Group 7

2022/5/12

```r
library(glmnet)
library(MASS)
library(class)
library(caret)
library(e1071)
library(mboost)
library(plyr)
library(import)
library(ipred)
library(LiblineaR)
library(naivebayes)
library(nnet)
library(randomForest)
library(gbm)
```

```r
set.seed(1082)
data = read.csv("haralick_average blur.csv", header = T)
data$Label <- factor(data$Label)
```

```r
trControl = trainControl(method = "cv", number = 5)
```

Penalized Logistic Regression

```
tuneGrid <- expand.grid(alpha = 1, lambda = seq(0, 5, by = 0.1))

model = train(Label ~ ., data = data, method = "glmnet", tuneGrid = tuneGrid,
    trControl = trControl)
```

```
## Warning: from glmnet C++ code (error code -97); Convergence for 97th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
## Warning: from glmnet C++ code (error code -95); Convergence for 95th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
model
```

```
## glmnet
##
## 1500 samples
##   52 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1199, 1202, 1199, 1201, 1199
## Resampling results across tuning parameters:
##
##   lambda  Accuracy   Kappa
##   0.0     0.9520326  0.9396204
##   0.1     0.6592893  0.5525209
##   0.2     0.5800124  0.4394372
##   0.3     0.4946911  0.3190068
##   0.4     0.2633380  0.0000000
##   0.5     0.2633380  0.0000000
##   0.6     0.2633380  0.0000000
##   0.7     0.2633380  0.0000000
##   0.8     0.2633380  0.0000000
##   0.9     0.2633380  0.0000000
##   1.0     0.2633380  0.0000000
##   1.1     0.2633380  0.0000000
##   1.2     0.2633380  0.0000000
##   1.3     0.2633380  0.0000000
##   1.4     0.2633380  0.0000000
##   1.5     0.2633380  0.0000000
##   1.6     0.2633380  0.0000000
##   1.7     0.2633380  0.0000000
##   1.8     0.2633380  0.0000000
##   1.9     0.2633380  0.0000000
##   2.0     0.2633380  0.0000000
##   2.1     0.2633380  0.0000000
##   2.2     0.2633380  0.0000000
##   2.3     0.2633380  0.0000000
##   2.4     0.2633380  0.0000000
##   2.5     0.2633380  0.0000000
##   2.6     0.2633380  0.0000000
##   2.7     0.2633380  0.0000000
##   2.8     0.2633380  0.0000000
##   2.9     0.2633380  0.0000000
##   3.0     0.2633380  0.0000000
##   3.1     0.2633380  0.0000000
##   3.2     0.2633380  0.0000000
##   3.3     0.2633380  0.0000000
##   3.4     0.2633380  0.0000000
##   3.5     0.2633380  0.0000000
##   3.6     0.2633380  0.0000000
##   3.7     0.2633380  0.0000000
##   3.8     0.2633380  0.0000000
##   3.9     0.2633380  0.0000000
##   4.0     0.2633380  0.0000000
##   4.1     0.2633380  0.0000000
##   4.2     0.2633380  0.0000000
##   4.3     0.2633380  0.0000000
##   4.4     0.2633380  0.0000000
##   4.5     0.2633380  0.0000000
##   4.6     0.2633380  0.0000000
##   4.7     0.2633380  0.0000000
##   4.8     0.2633380  0.0000000
##   4.9     0.2633380  0.0000000
##   5.0     0.2633380  0.0000000
##
## Tuning parameter 'alpha' 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 alpha = 1 and lambda = 0.
```

```
confusionMatrix(model, norm = "none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 384  11   1  10   0   0
##          1   6 274   0   8   1   3
##          2   0   1  57   0   0   0
##          3   5   8   0 207   5   1
##          4   0   4   0   2 140   1
##          5   0   5   0   0   0 366
##
##  Accuracy (average) : 0.952
```

KNN

```
knn.fit <- train(Label ~ ., method = "knn", tuneGrid = expand.grid(k = 5),
    trControl = trControl, metric = "Accuracy", data = data)
knn.fit
```

```
## k-Nearest Neighbors
##
## 1500 samples
##   52 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1199, 1199, 1201, 1201, 1200
## Resampling results:
##
##   Accuracy   Kappa
##   0.7167353  0.6399852
##
## Tuning parameter 'k' was held constant at a value of 5
```

Random Forest

```
rf.fit <- train(Label ~ ., method = "rf", trControl = trControl,
    metric = "Accuracy", data = data)

rf.fit
```

```
## Random Forest
##
## 1500 samples
##   52 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1198, 1201, 1200, 1201, 1200
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9280234  0.9090836
##   27    0.9287012  0.9100082
##   52    0.9160498  0.8941224
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
confusionMatrix(rf.fit, norm = "none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 383   7   3  26   3   0
##          1   8 277   0   9   6   7
##          2   1   1  55   0   0   0
##          3   3  10   0 186   8   1
##          4   0   1   0   6 129   0
##          5   0   7   0   0   0 363
##
##  Accuracy (average) : 0.9287
```

Boosting Tree

```
treeboost.fit <- train(Label ~ ., method = "gbm", verbose = FALSE,
    trControl = trControl, metric = "Accuracy", data = data)
treeboost.fit
```

```
## Stochastic Gradient Boosting
##
## 1500 samples
##   52 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1201, 1201, 1200, 1199, 1199
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8846279  0.8539202
##   1                  100      0.9193015  0.8980678
##   1                  150      0.9246594  0.9048058
##   2                   50      0.9246549  0.9048024
##   2                  100      0.9326639  0.9150280
##   2                  150      0.9393395  0.9234588
##   3                   50      0.9339794  0.9166270
##   3                  100      0.9413417  0.9260147
##   3                  150      0.9419973  0.9268347
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
confusionMatrix(treeboost.fit, norm = "none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 384   8   4  20   1   0
##          1   6 280   0   8   3   1
##          2   0   1  51   0   0   0
##          3   5   9   3 194   6   0
##          4   0   1   0   5 136   2
##          5   0   4   0   0   0 368
##
##  Accuracy (average) : 0.942
```

SVM

```r
svm.fit <- train(Label ~ ., method = "svmRadial", trControl = trControl,
    metric = "Accuracy", data = data)
svm.fit
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1500 samples
##   52 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1200, 1200, 1200, 1200, 1200
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8273333  0.7783192
##   0.50  0.8813333  0.8491714
##   1.00  0.9073333  0.8827809
##
## Tuning parameter 'sigma' was held constant at a value of 0.01747041
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01747041 and C = 1.
```

```r
confusionMatrix(svm.fit, norm = "none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 371   8   4  46   5   0
##          1  17 279   0  21   5   1
##          2   4   0  54   0   0   0
##          3   2   6   0 156   3   2
##          4   1   3   0   4 133   0
##          5   0   7   0   0   0 368
##
##  Accuracy (average) : 0.9073
```

Summary

Model	Predictor	Parameter	Accuracy
Penalized Logistic Regression	52	$\lambda = 0$	0.9520
KNN	52	K=5	0.7167
Random Forest	52	mtry = 27	0.9287
Boosting Tree	52	shrinkage = 0.1	0.9420
SVM	52	sigma = 0.0174, C = 1	0.9073