PROJECT2

We are performing different classification techniques for the given data set ‘project2.txt’ and estimating the misclassification rate by using cross-validation.

The given data set is project2.txt first we load the data using read.csv and there are no headers so header=FALSE and the columns are separated with ‘,’ so sep=’,’.

Now the first four columns represent four quantitative predictors. The fifth column stands for a categorical response, with two categories coded as 0 and 1.

The Dimension of the data set is 1375 rows and 5 columns.
we produce a matrix that contains all the pair wise correlations among the predictors in data set.

Here, we performed 4 types of classification:

- Logistic Regression
- QDA - Quadratic Discriminant Analysis
- LDA - Linear Discriminant Analysis
- KNN - K nearest neighbor

And done the K-Fold validation for all the four classifications from k=1 to k=10

sdm.prj2 = read.csv('/Users/vinithavudhayagiri/Downloads/project2.txt',sep = ",",header=FALSE)
dim(sdm.prj2)
names(sdm.prj2)

```
> sdm.prj2 = read.csv('/Users/vinithavudhayagiri/Downloads/project2.txt',sep = ",",header=FALSE)
> dim(sdm.prj2)
[1] 1372    5
> names(sdm.prj2)
[1] "V1" "V2" "V3" "V4" "V5"
```

summary(sdm.prj2)

```
> summary(sdm.prj2)
       V1                V2                V3                V4                V5
 Min.   :-7.0421   Min.   :-13.773   Min.   :-5.2861   Min.   :-8.5482   Min.   :0.0000
 1st Qu.:-1.7730   1st Qu.: -1.708   1st Qu.:-1.5750   1st Qu.:-2.4135   1st Qu.:0.0000
 Median : 0.4962   Median :  2.320   Median : 0.6166   Median :-0.5867   Median :0.0000
 Mean   : 0.4337   Mean   :  1.922   Mean   : 1.3976   Mean   :-1.1917   Mean   :0.4446
 3rd Qu.: 2.8215   3rd Qu.:  6.815   3rd Qu.: 3.1793   3rd Qu.: 0.3948   3rd Qu.:1.0000
 Max.   : 6.8248   Max.   : 12.952   Max.   :17.9274   Max.   : 2.4495   Max.   :1.0000
```

attach(sdm.prj2)

```
> attach(sdm.prj2)
The following objects are masked from sdm.prj2 (pos = 7):

    V1, V2, V3, V4, V5

The following objects are masked from sdm.prj2 (pos = 12):

    V1, V2, V3, V4, V5

The following objects are masked from sdm.prj2 (pos = 13):

    V1, V2, V3, V4, V5
```

cor(sdm.prj2)

```
> cor(sdm.prj2)
           V1         V2         V3          V4          V5
V1  1.0000000  0.2640255 -0.3808500  0.27681670 -0.72484314
V2  0.2640255  1.0000000 -0.7868952 -0.52632084 -0.44468776
V3 -0.3808500 -0.7868952  1.0000000  0.31884089  0.15588324
V4  0.2768167 -0.5263208  0.3188409  1.00000000 -0.02342368
V5 -0.7248431 -0.4446878  0.1558832 -0.02342368  1.00000000
```

LOGISTIC REGRESSION

Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurrence.

Logistic = glm (V5 ~ V1 + V2 + V3 + V4 , family = binomial ,data = sdm.prj2)
coef (Logistic)

```
> ##Logistic regression
> Logistic = glm ( V5 ~ V1 + V2 + V3 + V4 , family = binomial ,data = sdm.prj2 )
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> coef (Logistic)
(Intercept)          V1          V2          V3          V4
   7.321805   -7.859330   -4.190963   -5.287431   -0.605319
```

summary (Logistic)

```
> summary (Logistic)

Call:
glm(formula = V5 ~ V1 + V2 + V3 + V4, family = binomial, data = sdm.prj2)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.70001   0.00000   0.00000   0.00029   2.24614

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   7.3218     1.5589   4.697 2.64e-06 ***
V1           -7.8593     1.7383  -4.521 6.15e-06 ***
V2           -4.1910     0.9041  -4.635 3.56e-06 ***
V3           -5.2874     1.1612  -4.553 5.28e-06 ***
V4           -0.6053     0.3307  -1.830   0.0672 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1885.122  on 1371  degrees of freedom
Residual deviance:   49.891  on 1367  degrees of freedom
AIC: 59.891

Number of Fisher Scoring iterations: 12
```

summary (Logistic)$coef

```
> summary (Logistic)$coef
             Estimate Std. Error   z value     Pr(>|z|)
(Intercept)  7.321805  1.5588603  4.696896 2.641448e-06
V1          -7.859330  1.7383123 -4.521242 6.147788e-06
V2          -4.190963  0.9041488 -4.635258 3.564919e-06
V3          -5.287431  1.1611830 -4.553486 5.276415e-06
V4          -0.605319  0.3307210 -1.830301 6.720497e-02
```

summary (Logistic)$coef[, 4]

```
> summary (Logistic)$coef[, 4]
 (Intercept)           V1           V2           V3           V4
2.641448e-06 6.147788e-06 3.564919e-06 5.276415e-06 6.720497e-02
```

probability = predict (Logistic , type = "response")
probability[1:10]

```
> probability = predict (Logistic , type = "response")
> probability[1:10]
           1            2            3            4            5            6            7            8            9           10 
2.220446e-16 2.220446e-16 2.185822e-10 2.220446e-16 4.579103e-01 2.220446e-16 2.220446e-16 1.435064e-11 2.220446e-16 2.220446e-16
```

predict = rep ("0", 1372)
predict[probability > .6] = "1"
mean (predict == V5)
table (predict , V5)

```
> predict = rep ("0", 1372)
> predict[probability > .6] = "1"
> mean (predict == V5)
[1] 0.9912536
> table (predict , V5)
       V5
predict   0   1
      0 757   7
      1   5 603
```

Now we are splitting our data to train.data as training data and test.data as test data as per the probability of 0.45 and 0.55 with respectively.

data.sample <- sample(c(TRUE, FALSE), nrow(sdm.prj2), replace=TRUE, prob=c(0.45,0.55))
train.data <- sdm.prj2[!data.sample,]
dim(train.data)

```
> #splitting training data and test data
> data.sample <- sample(c(TRUE, FALSE), nrow(sdm.prj2), replace=TRUE, prob=c(0.45,0.55))
> train.data <- sdm.prj2[!data.sample, ]
> dim(train.data)
[1] 723   5
```

test.data<- sdm.prj2[data.sample,]
dim(test.data)

```
> test.data<- sdm.prj2[data.sample, ]
> dim(test.data)
[1] 649   5
```

final <- predict (Logistic , newdata = test.data, type = "response")

final <- ifelse(final > 0.5,1,0)
Error.data <- mean(final != test.data$V5)
print(paste('Accuracy', 1 - Error.data))

```
> final <- predict (Logistic , newdata = test.data, type = "response")
> final <- ifelse(final > 0.5,1,0)
> Error.data <- mean(final != test.data$V5)
> print(paste('Accuracy', 1 - Error.data))
[1] "Accuracy 0.99075500770416"
```

K-FOLD FROM 1 TO 10 CROSS VALIDATION FOR LOGISTIC REGRESSION

Here, we are validating our data set using logistic regression with the help of K-FOLD validation.

library(caret)
library(dplyr)
set.seed(100)
train.ctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
Logistic.fit<- train(factor(V5) ~ V1 + V2 + V3 + V4, data = sdm.prj2, method = "glm", family = "binomial", trControl=train.ctrl, tuneLength = 0)

```
> library(caret)
> library(dplyr)
> set.seed(100)
> train.ctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
> Logistic.fit<- train(factor(V5) ~ V1 + V2 + V3 + V4, data = sdm.prj2, method = "glm", family = "binomial", trControl=train.ctrl, tuneLength = 0)
There were 11 warnings (use warnings() to see them)
```

Logistic.fit

```
> Logistic.fit
Generalized Linear Model

1372 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1235, 1235, 1234, 1235, 1235, 1235, ...
Resampling results:

  Accuracy   Kappa
  0.9897916  0.9793315
```

Here the accuracy for logistic regression is **0.9897916**.

predict <- Logistic.fit$pred
predict$equal <- ifelse(predict$pred == predict$obs, 1,0)

```
fold <- predict %>%
  group_by(Resample) %>%
  summarise_at(vars(equal),
               list(Accuracy = mean))
fold
```

```
> predict <- Logistic.fit$pred
> predict$equal <- ifelse(predict$pred == predict$obs, 1,0)
> fold <- predict %>%
+   group_by(Resample) %>%
+   summarise_at(vars(equal),
+                list(Accuracy = mean))
> fold
# A tibble: 10 x 2
   Resample Accuracy
   <chr>       <dbl>
 1 Fold01      1
 2 Fold02      0.978
 3 Fold03      1
 4 Fold04      0.993
 5 Fold05      0.993
 6 Fold06      0.985
 7 Fold07      0.985
 8 Fold08      0.993
 9 Fold09      0.986
10 Fold10      0.985
```

QUADRATIC DISCRIMINANT ANALYSIS

Quadratic Discriminant Analysis (QDA) is a generative model. QDA assumes that each class follow a Gaussian distribution. The class-specific prior is simply the proportion of data points that belong to the class. The class-specific mean vector is the average of the input variables that belong to the class.

```
library(MASS)
library(ggplot2)
qda.fit = qda(V5~V1+V2+V3+V4, data=train.data)
qda.fit
```

```
> library(MASS)
> library(ggplot2)
> qda.fit = qda(V5~V1+V2+V3+V4, data=train.data)
> qda.fit
Call:
qda(V5 ~ V1 + V2 + V3 + V4, data = train.data)

Prior probabilities of groups:
        0         1 
0.5726141 0.4273859 

Group means:
         V1        V2        V3        V4
0  2.243078  4.181150 0.8822994 -1.108171
1 -1.843519 -1.372205 2.4850359 -1.031985
```

qda.class=predict(qda.fit,train.data)$class
table(qda.class)
mean(qda.class==V5)

```
> qda.class=predict(qda.fit,train.data)$class
> table(qda.class)
qda.class
  0   1 
404 319 
> mean(qda.class==V5)
[1] 0.5
Warning messages:
1: In `==.default`(qda.class, V5) :
  longer object length is not a multiple of shorter object length
2: In is.na(e1) | is.na(e2) :
  longer object length is not a multiple of shorter object length
```

K-FLOD FROM 1 TO 10 CROSS VALIDATION FOR QUADRATIC DISCRIMINANT ANALYSIS

We are validating our data set using Quadratic discriminant analysis with the help of K-FOLD validation.

library(caret)
library(dplyr)
set.seed(100)
train.ctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
qda.fit<- train(factor(V5) ~ V1 + V2 + V3 + V4, data = sdm.prj2, method = "qda", trControl=train.ctrl, tuneLength = 0)
qda.fit

```
> library(caret)
> library(dplyr)
> set.seed(100)
> train.ctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
> qda.fit<- train(factor(V5) ~ V1 + V2 + V3 + V4, data = sdm.prj2, method = "qda", trControl=train.ctrl, tuneLength = 0)
> qda.fit
Quadratic Discriminant Analysis

1372 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1235, 1235, 1234, 1235, 1235, 1235, ...
Resampling results:

  Accuracy   Kappa
  0.9839733  0.9676992
```

Here, the accuracy for Quadratic discriminant analysis is **0.9839733** comparing with logistic regression it has less accuracy.

predict <- qda.fit$pred
predict$equal <- ifelse(predict$pred == predict$obs, 1,0)

fold <- predict %>%
group_by(Resample) %>%
summarise_at(vars(equal),
list(Accuracy = mean))
fold

```
> predict <- qda.fit$pred
> predict$equal <- ifelse(predict$pred == predict$obs, 1,0)
> fold <- predict %>%
+   group_by(Resample) %>%
+   summarise_at(vars(equal),
+                list(Accuracy = mean))
> fold
# A tibble: 10 × 2
   Resample Accuracy
   <chr>       <dbl>
 1 Fold01      0.993
 2 Fold02      0.993
 3 Fold03      0.986
 4 Fold04      0.985
 5 Fold05      0.971
 6 Fold06      0.978
 7 Fold07      0.978
 8 Fold08      1
 9 Fold09      0.971
10 Fold10      0.985
```

LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis or Normal Discriminant Analysis or Discriminant Function Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modelling differences in groups i.e., separating two or more classes.

sdm.prj2[1:4] = scale(sdm.prj2[1:4])
lda <- lda(V5 ~ V1 + V2 + V3 + V4, data=train.data)
lda
lda.pred <- predict(lda, train.data)

```
> sdm.prj2[1:4] = scale(sdm.prj2[1:4])
> lda <- lda(V5 ~ V1 + V2 + V3 + V4, data=train.data)
> lda
Call:
lda(V5 ~ V1 + V2 + V3 + V4, data = train.data)

Prior probabilities of groups:
        0         1
0.5726141 0.4273859

Group means:
         V1        V2        V3        V4
0  2.243078  4.181150 0.8822994 -1.108171
1 -1.843519 -1.372205 2.4850359 -1.031985

Coefficients of linear discriminants:
           LD1
V1 -0.83944411
V2 -0.44353666
V3 -0.57038065
V4  0.02391648
> lda.pred <- predict(lda, train.data)
```

names(lda.pred)

```
> names(lda.pred)
[1] "class"     "posterior" "x"
```

head(lda.pred$class)

```
> head(lda.pred$class)
[1] 0 0 0 0 0 0
Levels: 0 1
```

head(lda.pred$posterior)

```
> head(lda.pred$posterior)
           0            1
3  0.9992789 7.211015e-04
8  0.9999053 9.473837e-05
9  0.9999998 1.869040e-07
10 0.9999923 7.711216e-06
12 0.9998563 1.437091e-04
15 0.9999999 6.659599e-08
```

mean(lda.pred$class == test.data$V5)

```
> mean(lda.pred$class == test.data$V5)
[1] 0.7966805
Warning messages:
1: In `==.default`(lda.pred$class, test.data$V5) :
  longer object length is not a multiple of shorter object length
2: In is.na(e1) | is.na(e2) :
  longer object length is not a multiple of shorter object length
```

K-FLOD FROM 1 TO 10 CROSS VALIDATION FOR LINEAR DISCRIMINANT ANALYSIS

We are validating our data set using Linear discriminant analysis with the help of K-FOLD validation.

library(caret)
library(dplyr)
set.seed(100)
train.ctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
lda.fit<- train(factor(V5) ~ V1 + V2 + V3 + V4, data = sdm.prj2, method = "lda", trControl=train.ctrl, tuneLength = 0)

lda.fit

```
> library(caret)
> library(dplyr)
> set.seed(100)
> train.ctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
> lda.fit<- train(factor(V5) ~ V1 + V2 + V3 + V4, data = sdm.prj2, method = "lda", trControl=train.ctrl, tuneLength = 0)
> lda.fit
Linear Discriminant Analysis

1372 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1235, 1235, 1234, 1235, 1235, 1235, ...
Resampling results:

  Accuracy   Kappa
  0.9766847  0.9530991
```

Here, the accuracy for Linear discriminant analysis is **0.9766847** comparing with logistic regression it has less accuracy.

predict <- lda.fit$pred
predict$equal <- ifelse(predict$pred == predict$obs, 1,0)

fold <- predict %>%
group_by(Resample) %>%
summarise_at(vars(equal),
list(Accuracy = mean))
fold

```
> predict <- lda.fit$pred
> predict$equal <- ifelse(predict$pred == predict$obs, 1,0)
>
> fold <- predict %>%
+   group_by(Resample) %>%
+   summarise_at(vars(equal),
+                list(Accuracy = mean))
> fold
# A tibble: 10 × 2
   Resample Accuracy
   <chr>       <dbl>
 1 Fold01      0.971
 2 Fold02      0.985
 3 Fold03      0.986
 4 Fold04      0.971
 5 Fold05      0.964
 6 Fold06      0.971
 7 Fold07      0.978
 8 Fold08      1
 9 Fold09      0.957
10 Fold10      0.985
```

KNN CLASSIFIER

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

library(e1071)
library(class)
library(caTools)

test = scale(test.data[, 1:4])
train = scale(train.data[, 1:4])
knn.clf = knn(train = train, test = test, cl = train.data$V5, k = 5)
clf<- table(test.data$V5,knn.clf)
clf

```
> library(class)
> library(caTools)
>
> test = scale(test.data[, 1:4])
> train = scale(train.data[, 1:4])
> knn.clf = knn(train = train, test = test, cl = train.data$V5, k = 5)
> clf<- table(test.data$V5,knn.clf)
> clf
   knn.clf
      0   1
  0 347   1
  1   0 301
```

mean (test.data$V5 ==knn.clf)

```
> mean (test.data$V5 ==knn.clf)
[1] 0.9984592
```

knn.clf = knn(train = train, test = test, cl = train.data$V5, k = 10)
clf<- table(test.data$V5,knn.clf)
clf

mean (test.data$V5 ==knn.clf)

```
> knn.clf = knn(train = train, test = test, cl = train.data$V5, k = 10)
> clf<- table(test.data$V5,knn.clf)
> clf
   knn.clf
      0   1
  0 346   2
  1   0 301
> mean (test.data$V5 ==knn.clf)
[1] 0.9969183
```

K-FOLD FROM 1 TO 10 CROSS VALIDATION FOR KNN

We are validating our data set using KNN with the help of K-FOLD validation.

library(caret)
library(dplyr)
set.seed(100)
train.ctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
knn.fit<- train(factor(V5) ~ V1 + V2 + V3 + V4, data = sdm.prj2, method = "knn", trControl=train.ctrl, tuneLength = 0)
knn.fit

```
> library(caret)
> library(dplyr)
> set.seed(100)
> train.ctrl <- trainControl(method = "cv", number = 10, savePredictions=TRUE)
> knn.fit<- train(factor(V5) ~ V1 + V2 + V3 + V4, data = sdm.prj2, method = "knn", trControl=train.ctrl, tuneLength = 0)
> knn.fit
k-Nearest Neighbors

1372 samples
   4 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1235, 1235, 1234, 1235, 1235, 1235, ...
Resampling results:

  Accuracy   Kappa
  0.9985454  0.9970583

Tuning parameter 'k' was held constant at a value of 5
```

Here, the accuracy for KNN is **0.9985454** comparing with other classifications it has high accuracy and low-test misclassification rate.

predict <- knn.fit$pred
predict$equal <- ifelse(predict$pred == predict$obs, 1,0)

fold <- predict %>%
group_by(Resample) %>%
summarise_at(vars(equal),
list(Accuracy = mean))
fold

```
> predict <- knn.fit$pred
> predict$equal <- ifelse(predict$pred == predict$obs, 1,0)
>
> fold <- predict %>%
+   group_by(Resample) %>%
+   summarise_at(vars(equal),
+                list(Accuracy = mean))
> fold
# A tibble: 10 × 2
   Resample Accuracy
   <chr>       <dbl>
 1 Fold01      1
 2 Fold02      1
 3 Fold03      1
 4 Fold04      1
 5 Fold05      1
 6 Fold06      1
 7 Fold07      1
 8 Fold08      0.993
 9 Fold09      0.993
10 Fold10      1
```

From the above four classification techniques we performed, KNN gives the best result as we have less misclassification rate.

Team Members:
Alekhya Monavarthi -50469035
Preethi Abhilasha Vaddi-50483865
Shanawaz Pathan-50471026
Vinitha Vudhayagiri-50478854