Group 7

2022/5/28

```
set.seed(1082)
data=read.csv("crop4x4.csv",header = T)
data.2=read.csv("testCrop4x4.csv",header = T)
data$Label <- factor(data$Label)
```

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

PCA(training data)

15 PCs can explain 89.77% 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
##          0 377   0   9  52   0   0
##          1   6 289   0  20  10   1
##          2  11   0  45   0   0   0
##          3   1  11   4 150  12   0
##          4   0   1   0   5 124   0
##          5   0   2   0   0   0 370
##
##  Accuracy (average) : 0.9033
```

```
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
##          0 329  10  19  60   6   0
##          1  30 276   0  23  10   4
##          2  33   0  39   1   0   0
##          3   3   8   0 119  18   0
##          4   0   1   0  24 112   2
##          5   0   8   0   0   0 365
##
##  Accuracy (average) : 0.8267
```

```
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
##          0 350  20  34  52   4   0
##          1  17 166   0  40  24  84
##          2   2   0  22   2   0   0
##          3  26   9   2  82  11   3
##          4   0   5   0  14  54  35
##          5   0 103   0  37  53 249
##
##  Accuracy (average) : 0.6153
```

```
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
##   52 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1200, 1201, 1201, 1200, 1198
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9440014  0.9293331
##   27    0.9553261  0.9437366
##   52    0.9519927  0.9395353
##
## 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
##          0 390   2   3  10   0   0
##          1   1 281   0  11   3   1
##          2   2   0  55   0   0   0
##          3   2  13   0 198   4   0
##          4   0   3   0   8 139   0
##          5   0   4   0   0   0 370
##
##  Accuracy (average) : 0.9553
```

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

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

Boosting Tree

```
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, 1200, 1200, 1200, 1201
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9179975  0.8962663
##   1                  100      0.9353153  0.9184022
##   1                  150      0.9433198  0.9285601
##   2                   50      0.9373264  0.9209071
##   2                  100      0.9526576  0.9403409
##   2                  150      0.9579954  0.9470945
##   3                   50      0.9539820  0.9420329
##   3                  100      0.9593199  0.9487675
##   3                  150      0.9639887  0.9546533
##
## 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(boosttree.fit,norm="none")
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are un-normalized aggregated counts)
##
##           Reference
## Prediction   0   1   2   3   4   5
##          0 391   1   4   5   0   0
##          1   0 285   0   6   4   1
##          2   2   0  54   0   0   0
##          3   2   9   0 211   7   0
##          4   0   4   0   5 135   0
##          5   0   4   0   0   0 370
```

```
##
##  Accuracy (average) : 0.964
```

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

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

Naive Bayes

```
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: 1198, 1199, 1202, 1200, 1201
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.8326203  0.7913409
##    TRUE      0.8159822  0.7706445
##
## 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.
```

```
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
##          0 301   2   1  53   3   2
##          1  25 266   0  16   1   2
##          2  66   0  57   3   0   0
##          3   3  23   0 137  21   0
##          4   0   8   0  18 121   0
##          5   0   4   0   0   0 367
##
##  Accuracy (average) : 0.8327
```

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

```
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] 0.003732408
```

```
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
##                0 385   7   4  24   0   0
##                1   1 286   0  15   2   1
##                2   3   0  54   0   0   0
##                3   6   6   0 183  10   0
##                4   0   2   0   5 134   0
##                5   0   2   0   0   0 370
```

```
## [1] 0.9413333
```

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

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

Forward Selection

```
# 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)
```

```
multi.fit.aic=multinom(formula = Label ~X2 + X19 + X18 + X11 + X5 + X6 + X12 +X49 + X39 + X50 + X32 + X4

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
##                  0 388   3   2  13   0   0
##                  1   0 288   0  12   0   0
##                  2   0   0  56   0   0   0
##                  3   7  11   0 197   7   0
##                  4   0   1   0   5 139   0
##                  5   0   0   0   0   0 371
```

```
## [1] 0.9593333
```

```
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
```

```
## 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, 1201, 1201, 1199
## Resampling results across tuning parameters:
##
##   decay  Accuracy   Kappa
##   0e+00  0.9072985  0.8835741
##   1e-04  0.9033250  0.8785589
##   1e-01  0.9139896  0.8914952
##
## 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
##          0 379   5   5  28   0   0
##          1   0 278   0  19   9   1
##          2   1   2  53   2   1   0
##          3  15  13   0 166  10   0
##          4   0   3   0  12 126   1
##          5   0   2   0   0   0 369
##
##  Accuracy (average) : 0.914
```

```
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
##          0 384  13  17  44   0   0
##          1   2 278   0  18   6   0
##          2   4   0  41   0   0   0
##          3   5   9   0 162  10   0
##          4   0   1   0   3 130   0
##          5   0   2   0   0   0 371
##
##  Accuracy (average) : 0.9107
```

```
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[,53]) ; mean(train_pred.mode==data[,53])
```

```
##
## train_pred.mode   0   1   2   3   4   5
##               0 395   2   0  24   1   0
##               1   0 292   0  16   1   1
##               2   0   0  58   0   0   0
##               3   0   7   0 185   5   0
##               4   0   1   0   2 139   0
##               5   0   1   0   0   0 370
```

```
## [1] 0.9593333
```

```
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

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

head(test_pred_response)
```

```
##      QDA_Label_4x4 LDA_Label_4x4 KNN_Label_4x4 Random forest_Label_4x4
## [1,] "5"           "5"           "1"           "5"
## [2,] "0"           "0"           "0"           "0"
## [3,] "0"           "0"           "0"           "0"
## [4,] "0"           "0"           "0"           "0"
## [5,] "0"           "0"           "0"           "0"
## [6,] "1"           "1"           "5"           "1"
##      Boosting Tree_Label_4x4 Naive Bayes_Label_4x4 LASSO_Label_4x4
## [1,] "5"                     "5"                   "5"
## [2,] "0"                     "2"                   "0"
## [3,] "3"                     "0"                   "0"
## [4,] "0"                     "1"                   "0"
## [5,] "0"                     "0"                   "0"
## [6,] "1"                     "1"                   "1"
##      Forward Selection_Label_4x4 Penalized Multinomial Regression_Label_4x4
## [1,] "5"                         "5"
## [2,] "0"                         "0"
## [3,] "3"                         "3"
## [4,] "0"                         "0"
## [5,] "0"                         "0"
## [6,] "1"                         "1"
##      SVM-radial_Label_4x4 Mode_Label_4x4
## [1,] "5"                  "5"
## [2,] "0"                  "0"
## [3,] "0"                  "0"
## [4,] "0"                  "0"
## [5,] "0"                  "0"
## [6,] "1"                  "1"
```

```
write.csv(test_pred_response,"4x4_label.csv", row.names = FALSE)
```