYS+EMP_VAR_RATE,
  data = TrainData,
  method = 'knn',
```

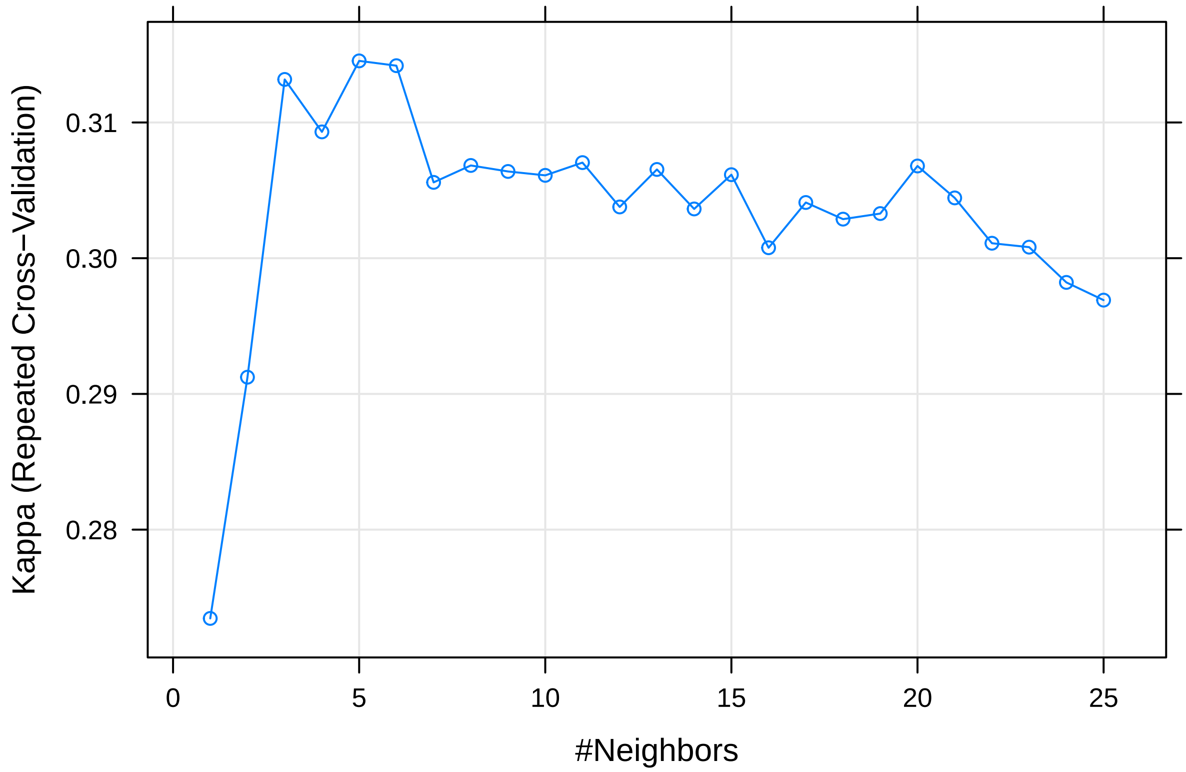
```r
  preProcess=c("center","scale"),
  tuneGrid=expand.grid(k=1:25),
  metric='Kappa',
  trControl=trainControl(
    method='repeatedcv',
    number=5,
    repeats=1))

TestPred3 <- predict(object= KnnModel3, newdata = TestData, type = "raw")
mean(TestPred3 == TestData$Y)
```

```
[1] 0.8913427
```

```r
confusionMatrix(data = TestPred3, reference = TestData$Y)
```

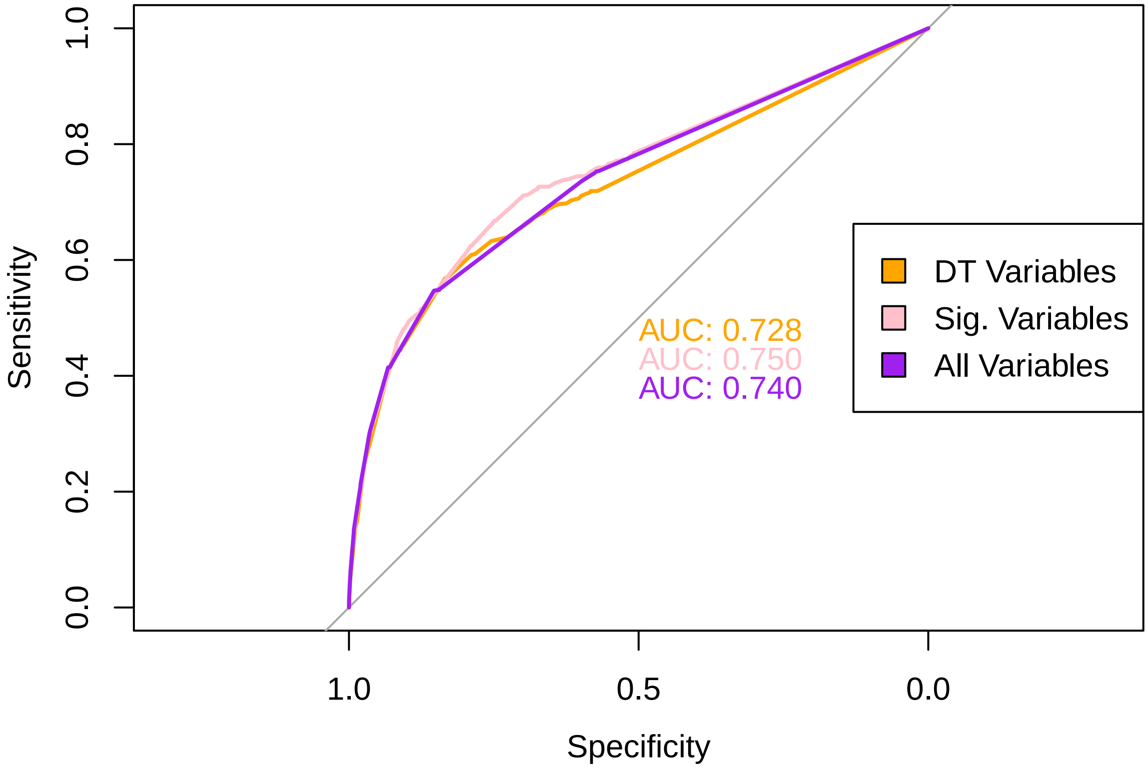
```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  7347  695
       yes  230  241

               Accuracy : 0.8913
                 95% CI : (0.8845, 0.8979)
    No Information Rate : 0.8901
    P-Value [Acc > NIR] : 0.3595

                  Kappa : 0.2903
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9696
            Specificity : 0.2575
         Pos Pred Value : 0.9136
         Neg Pred Value : 0.5117
             Prevalence : 0.8901
         Detection Rate : 0.8630
   Detection Prevalence : 0.9447
      Balanced Accuracy : 0.6136

       'Positive' Class : no
```

```r
plot(KnnModel3)
```



ROC Models

```r
#ROC Curve Plots
DTROC <- roc(response = TestData$Y, predictor = predict (KnnModel3,newdata = TestData,type="prob")[,2])
DTROC2 <- roc(response = TrainData$Y, predictor = predict (KnnModel3,newdata = TrainData,type="prob")[,
LOGROC <- roc(response = TestData$Y, predictor = predict (KnnModel2,newdata = TestData,type="prob")[,2]
ALLROC <- roc(response = TestData$Y, predictor = predict (KnnModel,newdata = TestData,type="prob")[,2])

plot.roc(DTROC, col = "orange",print.auc = T)
plot.roc(LOGROC, col = "pink",print.auc = T,add = TRUE, print.auc.x = 0.5 , print.auc.y = 0.45)
plot.roc(ALLROC, col = "purple",print.auc = T,add = TRUE, print.auc.x = 0.5 , print.auc.y = 0.40)
legend("right", legend = c("DT Variables","Sig. Variables","All Variables"),
       fill = c("orange","pink","purple","red"))
```




```
#Sensitivty ia model's ability to detect true positive
#Specificty is model's ability to discount a true negative
#Train data performs well as model was built off this model
#Model doesn't perform well on test data hence not a ribust model
#Model can't be used generally
#To improve we use our best model with the most important varaibles as below
```

Metric Calculations

```
#All Variables
KNNAllPredict <- predict(KnnModel,newdata = TrainData )
KNNAllPredictcf<-confusionMatrix(KNNAllPredict, TrainData$Y )
as.matrix(KNNAllPredictcf)
```

```
       no  yes
no  28401 2509
yes   570 1195
```

```
KNNAllPredict2 <- predict(KnnModel,newdata = TestData )
KNNAllPredict2cf<-confusionMatrix(KNNAllPredict2, TestData$Y )
as.matrix(KNNAllPredict2cf)
```

```
      no yes
no  7356 697
yes  221 239
```

```r
#The confusion matrix above clearly shows us that on the test data the model performed poorly and only f
```

```r
#Decision Tree Varaibles
KNNDTPredict <- predict(KnnModel3,newdata = TrainData )
KNNDTPredictcf<-confusionMatrix(KNNDTPredict, TrainData$Y)
as.matrix(KNNDTPredictcf)
```

```
       no  yes
no  28389 2295
yes   582 1409
```

```r
KNNDTPredict2 <- predict(KnnModel3,newdata = TestData)
KNNDTPredictcf2<-confusionMatrix(KNNDTPredict2, TestData$Y )
as.matrix(KNNDTPredictcf2)
```

```
      no yes
no  7352 695
yes  225 241
```

```r
#The above confusion matrix shows a vast improvement from the first model as it used only the important
```

```r
#GLM Variables
KNNGLMPredict <- predict(KnnModel2,newdata = TrainData )
KNNGLMPredictcf<-confusionMatrix(KNNGLMPredict, TrainData$Y )
as.matrix(KNNGLMPredictcf)
```

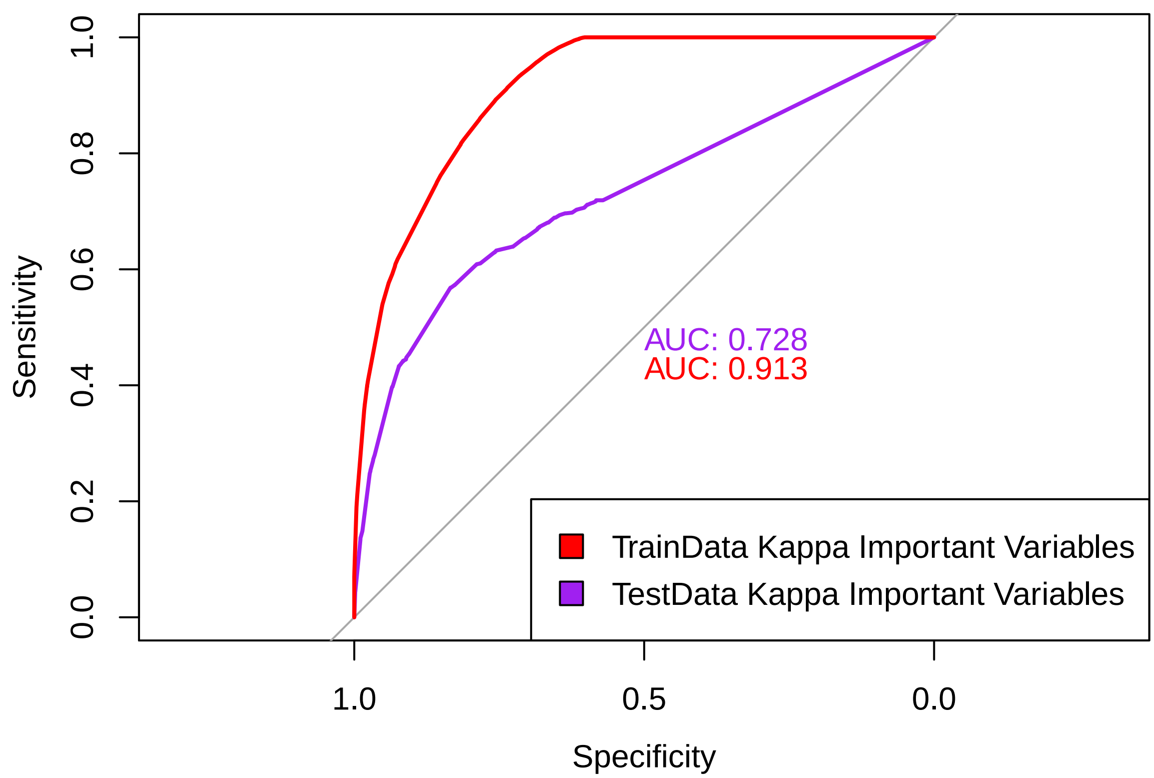
```
       no  yes
no  28381 2479
yes   590 1225
```

```r
KNNGLMPredict2 <- predict(KnnModel2,newdata = TestData )
KNNGLMPredictcf2<-confusionMatrix(KNNGLMPredict2, TestData$Y )
as.matrix(KNNGLMPredictcf2)
```

```
      no yes
no  7382 710
yes  195 226
```

```r
#The above confusion matrix shows a vast improvement from the first model but slightly worse than the s
```

```r
#Best ROC Curve
plot.roc(DTROC, col = "purple",print.auc = T, print.auc.x = 0.5 , print.auc.y = 0.5)
plot.roc(DTROC2, col = "red",print.auc = T,add = TRUE, print.auc.x = 0.5 , print.auc.y = 0.45)
legend("bottomright",legend = c("TrainData Kappa Important Variables","TestData Kappa Important Variabl
       fill = c("red","purple"))
```



The ROC graph measures how well the model performs in terms of specificty and sensitivity. Sensitivity measures how well the model correctly predicts that a person will deafult whilst specificity measures how well the model correctly predicts that a person won't default. However, there needs to be a balance between the two as an over use of one would be too much and reduce the other throwing it into an imbalance. The best models are those that bend out the furthest from the linear line. As seen in the graph above, the model with the test data performs worse than the model with train data. The model isn't good as it isn't robust hence not performing similar on the test and tain data sets. The model isn't very good as both the test and train data sets have different AUC's with the Train Data having an AUC of 93.5% and the Test Data having an AUC of 70.1%. The AUC (Area Under The Curve) is a metric for accuracy. However, the model's accuracy is not the best on foreign data and predicts "no" and the model should not be used.

Foreign Data Predictions

```
#New Data Frame Created To Use In Prediction Function Below

dummydata <- data.frame("JOB" = c("services","unemployed","blue-collar"),
                        "DEFAULTCREDIT" = c("no","no","yes"),
                        "MARITAL"=c("married","single","divorced"),
                        "HOUSING"=c("yes","yes","no"),
                        "CONTACT" = c("cellular" , "telephone" , "telephone"),
                        "MONTH" = c ("may","oct","aug"), "DAY_OF_WEEK" = c("tue","wed","mon"),
                        "CAMPAIGN" = c(1,1,1), "PDAYS" = c(1000, 999, 7),
                        "AGE"=c(42,26,78),
                        "LOAN"=c("yes","no","yes"),
                        "PREVIOUS"=c(0,1,0),
                        "EURIBOR3M"= c(1,1.5,0.9),
```

```r
                        "POUTCOME" = c("failure","nonexistent","success"),
                        "EMP_VAR_RATE" = c(1.1,-2.0,2.3),
                        "CONS_PRICE_IDX" = c(98.9444, 95.423,92.102),
                        "CONS_CONF_IDX" = c(-36.4, -47.5, -33.90),"NR_EMPLOYED" = c(5200, 5000, 6100),"
)

dummydata
```

```
##           JOB DEFAULTCREDIT  MARITAL HOUSING   CONTACT MONTH DAY_OF_WEEK
## 1    services            no  married     yes  cellular   may         tue
## 2  unemployed            no   single     yes telephone   oct         wed
## 3 blue-collar           yes divorced      no telephone   aug         mon
##   CAMPAIGN PDAYS AGE LOAN PREVIOUS EURIBOR3M    POUTCOME EMP_VAR_RATE
## 1        1  1000  42  yes        0       1.0     failure          1.1
## 2        1   999  26   no        1       1.5 nonexistent         -2.0
## 3        1     7  78  yes        0       0.9     success          2.3
##   CONS_PRICE_IDX CONS_CONF_IDX NR_EMPLOYED         EDUCATION   Y
## 1        98.9444         -36.4        5200          basic.9y  no
## 2        95.4230         -47.5        5000          basic.4y yes
## 3        92.1020         -33.9        6100 university.degree yes
```

```r
Predict <- function(foreigndata,model,displaytype){
 if(displaytype == "Class"){
   prob <- predict(object = model,newdata = foreigndata, type = "raw")
   return(prob)

 }
 else if (displaytype == "Prob")
 prob<- predict(object = model,newdata = foreigndata,type = "prob")
 prob[,2]
}


Predict(dummydata,KnnModel3,"Class")
```

```
## [1] no no no
## Levels: no yes
```

```r
#Tabulated Prediction Data
newdf <- data.frame("CAMPAIGN" = c(1,1,1), "PDAYS" = c(1000, 999, 7),
                        "AGE"=c(42,26,78),
                        "LOAN"=c("yes","no","yes"),
                        "PREVIOUS"=c(0,1,0),
                        "EURIBOR3M"= c(1,1.5,0.9),"Loan Status"=c("no","no","no"))
kable(newdf,align = "c")
```

CAMPAIGN	PDAYS	AGE	LOAN	PREVIOUS	EURIBOR3M	Loan.Status
1	1000	42	yes	0	1.0	no
1	999	26	no	1	1.5	no
1	7	78	yes	0	0.9	no

The table above shows a few of the variables used to make predictions on a customer's loan default prediction. The model failed to predict someone defaulting even with variables of that someone who would default.

Conclusion

The model is good aas it uses important variables when predicting customer default however the overrall model doesn't perform well as its accuracy is less than the baseline accuracy. Most of the customers are predicted to not default and fewer were predicted to default. This is not good for the business as the telemarketing company will be giving loans to too many people that will default on their loan. The model shouldn't be used and a more accurate model would be the decision tree model rather than the KNN model.