Ensemble Learning (Analytics Vidhya)

Anirudh Pillai

March 31, 2017

Load Libraries

```
# Load the required library
library('caret')
library('RANN')
# Set seed to Random
set.seed(1)
```

Structure of Data

```
setwd('C:/Users/aanirudh/Downloads')
# Load the Dataset
data<-read.csv('loan.csv')

#Structure of Data
str(data)
```

```
## 'data.frame':    614 obs. of  13 variables:
##  $ Loan_ID          : Factor w/ 614 levels "LP001002","LP001003",..: 1 2 3
4 5 6 7 8 9 10 ...
##  $ Gender           : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 3
3 3 3 ...
##  $ Married          : Factor w/ 3 levels "","No","Yes": 2 3 3 3 2 3 3 3 3
3 ...
##  $ Dependents       : Factor w/ 5 levels "","0","1","2",..: 2 3 2 2 2 4 2
5 4 3 ...
##  $ Education        : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1
2 1 1 2 1 1 1 ...
##  $ Self_Employed    : Factor w/ 3 levels "","No","Yes": 2 2 3 2 2 3 2 2 2
2 ...
##  $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006
12841 ...
##  $ CoapplicantIncome: num  0 1508 0 2358 0 ...
##  $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
##  $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
##  $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
##  $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3
3 3 2 3 2 ...
##  $ Loan_Status      : Factor w/ 2 levels "N","Y": 2 1 2 2 2 2 2 1 2 1 ...
```

Check Missing Values

```
sum(is.na(data))
```

```
## [1] 86

#Imputing missing values using median
preProcValues <- preProcess(data, method =
c("medianImpute","center","scale"))
data_preprocessed <- predict(preProcValues, data)
sum(is.na(data_preprocessed))

## [1] 0

index <- createDataPartition(data_preprocessed$Loan_Status, p =0.75, list =
FALSE)
trainset <- data_preprocessed[index,]
testset <- data_preprocessed[-index,]

# Defining training controls for multiple models

fitControl <- trainControl(method = "cv",number = 5,savePredictions =
'final',classProbs = T)

#Define predictors and outcome
predictors<-c("Credit_History", "LoanAmount", "Loan_Amount_Term",
"ApplicantIncome",
              "CoapplicantIncome")
outcomeName <- 'Loan_Status'
```

Random Forest

```
#Training the random forest model
model_rf <- train(trainset[,predictors],trainset[,outcomeName], method = 'rf'
,trControl = fitControl, tuneLength = 3)
#Predicting using random forest model
testset$pred_rf <- predict(object = model_rf , testset[,predictors])

confusionMatrix(testset$Loan_Status, testset$pred_rf)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 29 19
##          Y  9 96
##
##                Accuracy : 0.817
##                  95% CI : (0.7465, 0.8748)
##     No Information Rate : 0.7516
##     P-Value [Acc > NIR] : 0.03458
##
##                   Kappa : 0.5495
##  Mcnemar's Test P-Value : 0.08897
##
```

```
##             Sensitivity : 0.7632
##             Specificity : 0.8348
##          Pos Pred Value : 0.6042
##          Neg Pred Value : 0.9143
##              Prevalence : 0.2484
##          Detection Rate : 0.1895
##    Detection Prevalence : 0.3137
##       Balanced Accuracy : 0.7990
##
##        'Positive' Class : N
##
```

K- Nearest Neighbor

```
model_knn <- train(trainset[,predictors], trainset[,outcomeName], method =
'knn', trControl = fitControl, tuneLength = 3)
#Predicting using KNN
testset$pred_knn <- predict(object = model_knn, testset[,predictors])
#Check confusion matrix of the KNN Model
confusionMatrix(testset$Loan_Status, testset$pred_knn)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  29  19
##          Y   2 103
##
##                Accuracy : 0.8627
##                  95% CI : (0.7979, 0.913)
##     No Information Rate : 0.7974
##     P-Value [Acc > NIR] : 0.0241694
##
##                   Kappa : 0.6473
##  Mcnemar's Test P-Value : 0.0004803
##
##             Sensitivity : 0.9355
##             Specificity : 0.8443
##          Pos Pred Value : 0.6042
##          Neg Pred Value : 0.9810
##              Prevalence : 0.2026
##          Detection Rate : 0.1895
##    Detection Prevalence : 0.3137
##       Balanced Accuracy : 0.8899
##
##        'Positive' Class : N
##
```

Logistic Regression

```
#Let's try Logistic regression model
model_lr <- train(trainset[,predictors], trainset[,outcomeName], method =
'glm', trControl = fitControl, tuneLength = 3)
#Predicitng using Logistic Regression
testset$pred_lr <- predict(object = model_lr, testset[,predictors])

confusionMatrix(testset$Loan_Status, testset$pred_lr)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  28  20
##          Y   2 103
##
##                Accuracy : 0.8562
##                  95% CI : (0.7904, 0.9076)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.0594340
##
##                   Kappa : 0.6282
##  Mcnemar's Test P-Value : 0.0002896
##
##             Sensitivity : 0.9333
##             Specificity : 0.8374
##          Pos Pred Value : 0.5833
##          Neg Pred Value : 0.9810
##              Prevalence : 0.1961
##          Detection Rate : 0.1830
##    Detection Prevalence : 0.3137
##       Balanced Accuracy : 0.8854
##
##        'Positive' Class : N
##
```

Classification based on Average

```
#Averaging

testset$pred_rf_prob<-predict(object =
model_rf,testset[,predictors],type='prob')
testset$pred_knn_prob <- predict(object = model_knn, testset[,predictors],
type = 'prob')
testset$pred_lr_prob <- predict(object = model_lr, testset[,predictors], type
= 'prob')

testset$pred_avg<-
(testset$pred_rf_prob$Y+testset$pred_knn_prob$Y+testset$pred_lr_prob$Y)/3
#Splitting into binary classes at 0.5
testset$pred_avg<-as.factor(ifelse(testset$pred_avg>0.5,'Y','N'))
```

```
#CONFUSION Matrix
confusionMatrix(testset$Loan_Status, testset$pred_avg)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  28  20
##          Y   2 103
##
##                Accuracy : 0.8562
##                  95% CI : (0.7904, 0.9076)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.0594340
##
##                   Kappa : 0.6282
##  Mcnemar's Test P-Value : 0.0002896
##
##             Sensitivity : 0.9333
##             Specificity : 0.8374
##          Pos Pred Value : 0.5833
##          Neg Pred Value : 0.9810
##              Prevalence : 0.1961
##          Detection Rate : 0.1830
##    Detection Prevalence : 0.3137
##       Balanced Accuracy : 0.8854
##
##        'Positive' Class : N
##
```

Classification based on Majoirty Votes

```
# Majority Voting

testset$pred_majority <- as.factor(ifelse(testset$pred_rf == 'Y' &
testset$pred_knn =='Y', 'Y',
                                          ifelse(testset$pred_rf == 'Y' &
testset$pred_lr == 'Y', 'Y',
                                                 ifelse(testset$pred_knn ==
'Y' & testset$pred_lr == 'Y', 'Y','N'))))
confusionMatrix(testset$Loan_Status, testset$pred_majority)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  28  20
##          Y   2 103
##
##                Accuracy : 0.8562
```

```
##                  95% CI : (0.7904, 0.9076)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.0594340
##
##                   Kappa : 0.6282
##  Mcnemar's Test P-Value : 0.0002896
##
##             Sensitivity : 0.9333
##             Specificity : 0.8374
##          Pos Pred Value : 0.5833
##          Neg Pred Value : 0.9810
##              Prevalence : 0.1961
##          Detection Rate : 0.1830
##    Detection Prevalence : 0.3137
##       Balanced Accuracy : 0.8854
##
##        'Positive' Class : N
##
```

Weighted Average Classifier

```
#Weighted Average
testset$pred_weighted_avg <- (testset$pred_knn_prob$Y * 0.25) +
(testset$pred_rf_prob$Y * 0.25) + (testset$pred_lr_prob$Y * 0.5)
#Splitting into binary classes at 0.5
testset$pred_weighted_avg<-
as.factor(ifelse(testset$pred_weighted_avg>0.5,'Y','N'))

confusionMatrix(testset$Loan_Status, testset$pred_weighted_avg)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   N   Y
##          N  28  20
##          Y   2 103
##
##                Accuracy : 0.8562
##                  95% CI : (0.7904, 0.9076)
##     No Information Rate : 0.8039
##     P-Value [Acc > NIR] : 0.0594340
##
##                   Kappa : 0.6282
##  Mcnemar's Test P-Value : 0.0002896
##
##             Sensitivity : 0.9333
##             Specificity : 0.8374
##          Pos Pred Value : 0.5833
##          Neg Pred Value : 0.9810
##              Prevalence : 0.1961
```

```
##          Detection Rate : 0.1830
##    Detection Prevalence : 0.3137
##       Balanced Accuracy : 0.8854
##
##        'Positive' Class : N
##
```