Objective: Develop a Classifier to predict whether a customer is going to open a deposit account.

Data: Bank Dataset (bank-test.csv; bank-train.csv)

See UCI Repository https://archive.ics.uci.edu/ml/datasets/Bank+Marketing (**with variable 11 – duration, excluded**) for more information

Part1: Build four separate classifiers, and compare their performance.

```
library(caret)
library(plyr)
library(dplyr)
library(C50)
library(kernlab)
library(RWeka)
trainingData <- read.csv("C:\\Users\\ludai\\Desktop\\BAN620\\Assignment3\\bank-training.csv", row.names = 1)
testData<-read.csv("C:\\Users\\ludai\\Desktop\\BAN620\\Assignment3\\bank-test.csv", row.names = 1)
nrow(trainingData)
nrow(testData)
prop.table(table(trainingData$y))
prop.table(table(testData$y))
```

```
[1] 3090
> nrow(testData)
[1] 1029
> prop.table(table(trainingData$y))

       no       yes
0.8902913 0.1097087
> prop.table(table(testData$y))

       no       yes
0.8911565 0.1088435
```

a) Decision Tree

```
TrainingParameters <- trainControl(method = "cv", number = 10, repeats = 5)
DecTreeModel <- caret::train(trainingData[,-20], trainingData$y,
               method = "C5.0",
               trControl= TrainingParameters,
               na.action = na.omit
               )
DecTreeModel
summary(DecTreeModel)
#Testing the Model
DTPredictions <-predict(DecTreeModel, testData, na.action = na.pass)
confusionMatrix(DTPredictions, testData$y)
confusionMatrix
```

```
> DecTreeModel
C5.0

3090 samples
  19 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2781, 2781, 2781, 2782, 2781, 2781, ...
Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa
  rules  FALSE    1      0.9019373  0.2603314
  rules  FALSE   10      0.8970840  0.2761122
  rules  FALSE   20      0.8980570  0.2905335
  rules   TRUE    1      0.8996720  0.2586340
  rules   TRUE   10      0.8974045  0.2706525
  rules   TRUE   20      0.8993494  0.3038061
  tree   FALSE    1      0.8996720  0.2647905
  tree   FALSE   10      0.8967541  0.2812278
  tree   FALSE   20      0.8980486  0.2909091
  tree    TRUE    1      0.8996720  0.2644463
  tree    TRUE   10      0.9003108  0.3024265
  tree    TRUE   20      0.8983712  0.3050800

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were trials = 1, model = rules and winnow = FALSE.
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  911  98
       yes   6  14

               Accuracy : 0.8989
                 95% CI : (0.8789, 0.9167)
    No Information Rate : 0.8912
    P-Value [Acc > NIR] : 0.2281

                  Kappa : 0.1852
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9935
            Specificity : 0.1250
         Pos Pred Value : 0.9029
         Neg Pred Value : 0.7000
             Prevalence : 0.8912
         Detection Rate : 0.8853
   Detection Prevalence : 0.9806
      Balanced Accuracy : 0.5592

       'Positive' Class : no
```

b) Naive Bayes

set.seed(100)
NBModel <- train(trainingData[,-20], trainingData$y, method = "nb",trControl= trainControl(method = "cv", number = 10, repeats = 5))
NBModel
NBPredictions <-predict(NBModel, testData)
confusionMatrix(NBPredictions, testData$y)

```
> NBModel
Naive Bayes

3090 samples
  19 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2782, 2781, 2781, 2781, 2781, 2781, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.8424239  0.3291233
   TRUE      0.8747759  0.3491600

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

```
> confusionMatrix(NBPredictions, testData$y)
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  871  71
       yes  46  41

               Accuracy : 0.8863
                 95% CI : (0.8653, 0.9051)
    No Information Rate : 0.8912
    P-Value [Acc > NIR] : 0.7121

                  Kappa : 0.3502
 Mcnemar's Test P-Value : 0.0265

            Sensitivity : 0.9498
            Specificity : 0.3661
         Pos Pred Value : 0.9246
         Neg Pred Value : 0.4713
             Prevalence : 0.8912
         Detection Rate : 0.8465
   Detection Prevalence : 0.9155
      Balanced Accuracy : 0.6580

       'Positive' Class : no
```

c) Suppor Vector Machines (SVM)

```
set.seed(120)
svm_model <- train(y~., data = trainingData,
  method = "svmPoly",
  trControl= trainControl(method = "cv", number = 10, repeats = 5),
  tuneGrid = data.frame(degree = 1,scale = 1,C = 1))
svm_model
SVMPredictions <-predict(svm_model, testData, na.action = na.pass)
confusionMatrix(SVMPredictions, testData$y)
```

```
> svm_model
Support Vector Machines with Polynomial Kernel

3090 samples
  19 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2781, 2780, 2781, 2781, 2782, 2781, ...
Resampling results:

  Accuracy   Kappa
  0.8986969  0.2777773

Tuning parameter 'degree' was held constant at a value of 1
Tuning parameter 'scale' was held constant at a value of
 1
Tuning parameter 'C' was held constant at a value of 1
```

```
> confusionMatrix(SVMPredictions, testData$y)
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  906  98
       yes  11  14

               Accuracy : 0.8941
                 95% CI : (0.8736, 0.9122)
    No Information Rate : 0.8912
    P-Value [Acc > NIR] : 0.406

                  Kappa : 0.1715
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9880
            Specificity : 0.1250
         Pos Pred Value : 0.9024
         Neg Pred Value : 0.5600
             Prevalence : 0.8912
         Detection Rate : 0.8805
   Detection Prevalence : 0.9757
      Balanced Accuracy : 0.5565

       'Positive' Class : no
```

d) Neural Network

```
nnmodel <- train(trainingData[,-20], trainingData$y, method = "nnet",
        trControl= trainControl(method = "cv", number = 10, repeats = 5))
nnmodel
nnetpredictions <-predict(nnmodel, testData, na.action = na.pass)
confusionMatrix(nnetpredictions, testData$y)
```

```
> nnmodel
Neural Network

3090 samples
  19 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 2781, 2781, 2782, 2781, 2781, 2780, ...
Resampling results across tuning parameters:

  size  decay  Accuracy   Kappa
  1     0e+00  0.8912598  0.0244378
  1     1e-04  0.8902921  0.0000000
  1     1e-01  0.9006481  0.3061129
  3     0e+00  0.8948197  0.1305268
  3     1e-04  0.8951465  0.0806496
  3     1e-01  0.9003203  0.3125602
  5     0e+00  0.8948228  0.0954000
  5     1e-04  0.8974118  0.1148964
  5     1e-01  0.8980559  0.3173886

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were size = 1 and decay = 0.1.
> 
```

```
> confusionMatrix(nnetpredictions, testData$y)
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  906  96
       yes  11  16

               Accuracy : 0.896
                 95% CI : (0.8757, 0.914)
    No Information Rate : 0.8912
    P-Value [Acc > NIR] : 0.33

                  Kappa : 0.1962
 Mcnemar's Test P-Value : 4.639e-16

            Sensitivity : 0.9880
            Specificity : 0.1429
         Pos Pred Value : 0.9042
         Neg Pred Value : 0.5926
             Prevalence : 0.8912
         Detection Rate : 0.8805
   Detection Prevalence : 0.9738
      Balanced Accuracy : 0.5654

       'Positive' Class : no
```

Summary: Evaluation Four Model

```
> t <- matrix(cbind(c(0.8989,0.9935,0.1250,0.9029),c(0.8863,0.9498,0.3661,0.9246),
+     c(0.8941,0.9880,0.1250,0.9024),c(0.8960,0.9880,0.1429,0.9042)),
+     nrow=4,dimnames=list(c('Accuracy','Sensitivity/Recall','Specificity','Precision'),
+     c('DecTree','NB','SVM','NN')))
> t
                   DecTree     NB    SVM     NN
Accuracy            0.8989 0.8863 0.8941 0.8960
Sensitivity/Recall  0.9935 0.9498 0.9880 0.9880
Specificity         0.1250 0.3661 0.1250 0.1429
Precision           0.9029 0.9246 0.9024 0.9042
>
```

Based on the above matrix, we found the Decision Tree Model has the highest accuracy and recall. But in this case, we hope the recall(sensitivity) is low since the positive class is ‘No’, which means a customer does not want to open a deposit account. The Naive Bayes Model has the lowest recall. Therefore, it is hard to say which model is better only from the above measures.

Part2. F Measure weight justification, F Measures, Best Model

```
> model = c("DecTree","NB","SVM","NN")
> recall = c(0.9935,0.9498,0.9880,0.9880)
> precision = c(0.9029,0.9246,0.9024,0.9042)
> fscore <- 2 * precision * recall / (precision + recall)
> fscore_table = data.frame(model,recall,precision,fscore)
> fscore_table
    model recall precision    fscore
1 DecTree 0.9935    0.9029 0.9460358
2      NB 0.9498    0.9246 0.9370306
3     SVM 0.9880    0.9024 0.9432620
4      NN 0.9880    0.9042 0.9442444
> beta=as.numeric(0.5)
> x=beta^2
> x
[1] 0.25
> fmeasure=((1+x)*precision*recall)/(x*precision+recall)
> fmeasure_table=data.frame(model,recall,precision,fmeasure)
> fmeasure_table
    model recall precision  fmeasure
1 DecTree 0.9935    0.9029 0.9196735
2      NB 0.9498    0.9246 0.9295325
3     SVM 0.9880    0.9024 0.9183125
4      NN 0.9880    0.9042 0.9198031
> |
```

```
> beta=as.numeric(2)
> x=beta^2
> fmeasure=((1+x)*precision*recall)/(x*precision+recall)
> fmeasure_table=data.frame(model,recall,precision,fmeasure)
> fmeasure_table
    model recall precision  fmeasure
1 DecTree 0.9935    0.9029 0.9739540
2      NB 0.9498    0.9246 0.9446507
3     SVM 0.9880    0.9024 0.9696050
4      NN 0.9880    0.9042 0.9700200
```

Precision can be thought of as a measure of exactness, whereas recall is a measure of completeness. In this case, the classification goal is to predict whether customers will subscribe to open deposit accounts. Here the positive class is 'no'. This means a customer does not subscribe to a deposit. So, it is important to choose a model with a low recall. In order to illustrate recall and precision for each model, we need to compute F-measure. The F measure is the harmonic mean of precision and recall. It gives equal weight to precision and recall. The Fβ measure is a weighted measure of precision and recall. It assigns β times as much weight to recall as to precision. I assigned β =0.5(which weights precision twice as much as recall), and β=2((which weights recall twice as much as precision) respectively. Based on the analysis for the above table, Naive Bayes method is the recommended classification

method as it contains lowest recall and F score and the highest precision in the F measure table and F_2 measure table. We do not hope a model that aggressively classifies a customer response as ‘no’, we want more customers to open a bank account.

Part3. Cost Sensitive Model

```
library(C50)
cmatrix <- cbind(c(0,1), c(10,0))
cmatrix
trainingData <- read.csv("C:\\Users\\ludai\\Desktop\\BAN620\\Assignment3\\bank-training.csv", row.names = 1)
testData<-read.csv("C:\\Users\\ludai\\Desktop\\BAN620\\Assignment3\\bank-test.csv", row.names = 1)
CostDTModel <- C5.0(y ~., data=trainingData,cost=cmatrix)
CostDTModel
summary(CostDTModel)
PredictionCostModel <-predict(CostDTModel, testData,na.action = na.pass)
confusionMatrix(PredictionCostModel, testData$y)
```

```
> CostDTModel

Call:
C5.0.formula(formula = y ~ ., data = trainingData, cost = cmatrix)

Classification Tree
Number of samples: 3090
Number of predictors: 19

Tree size: 61

Non-standard options: attempt to group attributes

Cost Matrix:
    no yes
no   0  10
yes  1   0
> 
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  605  39
       yes 312  73

               Accuracy : 0.6589
                 95% CI : (0.629, 0.6879)
    No Information Rate : 0.8912
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1505
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6598
            Specificity : 0.6518
         Pos Pred Value : 0.9394
         Neg Pred Value : 0.1896
             Prevalence : 0.8912
         Detection Rate : 0.5879
   Detection Prevalence : 0.6259
      Balanced Accuracy : 0.6558

       'Positive' Class : no
```

Profit=73*10(revenue)-(312+73) *1=345

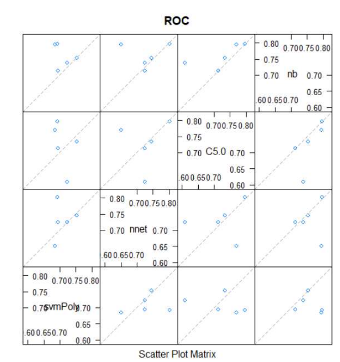
Based on cost matrix and confusion matrix, since the cost of a phone call is 1 unit and lost business is 10units, the revenue will be 73*10(True Negative), and the cost is 1*(312+73). The profit is 345.

Part4: Extra credit

a) Models Correlation

```
econtrol <- trainControl(method="cv", number=5, summaryFunction = twoClassSummary, savePredictions=TRUE,
classProbs=TRUE)
models <- caretList(y ~., data=trainingData,
           methodList=c("svmPoly", "nnet", "C5.0", "nb"),
           trControl = econtrol )
results <- resamples(models)
results$values
results$metrics
summary(results)
mcr <-modelCor(results)
mcr
```

```
> summary(results)

Call:
summary.resamples(object = results)

Models: svmPoly, nnet, C5.0, nb
Number of resamples: 5

ROC
             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
svmPoly 0.6849932 0.6912834 0.6942724 0.7091847 0.7228342 0.7525401    0
nnet    0.6497151 0.7238770 0.7242447 0.7288615 0.7450267 0.8014439    0
C5.0    0.6094652 0.7130218 0.7333155 0.7246515 0.7705699 0.7968850    0
nb      0.7129017 0.7381684 0.7523663 0.7589476 0.7939620 0.7973396    0

Sens
             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
svmPoly 0.9818512 0.9836364 0.9836364 0.9854611 0.9872727 0.9909091    0
nnet    0.9564428 0.9745455 0.9836364 0.9771067 0.9836364 0.9872727    0
C5.0    0.9509091 0.9818182 0.9836364 0.9792767 0.9891107 0.9909091    0
nb      1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    0

Spec
             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
svmPoly 0.1323529 0.2352941 0.2352941 0.2273047 0.2500000 0.2835821    0
nnet    0.1470588 0.1764706 0.2500000 0.2509219 0.3134328 0.3676471    0
C5.0    0.1323529 0.2058824 0.2089552 0.2359087 0.2352941 0.3970588    0
nb      0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000    0

> mcr <-modelCor(results)
> mcr
           svmPoly       nnet       C5.0          nb
svmPoly  1.0000000 0.21416392 -0.4204132 -0.34446113
nnet     0.2141639 1.00000000  0.1514213  0.02842688
C5.0    -0.4204132 0.15142135  1.0000000  0.69102423
nb      -0.3444611 0.02842688  0.6910242  1.00000000
>
```



Based on result of Models correlation, I find these four models have low correlation (<0.75) by using resample and modelCor functions. C5.0 and NB have the highest positive correlation; and nnet and NB have the least correlation.

b) Ensemble Methods

 # subset models to use
smallmodels <- c(models$svmPoly, models$C5.0, models$nb,models$nnet)
 # Create stacked models. Try rpart
enstackmodel <- caretStack(models, method = "rpart")
print(enstackmodel)
enstackmodel <- caretStack(models, method = "C5.0", metric="Sens",trControl = trainControl(number = 5, summaryFunction = twoClassSummary,classProbs = TRUE))
print(enstackmodel)
#Predict
enstackpredictions <-predict(enstackmodel, testData, na.action = na.omit)
 # Create confusion matrix
cmBank <-confusionMatrix(enstackpredictions, testData$y, mode="everything", positive = "no")
cmBank

```
> print(enstackmodel)
A rpart ensemble of 2 base models: svmPoly, nnet, C5.0, nb

Ensemble results:
CART

3090 samples
   4 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 3090, 3090, 3090, 3090, 3090, 3090, ..
Resampling results across tuning parameters:

  cp           Accuracy   Kappa
  0.006637168  0.8873470  0.2709128
  0.008849558  0.8900828  0.2786850
  0.058997050  0.8916057  0.1780602
```

```
> print(enstackmodel)
A C5.0 ensemble of 2 base models: svmPoly, nnet, C5.0, nb

Ensemble results:
C5.0

3090 samples
   4 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Bootstrapped (5 reps)
Summary of sample sizes: 3090, 3090, 3090, 3090, 3090
Resampling results across tuning parameters:

  model  winnow  trials  ROC        Sens       Spec
  rules  FALSE    1      0.6190529  0.9845063  0.1830710
  rules  FALSE   10      0.7421883  0.9751225  0.2272290
  rules  FALSE   20      0.7471149  0.9753644  0.2486935
  rules   TRUE    1      0.6191909  0.9787190  0.2504490
  rules   TRUE   10      0.6646853  0.9752259  0.2595383
  rules   TRUE   20      0.6646853  0.9752259  0.2595383
  tree   FALSE    1      0.7034884  0.9757112  0.2082679
  tree   FALSE   10      0.7436723  0.9728238  0.2472836
  tree   FALSE   20      0.7473011  0.9724399  0.2438993
  tree    TRUE    1      0.6641840  0.9775178  0.2519994
  tree    TRUE   10      0.6906682  0.9771411  0.2489459
  tree    TRUE   20      0.6906682  0.9771411  0.2489459
```

```
> cmBank
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  906  98
       yes  11  14

               Accuracy : 0.8941
                 95% CI : (0.8736, 0.9122)
    No Information Rate : 0.8912
    P-Value [Acc > NIR] : 0.406

                  Kappa : 0.1715
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9880
            Specificity : 0.1250
         Pos Pred Value : 0.9024
         Neg Pred Value : 0.5600
              Precision : 0.9024
                 Recall : 0.9880
                     F1 : 0.9433
             Prevalence : 0.8912
         Detection Rate : 0.8805
   Detection Prevalence : 0.9757
      Balanced Accuracy : 0.5565

       'Positive' Class : no
```

econtrol <- trainControl(method="cv", number=5, summaryFunction = twoClassSummary, savePredictions=TRUE, classProbs=TRUE)

models <- caretList(y ~., data=trainingData,
    methodList=c("nnet", "nb"),
    trControl = econtrol)
smallmodels <- c(models$nb,models$nnet)
enstackmodel <- caretStack(models, method = "rpart")

print(enstackmodel)
enstackmodel <- caretStack(models, method = "C5.0", metric="Sens",trControl = trainControl(number = 5, summaryFunction = twoClassSummary,classProbs = TRUE))
print(enstackmodel)
enstackpredictions <-predict(enstackmodel, testData, na.action = na.omit)
cmBank <-confusionMatrix(enstackpredictions, testData$y, mode="everything", positive = "no")
cmBank

```
A C5.0 ensemble of 2 base models: nnet, nb

Ensemble results:
C5.0

3090 samples
   2 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Bootstrapped (5 reps)
Summary of sample sizes: 3090, 3090, 3090, 3090, 3090
Resampling results across tuning parameters:
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  905  95
       yes  12  17

               Accuracy : 0.896
                 95% CI : (0.8757, 0.914)
    No Information Rate : 0.8912
    P-Value [Acc > NIR] : 0.33

                  Kappa : 0.2056
 Mcnemar's Test P-Value : 2.241e-15

            Sensitivity : 0.9869
            Specificity : 0.1518
         Pos Pred Value : 0.9050
         Neg Pred Value : 0.5862
              Precision : 0.9050
                 Recall : 0.9869
                     F1 : 0.9442
             Prevalence : 0.8912
         Detection Rate : 0.8795
   Detection Prevalence : 0.9718
      Balanced Accuracy : 0.5693

       'Positive' Class : no
```

Ensemble model by using all four models:
Accuracy: 0.8941
Sensitivity: 0.9880
Specificity: 0.1250
Precision: 0.9024

Ensemble model by using least correlation nnet and nb models:
Accuracy: 0.8960
Sensitivity: 0.9869
Specificity: 0.1518
Precision: 0.9050

In ensemble method, we can combine predictors to improve classification accuracy. In this case, I used model stacking to create ensemble model. I find the accuracy is improved by using least correlation models.

C) Develop an ensemble model that returns a profit greater than 425 on the test data

```
> # Create models with sensitivity as optimization metric instead of accuracy
> ensmodel2 <- caretEnsemble(models,
+                            metric = "Sens",
+                            trControl = trainControl(number = 2, summaryFunction = twoClassSummary,classProbs = TRUE)
+ )
> summary(ensmodel2)
The following models were ensembled: svmPoly, nnet, C5.0, nb
They were weighted:
-2.9794 -0.2833 3.7614 2.4072 -24190.0965
The resulting Sens is: 0.9895
The fit for each individual model on the Sens is:
  method      Sens       SensSD
 svmPoly 0.9986372 0.0009085423
    nnet 1.0000000 0.0000000000
    C5.0 0.9858248 0.0043332664
      nb 1.0000000 0.0000000000
> enstackpredictions <-predict(ensmodel2, testData, na.action = na.omit)
> cmBank2 <-confusionMatrix(enstackpredictions, testData$y, mode="everything", positive = "no")
> cmBank2
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  894  83
       yes  23  29

               Accuracy : 0.897
                 95% CI : (0.8768, 0.9149)
    No Information Rate : 0.8912
    P-Value [Acc > NIR] : 0.2941

                  Kappa : 0.3057
 Mcnemar's Test P-Value : 1.001e-08

            Sensitivity : 0.9749
            Specificity : 0.2589
         Pos Pred Value : 0.9150
         Neg Pred Value : 0.5577
              Precision : 0.9150
                 Recall : 0.9749
                     F1 : 0.9440
             Prevalence : 0.8912
         Detection Rate : 0.8688
   Detection Prevalence : 0.9495
```

```
> CostDTModel

Call:
C5.0.formula(formula = y ~ ., data = trainingData, cost = cmatrix)

Classification Tree
Number of samples: 3090
Number of predictors: 19

Tree size: 61

Non-standard options: attempt to group attributes

Cost Matrix:
    no yes
no   0  10
yes  1   0
```

```
 Confusion Matrix and Statistics

            Reference
 Prediction no yes
        no 605 31
        yes 312 81
```

Profit= 81*10-(312+81)*1=417

Ensemble method can improve model accuracy by using bagging, booting, and stacking. In this case, I used resample, stack models to improve models. I find the profit increase. (no more than 425, I did not find reasons.)