1. Use the below given data set

Data Set

2. Perform the below given activities:

a. Create classification model using logistic regression model

# Load data

# install.packages('mlbench')

data(caret, package="mlbench")

bc <- BreastCancer[complete.cases(BreastCancer), ] # keep complete rows

# remove id column

bc <- bc[,-1]

# convert to numeric

for(i in 1:9) {

bc[, i] <- as.numeric(as.character(bc[, i]))

}

# Change Y values to 1's and 0's

bc$Class <- ifelse(bc$Class == "malignant", 1, 0)

bc$Class <- factor(bc$Class, levels = c(0, 1))

# Prep Training and Test data.

library(caret)

'%ni%' <- Negate('%in%') # define 'not in' func

options(scipen=999) # prevents printing scientific notations.

set.seed(100)

trainDataIndex <- createDataPartition(bc$Class, p=0.7, list = F)

trainData <- bc[trainDataIndex, ]

testData <- bc[-trainDataIndex, ]

# Class distribution of train data

table(trainData$Class)

# Down Sample

set.seed(100)

down\_train <- downSample(x = trainData[, colnames(trainData) %ni% "Class"],

y = trainData$Class)

table(down\_train$Class)

# Up Sample (optional)

set.seed(100)

up\_train <- upSample(x = trainData[, colnames(trainData) %ni% "Class"],

y = trainData$Class)

table(up\_train$Class)

# Build Logistic Model

logitmod <- glm(Class ~ Cl.thickness + Cell.size + Cell.shape, family = "binomial", data=down\_train)

summary(logitmod)

pred <- predict(logitmod, newdata = testData, type = "response")

pred

# Recode factors

y\_pred\_num <- ifelse(pred > 0.5, 1, 0)

y\_pred <- factor(y\_pred\_num, levels=c(0, 1))

y\_act <- testData$Class

# Accuracy

mean(y\_pred == y\_act) # 94%

b. verify model goodness of fit

##create two binary vectors of length 100   
x=sample(c(0,1),100, replace=T)  
y=sample(c(0,1),100, replace=T)

> ##create a 2x2 table with counts  
> xytab=table(x,y)  
> xytab  
   y  
x    0  1  
  0 24 29  
  1 26 21

> count=cbind(xytab[,2],xytab[,1])  
> count  
  [,1] [,2]  
0   29   24  
1   21   26

> xfactor=factor(c("0","1"))  
> xfactor  
[1] 0 1  
Levels: 0 1

> tmp3=glm(count~xfactor, family=binomial("logit"))  
> tmp3  
  
Call:  glm(formula = count ~ xfactor, family = binomial("logit"))  
  
Coefficients:  
(Intercept)     xfactor1    
     0.1892      -0.4028    
  
Degrees of Freedom: 1 Total (i.e. Null);  0 Residual  
Null Deviance:        1.005   
Residual Deviance: -4.441e-15     AIC: 12.72

> xydata=cbind(x,y)  
> xydata ## 100 rows, we are showing first 7  
       x y  
  [1,] 0 1  
  [2,] 0 1  
  [3,] 0 0  
  [4,] 0 1  
  [5,] 1 0  
  [6,] 0 1  
  [7,] 0 0.....

**> tmp1=glm(y~x, family=binomial("logit"))  
> tmp1**Call:  glm(formula = y ~ x, family = binomial("logit"))  
  
Coefficients:  
(Intercept)            x    
     0.1892      -0.4028    
  
Degrees of Freedom: 99 Total (i.e. Null);  98 Residual  
Null Deviance:        138.6   
Residual Deviance: 137.6     AIC: 141.6

c. Report the accuracy measures

d. Report the variable importance

e. Report the unimportant variables

f. Interpret the results

g. Visualize the results

options(digits = 7)

library(caret)

training <- read.csv("pml-training.csv")

testing <- read.csv("pml-testing.csv")

dim(training)

str(training$classe)

summary(training$classe)

names(training)[names(testing) != names(training)]

names(testing)[names(testing) != names(training)]

sum(is.na(training))

sum(is.na(testing))

predictorIdx <- c(grep("^accel", names(training)), grep("^gyros", names(training)), grep("^magnet", names(training)), grep("^roll", names(training)), grep("^pitch", names(training)), grep("^yaw", names(training)), grep("^total", names(training))) trainPredSet <- training[, c(predictorIdx, 160)] testPredSet <- testing[, c(predictorIdx, 160)] length(predictorIdx)

sum(names(testing)[predictorIdx] != names(training)[predictorIdx])

sum(is.na(trainPredSet))

sum(is.na(testPredSet))

nearZeroVar(trainPredSet[, -53], saveMetric = TRUE)

qplot(x = trainPredSet[, "accel\_belt\_x"], y = trainPredSet[, "accel\_arm\_x"], color = trainPredSet$classe)

set.seed(125) inTrain <- createDataPartition(y = trainPredSet$classe, p = 0.8, list = FALSE) cvTrain <- trainPredSet[inTrain, ] cvTest <- trainPredSet[-inTrain, ]

fitCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 10)

set.seed(125) modFit <- train(classe ~ ., data = cvTrain, method = "qda", preProcess = c("center", "scale"), trControl = fitCtrl)

print(modFit)

confusionMatrix(data = ptrain, reference = cvTrain$classe)

ptest <- predict(modFit, newdata = cvTest) equalPredTest <- (ptest == cvTest$classe) print(sum(equalPredTest)/length(equalPredTest))

confusionMatrix(data = ptest, reference = cvTest$classe)

testPrediction <- predict(modFit, newdata = testing) print(rbind(testing[1:20, 160], as.character(testPrediction)))