```
> ##############################################
> # Statistical Modelling & Machine Learning #
> #             R Example5                     #
> ##############################################
>
> options(warn = -1)  # Turn off warning message
>
> # Data generation
> install.packages('mvtnorm')
> library(mvtnorm)
>
> set.seed(10)
> n1 = 500; n2 = 50
> mean1 = c(6,6)
> mean2 = c(4,5)
> sig1 = matrix(c(2,0,0,2),2,2)
> sig2 = matrix(c(2,-0.8,-0.8,2),2,2)
>
> x = rbind(rmvnorm(n1,mean1,sig1),rmvnorm(n2,mean2,sig2))
> y = as.factor(c(rep(0,n1),rep(1,n2)))
> train = data.frame(y,x)
>
> x = rbind(rmvnorm(n1,mean1,sig1),rmvnorm(n2,mean2,sig2))
> y = as.factor(c(rep(0,n1),rep(1,n2)))
> test = data.frame(y,x)
>
>
> # Scatter plot for the training data
> plot(cbind(train$X1,train$X2),
+     col=(3-as.numeric(train$y)),xlab='X1',ylab='X2')
>
> # Classifier: Logistic regression
> fit = glm(y~., family=binomial, data=train)
> phat.test = predict(fit, test, type='response')
> yhat.test = ifelse(phat.test > 0.5, 1, 0)
>
> cm  = table(true = test$y, predict=yhat.test)
> cm
    predict
true   0   1
   0 492   8
   1  42   8
>
>
> misclass = function(cm) 1 - sum(diag(cm))/sum(cm)
> # cm: confusion matrix (higher value = positive class)
> fmeasure = function(cm)
+ {
+   TPR = cm[2,2]/sum(cm[2,])
+   PPV = cm[2,2]/sum(cm[,2])
+   return((2*TPR*PPV)/(TPR + PPV))
+ }
>
> # Test misclassification rate
> misclass(cm)
[1] 0.09090909
>
> # Test F-measure
> fmeasure(cm)
[1] 0.2424242
>
> # ROC curve ---------------------------------------
> install.packages('pROC')
```

```
> library(pROC)
>
> phat.tr = predict(fit, train, type='response')
>
> lr.roc = roc(train$y ~ phat.tr)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> plot(lr.roc)
>
> # AUC
> auc(lr.roc)
Area under the curve: 0.8943
>
> ########## Alternate cut-off (using training dataset) ##########
>   # Fird the closest point on ROC curve to (1,1).
> th = coords(lr.roc,x='best', best.method = 'closest.topleft')
> th
   threshold specificity sensitivity
1 0.06872988       0.772         0.9
>
> # Evaluation for the new cut-off
> yhat.test1 = ifelse(phat.test > th$threshold, 1, 0)
>
> cm1 = table(true = test$y, predict=yhat.test1)
> cm1
    predict
true   0   1
   0 380 120
   1  10  40
>
> # Test misclassification rate
> misclass(cm1)
[1] 0.2363636
>
> # Test F-measure
> fmeasure(cm1)
[1] 0.3809524
>
> ################### Adjusting prior prob. ####################
> library(MASS)
>
> fit = lda(y~., data=train)
> yhat.te = predict(fit,test)$class
>
> cm = table(true = test$y, predict=yhat.te)
> cm
    predict
true   0   1
   0 492   8
   1  41   9
>
> # Test misclassification rate
> misclass(cm)
[1] 0.08909091
>
> # Test F-measure
> fmeasure(cm)
[1] 0.2686567
>
> # Adjust prior prob.
> fit1 = lda(train$y, x = as.matrix(train[,-1]), prior=c(0.6,0.4))
> yhat.te1 = predict(fit1 ,as.matrix(test[,-1]))$class
>
```

```
> cm1 = table(true = test$y, predict=yhat.te1)
> cm1
    predict
true   0   1
   0 427  73
   1  17  33
>
> # Test misclassification rate
> misclass(cm1)
[1] 0.1636364
>
> # Test F-measure
> fmeasure(cm1)
[1] 0.4230769
>
> ###################### Sampling methods ####################
> install.packages('caret')
> library(caret)
>
> # SVM model from the original imbalaned data
> install.packages('e1071')
> library(e1071)
>
> cv.fit = tune(svm, y~., data=train, kernel='radial',
+             ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
+                         gamma=c(0.01,0.1,0.5,1,2,3,4)))
> summary(cv.fit)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
   10   0.1

- best performance: 0.07636364

- Detailed performance results:
    cost gamma     error dispersion
1  1e-03  0.01 0.09090909 0.03428397
2  1e-02  0.01 0.09090909 0.03428397
3  1e-01  0.01 0.09090909 0.03428397
4  1e+00  0.01 0.09090909 0.03428397
5  5e+00  0.01 0.09090909 0.03428397
6  1e+01  0.01 0.09090909 0.03736008
7  1e+02  0.01 0.08000000 0.03554637
8  1e-03  0.10 0.09090909 0.03428397
9  1e-02  0.10 0.09090909 0.03428397
10 1e-01  0.10 0.09090909 0.03428397
11 1e+00  0.10 0.09090909 0.03736008
12 5e+00  0.10 0.07818182 0.03325059
13 1e+01  0.10 0.07636364 0.02944232
14 1e+02  0.10 0.07818182 0.02717153
15 1e-03  0.50 0.09090909 0.03428397
16 1e-02  0.50 0.09090909 0.03428397
17 1e-01  0.50 0.09090909 0.03428397
18 1e+00  0.50 0.07636364 0.02944232
19 5e+00  0.50 0.08000000 0.03113996
20 1e+01  0.50 0.08363636 0.03229797
21 1e+02  0.50 0.08727273 0.02816715
22 1e-03  1.00 0.09090909 0.03428397
23 1e-02  1.00 0.09090909 0.03428397
```

```
24 1e-01  1.00 0.09090909 0.03428397
25 1e+00  1.00 0.08000000 0.02868402
26 5e+00  1.00 0.08909091 0.03024236
27 1e+01  1.00 0.08727273 0.03066451
28 1e+02  1.00 0.08363636 0.02868402
29 1e-03  2.00 0.09090909 0.03428397
30 1e-02  2.00 0.09090909 0.03428397
31 1e-01  2.00 0.09090909 0.03428397
32 1e+00  2.00 0.08363636 0.02993719
33 5e+00  2.00 0.08727273 0.02542567
34 1e+01  2.00 0.08909091 0.02339425
35 1e+02  2.00 0.09272727 0.02339425
36 1e-03  3.00 0.09090909 0.03428397
37 1e-02  3.00 0.09090909 0.03428397
38 1e-01  3.00 0.09090909 0.03428397
39 1e+00  3.00 0.08727273 0.03297326
40 5e+00  3.00 0.09272727 0.02339425
41 1e+01  3.00 0.09454545 0.02393748
42 1e+02  3.00 0.10545455 0.02816715
43 1e-03  4.00 0.09090909 0.03428397
44 1e-02  4.00 0.09090909 0.03428397
45 1e-01  4.00 0.09090909 0.03428397
46 1e+00  4.00 0.08909091 0.03476274
47 5e+00  4.00 0.09454545 0.02393748
48 1e+01  4.00 0.09818182 0.02454358
49 1e+02  4.00 0.10545455 0.03716293
>
> # SVM with the best parameters
> best.fit = cv.fit$best.model
>
> # Prediction for test data
> yhat.te = predict(best.fit, test)
> cm = table(true = test$y, predict=yhat.te)
>
> misclass(cm)
[1] 0.08545455
>
> fmeasure(cm)
[1] 0.2539683
>
>
> # Upsampling ----------------------------------------------------
> # Sampling with replacement from the small class.
> set.seed(10)
> uptrain = upSample(x = train[,-1], y=train$y, yname='y')
>
> dim(uptrain)
[1] 1000   3
>
> table(uptrain$y)

  0   1
500 500
>
> # Scatter plot for the upsampled training data
> plot(cbind(uptrain$X1,uptrain$X2),
+     col=(3-as.numeric(uptrain$y)),xlab='X1',ylab='X2')
>
>
> # SVM for upsampled data.
> set.seed(10)
> cv.fit1 = tune(svm, y~., data=uptrain, kernel='radial',
+             ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
```

```
+                         gamma=c(0.01,0.1,0.5,1,2,3,4)))
> summary(cv.fit1)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  100     4

- best performance: 0.098

- Detailed performance results:
    cost gamma error dispersion
1  1e-03  0.01 0.416 0.17494126
2  1e-02  0.01 0.416 0.17494126
3  1e-01  0.01 0.177 0.03433495
4  1e+00  0.01 0.177 0.03335000
5  5e+00  0.01 0.184 0.03687818
6  1e+01  0.01 0.185 0.03205897
7  1e+02  0.01 0.173 0.03335000
8  1e-03  0.10 0.410 0.18312109
9  1e-02  0.10 0.172 0.03190263
10 1e-01  0.10 0.175 0.02677063
11 1e+00  0.10 0.167 0.02945807
12 5e+00  0.10 0.177 0.02907844
13 1e+01  0.10 0.183 0.03301515
14 1e+02  0.10 0.169 0.02330951
15 1e-03  0.50 0.408 0.18593906
16 1e-02  0.50 0.176 0.02913570
17 1e-01  0.50 0.169 0.02424413
18 1e+00  0.50 0.164 0.02221111
19 5e+00  0.50 0.162 0.01686548
20 1e+01  0.50 0.165 0.02121320
21 1e+02  0.50 0.155 0.02415229
22 1e-03  1.00 0.408 0.18504054
23 1e-02  1.00 0.184 0.02633122
24 1e-01  1.00 0.173 0.02110819
25 1e+00  1.00 0.161 0.01911951
26 5e+00  1.00 0.153 0.02162817
27 1e+01  1.00 0.150 0.01825742
28 1e+02  1.00 0.140 0.02708013
29 1e-03  2.00 0.408 0.18540047
30 1e-02  2.00 0.230 0.05537749
31 1e-01  2.00 0.165 0.01433721
32 1e+00  2.00 0.157 0.02263233
33 5e+00  2.00 0.137 0.02451757
34 1e+01  2.00 0.138 0.02740641
35 1e+02  2.00 0.118 0.02573368
36 1e-03  3.00 0.402 0.19286149
37 1e-02  3.00 0.332 0.14335659
38 1e-01  3.00 0.161 0.01728840
39 1e+00  3.00 0.147 0.02406011
40 5e+00  3.00 0.134 0.03134042
41 1e+01  3.00 0.121 0.02514403
42 1e+02  3.00 0.109 0.02846050
43 1e-03  4.00 0.400 0.19527758
44 1e-02  4.00 0.398 0.19401031
45 1e-01  4.00 0.157 0.02359378
46 1e+00  4.00 0.140 0.03055050
47 5e+00  4.00 0.121 0.02514403
48 1e+01  4.00 0.114 0.02913570
```

```
49 1e+02  4.00 0.098 0.02529822
> 
> # SVM with the best parameters
> best.fit1 = cv.fit1$best.model
> 
> # Prediction for test data
> yhat.te1 = predict(best.fit1, test)
> cm1 = table(true = test$y, predict=yhat.te1)
> 
> misclass(cm1)
[1] 0.2163636
> 
> fmeasure(cm1)
[1] 0.2699387
> 
> 
> # Dwonsampling ------------------------------------------
> # Randomly remove obs. in the large class.
> set.seed(1)
> dntrain = downSample(x = train[,-1], y=train$y, yname='y')
> 
> dim(dntrain)
[1] 100   3
> 
> table(dntrain$y)

 0  1 
50 50 
> 
> # Scatter plot for the downsampled training data
> plot(cbind(dntrain$X1,dntrain$X2),
+     col=(3-as.numeric(dntrain$y)),xlab='X1',ylab='X2')
> 
> 
> # SVM for downsampled data.
> set.seed(10)
> cv.fit2 = tune(svm, y~., data=dntrain, kernel='radial',
+              ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
+                          gamma=c(0.01,0.1,0.5,1,2,3,4)))
> summary(cv.fit2)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    5   0.1

- best performance: 0.2

- Detailed performance results:
    cost gamma error dispersion
1  1e-03  0.01  0.59  0.1523884
2  1e-02  0.01  0.59  0.1523884
3  1e-01  0.01  0.59  0.1523884
4  1e+00  0.01  0.23  0.2213594
5  5e+00  0.01  0.23  0.2213594
6  1e+01  0.01  0.22  0.2097618
7  1e+02  0.01  0.21  0.2183270
8  1e-03  0.10  0.57  0.1888562
9  1e-02  0.10  0.57  0.1888562
10 1e-01  0.10  0.22  0.2299758
```

```
11 1e+00  0.10  0.21  0.2183270
12 5e+00  0.10  0.20  0.2108185
13 1e+01  0.10  0.20  0.2108185
14 1e+02  0.10  0.21  0.2078995
15 1e-03  0.50  0.57  0.1888562
16 1e-02  0.50  0.57  0.1888562
17 1e-01  0.50  0.22  0.2299758
18 1e+00  0.50  0.21  0.2183270
19 5e+00  0.50  0.22  0.2097618
20 1e+01  0.50  0.22  0.2149935
21 1e+02  0.50  0.30  0.1885618
22 1e-03  1.00  0.57  0.1888562
23 1e-02  1.00  0.57  0.1888562
24 1e-01  1.00  0.23  0.2496664
25 1e+00  1.00  0.21  0.2183270
26 5e+00  1.00  0.27  0.2311805
27 1e+01  1.00  0.28  0.2149935
28 1e+02  1.00  0.29  0.1969207
29 1e-03  2.00  0.57  0.1888562
30 1e-02  2.00  0.57  0.1888562
31 1e-01  2.00  0.42  0.1686548
32 1e+00  2.00  0.25  0.2173067
33 5e+00  2.00  0.25  0.1715938
34 1e+01  2.00  0.25  0.1840894
35 1e+02  2.00  0.28  0.1686548
36 1e-03  3.00  0.57  0.1888562
37 1e-02  3.00  0.57  0.1888562
38 1e-01  3.00  0.57  0.1946507
39 1e+00  3.00  0.24  0.1837873
40 5e+00  3.00  0.23  0.1766981
41 1e+01  3.00  0.22  0.1619328
42 1e+02  3.00  0.32  0.1398412
43 1e-03  4.00  0.58  0.1813529
44 1e-02  4.00  0.58  0.1813529
45 1e-01  4.00  0.59  0.1852926
46 1e+00  4.00  0.25  0.1840894
47 5e+00  4.00  0.21  0.1449138
48 1e+01  4.00  0.25  0.1581139
49 1e+02  4.00  0.37  0.1636392
>
> # SVM with the best parameters
> best.fit2 = cv.fit2$best.model
>
> # Prediction for test data
> yhat.te2 = predict(best.fit2, test)
> cm2 = table(true = test$y, predict=yhat.te2)
>
> misclass(cm2)
[1] 0.2345455
>
> fmeasure(cm2)
[1] 0.3943662
>
> # SMOTE ------------------------------------------------
> install.packages('DMwR')
> library(DMwR)
>
> set.seed(1)
> smtrain = SMOTE(y~., data=train, perc.over=200, k = 5, perc.under=200)
>
> dim(smtrain)
[1] 350   3
> table(smtrain$y)
```

```
  0   1
200 150
>
>
> # Scatter plot for the training data from SMOTE
> plot(cbind(smtrain$X1,smtrain$X2),
+     col=(3-as.numeric(smtrain$y)),xlab='X1',ylab='X2')
>
>
>
> # SVM for data from SMOTE.
> set.seed(10)
> cv.fit3 = tune(svm, y~., data=smtrain, kernel='radial',
+              ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
+                          gamma=c(0.01,0.1,0.5,1,2,3,4)))
> summary(cv.fit3)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  100     2

- best performance: 0.1514286

- Detailed performance results:
    cost gamma    error dispersion
1  1e-03  0.01 0.4285714 0.06598289
2  1e-02  0.01 0.4285714 0.06598289
3  1e-01  0.01 0.4257143 0.06926894
4  1e+00  0.01 0.1800000 0.05395892
5  5e+00  0.01 0.1857143 0.06208764
6  1e+01  0.01 0.2028571 0.06237913
7  1e+02  0.01 0.1942857 0.03761603
8  1e-03  0.10 0.4285714 0.06598289
9  1e-02  0.10 0.4285714 0.06598289
10 1e-01  0.10 0.1942857 0.05993193
11 1e+00  0.10 0.2114286 0.06053428
12 5e+00  0.10 0.2057143 0.03512207
13 1e+01  0.10 0.2000000 0.04467063
14 1e+02  0.10 0.2000000 0.04467063
15 1e-03  0.50 0.4285714 0.06598289
16 1e-02  0.50 0.4285714 0.06598289
17 1e-01  0.50 0.2000000 0.06734350
18 1e+00  0.50 0.1971429 0.05119878
19 5e+00  0.50 0.2028571 0.05462716
20 1e+01  0.50 0.1914286 0.04675405
21 1e+02  0.50 0.1828571 0.04085259
22 1e-03  1.00 0.4285714 0.06598289
23 1e-02  1.00 0.4285714 0.06598289
24 1e-01  1.00 0.2085714 0.07505100
25 1e+00  1.00 0.1914286 0.04675405
26 5e+00  1.00 0.1857143 0.04714045
27 1e+01  1.00 0.1742857 0.05626307
28 1e+02  1.00 0.1714286 0.03299144
29 1e-03  2.00 0.4285714 0.06598289
30 1e-02  2.00 0.4285714 0.06598289
31 1e-01  2.00 0.2142857 0.07158713
32 1e+00  2.00 0.1885714 0.05251066
33 5e+00  2.00 0.1771429 0.03512207
```

```
34 1e+01  2.00 0.1685714 0.03676240
35 1e+02  2.00 0.1514286 0.03026714
36 1e-03  3.00 0.4285714 0.06598289
37 1e-02  3.00 0.4285714 0.06598289
38 1e-01  3.00 0.2114286 0.06625725
39 1e+00  3.00 0.1857143 0.05259696
40 5e+00  3.00 0.1685714 0.03676240
41 1e+01  3.00 0.1571429 0.02776644
42 1e+02  3.00 0.1685714 0.03420626
43 1e-03  4.00 0.4285714 0.06598289
44 1e-02  4.00 0.4285714 0.06598289
45 1e-01  4.00 0.2085714 0.05225092
46 1e+00  4.00 0.1771429 0.03995462
47 5e+00  4.00 0.1657143 0.03761603
48 1e+01  4.00 0.1600000 0.02409354
49 1e+02  4.00 0.1600000 0.04704415
>
> # SVM with the best parameters
> best.fit3 = cv.fit3$best.model
>
> # Prediction for test data
> yhat.te3 = predict(best.fit3, test)
> cm3 = table(true = test$y, predict=yhat.te3)
>
> misclass(cm3)
[1] 0.1727273
>
> fmeasure(cm3)
[1] 0.3708609
>
> ################## One-class learning ##################
> # Support vector data description (SVDD) -----------------
>
> # Training data for the large class.
> train.x0 = train[train$y == 0,-1]
>
> result = NULL
> for (nu in c(0.001,0.01,0.1,0.3,0.5,0.7,0.9))
+ {
+   for (gamma in c(0.01,0.1,0.5,1,2,3,5))
+   {
+     svddfit = svm(x=train.x0, type='one-classification', kernel='radial',
+                   nu=nu, gamma=gamma)
+
+     minor = predict(svddfit,test[,-1])
+     # predict: TRUE: small class(outlier), FALSE: large class
+     yhat.te = numeric(nrow(test))
+     yhat.te[minor==TRUE] = 1
+     cm = table(true = test$y, predict=yhat.te)
+     result = rbind(result, c(nu,gamma,fmeasure(cm)))
+   }
+ }
> # Best result for F-measure.
> names(result) <- c('nu','gamma','F-measure')
> result[which.max(result[,3]),]
[1] 0.0010000 0.0100000 0.1548822
>
>
>
> ############### Cost-sensitive Learning ################
>
> # Class weighted SVM ----------------------------
> # svm function using 'class.weight' option
```

```
>
> wts = 500 / table(train$y)
>
> set.seed(10)
> cv.fit = tune(svm, y~., data=train, kernel='radial',
+             class.weights=wts,
+             ranges=list(cost=c(0.001,0.01,0.1,1,5,10,100),
+                         gamma=c(0.01,0.1,0.5,1,2,3,4)))
> summary(cv.fit)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  0.1  0.01

- best performance: 0.1890909

- Detailed performance results:
    cost gamma    error dispersion
1  1e-03  0.01 0.4436364 0.42839526
2  1e-02  0.01 0.6072727 0.41745215
3  1e-01  0.01 0.1890909 0.07675704
4  1e+00  0.01 0.2000000 0.06583503
5  5e+00  0.01 0.2163636 0.05103160
6  1e+01  0.01 0.2145455 0.05269592
7  1e+02  0.01 0.2272727 0.06014980
8  1e-03  0.10 0.4436364 0.42839526
9  1e-02  0.10 0.1963636 0.07109274
10 1e-01  0.10 0.2018182 0.05778272
11 1e+00  0.10 0.2254545 0.05297400
12 5e+00  0.10 0.2454545 0.05765545
13 1e+01  0.10 0.2490909 0.06063636
14 1e+02  0.10 0.2527273 0.06382368
15 1e-03  0.50 0.4436364 0.42839526
16 1e-02  0.50 0.2272727 0.06075739
17 1e-01  0.50 0.2327273 0.05474710
18 1e+00  0.50 0.2345455 0.05778272
19 5e+00  0.50 0.2345455 0.05314706
20 1e+01  0.50 0.2254545 0.05825753
21 1e+02  0.50 0.2218182 0.05925773
22 1e-03  1.00 0.4436364 0.42839526
23 1e-02  1.00 0.2418182 0.08573134
24 1e-01  1.00 0.2363636 0.05213530
25 1e+00  1.00 0.2072727 0.05825753
26 5e+00  1.00 0.2181818 0.06180630
27 1e+01  1.00 0.2072727 0.06132902
28 1e+02  1.00 0.2400000 0.07559984
29 1e-03  2.00 0.4436364 0.42839526
30 1e-02  2.00 0.3818182 0.20694995
31 1e-01  2.00 0.2236364 0.06242718
32 1e+00  2.00 0.2109091 0.06072715
33 5e+00  2.00 0.2236364 0.06301282
34 1e+01  2.00 0.2345455 0.08106273
35 1e+02  2.00 0.2163636 0.06718804
36 1e-03  3.00 0.4436364 0.42839526
37 1e-02  3.00 0.5600000 0.38386177
38 1e-01  3.00 0.2109091 0.05950515
39 1e+00  3.00 0.2090909 0.05504819
40 5e+00  3.00 0.2272727 0.07820296
41 1e+01  3.00 0.2272727 0.06652878
```

```
42 1e+02  3.00 0.2054545 0.06641827
43 1e-03  4.00 0.4436364 0.42839526
44 1e-02  4.00 0.6072727 0.41745215
45 1e-01  4.00 0.2181818 0.06239776
46 1e+00  4.00 0.2054545 0.05286989
47 5e+00  4.00 0.2145455 0.06402479
48 1e+01  4.00 0.2218182 0.06459594
49 1e+02  4.00 0.1963636 0.06737911
>
> # SVM with the best parameters
> best.fit = cv.fit$best.model
>
> # Prediction for test data
> yhat.te = predict(best.fit, test)
> cm = table(true = test$y, predict=yhat.te)
>
> misclass(cm)
[1] 0.1945455
>
> fmeasure(cm)
[1] 0.4153005
>
> ################## Ensemble-based methods ###################
> install.packages('ebmc')
> library(ebmc)
>
> # SMOTE Boost ----------------------------------------
> set.seed(10)
>
> fit1 = sbo(y~., data=train, size=200, alg='cart', over=300)
> # y should be encoded by (0,1); 0 large class, 1 small class
> # size: # of boosting iterations
> # alg: weak learner
> # over: oversampling rate (multiple of 100 is only acceptible)
>
> yhat.te = predict(fit1, test, type='class')
> cm = table(true = test$y, predict=yhat.te)
>
> misclass(cm)
[1] 0.1454545
>
> fmeasure(cm)
[1] 0.3220339
>
>
> # SMOTE Bagging -------------------------------------
> set.seed(10)
>
> fit2 = sbag(y~., data=train, size=300, alg='cart')
> # y should be encoded by (0,1); 0 large class, 1 small class
>
> yhat.te = predict(fit2, test, type='class')
> cm = table(true = test$y, predict=yhat.te)
>
> misclass(cm)
[1] 0.2054545
>
> fmeasure(cm)
[1] 0.3542857
```