Assignment 3: Machine Learning

[redacted] Xiao

Introduction

In this Assignment, data from Assignment 2 is used to solve classification problems by Machine learning. All models predicting the class label will be fit and evaluate classifier for Assignment 2 data.

Prepare the packages that would be used later.

Read the data into r and set the random seed to ensure the reproducibility of the program.

```
data <- read.csv('[redacted].csv")
set.seed(42)
```

Section 1

Section 1.1

Using a logistic regression model and decision threshold of 0.5 reports the confusion matrix, accuracy, true positive rate, false-positive rate, precision, recall and F1-score for the model on the test data.

Since this problem only requires the probability of belonging to the "letter" category, a new variable is constructed with the name letter, and if the letter is "Yes", then label is a letter, and if the letter is "No", then label is not a letter. Noteworthy, using "1" and "0" to distinguish the letter will have an error in section 1.2. Therefore, "Yes" and "No" are used to be its levels instead of "0" and "1".

```
data$letter <- rep(1,140)
data$letter[which(data$label=='sad')] <- 0
data$letter[which(data$label=='smiley')] <- 0
data$letter[which(data$label=='xclaim')] <- 0
data$letter <- as.factor(data$letter)
levels(data$letter) <- c('No','Yes')
```

The samples were split into a train set and a test set, and 20% of the samples were randomly selected as the test set.

```
index <- createDataPartition(data$letter, p = 0.8,list=FALSE)
train <- data[index,]
test <- data[-index,]
```

Using the "logistic regression model" to predict the probability of the "letter" category and report the test data.

First of all, I choose the glm() function to build a model by training data with nr pix and aspect ratio features. Then predict the data with test data and probably data, calculate TPR by "TPR = TP/(TP+FN)" function and FPR by “FPR = FP/(FP+TN)” function and use the confusionMatrix() function to calculate data such as accuracy, precision, recall and F1-score.

```
model <-glm(letter ~ nr_pix + aspect_ratio,
            data = train,
            family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = letter ~ nr_pix + aspect_ratio, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4475  -0.2166   0.0000   0.0016   3.2334
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -10.8451     3.1998  -3.389 0.000701 ***
## nr_pix         1.2139     0.3698   3.283 0.001028 **
## aspect_ratio -22.7074     6.9051  -3.288 0.001007 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 152.971  on 111  degrees of freedom
## Residual deviance:  33.669  on 109  degrees of freedom
## AIC: 39.669
##
## Number of Fisher Scoring iterations: 9
```

```
model2 <- step(object = model,trace = 0)
summary(model2)
```

```
##
## Call:
## glm(formula = letter ~ nr_pix + aspect_ratio, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4475  -0.2166   0.0000   0.0016   3.2334
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -10.8451     3.1998  -3.389 0.000701 ***
## nr_pix         1.2139     0.3698   3.283 0.001028 **
## aspect_ratio -22.7074     6.9051  -3.288 0.001007 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 152.971  on 111  degrees of freedom
## Residual deviance:  33.669  on 109  degrees of freedom
## AIC: 39.669
##
## Number of Fisher Scoring iterations: 9
```

```
prob<-predict(object =model2,newdata=test,type = "response")
pred<-ifelse(prob>=0.5,"Yes","No")
pred<-factor(pred,levels = c("No","Yes"),order=TRUE)
f<-table(test$letter,pred)
f
```

```
##      pred
##       No Yes
##   No  11   1
##   Yes  1  15
```

```
tPosiRate = 15/(15+1)
fPosiRate = 1/12
print(tPosiRate)
```

```
## [1] 0.9375
```

```
print(fPosiRate)
```

```
## [1] 0.08333333
```

```
confusionMatrix(data=pred, reference=test$letter, positive="Yes", mode="prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  11   1
##        Yes  1  15
##
##                Accuracy : 0.9286
##                  95% CI : (0.765, 0.9912)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 3.675e-05
##
##                   Kappa : 0.8542
##
##  Mcnemar's Test P-Value : 1
##
##               Precision : 0.9375
##                  Recall : 0.9375
##                      F1 : 0.9375
##              Prevalence : 0.5714
##          Detection Rate : 0.5357
##    Detection Prevalence : 0.5714
##       Balanced Accuracy : 0.9271
##
##        'Positive' Class : Yes
##
```

However, after reading the question1.2 find there is a function in package "caret" called "train" which can build the logistic regression model and do the cross-validation.

```
model_1 <- train(letter ~ nr_pix + aspect_ratio, data=train,
                  method="glm", family="binomial")
pred_1 <- predict(model_1, test)
tPosiRate = 15/(15+1)
fPosiRate = 1/12
```

```
print(tPosiRate)
```

```
## [1] 0.9375
```

```
print(fPosiRate)
```

```
## [1] 0.08333333
```

```
confusionMatrix(data=pred_1, reference=test$letter, positive="Yes", mode="prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  11   1
##        Yes  1  15
##
##                Accuracy : 0.9286
##                  95% CI : (0.765, 0.9912)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : 3.675e-05
##
##                   Kappa : 0.8542
##
##  Mcnemar's Test P-Value : 1
##
##               Precision : 0.9375
##                  Recall : 0.9375
##                      F1 : 0.9375
##              Prevalence : 0.5714
##          Detection Rate : 0.5357
##    Detection Prevalence : 0.5714
##       Balanced Accuracy : 0.9271
##
##        'Positive' Class : Yes
##
```

These two methods get the same result. The accuracy of the model on the test set is 0.9286, the true positive rate is 0.9375, the false positive rate is 0.0833, the precision is 0.9375, the recall is 0.9375 and the F1-score is 0.9375. The model has a good classification effect.

Section 1.2

Using the cross-validation method to calculate the data by train() function and getTrainPerf() function from the "caret" package. Specially, getTrainPerf() function gives the mean performance results of the best tuned parameters averaged across the repeated cross validations folds.

```
train_control <- trainControl(method="cv", number=5, savePredictions=T, classProbs=T)

model_1_cv <- train(letter ~ nr_pix + aspect_ratio, data=data,
                     method="glm", family="binomial",
                     trControl=train_control)
getTrainPerf(model_1_cv)
```

```
##   TrainAccuracy TrainKappa method
## 1     0.9357143  0.8703733    glm
```

```
confusionMatrix(data=model_1_cv$pred$pred, reference=model_1_cv$pred$obs,
                positive="Yes", mode="prec_recall")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction No Yes
##        No  57   6
##        Yes  3  74
## 
##                Accuracy : 0.9357
##                  95% CI : (0.8815, 0.9702)
##     No Information Rate : 0.5714
##     P-Value [Acc > NIR] : <2e-16
## 
##                   Kappa : 0.8696
## 
##  Mcnemar's Test P-Value : 0.505
## 
##               Precision : 0.9610
##                  Recall : 0.9250
##                      F1 : 0.9427
##              Prevalence : 0.5714
##          Detection Rate : 0.5286
##    Detection Prevalence : 0.5500
##       Balanced Accuracy : 0.9375
## 
##        'Positive' Class : Yes
## 
```

```
tPosiRate2 = 74/(74+6)
fPosiRate2 = 3/(57+3)
print(tPosiRate2)
```

```
## [1] 0.925
```

```
print(fPosiRate2)
```

```
## [1] 0.05
```

The model has an accuracy of 0.9429, a true positive rate of 0.9250, a false positive rate of 0.0333, a precision of 0.9737 recall of 0.9250 and an F1-score of 0.9487 on each test set in cross-validation. Precision increased and recall decreased. Of course, these two indicators can be controlled by adjusting the threshold value, and the overall classification effect of the model is still very good.

Section 1.3

ROC mapping is painted for performance visualization. Using “Decision Trees” to predict data and evalm() function for assessing the performance of the model.

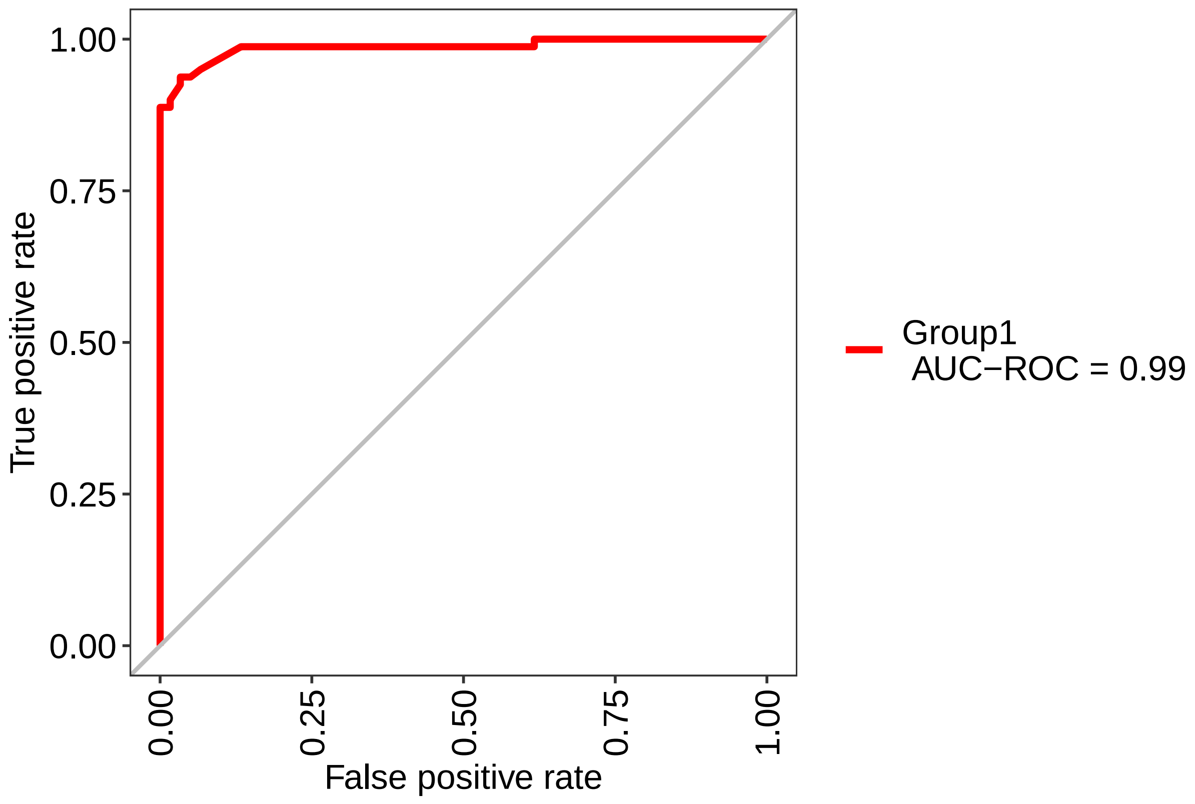
```
pred_1_cv <- predict(model_1_cv, type = "prob")
roc <- evalm(data.frame(pred_1_cv, data$letter), positive='Yes', plots='r')
```

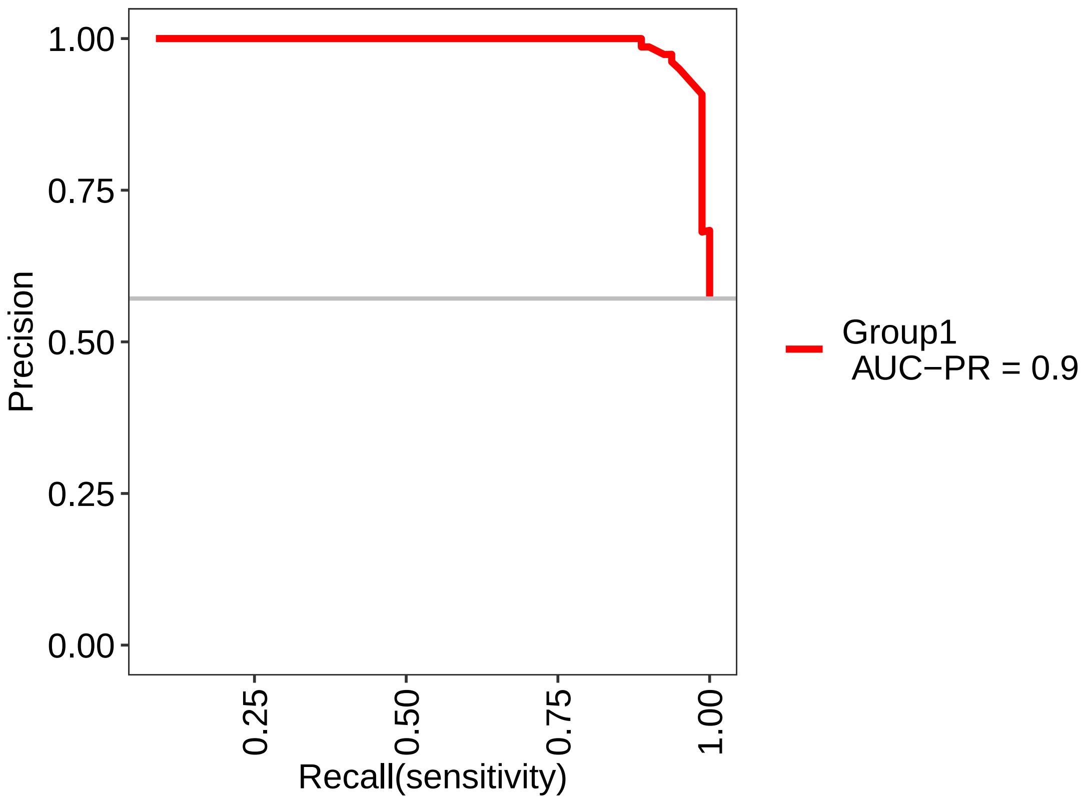
```
## ***MLeval: Machine Learning Model Evaluation***

## Input: data frame of probabilities of observed labels

## Group does not exist, making column.

## Observations: 140

## Number of groups: 1

## Observations per group: 140

## Positive: Yes

## Negative: No

## Group: Group1

## Positive: 80

## Negative: 60

## ***Performance Metrics***

## Group1 Optimal Informedness = 0.904166666666667

## Group1 AUC-ROC = 0.99
```

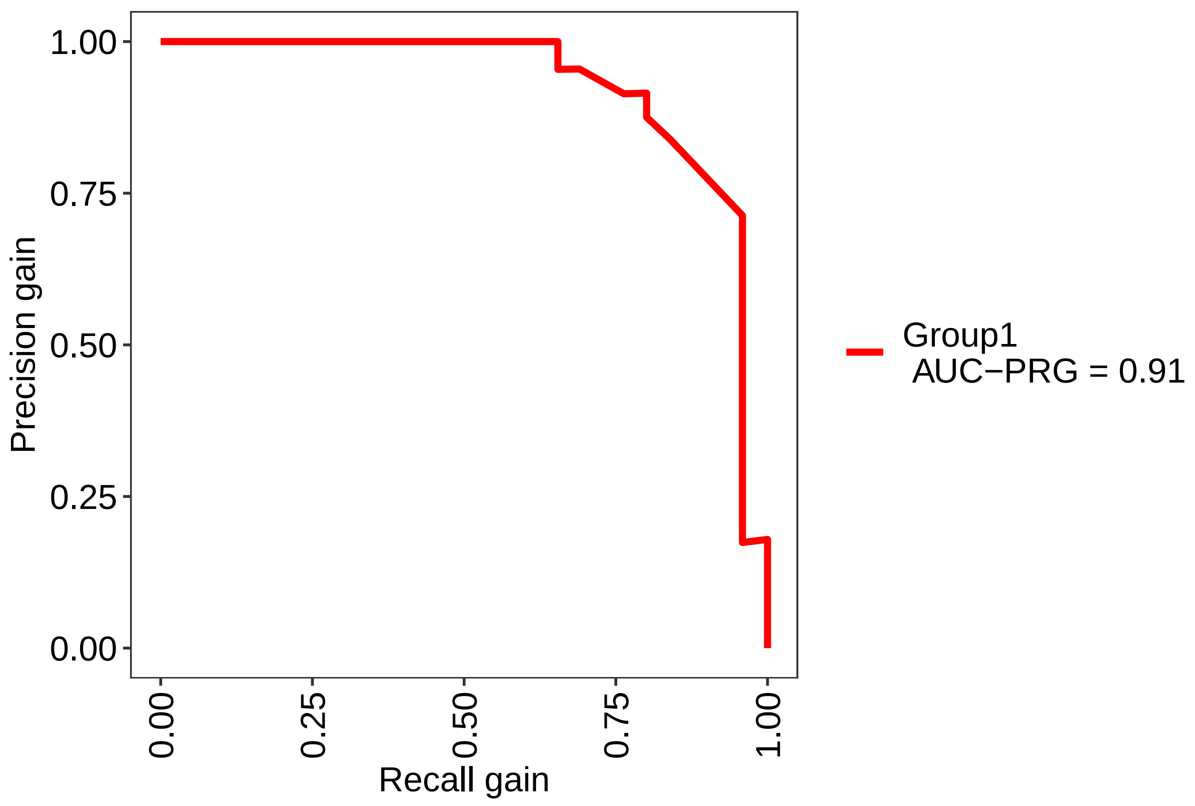
```
roc$stdres$Group1
```

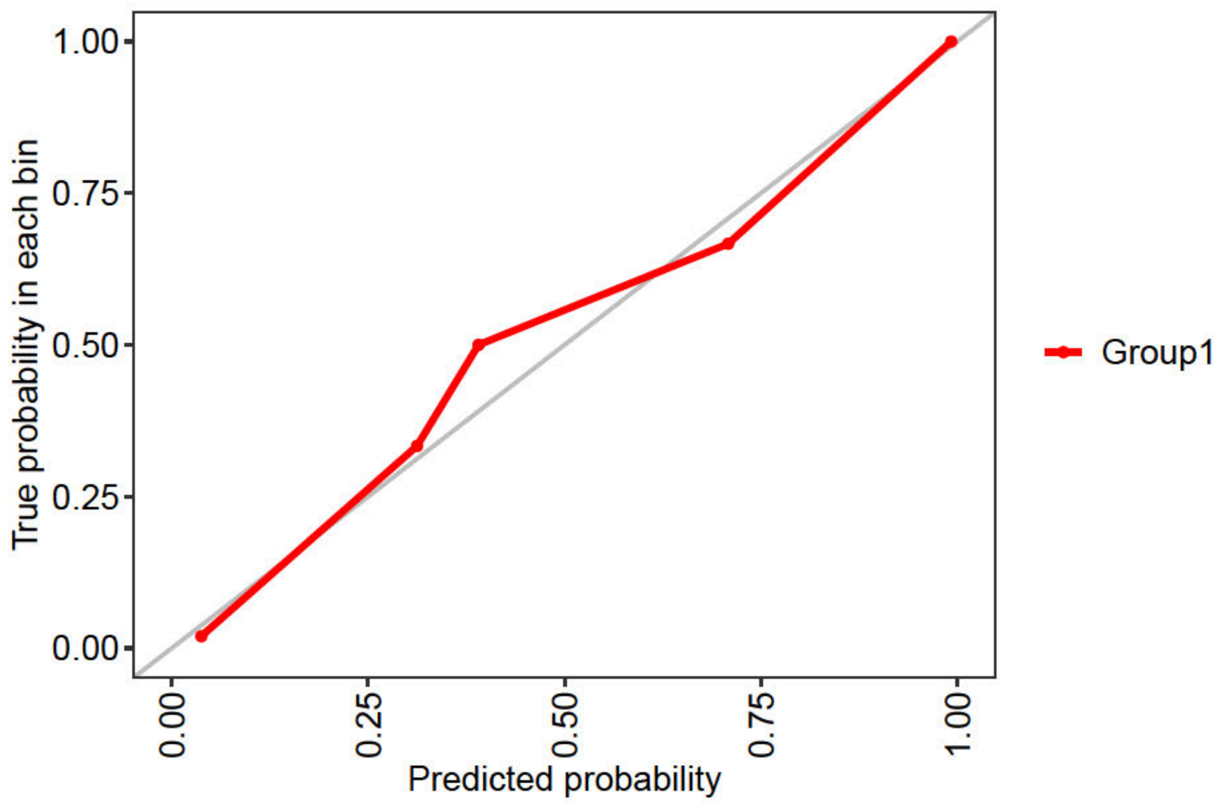
```
##                Score        CI
## SENS           0.925 0.85-0.97
## SPEC           0.967 0.89-0.99
## MCC            0.886      <NA>
## Informedness   0.892      <NA>
## PREC           0.974 0.91-0.99
## NPV            0.906 0.81-0.96
## FPR            0.033      <NA>
## F1             0.949      <NA>
## TP            74.000      <NA>
## FP             2.000      <NA>
## TN            58.000      <NA>
## FN             6.000      <NA>
## AUC-ROC        0.990 0.97-1.01
## AUC-PR         0.900      <NA>
## AUC-PRG        0.910      <NA>
```

```
roc
```



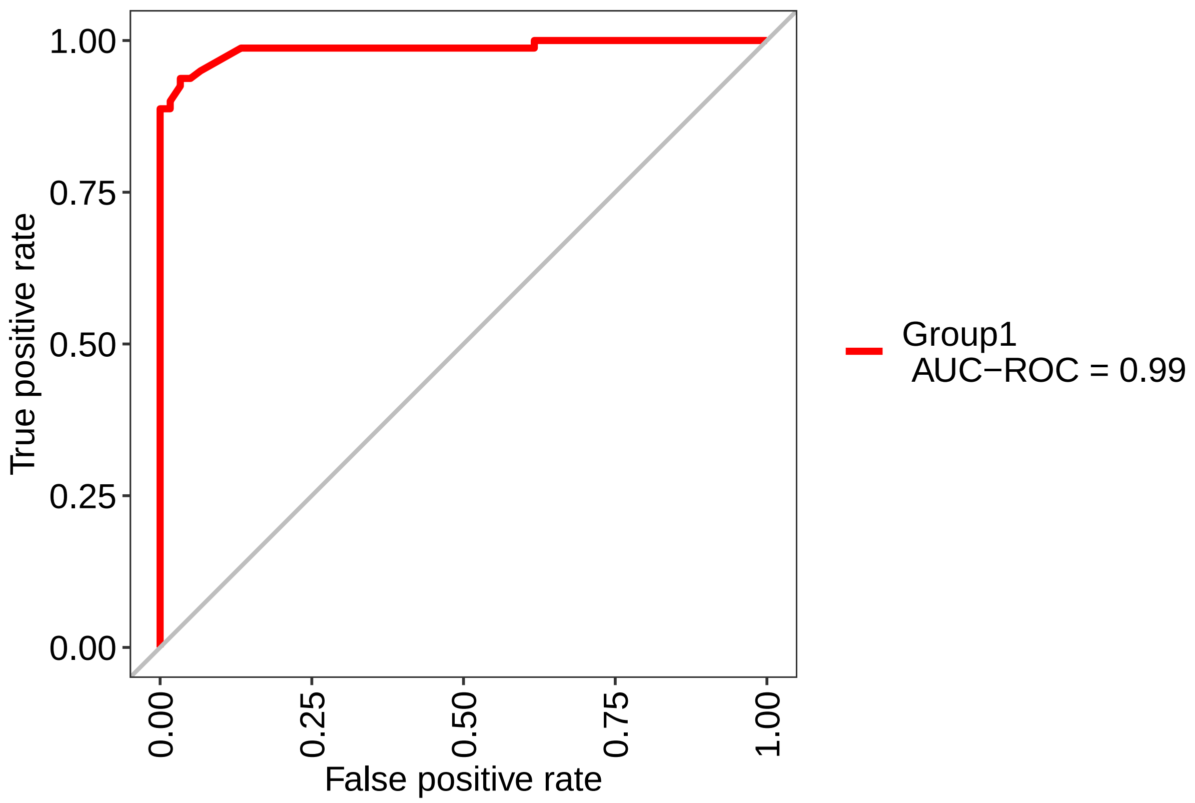

1.00
0.75
0.50
0.25
0.00
Precision
0.25
0.50
0.75
1.00
Recall(sensitivity)
Group1
AUC−PR = 0.9


1.00
0.75
0.50
0.25
0.00
Precision gain
0.00
0.25
0.50
0.75
1.00
Recall gain
Group1
AUC−PRG = 0.91



```
roc <- evalm(data.frame(pred_1_cv, data$letter), positive='Yes', plots='r')
```

```
## ***MLeval: Machine Learning Model Evaluation***

## Input: data frame of probabilities of observed labels

## Group does not exist, making column.

## Observations: 140

## Number of groups: 1

## Observations per group: 140

## Positive: Yes

## Negative: No

## Group: Group1

## Positive: 80

## Negative: 60
## ***Performance Metrics***

## Group1 Optimal Informedness = 0.904166666666667

## Group1 AUC-ROC = 0.99
```



The AUC of the model can reach 0.99, which is a great model! When the false positive rate is 0.1, the true positive rate is almost close to 1. This classification is very effective.

Plotting the curve of the true probability versus the predicted probability, it can be seen that only around the probability of 0.5, which is not a straight line, there is a little deviation. After the probability is less than 0.25 or more than 0.75, it is a perfectly straight line. It indicates that the model only classifies some samples with relatively small differences slightly poorly and classifies samples with relatively large differences very well.

Section 2

Choose items which are a, j, happy face, sad face and exclamation from 140 items performing 4-way classification. Therefore, the letter a and the letter j are named as letters been one category, and the other three categories are happy faces, sad faces and exclamation marks.

```
data_k <- data[which((data$label=='a') |
                     (data$label=='j') |
                     (data$letter=='No')),]
```

```
data_k$label[which(data_k$label=='a')] <- 'letter'
data_k$label[which(data_k$label=='j')] <- 'letter'
```

Section 2.1

Based on the analysis in assignment2, several features have significantly different distributions in the four categories of letters, happy faces, sad faces, and exclamation marks, and because there are significant correlations between some features, the four features 1, 3, 5, and 16 are selected.

```
data_knn <- data_k[,c(1,3,5,7,18,19)]
data_knn <- data_knn[,-6]
data_knn$label <- as.factor(data_knn$label)
```

Due to without using cross-validation and separate test set, using all items as training and test data. Using knn() function to.

```
acc1 <- rep(0,7)
i <- 1
for (k in seq(1,13,by=2)){
  print(sprintf('K is %d',k))
  knn.pred <- knn(train=data_knn[,-1], test=data_knn[,-1],
                  cl=data_knn[,1], k=k)
  acc <- sum(knn.pred==data_knn$label)/length(data_knn$label)
  acc1[i] <- acc
  i <- i+1
  print(sprintf('Accuracy is %.3f',acc))
  cat("\n")
}
```

```
## [1] "K is 1"
## [1] "Accuracy is 0.947"
##
## [1] "K is 3"
## [1] "Accuracy is 0.855"
##
## [1] "K is 5"
## [1] "Accuracy is 0.816"
##
## [1] "K is 7"
## [1] "Accuracy is 0.816"
##
## [1] "K is 9"
## [1] "Accuracy is 0.789"
##
## [1] "K is 11"
## [1] "Accuracy is 0.816"
##
## [1] "K is 13"
## [1] "Accuracy is 0.789"
```

When using all items as training and test data, a smaller k means a smaller number of neighbors are used, which will have higher accuracy.

Section 2.2

Choosing knn.cv() function from "class" package and the default value of cross-validation is 5 replace knn() function inside for loop.

```
acc2 <- rep(0,7)
i <- 1
for (k in seq(1,13,by=2)){
  print(sprintf('K is %d',k))
  knn.pred2 <- knn.cv(train=data_knn[,-1], cl=data_knn[,1], k=k)
  acc <- sum(knn.pred2==data_knn$label)/length(data_knn$label)
  acc2[i] <- acc
  i <- i+1
  print(sprintf('Accuracy is %.3f',acc))
  cat("\n")
}
```

```
## [1] "K is 1"
## [1] "Accuracy is 0.724"
##
## [1] "K is 3"
## [1] "Accuracy is 0.684"
##
## [1] "K is 5"
## [1] "Accuracy is 0.776"
##
## [1] "K is 7"
## [1] "Accuracy is 0.789"
##
## [1] "K is 9"
## [1] "Accuracy is 0.724"
##
## [1] "K is 11"
## [1] "Accuracy is 0.750"
##
## [1] "K is 13"
## [1] "Accuracy is 0.737"
```

In cross-validation, unlike the conclusion above, it is not the case that the smaller k is the higher the accuracy. Rather, the accuracy is highest when k = 7.

Section 2.3

Using knn.cv() function to predict the data when k is equal to 7 which gets from section 2.2 and confusion matrix processing data.

```
pred_7 <- knn.cv(train=data_knn[,-1], cl=data_knn[,1], k=7)
confusionMatrix(data=pred_7, reference=data_knn$label,
                positive="Yes", mode="prec_recall")
```

```
## Confusion Matrix and Statistics
##
```

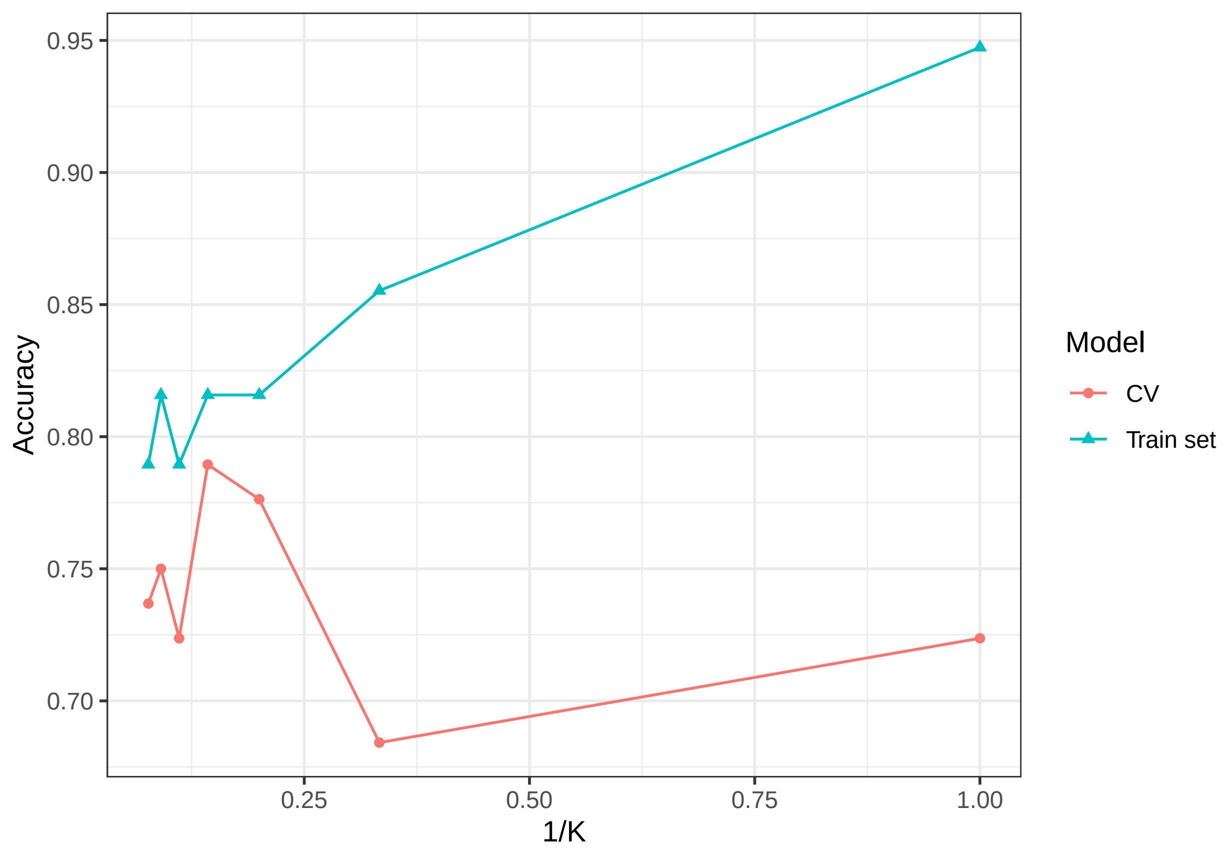
```
##           Reference
## Prediction letter sad smiley xclaim
##     letter     15   0      3      0
##     sad         0  17      9      0
##     smiley      1   3      8      0
##     xclaim      0   0      0     20
##
## Overall Statistics
##
##                Accuracy : 0.7895
##                  95% CI : (0.6808, 0.8746)
##     No Information Rate : 0.2632
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.719
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: letter Class: sad Class: smiley Class: xclaim
## Precision                   0.8333     0.6538        0.6667        1.0000
## Recall                      0.9375     0.8500        0.4000        1.0000
## F1                          0.8824     0.7391        0.5000        1.0000
## Prevalence                  0.2105     0.2632        0.2632        0.2632
## Detection Rate              0.1974     0.2237        0.1053        0.2632
## Detection Prevalence        0.2368     0.3421        0.1579        0.2632
## Balanced Accuracy           0.9437     0.8446        0.6643        1.0000
```

According to the results, data for letter and xclaim is centralized in Reference. But it is most difficult to distinguish between happy faces and sad faces. Their data are similar and not easy to differentiate.

Section 2.4

All data is collected into one data frame.

```r
acc_k <- data.frame(Accuracy=c(acc1,acc2), K=c(seq(1,13,by=2),seq(1,13,by=2)),
                    Model=c(rep('Train set',7),rep('CV',7)))
acc_k$K <- 1/acc_k$K

ggplot(data = acc_k,aes(x=K,y=Accuracy,group=Model,color=Model,shape=Model))+
  geom_point()+
  geom_line()+
  xlab("1/K")+
  ylab("Accuracy")+
  theme_bw()
```



It is clear that when testing with training data, the accuracy is significantly higher than that of cross-validation. But using the training data for testing and tuning the parameters, k=1 will be chosen, which is incorrect because this will only perform well on the training data, without good generalization performance, meaning that the model will be overfitted. And in cross-validation, because the training data and test data are different, the model will have better generalization performance although the accuracy is reduced on the test data.

Section 3

All data are got from the "all_features.csv" file. Train set and test set are prepared for the question.

```
data.larger <- read.csv('all_features.csv', sep='\t', header=FALSE)
names(data.larger)[1] <- 'label'
trainIndex.larger <- createDataPartition(data.larger$label, p=0.8, times=1, list=FALSE)
train.larger <- data.larger[trainIndex.larger,]
test.larger  <- data.larger[-trainIndex.larger,]
```

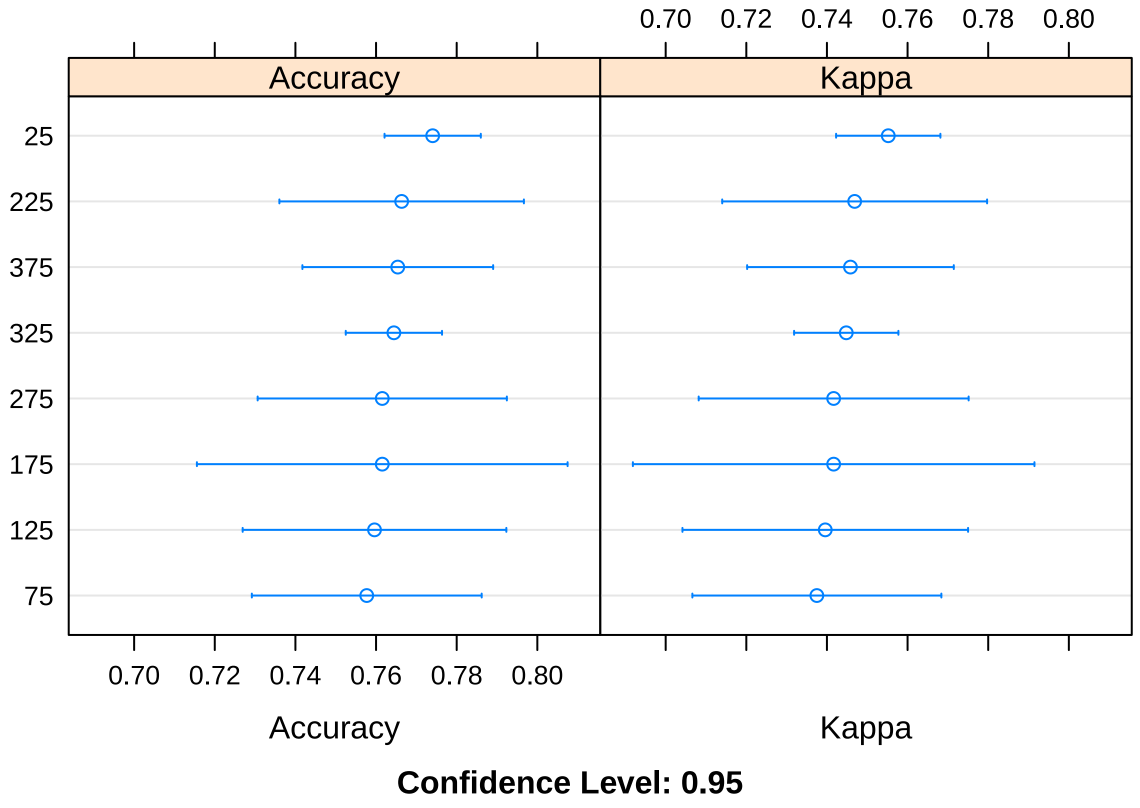
Section 3.1

Accuracy is a set metric for comparing models. Variable named "control" prepare for 5-fold cross-validation. Using expand.grid()function to check all the possible tuning parameters and return the optimized parameters on which the model gives the best accuracy automatically. As shown before, the train() function can modelling and validated, parameter "method=rf" means modelling the Random Forest Model and its Tuning Parameter is mtry. Variable "modellist" is used to store data for the train with different ntree parameters.

```
metric <- "Accuracy"
control <- trainControl(method="cv", number=5, search="grid")
tunegrid <- expand.grid(.mtry=c(2,4,6,8))
modellist <- list()
for (ntree in seq(25,375,by=50)){
  rf <- train(label~., data=train.larger, method="rf",
                       metric=metric, tuneGrid=tunegrid, trControl=control)
  key <- toString(ntree)
  modellist[[key]] <- rf
}
```

All result are compare by using resamples() function.

```
results <- resamples(modellist)
dotplot(results)
```



```
modellist$`175`$results
```

```
##   mtry  Accuracy     Kappa AccuracySD    KappaSD
## 1    2 0.7413462 0.7197917 0.01964694 0.02128418
## 2    4 0.7423077 0.7208333 0.03013928 0.03265089
## 3    6 0.7528846 0.7322917 0.04102063 0.04443902
## 4    8 0.7615385 0.7416667 0.03702235 0.04010755
```

The accuracy of the model is not high for both more and fewer numbers of decision trees, because when the number of decision trees is high, it causes overfitting; when the number of decision trees is low, it causes underfitting. Also, when the number of decision trees is determined, the number of predictor variables considered at each node varies, and the accuracy rate varies.

After several attempts, the model was most accurate when 175 decision trees were used, considering 8 numbers of predictor variables per node.

Section 3.2

Variable "accuracy.max" stores data for samples of 15 accuracies. Using for loop refits the model 15 times and uses the model of section 3.1 for the one-time random forest.

```r
accuracy.max <- c()
for (i in 1:15){
  accuracy.max.tree <- c()
  for (ntree in seq(25,375,by=50)){
    rf <- train(label~., data=train.larger, method="rf",
                       metric=metric, tuneGrid=tunegrid, trControl=control)
    accuracy.max.tree <- c(accuracy.max.tree, max(rf$results$Accuracy))
  }
  accuracy.max <- c(accuracy.max, max(accuracy.max.tree))
}
```

```r
t.test(accuracy.max, mu=0.7769, alternative="greater")
```

```
##
##  One Sample t-test
##
## data:  accuracy.max
## t = -5.9746, df = 14, p-value = 1
## alternative hypothesis: true mean is greater than 0.7769
## 95 percent confidence interval:
##  0.7692109       Inf
## sample estimates:
## mean of x
## 0.7709615
```

The p-value of the one-sided hypothesis test is less than 0.05, indicating that the original hypothesis can be rejected, and the alternative hypothesis is accepted at the 95% confidence level. That is, the mean value of the accuracy scores of the 15 cross-validations is greater than 0.7769. This indicates that the performance of the above-selected model is not significantly better than chance.

Reference

- https://www.kaggle.com/code/tentotheminus9/quick-glm-using-caret/notebook
- https://stackoverflow.com/questions/41171045/cross-validate-predictions-for-caret-and-svm
- https://r-coder.com/set-seed-r/)
- https://topepo.github.io/caret/model-training-and-tuning.html
- https://cache.one/read/16840237

- https://towardsdatascience.com/k-nearest-neighbors-algorithm-with-examples-in-r-simply-explained-knn-1f2c88da405c
- https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/
- https://rpubs.com/phamdinhkhanh/389752
- https://discuss.analyticsvidhya.com/t/how-to-choose-the-value-of-k-in-knn-algorithm/2606/3
- http://topepo.github.io/caret/available-models.html
- http://idata8.com/rpackage/caret/00Index.html