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_bilateralFilter.csv", header = T)
data$Label <- factor(data$Label)
```

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

Penalized Logistic Regression

```r
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 -92); Convergence for 92th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
## Warning: from glmnet C++ code (error code -86); Convergence for 86th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
## Warning: from glmnet C++ code (error code -100); Convergence for 100th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```r
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     0.9466970  0.9328408
##   0.1     0.6612783  0.5562252
##   0.2     0.5113158  0.3461537
##   0.3     0.2633380  0.0000000
##   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
##          0 387  11   2  13   0   0
##          1   6 275   0   8   3   8
##          2   0   0  56   0   0   2
##          3   2   7   0 203   4   0
##          4   0   1   0   3 139   1
##          5   0   9   0   0   0 360
##
##  Accuracy (average) : 0.9467
```

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
##   0.6620748  0.5694973
##
## Tuning parameter 'k' was held constant at a value of 5
```

Random Forest

```r
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    0.9360014  0.9191475
##   27    0.9346792  0.9175265
##   52    0.9226701  0.9023659
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
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
##          0 391   9   1  16   1   0
##          1   2 264   0  20   5   2
##          2   0   1  57   0   0   0
##          3   2   7   0 189   5   0
##          4   0   0   0   2 134   0
##          5   0  22   0   0   1 369
##
##  Accuracy (average) : 0.936
```

Boosting Tree

```r
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: 1199, 1199, 1200, 1200, 1202
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9059786  0.8810292
##   1                  100      0.9293457  0.9108321
##   1                  150      0.9326880  0.9150877
##   2                   50      0.9339990  0.9166993
##   2                  100      0.9400035  0.9243198
##   2                  150      0.9473392  0.9336044
##   3                   50      0.9413324  0.9259739
##   3                  100      0.9426702  0.9276991
##   3                  150      0.9453258  0.9310493
##
## 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 =
##  2, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
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
##          0 387   9   2  13   2   0
##          1   4 276   0  10   4   1
##          2   1   0  56   0   0   0
##          3   3   7   0 200   4   0
##          4   0   2   0   4 134   2
##          5   0   9   0   0   2 368
##
##  Accuracy (average) : 0.9473
```

SVM

```
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: 1199, 1200, 1199, 1201, 1201
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8647032  0.8278395
##   0.50  0.8980146  0.8708379
##   1.00  0.9180215  0.8963713
##
## Tuning parameter 'sigma' was held constant at a value of 0.01940088
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01940088 and C = 1.
```

```
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
##          0 379  10   5  33   2   0
##          1  14 272   0  16   4   5
##          2   2   1  53   0   0   0
##          3   0   8   0 176   5   3
##          4   0   0   0   2 134   0
##          5   0  12   0   0   1 363
##
##  Accuracy (average) : 0.918
```

Summary

Model	Predictor	Parameter	Accuracy
Penalized Logistic Regression	52	$\lambda = 0$	0.9467
KNN	52	K=5	0.6621
Random Forest	52	mtry = 2	0.9360
Boosting Tree	52	shrinkage = 0.1	0.9473
SVM	52	sigma = 0.0194, C = 1	0.9180