Group 7

2022/5/28

```
set.seed(1082)
data=read.csv("crop3x3.csv",header = T)
data.2=read.csv("testCrop3x3.csv",header = T)
data$Label <- factor(data$Label)
```

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

PCA(training data)

15 PCs can explain 90.43 variation.

```
pca = princomp(data[,1:52],cor=T)
#summary(pca)
```

```
z1 <- pca$scores[,1]
z2 <- pca$scores[,2]
z3 <- pca$scores[,3]
z4 <- pca$scores[,4]
z5 <- pca$scores[,5]
z6 <- pca$scores[,6]
z7 <- pca$scores[,7]
z8 <- pca$scores[,8]
z9 <- pca$scores[,9]
z10 <- pca$scores[,10]
z11 <- pca$scores[,11]
z12 <- pca$scores[,12]
z13 <- pca$scores[,13]
z14 <- pca$scores[,14]
z15<- pca$scores[,15]
pca_data_train = data.frame(z1 = z1, z2 = z2, z3 = z3, z4 = z4, z5 = z5,z6 = z6, z7 = z7, z8 = z8, z9 =
pca_data_train$Label = data$Label
```

PCA(testing data)

```
pca.test <- predict(pca, newdata = data.2[,1:52])
pca.test=pca.test[,1:15]
colnames(pca.test) <- c("z1", "z2", "z3", "z4", "z5", "z6", "z7",
                        "z8", "z9", "z10", "z11", "z12", "z13","z14","z15")
```

QDA

```
qda.fit <- train(Label ~ ., method = "qda"
                 , trControl = trControl
                 ,metric = "Accuracy"
                 , data = pca_data_train)
confusionMatrix(qda.fit,norm="none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 367   0   6  52   0   0
##          1   9 288   0  18   6   8
##          2  13   0  48   0   0   0
##          3   6  11   4 154  16   0
##          4   0   2   0   3 124   0
##          5   0   2   0   0   0 363
##
##  Accuracy (average) : 0.896
```

```
pred.qda = predict(qda.fit,newdata = pca.test)
#pred.qda
```

```
write.csv(pred.qda, "QDA_Label.csv", row.names = FALSE)
```

LDA

```
lda.fit <- train(Label ~ .
                 , method = "lda"
                 , trControl = trControl
                 ,metric = "Accuracy"
                 , data = pca_data_train)
confusionMatrix(lda.fit,norm="none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
```

```
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 346   8  19  62   3   0
##          1  18 272   0  23  10   1
##          2  31   0  37   0   0   0
##          3   0  11   2 121  15   1
##          4   0   6   0  21 118   0
##          5   0   6   0   0   0 369
##
##  Accuracy (average) : 0.842
```

```r
pred.lda = predict(lda.fit,newdata = pca.test)
#pred.lda
```

```r
write.csv(pred.lda, "LDA_Label.csv", row.names = FALSE)
```

KNN

```r
knn.fit <- train(Label ~ .
                 , method = "knn"
                 , tuneGrid = expand.grid(k = 5)
                 ,trControl = trControl
                 , metric = "Accuracy"
                 , data = data)
confusionMatrix(knn.fit,norm="none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 365   6  33  44   1   0
##          1   4 154   0  28  20  85
##          2   3   0  21   5   0   0
##          3  22  11   4  88  12   5
##          4   0   8   0  14  52  35
##          5   1 124   0  48  61 246
##
##  Accuracy (average) : 0.6173
```

```r
pred.knn = predict(knn.fit,newdata = data.2)
#pred.knn
```

```r
write.csv(pred.knn, "KNN_Label.csv", row.names = FALSE)
```

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: 1200, 1200, 1199, 1200, 1201
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9520020  0.9394311
##   27    0.9539887  0.9420148
##   52    0.9533220  0.9411479
##
## 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
##          0 391   1   5   9   0   0
##          1   0 281   0  13   3   1
##          2   1   0  53   0   0   0
##          3   3  12   0 199   6   0
##          4   0   3   0   6 137   0
##          5   0   6   0   0   0 370
##
##  Accuracy (average) : 0.954
```

```
pred.rf=predict(rf.fit,newdata = data.2)
#pred.rf
```

```
write.csv(pred.rf,"Random forest_Label.csv", row.names = FALSE)
```

Boosting Tree

```r
boosttree.fit <- train(Label ~ .,method = "gbm"
                       ,verbose = FALSE
                       ,trControl= trControl
                       ,metric = "Accuracy"
                       ,data = data)
boosttree.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, 1201, 1200, 1199, 1201
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9126659  0.8896490
##   1                  100      0.9346705  0.9175208
##   1                  150      0.9373417  0.9209924
##   2                   50      0.9426861  0.9277212
##   2                  100      0.9500084  0.9369835
##   2                  150      0.9526840  0.9403502
##   3                   50      0.9473595  0.9336617
##   3                  100      0.9513596  0.9386888
##   3                  150      0.9526863  0.9403612
##
## 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.
```

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

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 388   1   6   9   1   0
##          1   0 285   1  11   4   4
##          2   1   0  51   0   0   0
##          3   6  10   0 203   6   0
##          4   0   3   0   4 135   0
##          5   0   4   0   0   0 367
```

```
##
##  Accuracy (average) : 0.9527
```

```r
pred.boosttree=predict(boosttree.fit,newdata = data.2)
#pred.boosttree
```

```r
write.csv(pred.boosttree,"Boosting Tree_label.csv", row.names = FALSE)
```

Naive Bayes

```r
naive.fit=train(Label ~ .,method = "naive_bayes",trControl= trControl,metric = "Accuracy",data = data)
naive.fit
```

```
## Naive Bayes
##
## 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, 1201, 1199, 1200, 1200
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.8253081  0.7824445
##    TRUE      0.8086837  0.7620053
##
## Tuning parameter 'laplace' 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 laplace = 0, usekernel = FALSE
##  and adjust = 1.
```

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

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 292   1   0  54   0   1
##          1  33 275   0  17   5   5
##          2  68   0  58   8   0   0
##          3   2  19   0 131  24   0
##          4   0   4   0  17 117   0
##          5   0   4   0   0   0 365
##
##  Accuracy (average) : 0.8253
```

```r
pred.naive=predict(naive.fit,newdata = data.2)
#pred.naive
```

```r
write.csv(pred.naive,"Naive Bayes_label.csv", row.names = FALSE)
```

LASSO

```
grid=seq (0,10,0.1)
x =model.matrix(Label ~ ., data)[,-1]
x.new=as.matrix(data.2)
y =data$Label
cv.out=cv.glmnet(x, y,family ="multinomial"
                 ,alpha =1,nfolds=5
                 ,type.multinomial="grouped")
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 0.003079428
```

```
train_pred.lasso <- predict(cv.out,s=bestlam,type = "class",newx =x)
# Confusion Matrix and Accuracy
table(train_pred.lasso,data[,53]) ; mean(train_pred.lasso==data[,53])
```

```
##
## train_pred.lasso   0   1   2   3   4   5
##                0 383   4   6  23   0   0
##                1   2 284   0  14   3   0
##                2   2   0  52   0   0   0
##                3   8   9   0 185  12   0
##                4   0   3   0   5 131   0
##                5   0   3   0   0   0 371
```

```
## [1] 0.9373333
```

```
lasso.pred=predict(cv.out,s=bestlam,type = "class",newx =x.new)
```

```
write.csv(lasso.pred,"LASSO_label.csv", row.names = FALSE)
```

Forward Selection

```r
# allNames <- names(data[,1:52])
# allVar <- paste("~", paste(allNames, collapse=" + "))
#
# multi.fit=multinom(Label~1,data=data, trace = F)
# stepAIC(multi.fit, direction = "forward",trace = FALSE,scope = allVar)
```

```r
multi.fit.aic=multinom(Label ~ X2 + X19 + X33 + X44 + X20 + X11 + X21 + X18 + X40 + X5 + X6 + X7 + X50 +

train_pred.forward <- predict(multi.fit.aic, data = data)
# Confusion Matrix and Accuracy
table(train_pred.forward,data[,53]) ; mean(train_pred.forward==data[,53])
```

```
##
## train_pred.forward   0   1   2   3   4   5
##                  0 385   1   1  15   0   0
##                  1   1 293   0  12   0   0
##                  2   0   0  57   0   0   0
##                  3   9   4   0 195   9   0
##                  4   0   5   0   5 137   0
##                  5   0   0   0   0   0 371
```

```
## [1] 0.9586667
```

```r
step.multi.pred=predict(multi.fit.aic,newdata=data.2)
```

```r
write.csv(step.multi.pred,"Forward Selection_label.csv", row.names = FALSE)
```

Penalized Multinomial Regression(Cross Validation)

```
multi.fit.2=train(Label ~ .
                  ,method = "multinom"
                  ,trControl=trControl
                  ,metric = "Accuracy"
                  , trace = F
                  ,data = data)
multi.fit.2
```

```
## Penalized Multinomial Regression
##
## 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:
##
##   decay  Accuracy   Kappa
##   0e+00  0.9040121  0.8792846
##   1e-04  0.9013477  0.8759478
##   1e-01  0.9147032  0.8925414
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.
```

```
confusionMatrix(multi.fit.2,norm="none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 378   5   4  25   0   0
##          1   2 275   0  17   3   2
##          2   1   0  52   2   1   1
##          3  14  11   2 170  12   0
##          4   0   9   0  13 130   1
##          5   0   3   0   0   0 367
##
##  Accuracy (average) : 0.9147
```

```
multi.fit.2.pred=predict(multi.fit.2,newdata=data.2)
```

```
write.csv(multi.fit.2.pred,"Penalized Multinomial Regression_label.csv", row.names = FALSE)
```

SVM

```r
svm.fit <- train(Label~.,method= "svmRadial",
                 trControl = trControl,
                 metric= "Accuracy",
                 data= data)
pred.svm=predict(svm.fit,newdata=data.2)
#pred.svm
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
##          0 384   6  17  36   1   0
##          1   2 287   0  19   4   0
##          2   2   0  41   0   0   0
##          3   7   6   0 168   9   0
##          4   0   2   0   4 132   0
##          5   0   2   0   0   0 371
##
##  Accuracy (average) : 0.922
```

```r
write.csv(pred.svm,"SVM-radial_Label.csv", row.names = FALSE)
```

mode

```r
train_pred_response <- cbind(
  matrix(predict(qda.fit, data = pca_data_train), ncol=1),
  matrix(predict(lda.fit, data = pca_data_train), ncol=1),
  matrix(predict(knn.fit, data = data), ncol=1),
  matrix(predict(rf.fit, data = data), ncol=1),
  matrix(predict(boosttree.fit, data = data), ncol=1),
  matrix(predict(naive.fit, data = data), ncol=1),
  matrix(predict(cv.out,s=bestlam,type = "class",newx =x), ncol=1),
  matrix(predict(multi.fit.aic, data = data), ncol=1),
  matrix(rep(NA,1500), ncol=1), # !!!
  #matrix(predict(multi.fit.2, data = data), ncol=1),
  matrix(predict(svm.fit, data = data), ncol=1))

Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

train_pred.mode <- apply(train_pred_response, 1, Mode)
# Confusion Matrix and Accuracy
table(train_pred.mode,data[,53]) ; mean(train_pred.mode==data[,53])
```

```
##
## train_pred.mode   0   1   2   3   4   5
##               0 394   2   0  21   0   0
##               1   1 294   0  13   1   0
##               2   0   0  58   0   0   0
##               3   0   5   0 190   7   0
##               4   0   1   0   3 138   0
##               5   0   1   0   0   0 371
```

```
## [1] 0.9633333
```

```r
test_pred_response <- cbind(matrix(pred.qda, ncol=1),
                            matrix(pred.lda, ncol=1),
                            matrix(pred.knn, ncol=1),
                            matrix(pred.rf, ncol=1),
                            matrix(pred.boosttree, ncol=1),
                            matrix(pred.naive, ncol=1),
                            lasso.pred,
                            matrix(step.multi.pred, ncol=1),
                            matrix(multi.fit.2.pred, ncol=1),
                            matrix(pred.svm, ncol=1))

pred.mode <- apply(test_pred_response, 1, Mode)
#pred.mode
test_pred_response <- cbind(test_pred_response, pred.mode)
```

all model

```r
colnames(test_pred_response)<- paste(c("QDA_Label", "LDA_Label", "KNN_Label", "Random forest_Label", "Bo

head(test_pred_response)
```

```
##      QDA_Label_3x3 LDA_Label_3x3 KNN_Label_3x3 Random forest_Label_3x3
## [1,] "5"           "5"           "5"           "5"
## [2,] "0"           "0"           "0"           "0"
## [3,] "0"           "0"           "0"           "3"
## [4,] "0"           "0"           "0"           "0"
## [5,] "0"           "0"           "0"           "0"
## [6,] "1"           "1"           "5"           "1"
##      Boosting Tree_Label_3x3 Naive Bayes_Label_3x3 LASSO_Label_3x3
## [1,] "5"                     "5"                   "5"
## [2,] "0"                     "2"                   "0"
## [3,] "3"                     "2"                   "0"
## [4,] "0"                     "0"                   "0"
## [5,] "0"                     "0"                   "0"
## [6,] "1"                     "1"                   "1"
##      Forward Selection_Label_3x3 Penalized Multinomial Regression_Label_3x3
## [1,] "5"                         "5"
## [2,] "0"                         "0"
## [3,] "3"                         "0"
## [4,] "0"                         "0"
## [5,] "0"                         "0"
## [6,] "1"                         "1"
##      SVM-radial_Label_3x3 Mode_Label_3x3
## [1,] "5"                  "5"
## [2,] "0"                  "0"
## [3,] "0"                  "0"
## [4,] "0"                  "0"
## [5,] "0"                  "0"
## [6,] "1"                  "1"
```

```r
write.csv(test_pred_response,"3x3_label.csv", row.names = FALSE)
```