Group 7

2022/6/2

```
set.seed(1082)
data <- read.csv("crop4x4all.csv", header = T)
data.2 <- read.csv("test_crop4x4all.csv", header = T)
data$Label <- factor(data$Label)
```

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

PCA(training data)

19 PCs can explain 95% variation.

```
pca = princomp(data[,1:832],cor=T)

cumulative.var <- c()
PoV <- pca$sdev^2/sum(pca$sdev^2)
for (i in 1:832){
  cumulative.var[i] <- sum(PoV[1:i])
}
#cumulative.var
```

```
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$scores[,15]
z16 <- pca$scores[,16]
z17 <- pca$scores[,17]
z18 <- pca$scores[,18]
z19 <- pca$scores[,19]

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[,1:832])
pca.test=pca.test[,1:19]
colnames(pca.test) <- c("z1", "z2", "z3", "z4", "z5", "z6", "z7","z8", "z9", "z10", "z11", "z12","z13",
```

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
##          0 368   1   3  55   0   0
##          1  20 277   0   8   1   1
##          2   7   0  52   0   0   0
##          3   0  21   3 162  14   0
##          4   0   2   0   2 131   0
##          5   0   2   0   0   0 370
##
##  Accuracy (average) : 0.9067
```

```
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
##          0 265  25   2  83  25   0
##          1  73 267   0  19   3   2
##          2  43   0  55   0   0   0
##          3  12   5   1 111   8   0
##          4   2   1   0  14 110   0
##          5   0   5   0   0   0 369
##
##  Accuracy (average) : 0.7847
```

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

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

KNN

```
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
##          0 376  60  16  56  15  13
##          1   0 218   0  15   4  88
##          2   7   0  41   0   0   0
##          3  12   9   1 140   5   7
##          4   0   1   0  11 121   0
##          5   0  15   0   5   1 263
##
##  Accuracy (average) : 0.7727
```

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

```
#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
##  832 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1201, 1201, 1199, 1198, 1201
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.9293742  0.9108259
##    40   0.9366966  0.9201735
##   832   0.9313277  0.9133956
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 40.
```

```
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   8   2  15   0   0
##          1   6 280   0  23   3   0
##          2   1   0  56   0   0   0
##          3   5   8   0 183  11   0
##          4   0   3   0   6 132   0
##          5   0   4   0   0   0 371
##
##  Accuracy (average) : 0.9367
```

```
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
##  832 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1199, 1200, 1200, 1201, 1200
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8559837  0.8176303
##   1                  100      0.8959995  0.8684332
##   1                  150      0.9099928  0.8863283
##   2                   50      0.9100173  0.8862370
##   2                  100      0.9286752  0.9099437
##   2                  150      0.9313263  0.9133810
##   3                   50      0.9239996  0.9039831
##   3                  100      0.9359996  0.9191539
##   3                  150      0.9373352  0.9209060
##
## 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
##          0 386  10   6  18   2   0
##          1   1 278   0  15   2   1
##          2   3   0  52   0   0   0
##          3   5  11   0 187   9   0
##          4   0   1   0   7 133   0
##          5   0   3   0   0   0 370
```

```
##
##  Accuracy (average) : 0.9373
```

```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
##  832 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1201, 1199, 1199, 1201, 1200
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.6840083  0.6174615
##    TRUE      0.7060218  0.6413643
##
## 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 = TRUE
##  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  97   1   0  24   0   0
##          1 105 259   0  18   1   3
##          2  76   0  57  13   0   0
##          3  71  23   1 157  24   0
##          4  46   6   0  15 121   0
##          5   0  14   0   0   0 368
##
##  Accuracy (average) : 0.706
```

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

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

LASSO

```r
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] 0.004544428
```

```r
train_pred.lasso <- predict(cv.out,s=bestlam,type = "class",newx =x)
# Confusion Matrix and Accuracy
table(train_pred.lasso,data[,833]) ; mean(train_pred.lasso==data[,833])
```

```
##
## train_pred.lasso   0   1   2   3   4   5
##                0 361  16   7  28  10   0
##                1  22 275   0  16   3   0
##                2   5   0  51   0   0   0
##                3   7   9   0 178   8   0
##                4   0   1   0   5 125   0
##                5   0   2   0   0   0 371
```

```
## [1] 0.9073333
```

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

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

Forward Selection

```
# allNames <- names(data[,1:832])
# allVar <- paste("~", paste(allNames, collapse=" + "))
#
# multi.fit=multinom(Label~1,data=data, trace = F)
# stepAIC(multi.fit, direction = "forward",trace = FALSE,scope = allVar)
```

```
multi.fit.aic=multinom(formula = Label ~ X262 + X757 + X271 + X38 + X317 + X89 + X135 + X289 + X813 + X7

train_pred.forward <- predict(multi.fit.aic, data = data)
# Confusion Matrix and Accuracy
table(train_pred.forward,data[,833]) ; mean(train_pred.forward==data[,833])
```

```
##
## train_pred.forward   0   1   2   3   4   5
##                  0 380   8   0  11   2   0
##                  1  12 286   0   9   0   1
##                  2   0   0  58   0   0   0
##                  3   2   7   0 204   5   0
##                  4   1   2   0   3 139   0
##                  5   0   0   0   0   0 370
```

```
## [1] 0.958
```

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

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

```
multi.fit.2.pred = rep(NA,1000) # !!!
#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

```
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
##          0 362  16  20  26   3   0
##          1  16 277   0  21   3   0
##          2   7   0  38   0   0   0
##          3  10   6   0 178   8   0
##          4   0   1   0   2 132   0
##          5   0   3   0   0   0 371
##
##  Accuracy (average) : 0.9053
```

```
svm.fit
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1500 samples
##  832 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1198, 1201, 1201, 1199, 1201
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8266768  0.7778135
##   0.50  0.8806712  0.8486894
##   1.00  0.9053184  0.8802044
##
## Tuning parameter 'sigma' was held constant at a value of 0.00181931
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.00181931 and C = 1.
```

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

mode

```
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[,833]) ; mean(train_pred.mode==data[,833])
```

```
##
## train_pred.mode   0   1   2   3   4   5
##               0 384   2   0  12   0   0
##               1   7 294   0  10   1   0
##               2   2   0  58   0   0   0
##               3   2   5   0 204   4   0
##               4   0   1   0   1 141   0
##               5   0   1   0   0   0 371
```

```
## [1] 0.968
```

```
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_4x4all LDA_Label_4x4all KNN_Label_4x4all
## [1,] "5"              "5"              "3"
## [2,] "0"              "2"              "0"
## [3,] "0"              "0"              "3"
## [4,] "0"              "0"              "0"
## [5,] "0"              "0"              "0"
## [6,] "1"              "1"              "1"
##      Random forest_Label_4x4all Boosting Tree_Label_4x4all
## [1,] "5"                        "5"
## [2,] "0"                        "0"
## [3,] "3"                        "3"
## [4,] "0"                        "0"
## [5,] "0"                        "0"
## [6,] "1"                        "1"
##      Naive Bayes_Label_4x4all LASSO_Label_4x4all Forward Selection_Label_4x4all
## [1,] "5"                      "5"                "5"
## [2,] "2"                      "0"                "0"
## [3,] "3"                      "3"                "3"
## [4,] "1"                      "0"                "0"
## [5,] "4"                      "0"                "0"
## [6,] "1"                      "1"                "1"
##      Penalized Multinomial Regression_Label_4x4all SVM-radial_Label_4x4all
## [1,] NA                                            "5"
## [2,] NA                                            "0"
## [3,] NA                                            "3"
## [4,] NA                                            "0"
## [5,] NA                                            "0"
## [6,] NA                                            "1"
##      Mode_Label_4x4all
## [1,] "5"
## [2,] "0"
## [3,] "3"
## [4,] "0"
## [5,] "0"
## [6,] "1"
```

```r
write.csv(test_pred_response,"4x4all_label.csv", row.names = FALSE)
```