Group 7

2022/5/28

```
set.seed(1082)
data <- read.csv("crop and blur.csv", header = T)
data.2 <- read.csv("test_crop and blur.csv", header = T)
data$Label <- factor(data$Label)
```

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

PCA(training data)

37 PCs can explain 90.33 variation.

```
pca = princomp(data[,1:364],cor=T)

cumulative.var <- c()
PoV <- pca$sdev^2/sum(pca$sdev^2)
for (i in 1:364){
  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$scores[,19]
z20 <- pca$scores[,20]
z21 <- pca$scores[,21]
z22 <- pca$scores[,22]
```

```
z23 <- pca$scores[,23]
z24 <- pca$scores[,24]
z25 <- pca$scores[,25]
z26 <- pca$scores[,26]
z27 <- pca$scores[,27]
z28 <- pca$scores[,28]
z29 <- pca$scores[,29]
z30 <- pca$scores[,30]
z31 <- pca$scores[,31]
z32 <- pca$scores[,32]
z33 <- pca$scores[,33]
z34 <- pca$scores[,34]
z35 <- pca$scores[,35]
z36 <- pca$scores[,36]
z37 <- pca$scores[,37]

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[,1:364])
pca.test=pca.test[,1:37]
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
##          0 383   1  29  47   0   0
##          1   9 282   0  17  13   6
##          2   1   0   6   0   0   0
##          3   2  15  23 158  20   0
##          4   0   1   0   5 113   0
##          5   0   4   0   0   0 365
##
##  Accuracy (average) : 0.8713
```

```
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
##          0 339   6   9  55   5   1
##          1  14 276   0  20  13   0
##          2  41   0  49   0   0   0
##          3   1   8   0 130  10   1
##          4   0   6   0  22 118   0
##          5   0   7   0   0   0 369
##
##  Accuracy (average) : 0.854
```

```
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
##          0 368  13  31  54   2   1
##          1   4 194   0  47  25  56
##          2   5   0  20   4   0   0
##          3  18  17   7  65   9  10
##          4   0  13   0  25 101  10
##          5   0  66   0  32   9 294
##
##  Accuracy (average) : 0.6947
```

```
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
##  364 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1198, 1199, 1200, 1202, 1201
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2   0.9059979  0.8806913
##   183   0.9433254  0.9286435
##   364   0.9406631  0.9252489
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 183.
```

```
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
##          0 386   1   4  13   0   0
##          1   0 278   0  12   3   1
##          2   4   0  54   0   0   0
##          3   5  15   0 195  11   0
##          4   0   4   0   7 132   0
##          5   0   5   0   0   0 370
##
##  Accuracy (average) : 0.9433
```

```
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
##  364 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1200, 1201, 1200, 1199, 1200
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.9266596  0.9074372
##   1                  100      0.9479997  0.9344326
##   1                  150      0.9526620  0.9403317
##   2                   50      0.9533287  0.9411797
##   2                  100      0.9593198  0.9487489
##   2                  150      0.9599865  0.9495966
##   3                   50      0.9499864  0.9369407
##   3                  100      0.9579820  0.9470593
##   3                  150      0.9606465  0.9504238
##
## 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
##          0 390   1   2   9   0   0
##          1   0 287   1  11   6   1
##          2   0   0  55   0   0   0
##          3   5  11   0 201   2   0
##          4   0   1   0   6 138   0
##          5   0   3   0   0   0 370
```

```
##
##  Accuracy (average) : 0.9607
```

```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

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

```
## Naive Bayes
##
## 1500 samples
##  364 predictor
##    6 classes: '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1201, 1199, 1200, 1200, 1200
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.8046481  0.7572691
##    TRUE      0.8006591  0.7515772
##
## 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
##          0 271   0   0  52   0   0
##          1  47 268   0  16   4   3
##          2  74   0  56   8   0   0
##          3   3  21   1 131  29   0
##          4   0  10   1  20 113   0
##          5   0   4   0   0   0 368
##
##  Accuracy (average) : 0.8047
```

```
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] 0.003400832
```

```
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[,1:364])
allVar <- paste("~", paste(allNames, collapse=" + "))

multi.fit=multinom(Label~1,data=data, trace = F)
stepAIC(multi.fit, direction = "forward",trace = FALSE,scope = allVar)
```

```
## Call:
## multinom(formula = Label ~ X106 + X71 + X2 + X50 + X119 + X40 +
##     X5 + X18 + X63 + X108 + X122 + X102 + X12 + X107 + X72 +
##     X75 + X7 + X111 + X212 + X116 + X3, data = data, trace = F)
##
## Coefficients:
##    (Intercept)       X106           X71         X2        X50        X119
## 1 -105.6741583  -46.03277 -1.539435e-05  28.321657  -8.720923   -5.472934
## 2  -30.7072027 -132.67911 -1.423777e-05 103.283801  -4.591100 -144.700475
## 3  116.1605462  -50.92121 -1.874511e-05  79.985244 -11.629071  189.962149
## 4  -58.2536330  -62.90922 -1.888960e-05  32.052754  -9.986750   49.314494
## 5   -0.8154962  -11.25022 -2.100487e-06   1.152902 -28.714336    6.758840
##          X40          X5         X18        X63       X108        X122
## 1 -0.5597764 -0.02675277 -0.02955327   46.13913   18.85873  0.06645599
## 2 -2.8752493  0.17013236  1.02468173  -13.72508  -34.03791 -1.03136159
## 3 -4.7998893 -0.12994252 -0.31297132  -20.81759 -136.11312  0.37175636
## 4  3.0819829 -0.06354627 -1.15890981  165.27932  -16.89195  1.26326779
## 5 14.1664132 -0.26129605 -1.12716442 -100.43114  113.14897  1.16984709
##        X102       X12       X107       X72        X75         X7       X111
## 1 13.779091  71.08599 -0.3523680  77.26973 -145.10298  77.395776  -64.82867
## 2 10.925465  12.46035 -4.8006904  23.34796 -134.30545 115.525343 -112.74880
## 3  8.739565 -68.46182  0.9247795 139.68480  -71.10533  54.295326  -70.95660
## 4  7.661693 118.20032  4.5915894 137.18947 -101.53678 109.115536 -110.51949
## 5 18.014610 -42.63227  4.2014884  89.88446  -24.27010   5.946447   10.65773
##         X212       X116         X3
## 1 -7.7021108  -2.932319  0.3436173
## 2  2.0739622 -71.936505  4.7851846
## 3 -0.9820592  80.874321 -0.8986826
## 4 -6.7095798  35.349075 -4.5817079
## 5  1.8266046 -90.438140 -4.1493329
##
## Residual Deviance: 228.9225
## AIC: 448.9225
```

```
multi.fit.aic=multinom(formula = Label ~ X106 + X71 + X2 + X50 + X119 + X40 + X5 + X18 + X63 + X108 + X1
```

```
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

```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
##          0 385   7  38  59   2   0
##          1   8 282   0  24  14   1
##          2   0   0  20   0   0   0
##          3   2   8   0 135  11   2
##          4   0   2   0   9 119   0
##          5   0   4   0   0   0 368
##
##  Accuracy (average) : 0.8727
```

```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[,365]) ; mean(train_pred.mode==data[,365])
```

```
##
## train_pred.mode   0   1   2   3   4   5
##               0 395   1   0  31   0   0
##               1   0 298   0  12   0   0
##               2   0   0  58   0   0   0
##               3   0   3   0 184   3   0
##               4   0   0   0   0 143   0
##               5   0   1   0   0   0 371
```

```
## [1] 0.966
```

```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

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

head(test_pred_response)
```

```
##      QDA_Label_cb LDA_Label_cb KNN_Label_cb Random forest_Label_cb
## [1,] "5"          "5"          "5"          "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_cb Naive Bayes_Label_cb LASSO_Label_cb
## [1,] "5"                    "5"                  "5"
## [2,] "0"                    "0"                  "0"
## [3,] "0"                    "0"                  "0"
## [4,] "0"                    "1"                  "0"
## [5,] "0"                    "0"                  "0"
## [6,] "1"                    "1"                  "1"
##      Forward Selection_Label_cb Penalized Multinomial Regression_Label_cb
## [1,] "5"                        NA
## [2,] "0"                        NA
## [3,] "3"                        NA
## [4,] "0"                        NA
## [5,] "0"                        NA
## [6,] "1"                        NA
##      SVM-radial_Label_cb Mode_Label_cb
## [1,] "5"                 "5"
## [2,] "0"                 "0"
## [3,] "0"                 "0"
## [4,] "0"                 "0"
## [5,] "0"                 "0"
## [6,] "1"                 "1"
```

```
write.csv(test_pred_response,"crop and blur_label.csv", row.names = FALSE)
```