```r
#Set up
setwd("/Users/sunanqi/Desktop/BIOS626/Midterm")
```

```r
#Read data
data <- readLines("training_data.txt")
header <- unlist(strsplit(data[1], "\\s{1,}"))
data <- data[-1]
df <- read.table(text = data, header = FALSE, sep = "\t")
colnames(df) <- header

data2 <- readLines("test_data.txt")
header2 <- unlist(strsplit(data2[1], "\\s{1,}"))
data2 <- data2[-1]
df2 <- read.table(text = data2, header = FALSE, sep = "\t")
colnames(df2) <- header2
```

```r
#Categorize the activity level to binary variable df$activity_type (task1)
##or variable of level 1-7 df$activity_type2 (task2)
df$activity_type <- ifelse(df$activity %in% c(4,5,6,7,8,9,10,11,12), 0, 1)
df$activity_type2 <- ifelse(df$activity %in% c(7,8,9,10,11,12), 7, df$activity)
df$activity_type_factor <- factor(df$activity_type2, levels=1:7)
```

```r
#task1 - apply GLM
logit_model <- glm(activity_type ~ ., data = df[, 3:564], family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
df2$pred_activity_prob <- predict(logit_model, newdata = df2, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
df2$pred_activity_type <- ifelse(df2$pred_activity_prob > 0.5, 1, 0)
write.table(df2$pred_activity_type, "binary_5823.txt", col.names = FALSE,
            row.names = FALSE, quote = FALSE)
```

```r
#final algorithm of task2: apply svm_2
library(e1071)
svm_model <- svm(df[, c(3:563)], df$activity_type_factor, type = "C-classification",
                 kernel = "linear")
df2$pred_activity_type2_svm2 <- predict(svm_model,df2[, c(2:562)])
df2$pred_activity_type2_svm2[1:50]
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 4 4 4 4 4 4 4 4 4 4 4 5 5 7 5 5 5 5 5 5 5
## [39] 5 5 5 5 5 5 5 5 5 7 7 7
## Levels: 1 2 3 4 5 6 7
```

```
write.table(df2$pred_activity_type2_svm2, "multiclass_7865.txt", col.names = FALSE,
            row.names = FALSE, quote = FALSE)
```

```
#first attempt of task2: apply randomForest
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
rf_model <- randomForest(activity_type_factor ~ ., df[, c(3:563, 566)])
df2$pred_activity_type2_rf <- predict(rf_model, df2)
df2$pred_activity_type2_rf[1:50]
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 5 5 4 4 5 4 4 4 4 5 5 5 5 7 5 5 5 5 5 5 5
## [39] 5 5 5 5 5 5 5 5 5 7 7 7
## Levels: 1 2 3 4 5 6 7
```

```
#Other trials after first attempt:
#Trial 1: apply svm_1
library(e1071)
svm_model <- svm(df[, c(3:563)], df$activity_type_factor)
df2$pred_activity_type2_svm1 <- predict(svm_model, df2[, c(2:562)])
df2$pred_activity_type2_svm1[1:50]
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 4 4 4 4 4 4 4 5 4 4 4 5 5 7 5 5 5 5 5 5 5
## [39] 5 5 5 5 5 5 5 5 5 7 7 7
## Levels: 1 2 3 4 5 6 7
```

```
#Trial 2: apply lda
library(MASS)
lda_model <- lda(activity_type_factor ~ ., df[, c(3:563, 566)])
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
df2$pred_activity_type2_lda <- predict(lda_model, df2[, c(2:562)])
df2$pred_activity_type2_lda$class[1:50]
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 4 4 4 4 4 4 4 4 4 4 4 5 5 7 5 5 5 5 5 5 5
## [39] 5 5 5 5 5 5 5 5 5 7 7 7
## Levels: 1 2 3 4 5 6 7
```

```
#Trial 3: apply knn (with k = 1, 3, 5, 10)
library(class)
knn_model <- knn(train = df[, 3:563], test = df2[, 2:562], cl = df$activity_type_factor, k = 10)
df2$pred_activity_type2_knn <- factor(knn_model, levels = 1:7)
df2$pred_activity_type2_knn[1:50]
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 2 5 4 4 5 5 4 4 5 5 4 5 5 5 7 5 5 5 5 5 5 5
## [39] 5 5 5 5 5 5 5 5 5 2 7 7
## Levels: 1 2 3 4 5 6 7
```

```r
#Trial 4: apply gbm
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```r
gbm_model <- gbm(
  formula = activity_type_factor ~ .,
  df[, c(3:563, 566)],
  distribution = "multinomial",
  n.trees = 100,
  interaction.depth = 2,
  shrinkage = 0.1,
  n.minobsinnode = 10,
  bag.fraction = 0.5,
  cv.folds = 5,
  keep.data = TRUE,
  verbose = TRUE
)
```

```
## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.9459             nan     0.1000    0.7441
##      2        1.5200             nan     0.1000    0.4282
##      3        1.2754             nan     0.1000    0.2879
##      4        1.1087             nan     0.1000    0.2590
##      5        0.9592             nan     0.1000    0.2250
##      6        0.8316             nan     0.1000    0.1539
##      7        0.7423             nan     0.1000    0.1334
##      8        0.6650             nan     0.1000    0.1113
##      9        0.6003             nan     0.1000    0.0846
##     10        0.5491             nan     0.1000    0.0879
##     20        0.2685             nan     0.1000    0.0195
##     40        0.1271             nan     0.1000    0.0032
##     60        0.0806             nan     0.1000    0.0006
##     80        0.0581             nan     0.1000    0.0010
##    100        0.0448             nan     0.1000    0.0003
```

```r
gbm_pred <- predict(gbm_model, newdata = df2[, c(2:562)], n.trees = 100)
df2$pred_activity_type2_gbm <- apply(gbm_pred, 1, which.max)
df2$pred_activity_type2_gbm[1:50]
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 2 4 5 5 4 4 4 5 5 5 5 5 5 7 5 5 5 5 5 5 5
## [39] 5 5 5 5 5 5 5 5 5 2 7 7
```

```r
#Trial 5: apply naive bayes
library(e1071)
nb_model <- naiveBayes(activity_type_factor ~ ., data = df[, c(3:563, 566)])
df2$pred_activity_type2_nb <- predict(nb_model, newdata = df2[, c(2:562)])
df2$pred_activity_type2_nb[1:50]
```

```
##  [1] 5 5 4 4 4 4 4 4 4 4 4 4 4 4 4 5 3 4 4 4 4 4 4 4 4 4 4 4 4 4 7 4 4 4 5 5 5 5
## [39] 5 4 4 4 4 4 4 4 7 7 7 7
## Levels: 1 2 3 4 5 6 7
```

```r
#Trial 6: apply decision tree
library(rpart)
library(rpart.plot)
tree_model <- rpart(activity_type_factor ~ ., data=df[,c(3:563,566)])
df2$pred_activity_type2_dt <- predict(tree_model, df2[, c(2:562)], type="class")
df2$pred_activity_type2_dt[1:50]
```

```
##  [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 5 5 5 5 5 5 5 5 5 5 5 5 5 7 5 5 5 5 5 5 5
## [39] 5 5 5 5 5 5 5 5 5 2 7 7
## Levels: 1 2 3 4 5 6 7
```

```r
#Comparing the result of svm_1 and svm_2 to see how many estimations are different
n = 0
for (i in 1:3162) {
  if (df2$pred_activity_type2_svm1[i] != df2$pred_activity_type2_svm2[i]) {
    n = n + 1
  }
}
n
```

```
## [1] 143
```

```r
#comparing the result of svm_1 and lda to see how many estimations are different
n = 0
for (i in 1:3162) {
  if (df2$pred_activity_type2_svm1[i] != df2$pred_activity_type2_lda$class[i]) {
    n = n + 1
  }
}
n
```

```
## [1] 127
```

```r
# Self-test: randomly split the training data into training set and test set
# to self-test the accuracy of the model
library(caret)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
## Loading required package: lattice
```

```r
set.seed(123)

# Split dataset into 70% train and 30% test sets
train_index <- createDataPartition(df$activity_type_factor, p = 0.7, list = FALSE)
train <- df[train_index, ]
test <- df[-train_index, ]
```

```r
#accuracy of svm_1
svm_model_1 <- svm(train[, c(3:563)], train$activity_type_factor)
trial_svm1 <- predict(svm_model_1, test[, c(3:563)])
n = 0
for (i in 1:length(test)) {
  if (trial_svm1[i] != test$activity_type_factor[i]) {
    n = n + 1
  }
}
1-n/length(test)
```

```
## [1] 0.9717314
```

```r
#accuracy of svm_2 (final algorithm)
svm_model_2 <- svm(train[, c(3:563)], train$activity_type_factor, type = "C-classification",
                   kernel = "linear")
trial_svm2 <- predict(svm_model_2, test[, c(3:563)])
n = 0
for (i in 1:length(test)) {
  if (trial_svm2[i] != test$activity_type_factor[i]) {
    n = n + 1
  }
}
1-n/length(test)
```

```
## [1] 0.9805654
```

```r
#accuracy of lda
lda_model_1 <- lda(activity_type_factor ~ ., train[, c(3:563,566)])
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```r
trial_lda <- predict(lda_model_1, test[, c(3:563)])
n = 0
for (i in 1:length(test)) {
  if (trial_lda$class[i] != test$activity_type_factor[i]) {
    n = n + 1
  }
}
1-n/length(test)
```

```
## [1] 0.9734982
```

```r
#accuracy of another svm model: svm_3
svm_model_3 <- svm(train$activity_type_factor ~ ., data = train[, c(3:563)],
                   kernel = "linear", cost = 1)

trial_svm3 <- predict(svm_model_3, test[, c(3:563)])
n = 0
for (i in 1:length(test)) {
  if (trial_svm3[i] != test$activity_type_factor[i]) {
    n = n + 1
  }
}
1-n/length(test)
```

```
## [1] 0.9805654
```

```r
#accuracy of random forest
library(randomForest)
rf_model_1 <- randomForest(activity_type_factor ~ ., train[, c(3:563, 566)])
trial_rf <- predict(rf_model_1, test[, c(3:563)])
n = 0
for (i in 1:length(test)) {
  if (trial_rf[i] != test$activity_type_factor[i]) {
    n = n + 1
  }
}
1-n/length(test)
```

```
## [1] 0.9646643
```

```r
#accuracy of another model: kernlab
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
svm_model_4 <- ksvm(train$activity_type_factor ~ ., train[, c(3:563)], kernel = "rbfdot")
trial_svm4 <- predict(svm_model_4, test[, c(3:563)])
n = 0
for (i in 1:length(test)) {
  if (trial_svm4[i] != test$activity_type_factor[i]) {
    n = n + 1
  }
}
1-n/length(test)
```

```
## [1] 0.9699647
```

```r
#accuracy of another svm model: svm5 using libsvm
svm_model_5 <- svm(train$activity_type_factor ~ ., data = train[, c(3:563)],
                   kernel = "radial")
trial_svm5 <- predict(svm_model_5, test[, c(3:563)])
n = 0
for (i in 1:length(test)) {
  if (trial_svm5[i] != test$activity_type_factor[i]) {
    n = n + 1
  }
}
1-n/length(test)
```

```
## [1] 0.9717314
```