EAS508 Classification Project

2023-11-29

Team Members:

Alifiya Aziz Batterywala (50539267)

Karthika Rajan Nair (50510469)

Meghna Ramesan Aalingil (50496345)

Prachi Vijay Patil (50539237)

Sai Shanmukh Puppala (50537991)

```r
library(MASS)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.2

## Loading required package: lattice
```

```r
library (glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.2

## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.3.2
```

```r
library(class)

# Load your data
data <- read.csv("project2.csv", header = TRUE)
colnames(data) <- c("X1", "X2", "X3", "X4", "Y")
```

Model 1- Linear Discriminant Analysis (LDA)

```r
# Splitting the data into training and testing sets
set.seed(1)
train <- sample(1:nrow(data), 0.8 * nrow(data))
train.data <- data[train, ]
test.data <- data[-train, ]

# Fit LDA model
lda.model <- lda(train.data[, -ncol(train.data)],
                 grouping = train.data[, ncol(train.data)])
# Make predictions
lda.predictions <- predict(lda.model, test.data[, -ncol(test.data)])
# Evaluate the model
lda.accuracy <- sum(test.data[,ncol(test.data)]==
                      lda.predictions$class)/nrow(test.data)
lda.accuracy
```

```
## [1] 0.9672727
```

```r
table(lda.predictions$class, test.data$Y)
```

```
##
##       0   1
##   0 141   0
##   1   9 125
```

```r
# Cross-Validation for misclassification Rate
# 10-fold cross-validation
control <- trainControl(method = "cv", number = 10)
cv_model <- train(as.factor(Y) ~ ., data = data, method = "lda", trControl = control)
cv_results <- cv_model$results
# Classification Rate
cv_classification_rate <- mean(cv_results$Accuracy)
cv_classification_rate
```

```
## [1] 0.9766529
```

```r
# Misclassification Rate
misclassification_rate <- 1 - mean(cv_results$Accuracy)
misclassification_rate
```

```
## [1] 0.02334709
```

LDA Model Accuracy : 0.9672727

Accuracy using Cross Validation with 10 folds : 0.9766529

Misclassification Rate using Cross Validation : 0.02334709

Model 2- Quadratic Discriminant Analysis (QDA)

```r
# Splitting the data into training and testing sets
set.seed(1)
train <- sample(1:nrow(data), 0.8 * nrow(data))
train.data <- data[train, ]
test.data <- data[-train, ]

# Fit QDA model
qda.model <- qda(train.data[, -ncol(train.data)],
                 grouping = train.data[, ncol(train.data)])
# Make predictions
qda.predictions <- predict(qda.model,
                           test.data[, -ncol(test.data)])
# Evaluate the model
qda.accuracy <- sum(test.data[,ncol(test.data)]
                    == qda.predictions$class) / nrow(test.data)
qda.accuracy
```

```
## [1] 0.9854545
```

```r
table(qda.predictions$class, test.data$Y)
```

```
##
##       0   1
##   0 146   0
##   1   4 125
```

```r
# Cross-Validation for misclassification Rate
# 10-fold cross-validation
control <- trainControl(method = "cv", number = 10)
cv_model <- train(as.factor(Y) ~ ., data = data,
                  method = "qda", trControl = control)
cv_results <- cv_model$results
# Classification Rate
cv_classification_rate <- mean(cv_results$Accuracy)
cv_classification_rate
```

```
## [1] 0.9839469
```

```r
# Misclassification Rate
misclassification_rate <- 1 - mean(cv_results$Accuracy)
misclassification_rate
```

```
## [1] 0.0160531
```

QDA Model Accuracy : 0.9854545

Accuracy using Cross Validation with 10 folds : 0.9846821

Misclassification Rate using Cross Validation : 0.01531789

Model 3- Logistic Regression

```
glm.fits <- glm(formula = Y ~ X1 + X2 + X3 + X4,data=data,family = "binomial")
```

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

```
summary(glm.fits)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2 + X3 + X4, family = "binomial", data = data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   7.3218     1.5589   4.697 2.64e-06 ***
## X1           -7.8593     1.7383  -4.521 6.15e-06 ***
## X2           -4.1910     0.9041  -4.635 3.56e-06 ***
## X3           -5.2874     1.1612  -4.553 5.28e-06 ***
## X4           -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: 1883.945  on 1370  degrees of freedom
## Residual deviance:   49.891  on 1366  degrees of freedom
## AIC: 59.891
##
## Number of Fisher Scoring iterations: 12
```

```
coef(glm.fits)
```

```
## (Intercept)          X1          X2          X3          X4
##    7.321805   -7.859330   -4.190963   -5.287431   -0.605319
```

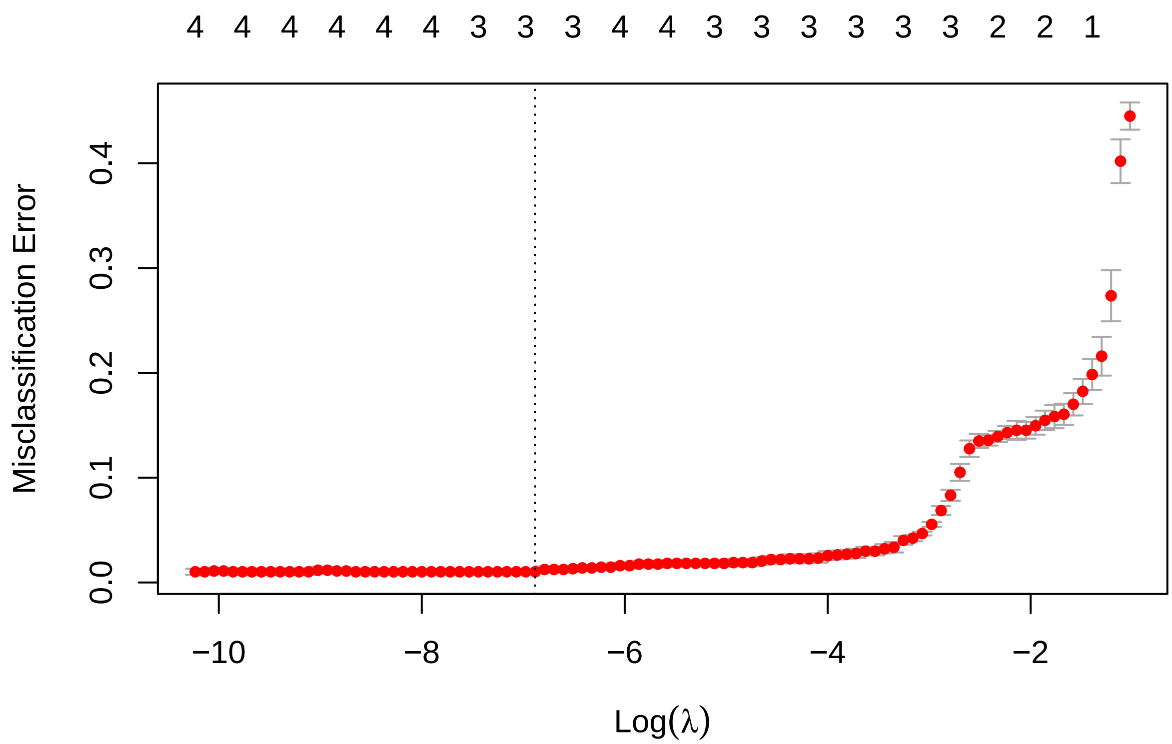
```
summary(glm.fits)$coef
```

```
##              Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)  7.321805  1.5588613  4.696893 2.641489e-06
## X1          -7.859330  1.7383134 -4.521239 6.147871e-06
## X2          -4.190963  0.9041494 -4.635255 3.564970e-06
## X3          -5.287431  1.1611837 -4.553483 5.276488e-06
## X4          -0.605319  0.3307211 -1.830300 6.720505e-02
```

```
predictions <- predict(glm.fits, type = "response")
predicted_classes <- ifelse(predictions > 0.5, 1, 0)
conf_matrix <- table(Actual = data$Y, Predicted = predicted_classes)
Accuracy <- accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
Accuracy*100
```

```
## [1] 99.19767
```

```r
# Cross-Validation for misclassification Rate
X <- as.matrix(data[, c('X1', 'X2', 'X3', 'X4')])
Y <- data$Y
cv.glmnet.fit <- cv.glmnet(x = X, y = Y, family = "binomial",
                           type.measure = "class", nfolds = 5)
best_lambda <- cv.glmnet.fit$lambda.min
plot(cv.glmnet.fit)
```




```r
predictions <- predict(cv.glmnet.fit, newx = X,
                       s = "lambda.min", type = "response")
predicted_classes <- ifelse(predictions > 0.5, 1, 0)
misclassification_error_rate <- mean(predicted_classes != as.numeric(Y))
misclassification_error_rate
```

```
## [1] 0.00948213
```

```r
1- misclassification_error_rate
```

```
## [1] 0.9905179
```

Accuracy for Logistic Regression : 99.19825%

Accuracy Rate using Cross Validation : 0.9905248

Misclassification Error Rate obtained : 0.009475219

Model 4- Naive Bayes

```r
set.seed(1)
# Create a trainControl object for 5-fold cross-validation
ctrl <- trainControl(method = "cv", number = 5)

# Train the Naive Bayes model using cross-validation
nb_model <- train(
  x = data[, 1:4],
  y = as.factor(data$Y),
  method = "naive_bayes",
  trControl = ctrl
)
nb_model
```

```
## Naive Bayes
##
## 1371 samples
##    4 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1097, 1097, 1097, 1096, 1097
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.8417200  0.6775942
##    TRUE      0.9175793  0.8321158
##
## 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 = TRUE
##  and adjust = 1.
```

```r
pred_nb <- predict(nb_model, newdata = data[, 1:4])
conf_matrix_nb <- confusionMatrix(pred_nb, as.factor(data$Y))
conf_matrix_nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 725  67
##          1  36 543
##
##                Accuracy : 0.9249
##                  95% CI : (0.9096, 0.9383)
##     No Information Rate : 0.5551
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.8471
##
##  Mcnemar's Test P-Value : 0.003117
##
##             Sensitivity : 0.9527
##             Specificity : 0.8902
##          Pos Pred Value : 0.9154
##          Neg Pred Value : 0.9378
##              Prevalence : 0.5551
##          Detection Rate : 0.5288
##    Detection Prevalence : 0.5777
##       Balanced Accuracy : 0.9214
##
##        'Positive' Class : 0
##
```

```r
misclassification_rate_nb <- 1 - conf_matrix_nb$overall["Accuracy"]
misclassification_rate_nb
```

```
##   Accuracy
## 0.07512764
```

Accuracy of the Naive Bayes model : 92.57%.

Misclassification rate using Cross Validation with 5 folds : 7.43%.

Sensitivity (True Positive Rate): 95.28%

Specificity (True Negative Rate): 89.18%

Model 5- KNN

```r
train_test_split <- sample(1:nrow(data), size = 0.8*nrow(data))
train.X <- data[train_test_split, c('X1', 'X2', 'X3', 'X4')]
test.X <- data[-train_test_split, c('X1', 'X2', 'X3', 'X4')]
train.Y <- data[train_test_split, 'Y']
test.Y <- data[-train_test_split, 'Y']
table(data$Y)
```

```
##
##   0   1
## 761 610
```

```r
table(train.Y)
```

```
## train.Y
##   0   1
## 613 483
```

```r
table(test.Y)
```

```
## test.Y
##   0   1
## 148 127
```

```r
#KNN model with k=1
knn.pred <- knn (train.X, test.X, train.Y , k = 1)
table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred   0   1
##        0 148   0
##        1   0 127
```

```r
(151+123)/275
```

```
## [1] 0.9963636
```

```r
#5-fold cross validation on KNN with k=1
data$Y <- as.factor(data$Y)
control <- trainControl(method="cv", number=5)
knnFit <- train(Y~., data=data, method="knn", trControl=control, tuneGrid=expand.grid(k=1))
print(knnFit)
```

```
## k-Nearest Neighbors
##
## 1371 samples
##    4 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1097, 1097, 1097, 1096, 1097
## Resampling results:
##
##   Accuracy   Kappa
##   0.9992701  0.9985236
##
## Tuning parameter 'k' was held constant at a value of 1
```

```r
#KNN model with k=3
knn.pred <- knn (train.X, test.X, train.Y , k = 3)
table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred   0   1
##        0 148   0
##        1   0 127
```

```r
(152+123)/275
```

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

```r
#5-fold cross validation on KNN with k=3
data$Y <- as.factor(data$Y)
control <- trainControl(method="cv", number=5)
knnFit <- train(Y~., data=data, method="knn", trControl=control, tuneGrid=expand.grid(k=3))
print(knnFit)
```

```
## k-Nearest Neighbors
##
## 1371 samples
##    4 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1097, 1097, 1097, 1097, 1096
## Resampling results:
##
##   Accuracy   Kappa
##   0.9992701  0.9985236
##
## Tuning parameter 'k' was held constant at a value of 3
```

Accuracy of KNN with k = 1 : 0.996

Accuracy of KNN with k = 3 : 1

Accuracy of 5-fold Cross Validation accuracy of KNN with k = 1 : 0.9992701

Misclassification rate of 5-fold Cross Validation accuracy of KNN with k = 1 : 0.0007

Accuracy of 5-fold Cross Validation accuracy of KNN with k = 3 : 0.9992727

Misclassification rate of 5-fold Cross Validation accuracy of KNN with k = 3 : 0.0007

Model 6- Support Vector Machine (SVM)

```r
# Split the data into training and testing sets (80-20 split)
set.seed(1)
i <- sample(1:nrow(data), 0.8 * nrow(data))
train_data <- data[i, ]
test_data <- data[-i, ]

# Split the training and testing sets into predictors (X) and response (y)
X_train <- train_data[, 1:4]
y_train <- train_data[, 5]
X_test <- test_data[, 1:4]
y_test <- test_data[, 5]
```

```r
# Create a trainControl object for 10-fold cross-validation
ctrl <- trainControl(method = "cv", number = 2)

# Train the SVM model using cross-validation
svm_model_cv <- train(
  x = X_train,
  y = as.factor(y_train),
  method = "svmRadial",
  trControl = ctrl,
  preProcess = c("center", "scale"),
  tuneLength = 2
)

# Print the cross-validated results
svm_model_cv
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1096 samples
##    4 predictor
##    2 classes: '0', '1'
##
## Pre-processing: centered (4), scaled (4)
## Resampling: Cross-Validated (2 fold)
## Summary of sample sizes: 549, 547
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.9945222  0.9889132
##   0.50  0.9963504  0.9926092
##
## Tuning parameter 'sigma' was held constant at a value of 0.4181747
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.4181747 and C = 0.5.
```

```r
# Make predictions on the test set
svm_predictions_cv <- predict(svm_model_cv, newdata = X_test)

# Confusion matrix for SVM model
svm_conf_matrix_cv <- confusionMatrix(svm_predictions_cv, as.factor(y_test))
svm_conf_matrix_cv
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 150   0
##          1   0 125
##
##                Accuracy : 1
##                  95% CI : (0.9867, 1)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5455
##          Detection Rate : 0.5455
##    Detection Prevalence : 0.5455
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

Accuracy with 10 fold Cross Validation: 1

Misclassification with 10 fold Cross Validation: 0

CONCLUSION

In conclusion, the classification project undertook a comprehensive assessment of various models to determine the most effective one based on minimal misclassification error and maximum accuracy.

The QDA, Logistic Regression, and KNN (with k = 3) models showed exceptionally high performance, with near-perfect accuracy. KNN with k = 3 stands out with a perfect accuracy score. Along with KNN, SVM also gives the prefect 100% accuracy with 0% misclassification error. QDA achieved an accuracy of 98.55%, Logistic Regression attained an impressive 99.20% accuracy with a mere 0.95% misclassification rate, and KNN with both k values exhibited near-perfect accuracy of 99.6% and 100%, respectively. The LDA model also performed well, but its accuracy is slightly lower than the others. Naive Bayes exhibited the lowest accuracy and highest misclassification rate, suggesting it may not be as suitable for this dataset. The model with perfect cross-validation results (100% accuracy and 0% misclassification) is ideal, but it's important to consider the risk of overfitting, especially with a perfect score. This could indicate that the model is too closely fitted to the specific nuances of the training data, potentially reducing its generalizability to new data. KNN with k = 3 appears to be the best predictive model for this dataset, given its perfect accuracy and excellent performance under cross-validation. However, this perfect score warrants a cautious approach to ensure that the model is not over-fitted.

In summary, while Support Vector Machines, as well as KNN (k = 3), seem to be the best models based on the given metrics, it's important to consider the context of the data and the potential for overfitting. The Logistic Regression model provides a slightly more conservative yet highly accurate alternative.